Task-based Adaptation of Graphical Content in Smart Visual Interfaces

Dissertation

zur
Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von

Dipl.-Inf. Georg Fuchs
aus Bad Oldesloe

Bad Oldesloe, 11. April 2011

urn:nbn:de:gbv:28-diss2011-0171-1

Principal Advisor:	Prof. Dr.-Ing. habil. Heidrun Schumann *University of Rostock, Germany*
External Reviewers:	Prof. Dr.-Ing. habil. Peter Forbrig *University of Rostock, Germany*
	Prof. Mag. Dr. Silvia Miksch *Vienna University of Technology, Austria*
Date of Defense:	June 16th, 2011

Abstract

Visual representation has always been an important communication medium to convey complex facts. In order to be effective, a visual representation must be adapted to its respective context of use. This is a crucial aspect especially in smart visual interfaces — context-sensitive user interfaces that present to the user as specifically as possible those information and with the level of detail required for her current situation. Of the influencing factors contributing to this context of use, the characteristics of the content from which the representation is generated and the user's task at hand are the most important ones.

The goal of this thesis is to provide a basic approach to the task-based adaptation of visual representations from graphical content in smart visual interfaces, and to detail concrete solution approaches to associated challenges. The objective of the approach thereby is to adapt the graphical content associated with the task at hand automatically in such a way that the content's visual representation provides a good initial view with respect to information relevant to the current working step.

In this thesis, two core concepts are introduced that facilitate the automatic generation of task-specific visual representations from different types of graphical content based on a suitable task description. Starting point of the approach is a task model representing a hierarchical decomposition of the task at hand into interrelated subtasks. The PAGE/FEATURE concept represents a method to enrich this model with the specification of graphical content associated with individual working steps as well as relevant representation aspects. An adaptation pipeline provides the bridge between the conceptual task context specification in the enriched task model and concrete display techniques comprising the functional building blocks of smart visual interfaces.

It is discussed how these concepts are applied to graphical content of each of the four principal visual data types raster images, 2D vector and 3D graphics as well as data visualizations. This includes a review on how existing display techniques can be integrated into the adaptation pipeline approach.

Moreover, several novel display techniques are introduced that address previously open challenges with respect to task-driven adaptation of visual representations in smart visual interface. The developed concepts and techniques have been practically employed in the scope of a joint industry-academia research project underlining their utility for smart visual interface design.

Keywords: smart visual interface, task-driven, visual representation, adaptation, scalable graphics, scalability, task model, smart graphics

CR Classification: H.5.2, I.3.3, I.3.6

Zusammenfassung

Visuelle Repräsentationen sind schon immer ein wichtiges Medium zur Kommunikation komplexer Fakten gewesen. Um dies effektiv zu gewährleisten, muss eine Repräsentation an ihren jeweiligen Nutzungskontext angepasst werden. Ein wichtiger Aspekt ist dies insbesondere im Umfeld sog. Smart Visual Interfaces — kontextsensitive Nutzungsschnittstellen, die dem Anwender möglichst genau die Informationen in dem Detailgrad darbietet, wie es seine momentane Situation erfordert. Von den Einflussfaktoren, welche diesen Nutzungskontext definieren, sind Eigenschaften des darzustellenden Inhalts sowie der aktuellen Nutzeraufgabe am wichtigsten.

Die vorliegende Arbeit hat zum Ziel, einen grundlegenden Ansatz zur aufgabenbasierten Anpassung der visuellen Repräsentation grafischer Inhalte innerhalb Smart Visual Interfaces zu entwerfen, sowie konkrete Lösungen zugehöriger Herausforderungen im Detail zu betrachten. Der vorgestellte Ansatz zielt dabei darauf ab, eine automatische Anpassung des graphischen Inhalts durchzuführen dergestalt, dass die Inhaltsrepräsentation eine gute initiale Ansicht in Bezug auf die für die aktuelle Aufgabe benötigten Informationen darstellt.

In dieser Arbeit werden dazu zwei grundlegende Konzepte vorgestellt, die eine solche Anpassung verschiedener Inhaltstypen auf Grundlage geeigneter Aufgabenbeschreibungen ermöglichen. Ausgangspunkt ist ein Aufgabenmodell, welches eine hierarchische Dekomposition der aktuellen Aufgabe in zueinander in Beziehung stehende Teilaufgaben darstellt. Das PAGE/FEATURE-Konzept ist eine Methode, dieses Ausgangsmodell um entsprechende Informationen zu einzelnen Arbeitsschritten zugeordneten grafischen Inhalten sowie darstellungsrelevanter Aspekte anzureichern. Eine Anpassungspipeline stellt dabei eine Verbindung zwischen der Spezifikation des Aufgabenkontextes auf der konzeptionellen Ebene des erweiterten Aufgabenmodells einerseits und konkreten Darstellungstechniken als funktionale Bausteine Smart Visual Interfaces anderseits dar.

Es wird weiterhin erörtert, wie diese Konzepte auf graphische Inhalte der vier prinzipiellen visuellen Datentypen Rastergrafiken, 2D Vektor- und 3D-Grafiken sowie Datenvisualisierung anwendbar sind. Dies schließt eine Diskussion darüber ein, wie bereits bestehende Darstellungstechniken in die Anpassungspipeline integriert werden können.

Weiterhin werden mehrere neue Darstellungstechniken vorgestellt, welche bislang offene Herausforderungen in Bezug auf die aufgabenbasierte Anpassung visueller Repräsentationen innerhalb Smart Visual Interfaces adressieren. Die hier entwickelten Ansätze und Techniken wurden im Rahmen eines industriell-universitärem Verbundprojekt verwendet, was ihre praktische Anwendbarkeit für den Entwurf von Smart Visual Interfaces belegt.

Acknowledgments

My sincerest thanks go to all the people who advised and encouraged me while conducting, and especially, writing up this thesis. First and foremost, special thanks go to my supervisor Heidi Schumann. Her constructive criticism of my work and her ability to put thinks in perspective have been very helpful along the long way from my first ideas to the completion of this thesis. I also want to thank Peter Forbrig from University of Rostock and and Silvia Miksch from Vienna University of Technology for their acceptance to review my thesis. In addition to my advisors and reviewers, I want to express my gratitude to my colleagues and friends from the Computer Graphics and Visual Computing Group. Not only have they been very enjoyable and inspiring company, they also gave me uncounted opportunities for fruitful discussion and helped in scrutinizing my work. Here I would like to mention by name Hans-Jörg Schulz, Conrad Thiede, Christian Tominski and Daniel Reichart with whom I shared many ideas. Furthermore, I thank all my students that contributed bringing several ideas to life through their research projects and master theses, and my student assistants Steffi Biederstädt, Daniela Stüwe and Mathias Schröder that helped create many of the software artifacts.

Lastly, and most importantly, I want to thank my beloved Gabi, my family and all friends for their mental and logistical support. This thesis could not have been written without you.

Contents

List of Figures

1 Introduction

Visual representation has always been an important communication medium to convey complex facts. A visual representation can convey particular information by stressing its message subject over the faithful representation of reality. This does not only apply to arts e.g., drawings, paintings and photographs, but even more so to technical and scientific applications, such as technical drawings or schematics, and visualization.

With the continuing pervasion of networked computer devices, the use of computer-generated visual representations for communication is also increasing. At the same time, today's computer-aided collecting, processing, and presenting of information results in ever larger data sets and more complex models. As a consequence, it is often no longer sufficient to generate a static "one-size-fits-all" visual representation for communication of these information. Rather, a visual representation must be adapted to its respective context of use.

For this reason, the field of so-called *Smart Graphics* has drawn increasing research interest over recent years. The aim of Smart Graphics is

> "the interdisciplinary approach to the design, generation, presentation and interaction with 2D and 3D graphical interfaces in a manner that is sensitive to technological, computational and cognitive constraints. [...] Smart Graphics aims to move beyond the current requirement that designers anticipate every data, task and technological scenario, and instead facilitate the dynamic generation and presentation of content" [BKO00].

Besides an improved visualization of large data, a key goal of Smart Graphics is the development of efficient graphical interfaces i.e., *Smart Visual Interfaces*. A Smart Visual Interface thus is a context-sensitive user interface that presents to the user exactly those information and with the level of detail required for her current situation. It must therefore be determined WHAT content (which information) must be presented WHY (which task) to WHOM (user), WHEN (in the context of composite workflows)and WHERE (which output device). Based on these questions a visual representation is generated, and as the context changes, the representation adapts accordingly.

These '5Ws' can be considered an informal, high-level categorization of the influencing factors defining the Smart Visual Interface's *Context of Use*. Their adequate consideration determines whether the intended communicative goal associated with a visual representation can be attained. However, the multitude of influencing factors usually does not permit a complete solution. For this reason, contemporary approaches for the systematic generation of visual representations focus on some aspects while neglecting others.

It is generally accepted that the WHAT and WHY are by far the most important factors to be addressed. This is also well reflected in related literature. Most approaches do

indeed consider the characteristics of the data to be displayed (cf. e.g., [Mac86, Shn96, FFIT00]). Surprisingly few, however, contemplate the goal a user pursues with a visual representation, and even those that do incorporate only tasks on a relatively low level like locating and comparing values (e.g., [Shn96, FFIT00, ZCF02]). This is despite the fact that the WHY aspect has a crucial impact on whether a given visual representation is in fact expressive and effective. A recent publication [Mun09] on a conceptual model for design and evaluation of visualizations even goes as far as stating that without adequate consideration of the user's task (as the outermost of four nested abstraction layers), a visual representation is bound to fail is communicative goal.

In the same paper, the authors note that the term *task* is deeply overloaded in literature as a catch-all phrase for widely different WHY aspects. They argue that user tasks can be understood on four different levels of granularity: high- and low-level domain, as well as high- and low-level abstract tasks. The relevant distinction here is that domain levels capture the semantics of a process (e.g. 'find a cure for a disease' as a high-level goal in the medical domain), whereas abstract tasks describe a work flow decomposition on a purely functional level (e.g. 'compare values' or 'correlate data points').

However, almost all approaches from literature either provide a concrete solution to a specific domain problem or, if attempting to provide a generic framework for visual representation design, contend themselves with task descriptions at the granularity of low-level abstract tasks. As a result, task decompositions are often strongly data-driven and do not capture the overall structure of a domain-specific work flow – like temporal or causal relationships between individual sub-tasks – well, if at all. This, too, is an accepted problem in the context of designing adequate visual representations. The improved consideration of "semantics of the visualization process" has consequently been identified as one of the top ten challenges in the field of Visual Analytics [TC05, Kei05, TC06].

Approach Outline and Thesis Contributions

In this thesis several contributions to this challenge are presented. Specifically, it investigates the dynamic adaptation of visual representations to a given task as a vital aspect of Smart Visual Interfaces. A focus is on applications where visual representations are consumed as a source of supplemental information in the course of complex work flows, like in so-called e-Manuals described in [FRSF06], rather than being the result of a deliberate user effort as in typical data analysis (exploratory visualization) scenarios. In the former, both the structure of the work flow and the data associated with it are largely predetermined. In the scope of Smart Visual Interfaces as defined above it is still important, though, to provide the user with a good initial view for her task at hand to minimize navigation workload. The main challenges, therefore, are:

- What is an adequate task description that can express both the high-level structure of a work flow as well as the data-driven, low-level aspects required to actually generate the visual representation accordingly?

- How is the adaptation process controlled by the task description? This is strongly influenced by the "visual data type" of the representation associated with the task at hand.

To address the first issue, task modeling approaches from other disciplines are utilized in the generation process. In the field of software design it is commonly accepted that software development has to start with an analysis of the problem domain users work in [FBD+05]. To describe this problem domain in the field of human computer interaction, task models are widely used [Con03]. They form the foundation of several proposals for User Interface (UI) design that utilize task models to formalize the specification, and automate generation of an application's user interface [Sta00]. In these model-driven software design (MDD) processes first a model of the task at hand is developed, from which an (abstract) dialog model is derived through model transformation. The latter may then be used to generate concrete dialogs based on platform-specific widgets [PS02].

However, almost all approaches in this field relate to modeling of so-called Windows, Icons, Menus, Pointing Device (WIMP) interfaces, and corresponding interaction through them. Although the need for "post-WIMP" UI specifically for graphic-intense applications such as Computer-Aided Design (CAD) is well recognized [Dam97], so far only few task modeling approaches tackle the dynamic adaptation of predominantly graphical data (as opposed to e.g., UIs for manipulating spreadsheets or data tables).

Therefore, concepts to bridge this "modeling gap" are presented here. The core idea is to use hierarchical task models that capture the structure and composition of domain tasks on different abstraction levels as a starting point. These models are then enriched with appropriate descriptions of the task context with respect to graphical content to support the adaptation of visual representations associated with the task at hand [FRSF06, FS06]. In this thesis in particular, the Concur Task Tree (CTT) notation [PMM97] is used, which is wide-spread in software engineering. This allows embedding the task-driven adaptation of visual representations in established MDD tool chains as a contribution towards creating Smart Visual Interfaces. These concepts have been practically employed in the scope of a joint industry-academia research project [LFSa, LFSb].

Further, the amount of control over the adaptation process depends on the "visual data type". In this thesis, the visual data type is a broad categorization of the input data for the adaptation process. This has a direct impact on what operations are supported for the adaptation of the visual representation, and what types of annotations are feasible in the CTT model. It is divided into raster graphics (only imaging operations) [FS06, FLH+06], vector graphics (manipulation of vector primitives) [FSS07], 3D graphics (added control over the rendering process) [FRSF06, FHS08], and information visualizations (control over the entire image generation process including data filtering and visual mapping) [TFS08b, TFS08a].

Thesis Structure

In the following Chapter 2, relevant basics will be discussed including a clarification of important terms and concepts which have overloaded meanings in literature. It also proposes a categorizations of both the influencing factors that constitute a visual representation's context of use, and of the four principal "visual data types" of graphical content according to their inherent scalability and thus, adaptability. Chapter 2 further reflects on related work regarding both task modeling approaches in interfaces design,

as well as on visual representations that are scalable with respect to their context of use according to the proposed categorization, the task at hand in particular.

Based on this review Chapter 3 derives a detailed problem analysis that identifies relevant challenges and open research questions in creating smart visual interfaces based on task models. As the main contribution, Chapter 3 then presents the basic approach to task-driven adaptation of visual representations that has been developed in this thesis. It is based on two core concepts, a scheme for enriching general task models to describe the task context, and an adaptation pipeline to effect adaptation of graphical content. This basic approach provides a framework for the design of smart visual interfaces utilizing different concrete display technique suitable for the four principal visual data types of the interface's graphical content.

To this end, building on the basic approach Chapters 4–7 examine how adaptation control is effected for the four visual data types raster images, vector graphics, 3D graphics and abstract data visualization, respectively. Each chapter includes a discussion on how the respective data type is prepared to provide the necessary scalability, and reviews corresponding authoring tools that have been developed in the course of this thesis to support content preparation. Each chapter further proposes novel "Smart-X" display techniques that have been designed specifically to provide task-driven adapted representations within smart visual interfaces.

Chapter 8 briefly reviews several prototypical implementations of the proposed basic approach for use cases that have been addressed in the scope of two industry-academia research projects that framed the work presented in this thesis.

Chapter 9 concludes with a summary of the developed concepts and gives an outlook on future research directions and possible subsequent developments.

2 Basics

In this chapter, first relevant terms that are used throughout the thesis are introduced and disambiguated where necessary. Then, two principal application scenarios are introduced that illustrate both the need for, and requirements to, scalable visual representations. Based on these, related work in the problem domain of task modeling and task-driven development of user interfaces is reviewed. Strengths and weaknesses of existing approaches with respect to the handling of visual representations are discussed.

2.1 Definition of Terms

As the title implies, the work presented here is concerned with the *task-driven adaptation* of graphical content of a particular *visual data type* within *Smart Visual Interfaces* . This means to provide visual representations that exhibit *scalability* with respect to their *Context of Use*, the *task at hand* in particular. In the scope of this thesis, these key terms are understood as follows.

2.1.1 Smart Visual Interface

The term Smart Graphics was coined when the first symposium of the same name was held in March 2000 [KB09]. It is defined as "the interdisciplinary approach to the generation, presentation and interaction with 2D and 3D graphical interfaces in a manner that is sensitive to technological, computational and cognitive constraints. Such interfaces aim to move beyond the current requirement that designers anticipate every data, task and technological scenario, and instead allow the dynamic generation and presentation of content in such a manner that: (1) engages the user and is aesthetically satisfying; (2) takes account of cognitive insights as to the use of external representations thereby minimizing potential for imprecision and ambiguity; (3) is sensitive to the real-time demands of the task in the context of the available computational resources; and (4) adapts the form of the output according to constraints placed on the presentation by the nature of the target media and available interaction devices" [BKO00].

In line with this definition, a *Smart Visual Interface* is understood here as a (software) system component that handles the presentation of, interaction with, and manipulation of graphical content within the given Context of Use. To emphasize this focus of the present thesis, "visual interface" is used here over the more general "graphical user interface" (GUI). The latter is in widespread use to denote software user interfaces that are not solely using text input and output but also graphical interaction facilities (called widgets), such as mouse-clickable buttons or menu commands. Graphical user interfaces employing this principle are often referred to under the acronym WIMP (Windows, Icons, Menus, Pointing Device) [Dam97]. Contrary to this, a visual interface emphasizes

the direct interaction with graphical content, although it may still make use of classic GUI elements where it is more appropriate. Where this distinction is not relevant, the terms Visual Interface (VI), User Interface (UI) and Graphical User Interface (GUI) will be used synonymously.

2.1.2 The '5Ws': Context of Use

The *Context of Use* (CoU) is a phrase designating the sum of all relevant influencing factors that affect the generation and presentation of content within Smart Visual Interfaces [AASRS03]. Thereby what constitutes a particular factor as relevant depends, at the very least, on the content/data and the communicative and design goals [CMS99, Mun09]. Butz et al. [BKO00] for example state not only technological and computational aspects, but also cognitive constraints and aesthetic considerations as influencing factors. These can, however, conceptually be grouped according to higher-level aspects of the Context of Use.

In particular, we here propose a categorization of influencing factors specifically for Smart Visual Interfaces, characterized as follows:

What are the characteristics and structure of the content from which the visual representation is generated, how is it (technically) organized and encoded? This also determines what visual encodings should be used and what tasks are meaningful to perform on the underlying data (see e.g., [Mac86, RM90, Shn96]).

Why is the visual representation being generated, i.e., what is the communicative goal of the representation so that the user can extract the information she requires to complete her task? Diverse tasks will require emphasis on different aspects of the data at varying levels of abstraction by applying differing visual encodings to the respective content elements or regions in the visual representation (cf. e.g., [WL90, Shn96, ZF98, Mun09]).

When is the representation shown? During composite work flows comprising multiple interdependent sub-tasks (with varying sub-goals), different visual representations may be appropriate at different points of time. Causal and temporal relations between tasks should therefore be considered to preserve what is referred to as the users "mental map" of the problem domain [ELMS91], and to best possibly match expectations regarding the system's behavior [HK07a].

Where is the visual representation rendered? This may impose limits on the resources available to create and interact with visual representations. For example, mobile devices differ significantly from typical workstations in terms of computational power as well as graphics and interaction capabilities [MKS02, KMS04, Ros06].

For Whom is the visual representation intended? Different users will have e.g. different preferences, (cognitive) abilities, and levels of expertise that should be taken into account (cf. [AD67, HR98]).

We will refer to this categorization as the '5Ws' defining the Context of Use of visual representations within Smart Visual Interfaces. Our chosen partition of the influencing factors is not arbitrary: it relates to the different dimensions of visual representation scalability [TC05] that is required to actually accommodate for changes to the Context of Use (see Section 2.1.3 and Figure 2.1).

The multitude of possible influencing factors make a generic, holistic description of the Context of Use infeasible. Most contemporary approaches address only selected aspects, as exemplified by the cited approaches in the above list. A decomposition into several sub-components is typically used to capture those context aspects relevant to the application domain (cf. e.g., [Pue96, AASRS03, WFRS07]).

Specifically, the approach presented in this thesis addresses WHY and the WHAT aspects of the Context of Use because these are generally accepted to be the most important to consider [FFIT00]. They also can not be contemplated in isolation: while the task at hand affects what graphical content (data) is called for at which level of abstraction, characteristics of the underlying graphical content affect how this can be visually represented.

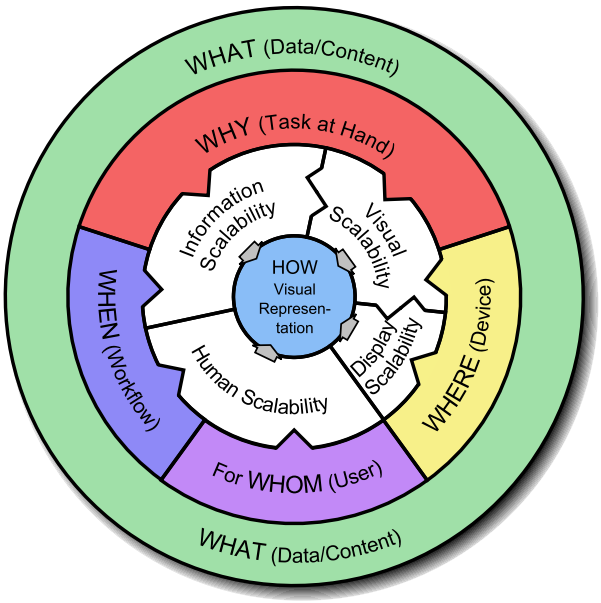


Figure 2.1: High-level categorization ('5Ws') of influencing factors defining the Context of Use (CoU) of visual representations. Different forms of scalability must be provided to accommodate for changes in influencing factors from the respective category to arrive at an efficient visual representation (HOW).

2.1.3 Scalability

This general capability of visualization methods to generate effective visual representations in different situations is commonly referred to as *scalability* [KLS00, Kei05, TC05]. Thomas and Cook [TC05] relate the need for scalability specifically to the challenges

that arise from massive, multi-dimensional and time-varying data sources – where it becomes imperative to find a suitable level of abstraction, or scale, for collecting, analyzing and especially, displaying such data. It is, however, also a suitable term to describe the desired ability to adapt visual representations to their Context of Use within Smart Visual Interfaces, as outlined in Section 2.1.1. When referring to user interfaces in general, *plasticity* is commonly used instead of scalability to circumscribe these properties (e.g., [SC01, CCT01, MSK03]) with the same intended meaning.

Thomas and Cook [TC05] identify five major scale issues that must be addressed: information scalability, visual scalability, display scalability, human scalability, and software scalability. These correspond to the ability of visual representations to accommodate for changes in one of the high-level categories (why, what, where, for whom, when) comprising a visual representation's Context of Use (cf. Figure 2.1):

Information scalability implies the capability to extract relevant information from raw data. This includes both a purely functional reduction (e.g., through filtering) to manageable data sizes, as well as the ability to generate appropriate levels of abstraction for a given analytic or communicative goal.

Visual Scalability is defined as the capability of effectively creating visual representations supporting a specific communicative goal. Factors affecting visual scalability include the visual metaphors used to represent information and the techniques used to interact with the visual representations. It is also linked to display and human scalability.

Display scalability denotes the ability to adapt a visual representation to a variety of display form factors, i.e., to make effective use of everything from a wall-sized display to PDA- or phone-sized screens. Thomas and Cook [TC05] observe that one major challenge is to develop not only visual representations, but also related interaction techniques so that they are display scale-independent. Display scalability is directly related to the WHERE aspects of the Context of Use.

Human Scalability, probably somewhat inaptly named, is not so much about "scaling" the human user in terms of her cognitive or analytic abilities, but rather relates to the number of concurrent users [TC05]. That is, visual analysis tools, and by extension, their visual interfaces, should be able accommodate for single-user up to multi-user collaborative work settings effectively. Human scalability addresses challenges arising from FOR WHOM and WHEN aspects of the Context of Use.

As indicated in Figure 2.1 requirements to these scalability aspects are determined primarily by the respective Context of Use categories, but there also exist interrelations between influence categories. In particular, information and visual scalability of Smart Visual Interfaces are both linked to the mutually dependent WHAT and WHY (cf. Section 2.1.2). Temporal and causal relation of the task at hand to previous working steps (WHEN) affect what levels of abstraction are appropriate for presented information (WHAT and WHY), likewise influencing information and visual scales. WHERE also affects WHY– the set of tasks performed at a given work location – as well as the display

scale dictated by the output device. Constraints on the display scale, in turn, have an impact on the viable levels of detail (information scale). The user's preferences and expertise (FOR WHOM) influence what constitutes appropriate abstractions and visual encodings of the representation, thus affecting information and visual scales.

Software scalability summarizes the desired properties that software tools should be capable of digesting heterogeneous data of arbitrary size, be of modular design to ease adoption of new analysis, visualization and interaction methods, and are capable to utilize different platforms. This is a cross-cutting concern that affects a software system's overall ability to accommodate changes in all five influencing factors of the Context of Use.

2.1.4 Adaptation

Adaptation as well as adaptable are two terms closely related to scalability that appear frequently in literature. The latter is often used synonymously in referring to different scalability aspects, especially visual/display scalability (e.g., [KMS04, MMS04, KJDG+06]) and software scalability (e.g., [PZB02, SKKM+05]).

On the other hand, *adaptation* is used consistently to refer to the actual process of modifying, respectively, the (graphical) content (e.g., [Kli09]) or the system configuration (e.g., [TTS09]) to the given Context of Use. Both terms are used here in line with these established meanings.

The task-driven adaptation of visual representations as discussed in this thesis primarily relates to issues of visual and information scalability. Depending on the output device, however, constraints with regard to the display scalability may also require consideration.

2.1.5 Visual data type

We thus use here a categorization of the graphical content handled by a visual interface into four classes, depending on the amount of control possible on the generation process of context-specific content representations. These four *visual data types* are:

Raster images (bitmaps) are given by a data structure representing a (generally rectangular) grid of pixels, or points of color. Only image space operations that modify pixel (color) values are possible as usually, semantic or structural information on the image content is not available[1] (Chapter 4).

2D vector graphics define the image content by a structured set of graphical primitives (rectangles, circles, etc.) with associated geometric transformations and graphical attributes (e.g., fill color, line stroke). Adaptation of the visual representation can thus be applied at the granularity of graphical primitives. Some formats, especially the XML-based Scalable Vector Graphics (SVG) W3C standard [FFJ03], allow to

[1] Note there are means for encoding meta data for raster images, e.g., using the generic MPEG-7 (ISO/IEC 15938) standard [MKP02] or through direct embedding [RFS08]. But these meta data are often not available for a given raster image and application context (see Section 2.2.1).

embed additional, application-specific information in custom primitive attributes (Chapter 5).

3D models describe virtual objects, typically as triangle meshes. Geometry and surface attributes are adaptable during the rendering process similar to vector graphics; in addition control of scene composition (i.e., lighting and view frustum setup) becomes possible (Chapter 6).

Abstract data has no inherent visual representation, rather it must be transformed into an image by a suitable visual mapping as part of the so-called visualization pipeline (see e.g., [HM90]). This enables operations in data space, such as filtering of values, as well as modification of the subsequent visual mapping and rendering stages (Chapter 7).

Adaptation strategies specific to these four content types of Smart Visual Interfaces will be discussed in detail in the Chapters indicated.

2.1.6 Task

The term *task* has a fairly large range of specific meanings across different disciplines. In colloquial language, it is used to describe "a piece of work to be done. Task implies work imposed by a person in authority or an employer or by circumstance" [DeV90], often in a definite quantity or amount. This further implies that the work is performed in order to attain a specific result, or task *goal*. Task is a synonym for activity although the latter carries a connotation of being possibly longer duration.

In this thesis a *task* is understood as a single, conceptually distinguishable but not necessarily atomic step within a composite activity or *work flow*. In particular, a task may be the root of a hierarchical decomposition into sub-tasks with subordinate (partial) goals. A partial order may be defined on sub-tasks in the form of preconditions (e.g., completed subordinate goals) and/or temporal dependencies (A 'happens-before' B) between sub-tasks. This hierarchical decomposition of high-level *composite tasks* and corresponding goals into subsequently more concrete sub-units is a common tenet in virtually all disciplines involved in task analysis and modeling, see for example [AD67, LV03, WFRS07, HR98, WVE98].

At the lowest level of decomposition are *basic tasks*, i.e., leafs of the task hierarchy according to the granularity at which tasks are specified on a conceptual level. Some task modeling approaches further specify *actions* of basic tasks, thus describing functional properties beyond the conceptual task decomposition [LV03]: an action is an atomic operation that is executed upon an artifact, by an entity that is involved in the completion of the task (user, computer, ...). In this thesis, we adopt this distinction. In particular, actions comprise user interaction with a task-specific visual representation as the respective basic task's artifact.

There is no agreement on an exact definition of *task goal*. Intuitively, a task's goal is the state of affairs the task is intended to produce, i.e., post conditions that hold after the task has been completed. In particular, within smart visual interfaces task-specific visual representations are employed to support the user in completing her tasks. This

requires the user to be able to extract required information from the representation. This property – conveying particular information by stressing its message subject over a faithful representation of reality – corresponds to the *communicative goal* of a visual representation.

Another term with contradicting meaning across disciplines is *abstract task*. In task analysis and modeling it is used as synonym to composite task, denoting a high-level conceptual specification of a domain task that must be further decomposed to arrive at basic tasks with concrete actions (e.g., [PMM97]). Contrary to this, in visualization abstract task usually designates low-level, generic data analysis tasks, i.e., tasks such as "compare values" that are applicable to (abstract from) different types of data [Mun09]. We here use composite task when referring to high-level (domain) tasks, and abstract task when referring to low-level data analysis tasks.

Moreover, references to the "task at hand" or "the user's current task" can be found in several publications (see e.g. [MPS02, TC05]). However, it is often not made explicit if this refers to a high-level composite task or to a leaf-level basic task (i.e., the current one in a sequence of working steps being performed). Here, we use *task at hand* to refer to both in general, and *current working step* when referring specifically to a leaf-level task.

Also note that the used definition of task is not exclusively user-centric – an activity can comprise tasks carried out by both human and the (computer) system the user is working with.

2.2 Application Background

The contributions of this thesis have been developed against the background of an applied research project "Landesforschungsschwerpunkt" (federal state research focus, LFS) as well as in cooperation with the research training school "Multimodal Smart Appliance ensembles for Mobile Applications" (MuSAMA) [MuS09]. The LFS comprised two funding periods with slightly different research focus, namely "Multimediales Content-Management in Mobilen Umgebungen mit Multimodalen Nutzungsschnittstellen" (Multimedia Content Management in Mobile environments with Multi-Modal user interfaces, M6C) [LFSa] and "Mobile Assistenz" (mobile assistance, LFS-MA) [LFSb], respectively. Both projects dealt with use cases that underline the utility of smart visual interface. These use cases are presented below with a discussion of the resulting challenges with respect to smart visual interfaces.

2.2.1 Mobile Information Systems: Maintenance Support

With the ongoing pervasion of mobile devices and wireless networks, applications such as location-based services and mobile information systems are also on the advance (e.g., see [NPF+04, Sie06, Kli09]). Their purpose is to provide a user with information specific to her current situation, task, and location on mobile devices such as Smartphones, Personal Digital Assistants (PDAs) or TabletPCs.

One use case of mobile information systems is mobile maintenance support, or so called "e-manuals" [FRSF06] (Figure 2.2). The design of such a system and correspond-

ing challenges were addressed in both the M6C [LFSa] and the LFS-MA [LFSb] subproject MAXIMA [MAX] for the concrete application scenario of HVAC[2] maintenance. HVAC units have a comparatively long operational life, and therefore often receive mid-life upgrades, for example to meet the latest energy efficiency standards. This poses a challenge for technicians to gain sufficient knowledge of all peculiarities of different units. Thus a technician requires on-site access to up-to-date schematics, operational and safety procedures as well as assembly instructions. Constantly amending printed (paper) manuals quickly becomes impractical in these situations. This is not only due to the required effort and expenses, but also simply due to the physical medium's increasing bulk [WKM+09].

Figure 2.2: E-manual applications are useful in maintenance scenarios where machinery- and task-specific knowledge is required (left). Small handheld devices are preferable as these can be carried along easily (center). The vast range of capabilities in this device class presents a challenge in designing the (visual) interface, though (right).

Presenting manuals as interactive content on a mobile device does overcome these limitations and principally offers greater versatility than paper. As wireless network access is becoming more and more widespread in enterprises as part of their LAN/WLAN infrastructure, providing the manual content from a centrally maintained repository on-demand is often feasible [Kli09]. This mitigates the problems of keeping any parts of the manual up to date. More importantly, using digital media makes available features such as keyword searches for specific content, context-based filters, as well as dynamic and interactive 2D or even 3D representations. The latter is especially beneficial for manual applications, as dynamic, interactive representations generally are more comprehensive than static illustrations. To this end, the content of an e-manual comprises three principal components: maintenance procedure (workflows, tasks) descriptions, visual (illustrations, schematics) and non-visual (text, tables) information, see Figure 2.3.

Because complex facts are communicated best using visual means, task-specific visual representations constitute the primary component of an e-manual's smart visual interface. A key aspect in this regard is the ability to adjust the presentation of information according to the user's task at hand as well as his proficiency, e.g., the chief engineer compared to a subcontractor's employee [WKM+09]. This requires suitable means for providing visual and information scalability of the manual contents, as discussed in Sec-

[2] Heating, Ventilation, Air Conditioning

tion 2.1.3. The main challenge here is graphical content from sources that do not provide a structured description of, and/or lack semantic information about, the depicted object(s). Common examples are raster images extracted from PDF files, and scanned illustrations from paper manuals. Since these acquisition methods often represent the primary sources for the e-manual's content, methods and tools have to be developed to enrich unstructured content to afford proper information and visual scalability [FRSF06].

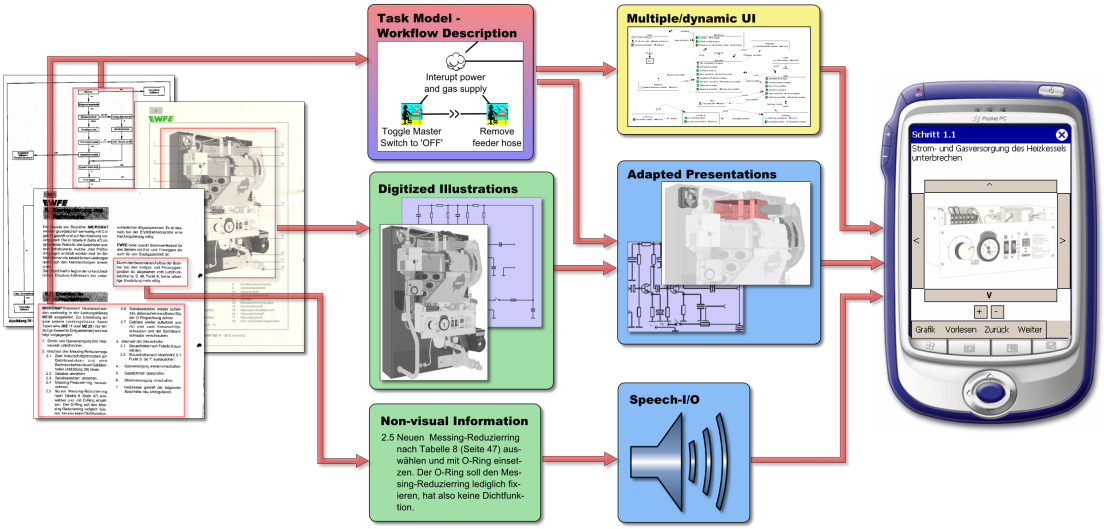


Figure 2.3: Principal components of an e-manual's contents are a description of the procedures involved, working step-specific illustrations plus auxiliary information (often presented via text-to-speech). Their extraction from printed media is largely a manual task necessitating tool support [FRSF06].

The challenges in creating effective task-specific visual representations are further compounded by the high variability in capabilities of mobile devices that might be used as clients (e.g., ranging from Smartphones to laptops) in terms of connectivity, processing power, graphical capabilities, and interaction facilities. This does not only call for efficient means to design and deploy so-called Multiple User Interface (MUI) to accommodate the different device classes. It also places a strong emphasis on the display scalability aspect of generated task-specific visual representations on top of the information and visual scalability requirements.

Since presenting graphical content on mobile devices means to cope primarily with the limited screen space [RTS06] some powerful approaches for display-scalable representations have been developed, e.g., for mobile web browsing [BMPW00, KRS03], museum guides [SW07] or maintenance support [HBP+07]. These approaches, however, generally do not address task-related requirements to visual representations.

Another approach to address limited screen space is to provide alternate interaction modes to augment the primary visual interface. To this end, e-manuals often make use of speech as an alternate input/output (I/O) mode [FRSF06, FS06] for non-graphical, auxiliary information, cf. Figure 2.3. The benefit of these so-called multi-modal interfaces is two-fold. First, using non-graphical means (e.g., speech) for output reduces the amount

of information that has to be encoded into the visual representation. Likewise, providing voice commands as an input method yields more screen space for visual representations as the corresponding GUI components (e.g., buttons, scroll bars) can be omitted. Second, multi-modal interfaces allow to adjust communication of relevant information to the user according to the work environment conditions. For example, high ambient light levels increase the utility of an alternate speech I/O mode as the screen becomes more difficult to read; whereas high ambient noise obviously favors visual communication of information. Such environmental conditions are quite common for mobile maintenance scenarios, for example during outdoor operations in direct sunlight or at noisy work spaces such as factory floors. The challenge here is to find an appropriate utilization of the primary visual and auxiliary modes to ensure the efficient communication of information for a given context of use (including the user's task at hand, the output device, and environmental conditions).

Of the challenges described above, the primary research question addressed in this thesis is how to enrich graphical content to afford information and visual scalability, and how to further associate the enriched content with the description of maintenance procedures to arrive at task-specific visual representations, as illustrated in Figure 2.3.

2.2.2 Smart Environments: Smart Meeting Room

Cook and Das [CD05] define a smart environment as "one that is able to acquire and apply knowledge about an environment and also to adapt to its inhabitants in order to improve their experience in that environment." Smart homes, smart class rooms, and smart meetings rooms (as shown in Figure 2.4) are specific examples of smart environments [EK05, AE06]. To this end, smart environments employ an ensemble of interconnected devices including computing devices (e.g., desktop computers, servers), output devices (e.g., projectors, monitors, flat panels), environmental devices (e.g., lights, blinds, air conditioning), and sensor devices (e.g., motion trackers, infrared beacons, light sensors) [HK05]. Some installations also strive to dynamically integrate mobile devices such as laptops, PDAs or smartphones that enter and leave the environment as they are carried along by the users [TTS09]. Device interconnection is driven by various technologies, including Bluetooth, wireless or hard-wired LAN, allowing for the necessary communication to accomplish tasks in a coordinated fashion [AE06].

A smart environment further implements software utilizing the device ensemble to provide "smart" support to users. This requires to constantly assess the current situation of the environment and that of its inhabitants. Based on an analysis of sensor measurements, user intentions and tasks are predicted [HK07b]. Preferably, the predictions would be accurate enough to allow fully automatic user support. In cases where this ideal can not be attained, users always have the possibility to revise the decisions made by the environment or to fine-tune the environment to their needs [HK07a].

Thus, how to accomplish "smart" support as a joint effort of several distributed software components in the face of uncertain, often incomplete and ambiguous situational knowledge is a challenging and actively investigated research question. Among others, two projects that address these issues are Multimodal Smart Appliance ensembles for

Figure 2.4: Typical example of a Smart Meeting Room with several output devices (displays, projectors) to support collaborative work.

Mobile Applications (MuSAMA) [MuS09] and MAIKE[3] [MAI], concentrating on fundamental and applied research, respectively.

Of particular interest in the scope of this thesis are those aspects that relate to the use of smart visual interfaces in smart environments. Multi-user collaboration in these environments is explicitly *not* limited to simply showing a single visual representation on more than one output device simultaneously [FTSS09]. Instead, available output devices need to be assigned to visual representations according to the users' current situation and task requirements. Visual representations must therefore be scalable to allow the support of different user goals and data sources, using displays of various sizes and resolutions.

There are some approaches to this end that adapt visual representations to the available resources on different target devices (e.g., [PZB02, SKKM+05, ZHHM07]). Most, however, emphasize technical and infrastructural issues, such as the utilization of multiple client platforms [SKKM+05] or the use of web services as the output distribution mechanism [ZHHM07]. In [TTS09], the authors propose a more general approach for distributed visualization. It uses a service-oriented architecture (SOA) to generate visual representations in a distributed fashion, including mobile devices that can dynamically enter or leave the smart room's ensemble. Most information visualization approaches, however, typically provide data- or task-specific solutions that are computed on a single machine for a fixed output device and a single user, as observed for example in [WPF04, TC05, Mun09]. They hence are ill-suited for information presentation in smart meeting rooms.

To summarize, in both scenarios outlined above smart visual interfaces can help to improve the respective system's utility. Thereby scalability of visual representations with

[3] **M**obile **A**ssistenzsysteme für **I**ntelligente **K**ooperierende Räume und **E**nsembles, a sub-project in the scope of the joint academia-industry project "Landesforschungsschwerpunkt LFS-MA" [LFSb].

respect to the task at hand is one of the most important aspects: in mobile maintenance scenarios, to present from the multitude of technical documentation only the immediately relevant information; and in smart environments, to enable (pro-)active support for the users' situation and requirements. Consideration of the user's task must therefore be integrated into the design process of the visual interface. To this end, the following section reviews task description and modeling approaches used in Human–Computer Interaction (HCI) and UI design, as well as contemporary approaches to scalable visual representations.

2.3 Related Work

Modeling is understood as abstraction of a real system by removing the irrelevant details in the current level of abstraction [WVE98]. As such, task models are a means to the systematic analysis of the problem domain users work in. They are used as tools in several different disciplines like project management, training [AD67, HR98], cognitive psychology, and software engineering.

Numerous task models have been proposed in the literature (see e.g. [Ben00] for a good overview). Limbourg and Vanderdonckt [LV03] alone list and compare 11 different models each representing a principal discipline involved with task analysis. As a result, the proposed concepts exhibit both differences in vocabulary as well as conceptual variations in terms of presentation, level of formality, complexity and expressiveness according to their intended use and objectives. All of them, however, are based on a common tenet: that tasks are performed to achieve a certain goal, and that complex tasks are decomposed into more basic units until the level of atomic tasks has been reached [LV03].

Prevalent examples include the pioneering Hierarchical Task Analysis (HTA) method [AD67]; the Goals, Operators, Methods, and Selection rules (GOMS) [CMN83, JK96] as an engineering model for human performance in cognitive psychology; as well as the User Action Notation (UAN) [HSH90], Concur Task Tree (CTT) [PMM97, Pat00], Unified Modeling Language (UML) activities [BJR00] and 'Yet Another Workflow Language' (YAWL) [HA05] in software engineering.

Task analysis and task modeling is a well-established research field in HCI and software engineering. From the perspective of this thesis, task models play an important role in contemporary user interface design methodology in particular.

2.3.1 Task Modeling in UI Design

Model-based User Interface Development (MBUID), as an important component of the Model-driven Design (MDD) approach [Alm06], has gained much attention by various researchers (see e.g., [SE96, MPS02, Luy04, GSSF04]) due to its ability to foster the integration of different viewpoints into the development process already in its early stages [Con03] and helps software designers to manage complexity by abstracting from low-level implementation details. It facilitates the development of efficient UIs by using different declarative models and the relationships between these models to describe the various facets of the UI at different development stages [Pue97, EVP01]. Thereby the

model-driven UI design process can be divided into four phases: (domain) task analysis, system design, implementation, and system evaluation (see e.g. [HR98, Sta00]).

During task analysis it is determined what tasks and goals are to be accomplished with the help of the software being designed, as well as how individual tasks are organized into an overall workflow. Therefore task models, as an explicit description of the overall workflow, are a commonly accepted starting point for the model-based UI development process [HR98, Kie04, Alm06]. These models define a decomposition of the task at hand into subtasks down to the level of conceptually atomic basic tasks at a granularity appropriate for the application domain. Typically this decomposition is represented as a hierarchical structure of nodes with parent-child relationships. Nodes within this structure represent individual tasks. Child nodes describe sub-tasks that have to be performed in order to complete the task of the parent node. Most task model formalisms also map temporal and causal relations in the form of preconditions, postconditions, or information transfer (e.g., the current selection from a list of items) between task nodes. A path traversing this structure therefore describes a sequence of basic tasks that are performed to complete the overall composite task represented by the hierarchy root node.

During the subsequent design phase, further information may be included to describe the Context of Use (CoU), i.e., an abstraction of the environmental circumstances of the task execution [Pue96]. To this end, most approaches specify a subset of user, domain (application), platform, dialog, layout and/or presentation models [SE96, Træ02]. Some approaches explicitly consider usability issues as part of their model [LJMM+05] to further promote user-centered interface design.

These information can then be utilized during the implementation phase. For this, first a dialog model is derived from the task model that describes the logical interface composition [LCCV03]. This includes grouping of related or simultaneously enabled tasks into dialogs, the definition of appropriate interaction objects to carry out the individual task actions, as well as deriving task goals and conditions that trigger transitions between dialogs. This dialog model is then used to generate concrete dialogs based on platform-specific widgets [PS02].

Model-based UI development can thus be seen as a series of model transformations [Sta00, Luy04], where abstract models (e.g., task, user, domain model) gradually evolve into more concrete models (e.g., dialog, layout, presentation model) which finally result in the actual implementation of the UI [Pue96, SWF+07], see Figure 2.5.

In fact, a key factor driving the proliferation of MBUID approaches is the ability to maintain platform independence (cf. [Luy04, Alm06]): A platform-independent model (PIM) can be used as input to transformation activities that lead to different alternative UI realizations that are implemented using different platform technologies [CCT+03, LVM+04]. The design process from platform-independent models to platform-specific realizations often entails the use of intermediate platform-specific models (PSM) [Alm06, WFRS07], cf. Figure 2.5. This is especially advantageous for creating mobile and multi-device UI for applications with very heterogeneous platform technologies and device capabilities like those outlined in Section 2.2. Model-based UI generation thus even allows to dynamically create and distribute user interfaces for services offered in such environments on demand [CCT01, RF05].

Since the design of task models is complex and error prone, tool support is needed

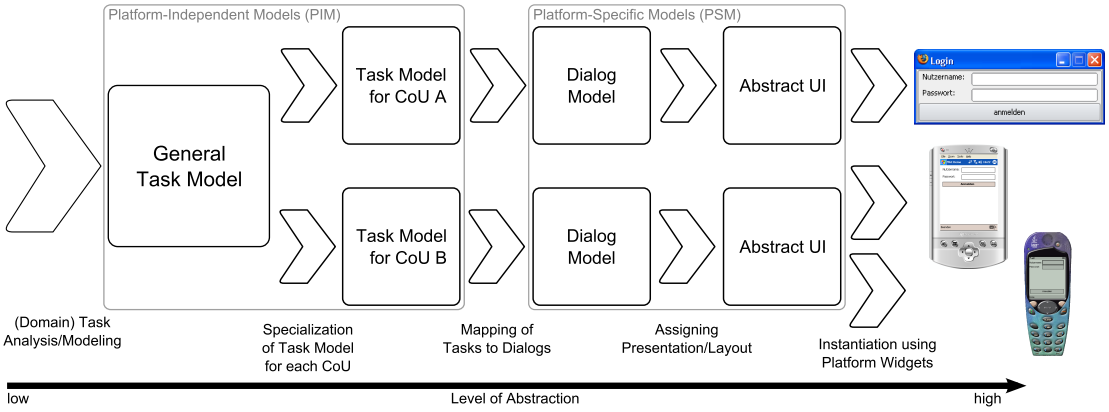


Figure 2.5: Schematic view of the Model-based User Interface Development (MBUID)) process for multiple UI using intermediate platform-independent and platform-specific models.

to carry out model-based UI development efficiently [Sch96, Pue97]. Especially tedious tasks can be supported or automated. Furthermore tool support is able to hide technical details of the used technologies, as the design should be made at a conceptual level.

Within the domain of user interface design, the *ConcurTaskTree* notation (CTT) developed by Paternò [PMM97] is one of the most popular [WPF04, Wur09]. This has been attributed to the fact that it contains the richest set of operators [SWF+07] and that tool support was available early-on through CTTE [MPS02], a graphical tool which facilitates the creation, visualization and sharing of CTT models. A comprehensive overview on CTT can be found in [Pat00].

In a nutshell, CTT are a graphical notation based on five concepts: tasks, objects, actions, operators, and roles. An activity is represented by a hierarchical, tree-like decomposition of a tasks into subtasks (cf. Figure 2.6). Each node in the decomposition tree represents a task. Compound tasks (called abstract tasks) are further decomposed up to the level of basic tasks, which are defined as tasks that could not be further decomposed conceptually (i.e., leafs of the hierarchy). Basic tasks are categorized based on whether a task is executed by the user, by the application, or by an interaction between user and application.

Moreover, objects as well as input/output actions modifying these objects are specified for each basic task. Task objects can be perceivable objects or internal (application data) objects. Application objects must be mapped onto perceivable objects in order to be presented to the user. This mapping is mainly directed toward the specification of on-screen interaction objects (interactors), e.g., labels, icons, buttons and menu entries; although [PMM97] also briefly mentions sound. Examples from [PS02] do consider images, but explicitly as static resources: images of different resolutions/quality must be provided and associated with the respective platform manually, thus completely forgoing scalability issues.

So-called operators link sibling tasks on the same level of decomposition by temporal and/or causal constraints. In this respect, CTT differs from most other hierarchical models like HTA [AD67], where operators are defined for parent-children relationships.

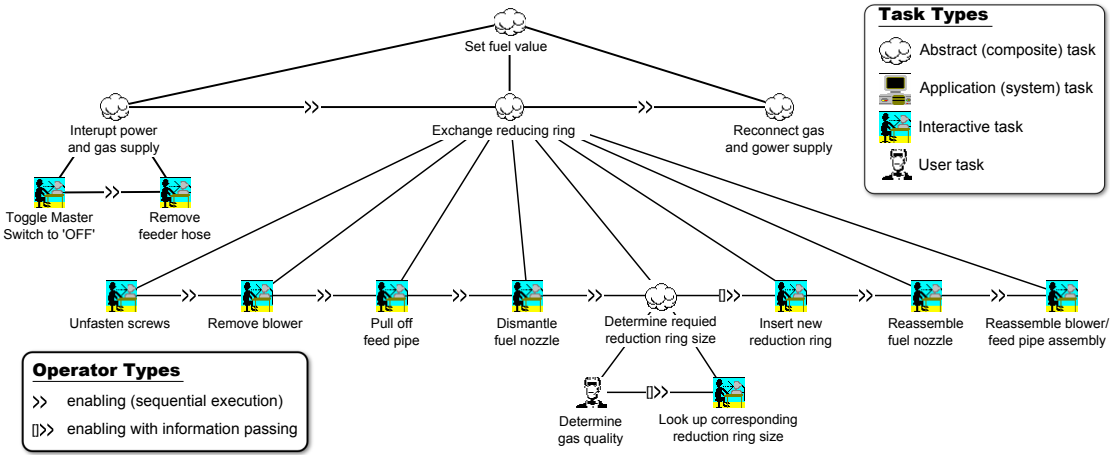


Figure 2.6: Example of a graphical CTT notation of a small maintenance task (from [FRSF06]). The task type is indicated by the node icon. Temporal operators are represented by horizontal links between sibling tasks adorned with the operator symbol.

CTT uses a formal definition for its temporal operators based on LOTOS (Language of Temporal Ordering Specification [BB87]). Paternò defines the temporal operators choice [], independent concurrency |||, concurrency with information exchange |[]|, order independence |=|, disabling [>, enabling (sequential) >>, enabling with information exchange []> and suspend/resume |>.

In addition, CTT provides the means to describe cooperative tasks. To describe such a task, the task model is composed of different task trees, one for the cooperative part and one for each role that is involved in the task.

Several extensions to the original notation have been proposed since to better facilitate specific requirements of different MDD aspects, foremost automatic UI generation [LCCV03] and multiple UI for different mobile devices [PS02]. For this purpose, CTT are *annotated* with additional information, e.g., with abstract UI descriptions in the case of the DYGIMES framework [LCCV03], meta operators to ensure model equivalence across model transformations [WSF08], or additional variables to capture the context of use resulting in different PSM during subsequent refinement [WFRS07].

In summary, using task modeling as integral part of UI design methodology offers opportunities to arrive at a better overall UI design, as well as to partially automate the development process, by explicitly capturing relevant influencing factors that comprise its context of use. These traits are equally desirable for the design of smart visual interfaces. In fact, the necessity to address scalability issues compounds even further the need to adequately consider the task at hand. Therefore, smart visual interface design can not do without integration of suitable task modeling approaches for visual representations. However, the vast majority of MBUID methods is geared towards the specification, creation and evaluation of WIMP interfaces. Graphical content is usually considered as a static resource that is displayed by virtue of a GUI widget, navigated by simple means like zooming and panning [RTS06]. As a result, these approaches do not

usually capture scalability aspects of visual representations corresponding to the '5Ws' (cf. Section 2.1.2) categorizing the context of use. Thus from the perspective of task-driven Smart Visual Interfaces according to this thesis' objective, at least the WHAT and WHY aspects must be incorporated (Chapter 3) for visual content of the four visual data types raster images (Chapter 4), vector graphics (Chapter 5), 3D graphics (Chapter 6), and visualization of abstract data (Chapter 7).

2.3.2 Scalable Visual Representations

In this thesis we distinguish these four visual data types according to their degree of scalability (Section 2.1.3). We will review previous work regarding task-driven adaptation, and the scalability issues thereby addressed, for each data type in the following.

Raster Graphics do not contain any structural or semantic information about the depicted content in the pixel grid itself. Unless a raster image is augmented by external meta data like MPEG-7 [MKP02], task-driven adaptation is therefore restricted to exploit visual and display scalability, but not information scalability.

Specifically, without knowledge on the depicted content, scalability is achieved by using Region of Interest (RoI) [RS99, LG05] based schemes: depending on the communicative goal, some image regions are more relevant than others. This relevance is usually expressed by a Degree of Interest (DoI) function that signifies the relative importances of regions [Kea98]. There are a number of RoI-based approaches that can be related to both visual and display scalability which we will review in the following.

The RoI-driven display of raster images has primarily been fueled by the requirements of image communication[4] in mobile environments [RS98, Ros06]. The limited bandwidth, comparatively high transmission costs, and small screen size of the typical output device mean that important image content should be prioritized both during transmission and display.

To address transmission efficiency, scalable image compression schemes have been devised that allow compressing the data once and then decompressing it at multiple data rates, spatial resolutions, and/or visual quality [LPKD01, JPE00] (also see [Ros06], pages 12 ff. for a good overview). Especially, progressive transmission and decoding enables visual scalability of raster images, by displaying the current RoI in detail first and only then increasing detail in the remainder of the image without incurring redundant transmission of data [NC98, Ros06]. Usually, the RoI used to prioritize this transmission order is specified interactively by the user (e.g., [RS99]). To this end, means for the efficient specification of arbitrary overlapping RoIs have been proposed [RRS01]. Some approaches use static RoI, i.e., a predetermined scheme of quality and/or resolution progression, to emphasize selected details chosen by an image author [JPE01a].

Moreover, the size of large images often exceeds the display area of the user's output device [RTS06], necessitating display scalability of raster images. This situation can

[4] Image communication is a collective term for the different operations performed from the supply of the image data at server side until its application at client side. A reasonable communication channel consists of at least three basic components – encoding, transmission and restoration (decoding) [Ros06].

be addressed in three principal ways: zooming and panning, multiple consecutive views (i.e., trading time for space by displaying multiple RoIs in sequence [LXMZ03]), and RoI-based distortion schemes (e.g., [HF01, RT03, GF04]).

The first two approaches achieve display scalability through resizing (a sub-region of) an image to a desired target size using cropping and scaling operations. This does not consider visual scalability, however: cropping tends to work well only for images containing single objects of importance since it can only remove pixels from the image periphery. Scaling the entire image reduces the size of important regions or even distorts them in the face of mismatching aspect ratios. More importantly, both approaches sacrifice the ability to see both the detailed RoI and an overview of the surrounding image context simultaneously.

This drawback is addressed by RoI-based distortion schemes. Here, one can distinguish two classes of techniques: *automatic image re-targeting* and (interactive) *Focus & Context representations*.

A compromise between image resizing and image cropping is to introduce a non-linear, data dependent scaling [LG05]. So-called content-aware, or automatic image re-targeting techniques [STR+05, AS07] aim to preserve important image features during resizing. These features can be detected either top-down or bottom-up to build a saliency map that determines what image regions to retain or discard [MZ03]. Top-down approaches attempt to identify complex features such as faces [VJ01], whereas bottom-up methods use measures of the low-level visual salience to identify aspects of the image that the eye may be drawn to [LG05]. The best known salience method is the framework of Itti et al. [IKN98] based on a detailed model of the human visual system, which however is computationally demanding and difficult to tune. Ma and Zhang [MZ03] argue that heuristic methods that are computationally more efficient and simpler to implement but still are effective for practical re-targeting problems. For example, Setlur et al. [STR+05] use a combination of a simple image segmentation algorithm and saliency heuristics to decompose an image into foreground features and background. The background is renewed (holes are filled) and scaled to target size, then the foreground segments are pasted onto the new background. Similarly, the Seam Carving approach presented by Avidan and Shamir [AS07] combines several low-level 'pixel energy' metrics with manually defined RoI of very high or very low importance, allowing the author to override what high-level image content to retain or discard, respectively, by the re-targeting algorithm (Figure 2.7).

These approaches enable a combination of display and visual scalability with respect to a communicative goal, depending on the feature detection algorithm used. However, related approaches from literature generally do not include an explicit task specification for parametrization.

Interactive Focus & Context representations [Kea98] have shown to be superior for a large range of image-related tasks [GF04] because they provide both overview and detail in the same view. They can be easily integrated into RoI-based display schemes whereby the RoI constitute the focus while the rest of the image represents the context [RS99]. Unlike re-targeting approaches, Focus & Context techniques typically specify the RoI explicitly [RRS01], rather than deriving it automatically from the image contents.

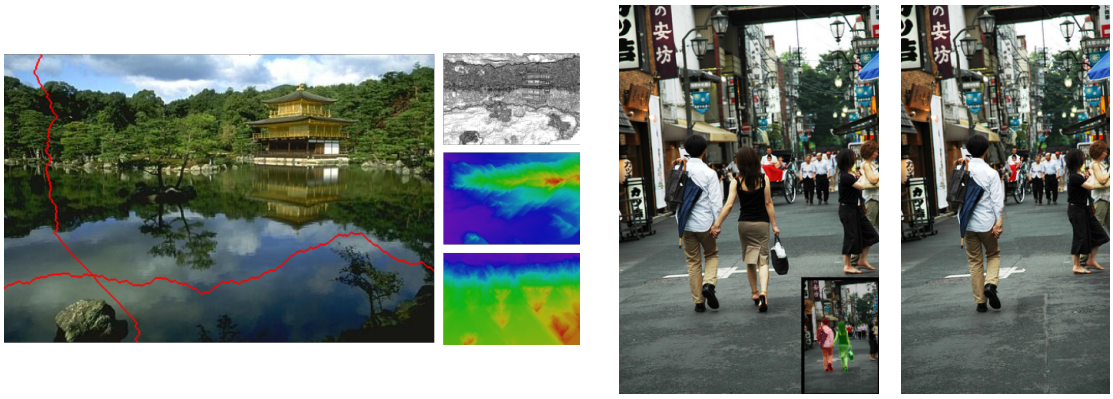


Figure 2.7: Seam carving is an image re-targeting method that successively removes or duplicates connected 'seams' of pixels (far left, shown in red) with the lowest total energy as determined by an energy map (second left). RoIs can be defined manually to either protect or prioritize image regions during seam removal (right two columns). (Figure from [AS07]).

So-called fisheye views are a specific type of Focus & Context representation that combine an undistorted, probably magnified focus region in full detail within a distorted context. Fisheye views have originally been introduced by Furnas [Fur86] to display large structures efficiently, but have been widely adapted to the display of raster images [RT03]. Rectangular Fisheye Views were introduced by Rauschenbach [Rau99] that use discrete Level of Detail (LoD) for distortion (see Figure 2.8). They are especially well-suited in mobile image communication scenarios as they allow efficient transmission of data and offer continuous transition between regions of varying distortion. Rauschenbach's general approach was later extended [Ros06] to make direct use of built-in features of the modern JPEG2000 image encoding standard [JPE00, JPE01a].

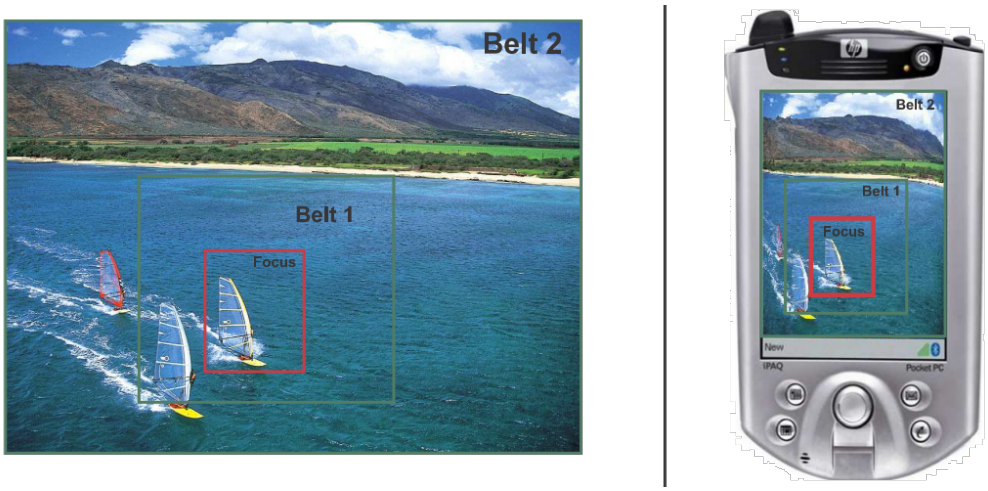


Figure 2.8: Example for the Rectangular Fisheye View [Rau99] RoI-based distortion technique using belts of constant LoD (i.e., magnification factors). Shown here is an implementation using built-in features of the JPEG200 encoding standard to achieve this effect (from [Ros06]).

In conclusion, scalable compression schemes and RoI-driven distortion techniques like Fisheye Views and image re-targeting afford both visual and display scalability of raster images. Most approaches, however, have been derived primarily to satisfy the needs of interactive mobile image communication, rather than the support of task-driven adaptation of raster images.

Vector Graphics are described by a set of graphical primitives with geometry defined in $\mathbb{R}^2$. As such, vector graphics are inherently resolution-independent as they are rendered prior to display at the desired target resolution. Vector graphics thus exhibit excellent display scalability that is the reason why they see increasing use especially in mobile application scenarios [Rei03, Kli09] with their wide range of display sizes.

Moreover, the content description by a set of individual primitives, usually organized hierarchically, and the conceptual separation of geometric information and its visual attributes (e.g., colors, strokes, fill patterns) is exploited by several approaches from literature to also address visual and information scalability.

In cartography, a large fraction of geographic data has long been stored as vector data such as the XML-based Geographic Markup Language (GML) format [Por07]. Transformation into a map representation would, at the least, include a selection of a subset of geographic objects [McE94, HG94] and rendering styles corresponding to the thematic setting of the map [McE94, Lup07]. Using XML transformation schemes like XSLT [Cla99], these two steps can be combined to generate several SVG representations from a single GML file holding the contents of a base map. Reichenbacher [Rei03] uses this general concept to provide adaptive map representations for location-based services on mobile devices. Notable of his approach is that it explicitly considers the context of use to support visual and even basic information scalability (i.e., the selection of predetermined map layers from the backing GML file). This includes the output device, location, but also a limited set of user goals and associated activities related to map use.

Contemporary vector graphic formats, while having some notion of content structure, nonetheless lack semantic information such as functional dependencies or topological constraints [FFJ03]. This poses a challenge during adaptation when content elements are resized or moved individually, or when several related elements should be replaced by a more complex aggregate representation to achieve visual scalability (often referred to as *semantic zooming*, see e.g. [BTM+01, MMT02, KHS04], Figure 2.9). Consequently, there have been several proposals to augment vector graphic formats with additional semantic information.

Among the first were Constraint Scalable Vector Graphics (CSVG) for simple figures [TMM00] and its extension [BTM+01] to more complex network diagrams (cf. Figure 2.9). CSVG uses a set of rules to constrain sizes, absolute and relative position as well as alternative visual representations ('switchgroups') of diagram elements. The rule definitions are embedded in the SVG file by making use of SVG's extension mechanism [FFJ03]. CSVG thereby functions as a container format; rules are evaluated on the client at runtime using the Cassowary system [BB98], creating adapted SVG documents in reaction to user interactions. Further research suggested a more restricted set of rules, called one-way constraints, is nonetheless sufficient for most applications

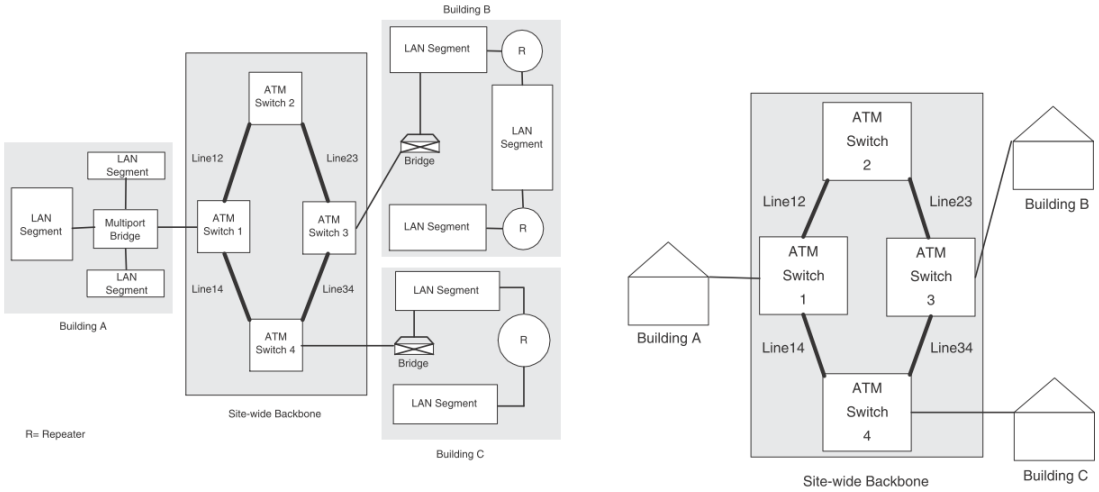


Figure 2.9: Layout and topological constraints added to CSVG vector graphics afford visual scalability or *semantic zooming* whereby parts of a diagram are replaced by simplified aggregate representations selected from a special 'switchnode' storing several alternatives (figure from [BTM+01]).

while computationally less expensive to resolve [MMM04]. All these approaches strive to directly embed the additional information within the vector graphics (SVG) file. Here again, a primary motivation for this often was the ability to perform purely client-side adaptation for mobile applications, to avoid server round trips [MMT02].

Klima et al. [KHS04] use a slightly different approach to additionally provide information scalability that uses an external "semantic tree" to select related content elements and their desired level of abstraction from a base XML data set. While this reduces the flexibility of client-side adaptation, it has the benefit that it allows to even swap a 2D SVG with a 3D Virtual Reality Markup Language (VRML) representation based on situation and device capabilities. Likewise, the GraSSML system [FD06] for accessibility adaptation of diagrams distinguishes three distinct levels of abstraction – structure, domain semantics, and presentation – that each reside in a separate file.

More recently, Marriott et al. [MMS04] proposed to combine constraint-enriched SVG with a Resource Description Framework (RDF) and a domain ontology description using the Web Ontology Language (OWL) [MH04] to generate adaptive diagrams in a larger context of collaborative work with multimedia documents. A very similar but more comprehensive approach has been presented by Klima [Kli09] for the adaptation of 2D and 3D graphical data in collaborative and mobile environments. It, too, uses an RDF and OWL to describe application-level semantics, but also includes abstract data types as an additional abstraction level on top. This reduces the required complexity of the ontology as well as the number of adaptation rules (i.e., constraints) that must be evaluated. Adaptation itself is driven by user queries (RoI definitions) and several different DoI functions [Kli09], but does not include task-based selection for either of both (see Figure 2.10).

In conclusion, vector graphics exhibit inherent device scalability that can be readily utilized for task-driven adaptation. Several approaches have been proposed that enable

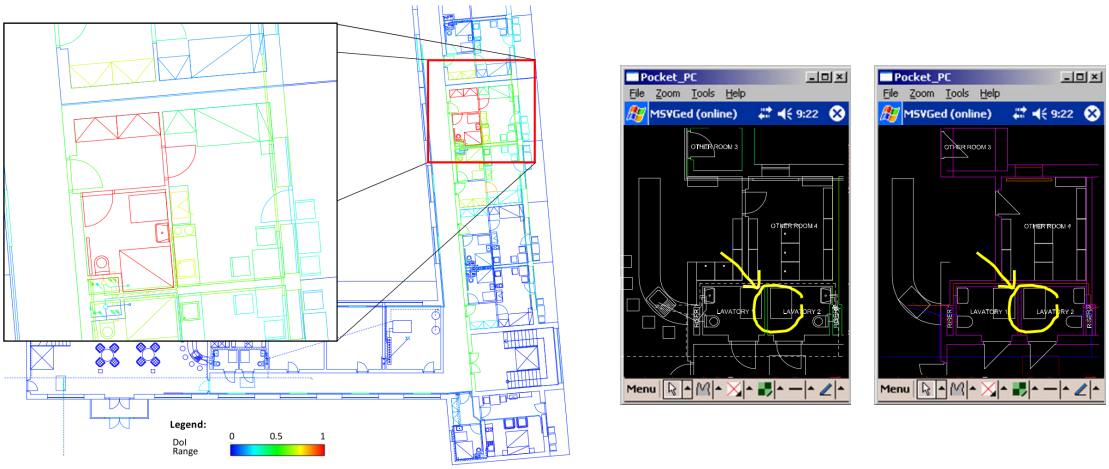


Figure 2.10: [Kli09]: An external RDF allows for structure- and/or semantic-based DoI propagation across elements of an SVG floor plan (left); resulting in different adapted representations of the same RoI for different users (right).

visual and even basic information scalability for vector graphics as well. Most of these target the XML-based SVG format, either by using SVG's extension mechanism, or by providing external meta data. With few exceptions, however, these approaches do not incorporate task specifications into their adaptation process.

3D Models, or objects in a virtual scene, describe graphical content by its geometry in $\mathbb{R}^3$ (usually as polygonal or triangle meshes) and associated visual attributes ("material"). Similar to 2D vector graphics, 3D models exhibit inherent display and visual scalability by virtue of the wide range of parameters that can be adjusted during rendering.

Display scalability is achieved by adjusting the pixel resolution of the scene's rendition to the screen. This may result in very small triangle sizes in image space (< 1 pixel) and it is often more necessary to achieve interactive framerates than showing fine details. Thus, Level of Detail techniques are an important part of many rendering techniques for large scenes and highly detailed objects [HS06]. To this end, various algorithms to simplify the surface mesh of polygonal 3D objects have been proposed. The majority of approaches can be distinguished by if they generate continuous LoD (e.g., [GWH01a]) or select from a pre-calculated set of discrete ones; and if they use view-independent or view-dependent measures (e.g., screen-space error [GH97]) to determine the appropriate LoD (see [LRC+02] for an overview). Others swap geometry with 2D impostors [Jes05] or surface samples (surfels) [HS07, Hol07] that require less memory. Common to all approaches, however, is that they try to maintain the perceived appearance of objects so as to minimize the visual impact of simplification.

Aside from basic display scalability to generally improve rendering efficiency, visual and information scalability are essential for a number of applications where 3D models are used to produce (interactive) representations with a defined communicative goal. Here, the photo-realistic depiction of a real object is not paramount, rather the task at hand requires to extract specific information from the 3D representation (cf. Fig-

ure 2.11). Examples include interactive illustration systems that provide interactive versions of educational graphics similar to that found in medical textbooks [BG05] or step-by-step assembly instructions [APH+03]. Augmented Reality (AR) and Virtual Reality (VR) approaches are employed to reduce time, cost and risks of specialist task training, like virtual trainer models for machine operators [MVL09, LM09] or virtual operation theaters for medical personnel [TIP05]. Such interactive tutoring systems need to communicate specific aspects of a complex model [PR02]. A primary question therefore is how to accentuate relevant object features while minimizing the visual impact of irrelevant or occluding components by a combination of visual and information scalability.

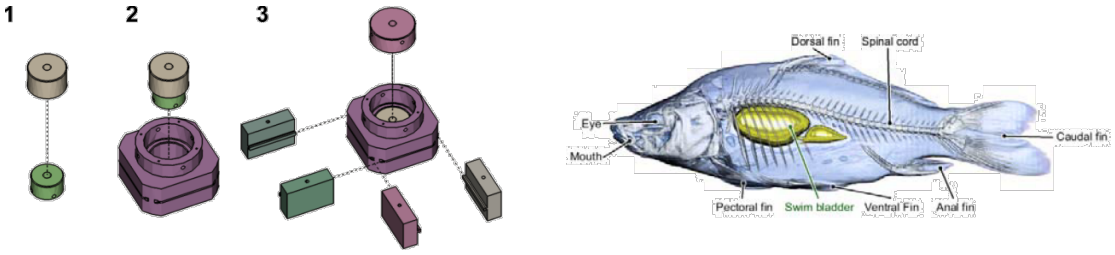


Figure 2.11: 3D illustrations with a specific communicative goal necessitate both visual and information scalability in order to adapt what aspects of a model are presented how, e.g during subsequent steps of assembly (left, from [APH+03]) or when highlighting specific organs in a medical illustration while subduing surrounding tissue (right, from VolumeShop [BG05]).

Straightforward visual scalability utilizes the scene setup (virtual camera, lighting) to alter both the perspective and the object's appearance. This can include local alteration of the selected LoD (see above), lighting and/or material properties (e.g., color) to accentuate object features [Hop99, PR02]. Some systems automatically determine optimal views which minimize occlusions [VÓ3, Vio05, GHS06] or plan camera movements to guide the interactive exploration [BRS01].

To complement this basic adaptation, a large number of more sophisticated rendering techniques has been developed over recent years. Non-photorealistic Rendering (NPR) styles [SS02] are a good way to highlight important object features. Specialized shading functions such as Gooch shading [GGSC98] may be applied to improved perception of shape and depth. Local transparency (ghost views) [DWE02, BG05] and cutaway views [BG05, DWE03] are very efficient methods to remove or subdue irrelevant occluding geometry, mimicking their long-established illustration technique counterparts from print media. Dynamic labeling is a method to further establish co-referential relations between textual information and visual elements [AHS05, LSC08].

All aforementioned approaches that enable visual scalability require additional information about the semantic composition of the 3D object e.g., to selectively apply NPR styles and ghosting. Such decomposition information further affords information scalability: components can be presented at a lower LoD or even removed from the representation entirely. To this end, external meta data formats are often used. For example, in [KLK03] an extension to the MPEG-21 DIA (Digital Item Adaptation) standard [Pat03] is presented for 3D content.

However, such meta data may not be available due to limitations of the data format, intellectual property protection considerations, or simply because no meta data was generated for the model components in the first place (cf. Section 2.2.1). Decomposing a model into meaningful components [ARSF07a] (Figure 2.12, left) and enhancing raw geometry with additional high-level shape information [ABF+06] (Figure 2.12, right) thus are important topics in both industry and academia. To this end, several segmentation methods have been devised [ABF+06]. Good recent overviews can be found in [AKM+06] and [Sha08].

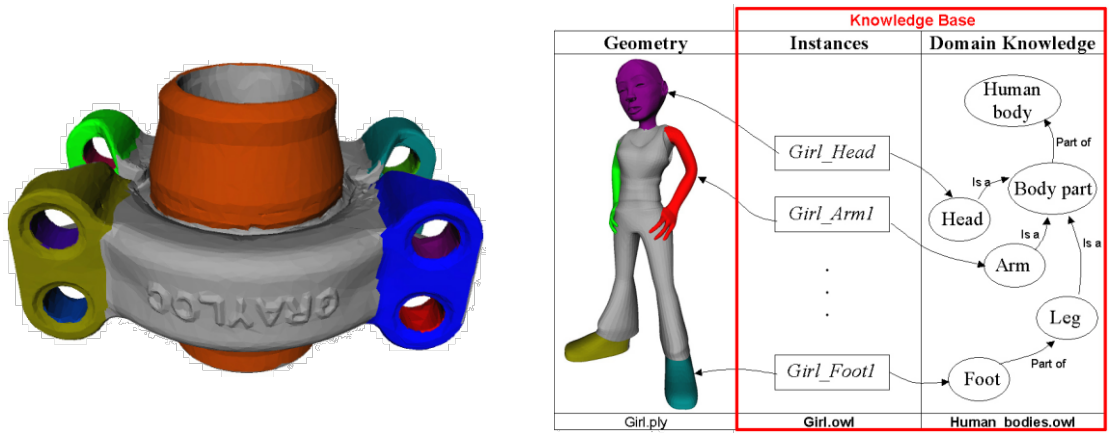


Figure 2.12: Without external meta data, mesh segmentation is necessary to extract meaningful components from the raw geometry (left). The segmented object can then be enriched with semantic information on individual segments and (topological, functional) relations between them to enable information scalability (from [ARSF07a]).

The segmentation of a surface mesh can be carried out either according to purely geometric aspects e.g., [MW99, GWH01b, AFS06]; or semantic-oriented e.g., [LZ04, MPS+04, KLT05]. In the former case the mesh is decomposed into patches that are equal with respect to a certain property (curvature, distance to reference plane) whereas latter methods try to identify object parts that correspond to relevant features of the object's shape. Geometry-based approaches are sometimes used as a pre-process to the detection of relevant features. Semantic-oriented approaches have enjoyed increasing attention in recent research because they support morphing, 3D surface reconstruction and skeleton extraction, which however cater primarily to the needs for efficient mesh storage and -retrieval rather than their visual representation.

Generally, different algorithm perform better for certain types of model features. For this reason, Attene et al. [ARSF07b] propose what they call *multi-segmentation*i.e., to use several segmentation algorithms in parallel and to mix-and-match their individual segmentation results.

There are some approaches that employ a combination of the general-purpose algorithms and techniques listed above to generate adapted 3D representations for a specific communicative intent. Selligmann and Feiner [SF91] propose IBIS (Intent-Based Illustration System) that treats the design of an illustration as a goal-driven process

within a system of constraints. The goal is to achieve the communicative intent; the constraints are the illustrative techniques an illustrator can apply. Later, IBIS was integrated into KARMA [FMS93], one of the first systems to utilize head-mounted displays for AR support of maintenance tasks. Krüger proposed a knowledge-based system called ARP [Krü98] for automating graphical abstractions of 3D models according to a presentation context. It, too, uses sets of attention and abstraction rules to support the human illustrator. More recently, Viola [Vio05, VFSG06] proposed an importance-driven scheme to generate expressive illustrations from volume data that is able to seamlessly compose different rendering styles and illustrative techniques for individual features.

In conclusion, there exist a wide variety of techniques that provide display and visual scalability for complex 3D models, as well as methods to enrich such models to enable information scalability. There also exists a number of approaches that integrate these into illustration generation systems that have the notion of a communicative goal; however, these usually provide rule-based systems that formalize illustration knowledge to support authors in creating static illustrations, rather than aiming at the automated adaptation of visual representations in the face of changes to the task at hand.

Abstract Data have no intrinsic visual representation, but rather must be transformed into an image through a visualization process. As such, visual representations of abstract data are, on principle, always scalable with respect to all five scalability issues as outlined in Section 2.1.3. Rather, contemporary visualization approaches can be distinguished in terms of the influencing factor categories ('5Ws', cf. Section 2.1.2) that are contemplated for the visualization process. Particularly relevant to this thesis are differences in how approaches consider data characteristics (WHAT) and visualization goals or user tasks (WHY). A review of the current state of the art yields the observation that approaches can be grouped into two categories in this regard.

In the first group are particular tools for a specific application/problem domain with a defined set of visualization and interaction techniques, e.g., meteorology [Tre99], electrical power system surveillance [OW00], stock market [MSK+06] and computer security analysis [Suo09]. Associated domain tasks and user requirements are typically derived through user interviews and evaluation/refinement cycles [Mun09] but often do not include a formal task specification on different granularities [WPF04].

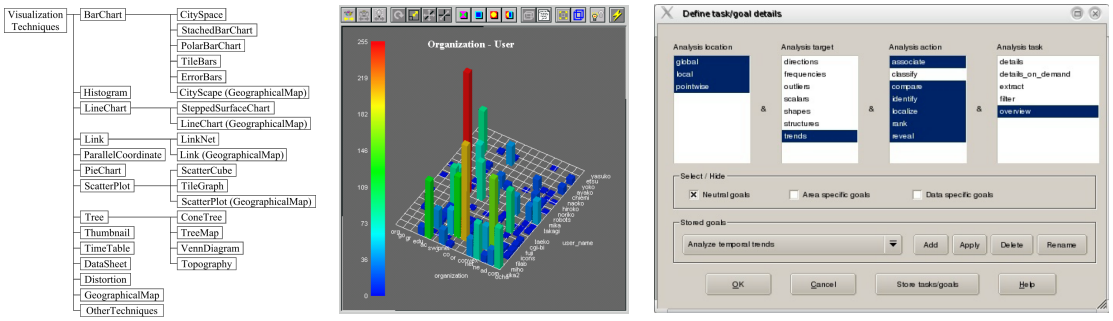
Another group comprises approaches that are founded on a systematical description of data classes and analysis tasks in order to arrive at an appropriate visualization design. The early works concentrated on the automatic design and generation of presentation graphics. Systems like APT [Mac86], BOZ [Cas91], SAGE [RKMG94], and VISAGE [RLS+96] were tools for generating 2D static diagrams, e.g., scatter plots and bar charts, taking relational data from arbitrary applications as input. Therefore, research effort was directed at finding suitable descriptions of data domains and their intrinsic relations to encode that input (i.e., selected WHAT aspects), as well as formalisms to specify visual encoding rules used to build appropriate visualizations.

There are also a variety of approaches to formalize WHY aspects in visualization systems and to relate them to data types and characteristics. A pioneering approach to

consider the task that should be supported by the visual representations was introduced by Casner [Cas91] for his BOZ tool. The main idea can be summarized as follows: The user specifies his task at hand by a sequence of predefined primitive logical operators (e.g., a query for a property of an object). This sequence of operators serves as the input to BOZ. Here, the logical operators are replaced by so-called perceptual operators (e.g., the lookup for a property of an object). Finally, appropriate visual attributes are automatically selected to realize the perceptual operators.

Roth and Mattis [RM90] introduced low-level information seeking goals like value look up, value comparison, or finding correlations. They use these goals together with meta data for relational data (e.g., scale type, cardinality of data records) as the dimensions of a taxonomy to characterize visualization expressiveness. Similarly, Wehrend and Lewis [WL90] separate goals into objects (e.g., scalars, positions) and operations. They define 11 operations (tasks) that might be performed on visualizations (e.g., identify, locate, distinguish or cluster). This separation allows specifying arbitrary targets (WHAT) independently from the type of action the user would like to perform (WHY). The paper also compares their task taxonomy with that of [RM90]. Keller and Keller [KK93] present a collection of visualization examples for various combinations of analysis goals and data types. Shneiderman [Shn96] has introduced his Task-by-Data type Taxonomy specifying seven generic tasks (e.g., overview, zoom, filter) and seven data types (e.g., 1D, 2D, Network). Notable of his approach is that the four tasks overview, zoom, filter and details on demand (the "information seeking mantra" [Shn96]) imply a hierarchical decomposition of both task scope and level of abstraction in the associated data. Fujishiro et al. [FFIT00] also describe analysis targets and explicitly differentiate them from analysis actions for their GADGET/IV system. They derive their taxonomy of visualization goals from an enriched Wehrend and Lewis matrix [WL90]. Furthermore, they combine these two categories with Shneiderman's tasks [Shn96]. GADGET/IV uses this information to propose and initialize a suitable visualization technique from a set of available ones, cf. Figure 2.13a, b. Nocke [NS04, Noc07] provides a systematic review of previous approaches to develop an analysis goal specification based on four main aspects (data-driven, user-driven, context and complexity, cf. Figure 2.13c) that incorporates many of the previous works. The intent of his approach, however, is more to offer a guide for semi-automatic visualization design rather than to provide concrete methods for automatic generation of visual representations. In a similar fashion, Amar et al.[AES05] argue for a specification of analysis goals as compound tasks, i.e., a combination of low-level analysis tasks, but do not address in detail how these should be applied to concrete visualization techniques. Andrienko and Andrienko [AA05] have developed a formal typology of visualization tasks at different levels of data granularity, but it, too, does not describe how to create concrete visualizations.

The aforementioned approaches consider a combination of WHAT and WHY aspects, but either specify tasks at a low level only, or specify higher-order goals only informally, generally as verbal lists. This makes it difficult to make reliable design choices satisfying all requirements for more complex task settings [Pla04]. For this reason, evaluation of visualization techniques and systems regarding their expressiveness and effectiveness has also garnered attention from the visualization community (e.g., [AS04, BC06, Mun09]). There is, however, some dispute as to the value of current evaluation methodologies,



(a) (b) (c)

Figure 2.13: Like many similar approaches, the GADGET/IV system [FFIT00] provides a predetermined set of visualization techniques (a), from which the most suitable one is selected based on low-level analysis goals and/or analysis actions (b). Similarly, the visualization design assistant in SIMENVVIS [Noc07] allows to specify analysis goals by combining several low-level task aspects (c).

considering most are based on informal test cases with small samples [Pla04, Nor06]. The general trend therefore is towards a more formalized approach to both design and subsequent evaluation of visualizations [AS04, BC06], not unlike methodologies employed in model-based UI design processes (cf. Section 2.3.1). An interesting work in this regard has been presented by Winckler et al. [WPF04] who use the CTT formalism [PMM97] to build task models based on the Wehrend and Lewis matrix [WL90] for structured testing and evaluation of interactive visualization techniques. A recent publication by Munzner [Mun09] proposes a nested model for visualization design and validation that puts the WHY and the WHAT aspects (user tasks and data, respectively) on the outermost – i.e., most important – layer.

In the same publication, Munzner further conceptually distinguishes four different levels of task granularity: high- and low-level domain as well as high- and low-level abstract tasks. Domain levels capture the semantics of a process (e.g., 'find a cure for a disease' as a high-level goal in the medical domain), whereas abstract tasks describe a work flow decomposition on a purely functional level as for example defined by Roth and Mattis [RM90] ('compare values', 'correlate data points'...). This important distinction is rarely explicitly made for the contemporary approaches reviewed above. This underlines the observation made here that their majority is either purpose-designed for a single to very few specific domain-level tasks with informally derived functional requirements, or concentrate on generic analysis tasks neglecting the application context.

In conclusion, virtually all visualization approaches address the characteristics of the data (WHAT). And although some approaches for visualization of abstract data consider isolated aspects of the task at hand (WHY), the adequate integration of the task at hand into the visualization process on multiple levels of abstraction and granularity (i.e., from high-level domain to low-level functional) is still a largely open research question. Most approaches are purpose-designed for a specific problem domain with an associated set of domain-level tasks. Those that address issues of systematic visualization design do, for the largest part, provide taxonomies for matching standard visualization techniques to

the fundamental data types (e.g., [RM90, WL90, Shn96]), or to comparatively low-level analysis tasks (e.g., [RM90, Cas91, AES05, Noc07]). One notable exception is the work presented in [AA05] that presents a visualization task typology based on a formal model, which however does not describe concrete means for how the actual visual representation should be generated.

2.4 Summary

In this chapter relevant terms and concepts that are used throughout the thesis have been introduced and disambiguated where necessary. Two application scenarios have been presented that illustrate both the need for, and requirements to, scalable visual representations: mobile maintenance support and smart meeting rooms. These two scenarios represent use cases from two academia-industry research projects providing the practical background to the contributions of this thesis. It has been shown that scalability of visual representations with respect to the representations context of use, in particular the task at hand, is one of the most important aspects in both scenarios. Consideration of the user's task must therefore be integrated into the design process of the visual interface.

To this end, as a first contribution we proposed in Section 2.1.2 a categorization of influencing factors into five **high-level aspect categories**. These **'5Ws'** – WHAT, WHY, WHEN, WHERE and FOR WHOM– define the context of use of visual representations within Smart Visual Interfaces. This partition of the influencing factors is not arbitrary: it relates to the different dimensions of visual representation scalability (Section 2.1.3) that is required to actually accommodate for changes to the context of use.

This categorization is used to derive a **classification of the graphical content** that constitutes the basis for task-specific visual representation within smart visual interfaces. In this thesis, for the purpose of task-driven adaptation we propose a distinction into **four principal visual data types** depending on their degree of inherent scalability: raster graphics (only imaging operations), vector graphics (manipulation of vector primitives), 3D graphics (added control over the rendering process), and information visualizations (control over the entire image generation process including visual mapping and data filtering).

Further, this chapter reviewed related work to the approach presented here. Related works thereby comprises two main areas. On the one hand, design methodologies based on formal models derived from (domain) task analysis are currently state of the art in the field of software engineering, UI design in particular. These approaches can potentially form the basis for the overall design process of smart visual interfaces. However, the vast majority of corresponding model-based UI design methods is geared towards the specification, creation and evaluation of so-called WIMP (Windows, Icons, Menus, Pointing Device) interfaces. Graphical content is usually considered as a *static* resource that is displayed by virtue of a GUI widget. As a result, model-based UI design methodologies do not usually capture the WHAT and WHY aspects pertaining to visual content

in sufficient detail to enable task-driven scalability of embedded visual representations.

On the other hand, there exists an large number of approaches to visual representations that are scalable with respect to their context of use according to the proposed categorization. Many of the reviewed contemporary approaches deal with scalable representation of graphical content of the visual data types raster images, vector graphics including maps, and 3D models. Only very few, however, include an explicit specification of the user's task at hand, relying instead either on interactive manipulation by the user, or on an implicit consideration by the content author. Similarly, although some approaches for visualization of abstract data consider isolated aspects of the task at hand, its adequate integration into the visualization process is still an open research question. Most approaches are either purpose-designed for a specific problem domain with an associated set of domain-level tasks, provide taxonomies for matching standard visualization techniques to the fundamental data types, or rely solely on low-level abstract analysis tasks detached from higher-level tasks and work flows in the application domain.

It is the aim of this thesis to propose first steps toward closing this "modeling gap" as a contribution to the design of task-driven Smart Visual Interfaces. To this end, the following chapter provides a detailed problem and requirement analysis for the dynamic adaptation of visual representations to the task at hand.

3 Basic Approach to Task-driven Adaptation of Visual Representations

Smart Visual Interfaces aim at the dynamic generation and presentation of visual content for a given context of use. The previous chapter discussed how the influencing factors constituting the context of use can be broken down into five higher-level categories ('5Ws', Section 2.1.2). This categorization is not arbitrary. It relates to different aspects of scalability that must be provided by the visual interface to accommodate for changes to the context of use, as illustrated by Figure 2.1 in Section 2.1.3.

The multitude of possible influencing factors in most problem domains make a holistic solution unfeasible. Rather, the following Section 3.1 explicates the objectives of our approach in particular, and which influencing factors thus to be considered. Section 3.2 analyses challenges and open research questions arising from resulting requirements. Section 3.3 then presents the basic concepts underlying the approach for task-based adaption of visual representation developed in the present thesis to address these challenges.

3.1 Scope and Objective of the Approach

Following the interpretation of *Smart Graphics* as "intelligently made graphics, which [...] enable humans to make much better sense of them" [KB09], in this thesis we consider *task-based adaptation of graphical content* according to the user's current task at hand as the most important function of smart visual interfaces. In other words, by visually presenting content and information in a way that is immediately useful to the user in the context of her current task, "smart" user support can be realized. With regards to this focus the different aspect categories of the '5Ws' that must be contemplated during the adaptation process vary in significance.

Clearly, the graphical content associated with the task at hand is the subject matter of adaptation. Its type and data format determines how it can be visually represented to the user. This is captured by the WHAT aspect. It must therefore always be considered for the adaptation process.

Depending on the task goal, the user needs to extract specific information from the visual representation associated with the task at hand. Thus, different aspects of the graphical content/data are relevant. The content's visual representation must not only show all relevant information at an appropriate level of abstraction (i.e., at an appropriate

information scale), but also visually emphasize them over contextual aspects to effectively convey this information (i.e., at an appropriate visual scale).

However, graphical content often was not created with the same communicative goal as that appropriate for the task at hand. This is especially the case where available 'raw' content (e.g., digitized illustrations, schematics, cf. Section 2.2.1) is associated with several different tasks. Therefore, to ensure relevant aspects of the data are suitably represented, adaptation of the raw graphical content is required according to WHY aspects.

Furthermore, the relevance of specific content aspects and information is not determined by the current working step alone. Rather, the task at hand is also defined through the current position in the superordinate composite workflow: information communicated through visual representations for previous tasks influence what information is currently required in what detail, and the context in which they are perceived. Thus, the WHEN aspects of the task at hand define an important portion of a visual representation's context of use.

Output device capabilities and work environment conditions further introduce constraints for this adaptation process. Device-specific properties such as available display space and color resolution directly affect the visual representation. The device's interaction facilities as well as environmental conditions, such as light or noise levels, influence the utility of visual communication compared to alternate interface modes; this in turn influences what information should be presented visually (see Section 2.2.1).

While constraints imposed by WHERE aspects must be regarded in choosing appropriate adaptation methods and techniques in terms of display scalability for a particular target device, a primarily device-*driven* adaptation of visual representations is not in the focus of this thesis. Note that this problem aspect has indeed been addressed in the scope of project MuSAMA. Refer to the PhD thesis of Thiede [Thi10] for a primarily device-driven adaptation approach as well as an extensive discussion of related WHERE aspects.

Incorporating the first four aspect categories for task-based adaptation also addresses several user-related issues (FOR WHOM). Considering content- (WHAT) and task-related (WHY) aspects results in visual representations that effectively communicates all required information without requiring the user to manually select and navigate the content. In addition, considering each task in the context of its superordinate workflow (WHEN) makes it more likely that adapted representations match the user's mental map of the problem domain across subsequent working steps. This cannot be ensured if adaptation is based solely on properties of isolated working steps. This increases the effectiveness of the adapted representations. Therefore, while we do not explicitly consider user-related aspects such as cognitive capabilities or domain expertise, several advantages in terms of user support are nonetheless provided implicitly.

The above set of influencing factors defines the context of use considered for the adaptation of visual representations in our approach. Figure 3.1 illustrates this scope with

respect to the overarching categorization of the context of use proposed in Section 2.1.2 (cf. Figure 2.1 on page 7) of visual representation within smart visual interfaces.

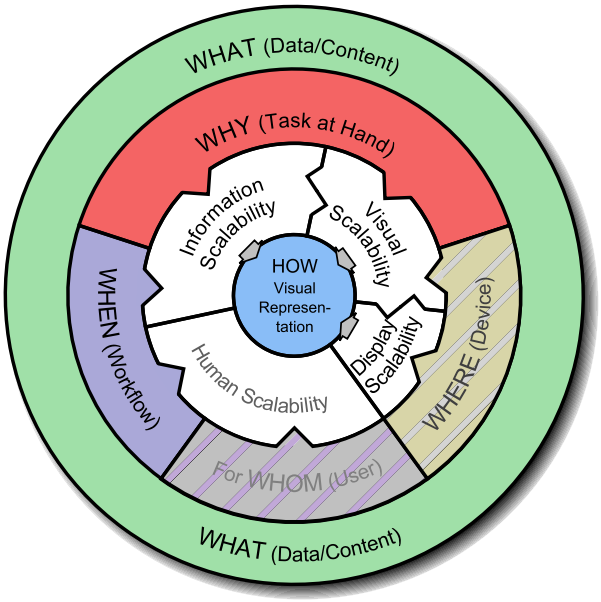


Figure 3.1: Scope of the approach presented in this thesis: provided visual content (WHAT) is adapted according to task requirements (WHY + WHEN). This primarily necessitates the provision of information and visual scalability. The output device (WHERE) imposes constraints on the adaptation process. User-related aspects (FOR WHOM) are only addressed implicitly.

Given this scope, the primary objective of the adaptation process in our approach is to provide an *initial representation* for each basic task that is effective and as efficient as possible. This means that all relevant information in the context of the task at hand are communicated to the user without necessitating active involvement. Automatic task-based adaptation of visual representations thus constitutes "smart" user support through the visual interface.

However the aim is *not* to restrict the user to that, and only that, automatically generated representation. Rather, task-based adaptation is to provide a good initial view on the information required for the task at hand. While striving to minimize the need for manual navigation, the user should always retain the ability to override the adaptation process in cases it does not meet her specific preferences and requirements. Thus, the user can always change and modify the current view interactively.

Besides the primary objective to provide task-based adaptation of visual representations as the smart visual interface's most important functionality, a secondary aspect is the integration of such methods into the design process of the overall system. The two application scenarios that stimulated the research presented here both put a strong emphasis on multiple user interface design to accommodate the large variety of devices typically encountered (compare Section 2.2). In this area Model-based User Interface Development (MBUID) approaches play out their particular strengths [Luy04]. In corre-

spondence, the further development of MBUID processes to efficiently support multiple, multi-modal user interfaces has been one of the main project areas (see for example [RF05, BDF+06]).

To effectively utilize visual communication through smart visual interfaces – as a specific type of user interface – in these application scenarios, their development has to be integrated into the superordinate model-based design process. The utility of this integration is two-fold:

- Model-based UI design processes use as their starting point a task model that describes a hierarchical decomposition of the workflow into sub-tasks and their interrelation. These models, as the result of a prior domain task analysis,[1] thereby provide a description of the WHEN aspect for each basic task. The main idea is to also use this model as the basis for the specification of adapted visual representations presented through the smart visual interface, rather than effecting the adaptation of graphical content solely on an unstructured set of low-level tasks.
- Task-based adaptation of different types of graphical content is but one aspect of complex information systems. As such smart visual interfaces augment, but do not replace, traditional GUI (WIMP interfaces). The task model provides the basis for a common MBUID process integrating multi-modal and smart visual interfaces development (cf. [BDF+06, FS06]).

For the two application scenarios outlined in Section 2.2 in particular, the Concur Task Tree (CTT) notation [PMM97] had been chosen as the task model formalism.

To summarize, the objective of task-based adaptation within the scope of this thesis is to adapt the **graphical content** (WHAT) associated with the **task at hand** (WHY + WHEN) in such a way that the content's **visual representation** (HOW) provides a **good initial view** with respect to the information relevant for the current working step. In doing so, the adaptation process is subject to output device constraints (WHERE). The automatic adaptation process significantly reduces exigency of manual navigation. In combination with suitable interaction functionality an even larger degree of user support can be attained by providing means to fine-tune the visual representation according to her expertise, preferences, and situational requirements. As a secondary aspect, the task-based adaptation and corresponding description of the influencing factors WHAT, WHY and WHEN should facilitate integration into an overall model-based UI design process. In light of the project background of this thesis in particular, for the underlying task model the CTT notation was predetermined.

3.2 Problem Statement

The scope outlined above can be succinctly summarized as "from task to visual representation." This can be distinguished into two challenges:

- The need to provide an adequate task description, and

[1] Task analysis is a complex field in its own (e.g., see [HR98]) and thus is well outside the scope of this thesis.

- providing means for an adequate adaptation control on the basis of this description.

3.2.1 Requirements for Task Descriptions

An adequate task description to this end further comprises two major aspects:

- the description of the graphical content associated with the task (WHAT), and
- the specification of the task-related aspects of the visual representation's context of use (WHY and WHEN), or *task context* for short, that provides the basis for adaptation.

The graphical content is the subject matter of task-based adaptation. Its task-specific visual representation is based on choosing appropriate information and visual scales of the graphical content with respect to the requirements of the task at hand. Therefore, requirements to the content description extend beyond that of a mere graphics storage format. Rather, it is necessary to provide a *scalable* description of graphical content independent of the underlying visual data type. This is a challenge because the four visual data types exhibit varying degrees of inherent scalability due to differences in their expressiveness, i.e., the amount of semantic and structural information available on depicted content. From the perspective of the adaptation process, it is therefore necessary to introduce an abstraction with respect to these type-specific variances so that the association of tasks with (self-contained units of) graphical content can be handled uniformly regardless of its type.

The scalability facilitated by such a suitable content description must then be utilized during the adaptation process. In particular, the specification of the task context must allow to determine the following characteristics of the visual representation with respect to the task's communicative goal:

- What information from the graphical content are required
- on what level of abstraction (i.e., the content's information scale), and
- how it should be visually encoded for communication (i.e., the content's visual scale).

The selection of relevant portions of the graphical content, i.e., what information are required for the task at hand, is determined primarily by the requirements for the current working step (WHY). This can be understood as the *local task context*.

However, the appropriate abstraction level of the presented information as well as its visual encoding can not be determined solely by contemplating individual low-level working steps in isolation. Instead, the task context also includes the position of the current working step in the composite workflow (WHEN). In particular, temporal and causal relations to other subtasks of the task at hand influence the relevance of particular information. This can be understood as the *extended task context*. For example, facts that have been previously communicated to the user potentially require less detail in subsequent steps. On the other hand, generating visual representations for a series of

successional working steps often involves the progressive refinement of an initial representation, i.e., by successively adding more detail information. Thus generally, visual representations supporting isolated low-level tasks are subject to different criteria than those for composite tasks comprising multiple working steps.

This claim has been substantiated in a user experiment [Son07]. Three different settings for the combination of a 3D visualization with different labeling styles were tested for three different tasks: two low-level tasks with the goal of locating a labeled object in 3D, and one more complex 3D scene exploration task. The latter task comprised two working steps, requiring participants to first identify an object by textual information before it had to be located in the 3D scene. The relative effectiveness of each setting was determined by comparing the time it took the participants to finish the task and the error rates in locating the objects. Participants were also asked to rate each setting according to the relative degree of support it offered for the respective task. In all cases, completion time, error rate and user acceptance matched. While the third visualization setting was rated to be the worst by half of the participants when performing the low-level tasks, a majority of 64 percent favored that very third setting for the complex exploration task. By contrast, the second setting, which was a close second-rated for the simple 'locate' tasks, was rated worst almost unanimously by 97 percent of the participants for the exploration task.

The majority of methods reviewed in Section 2.3.2 aimed primarily at providing scalable representations of graphical content without considering the task at hand at all. Integration of user tasks into the visualization process for abstract data has been investigated more extensively, nonetheless contemporary approaches are based on taxonomies of low-level analysis tasks only. This means that even those approaches contemplate only the local task context.

Therefore, it can be stated that the methods outlined in Section 2.3.2 do *not*, in and of themselves, provide an adequate basis for adaptation of graphical content within smart visual interfaces.

Instead, for the specification of the extended task context that is suitable for task-based adaptation, the further integration of the local task context with a description of the superordinate workflow is required. In particular, this workflow description must provide a decomposition of the task at hand into working steps that are each associated with a visual representations capturing causal and temporal dependencies between them (WHEN).

Corresponding task models developed for model-based user interface design provide a description of the workflow which the designed system is intended to support as a tool. These models – including the CTT formalism predetermined by the application background – describe the decomposition of the task at hand as a hierarchical structure of *task nodes* with parent-child as well as temporal and causal relationships between task nodes (cf. Section 2.3.1). Thus a path through this structure up to the basic task node of the current working step comprises the extended task context for that working step.

What these modeling approaches do *not* provide, however, are means to describe scalable graphical content. Although several models include the concept of 'task objects' at the level of basic tasks, and some further distinguish these into 'application objects'

from the application data model and 'perceivable objects' as their representation in the UI, contemporary approaches are geared towards the specification of user-machine interaction using typical WIMP GUI. In particular, perceivable objects (interactors) are assumed to be commonly available controls such as text fields, buttons or menu entries. Graphical content is limited to static 2D images, whereby scalability requires the specification of multiple predetermined versions of the image, of which one is selected during subsequent transformation of the task model into the presentation model (i.e., the abstract UI).

Thus, task models used for general model-based UI can be used as the basis for task-based adaptation of visual representations within smart visual interfaces. They already provide a suitable description of the superordinate workflow that defines the extended task context for adaptation. However, in order to incorporate scalable graphical content of the different visual data types – as the subject matter of adaptation – these task models must be extended. Since task models are specified at the conceptual level, these extensions must maintain abstraction from technical peculiarities of the visual data types.

3.2.2 Adaptation Control

Based on a suitable description of the graphical content (WHAT) and the task context (WHY, WHEN) the adaptation process must be steered so that a good initial view (HOW) for the current working step can be tailored automatically. Ideally, the adaptation process can utilize existing presentation techniques to this end. Although there is a variety of presentation techniques available to generate scalable visual representations based on the four visual data types, these generally have no concept of an explicit communicative goal which could be utilized to effect task-based adaptation. Instead, most approaches rely on the interactive manipulation of presentation parameters by the user.

For the task-based adaptation is therefore necessary to map the task context onto a suitable parameterization of individual presentation techniques. The extended task model then must be enriched in a suitable place with these adaptation control information. The challenge in doing so is to abstract from each specific techniques and their respective parameter space so that the enriched task model can still be described on a conceptual level, i.e., without introducing data type- or presentation technique-specific implementation details.

3.2.3 Open Research Questions

The above-identified challenges can be summarized as that there currently exists a 'modeling gap' between scalable display techniques on the one hand, and task modeling approaches for model-based UI design on the other hand. The former insufficiently contemplate task-related influencing factors of the context of use for visual representations. The latter by contrast widely neglect the integration of scalable graphical content that is the foundation of task-specific visual representations. Thus, closing this modeling gap requires to address three elementary questions:

- What is a suitable description of scalable graphical content in a way that it can be integrated into a high-level (domain) task model? In particular, the description should accommodate graphical content of all for visual data type while abstracting from implementation and data format details at the task modeling level.

- This model must further be extended (*enriched*) to capture the information and visual scale requirements to task-specific visual representations. What is a suitable model-level abstraction of WHY aspects specific to graphical content and its visual representation that is applicable to all visual data types?

- Finally, the information from the extended task model is employed to effect the adaptation process so that a good initial view is automatically generated for the current working step. To utilize existing display techniques for scalable representations to this end, a bridge must be provided between the model-level abstractions of the WHAT and WHY aspects, and concrete implementations for the particular visual data type. What is a suitable framework to facilitate this catenation for arbitrary display techniques?

3.3 PAGE-FEATURE Concept

In this section, we present our basic approach to address the three challenges outlined above as an essential step towards closing this 'modeling gap'. In doing so, our approach provides a means for the model-based development of smart visual interfaces and their integration into a superordinate MBUID process.

3.3.1 Concept Overview

Starting point is the general task model resulting from the domain task analysis phase (cf. Figure 2.5). Conceptually, this model is a hierarchy of distinct task nodes describing the extended task context (WHEN) for each basic task (leaf task nodes). Subsequent examples will use the CTT notation as a representative of this type of models.

The extension of this general task model required to accommodate task-specific visual representations occurs as part of the model-based smart visual interface design process. This comprises two steps:

- Definition of units of graphical content (WHAT) and their association with tasks, and

- further detailing of the task context (WHY) for each basic task with respect to task-based adaption of visual representations.

To support these two steps on the conceptual level of task models, in this thesis the concept of PAGES and FEATURES has been developed. These are introduced to provide the necessary abstraction from implementation-related aspects of the four visual data types. PAGES encapsulate the description of scalable content, while FEATURES are used as a means to designate relevant aspects of the content with respect to the task context.

Thereby, PAGES and FEATURES constitute the basic building blocks for the extension of the general task model towards task-specific visual representations.

Extending the task model is done by *annotating* task nodes with these building blocks. This is in line with several approaches that require extensions to the original notation, cf. Section 2.3.1 on page 19. In the following two subsections we will introduce both concepts.

3.3.2 WHAT-oriented aspect – PAGES

The purpose of a PAGE is to provide a conceptually self-contained unit of graphical content that abstracts from the intricacies of the underlying visual data type. This abstraction barrier offers two advantages:

- It facilitates the association of multiple tasks with the same graphical content. Multiple working steps of a composite tasks often require adapted representations derived from the same graphical content. In these cases, a single PAGE definition can be associated with the corresponding basic tasks, rather than requiring to specify a data format-specific description for each task individually.

- It decouples the creation or acquisition (e.g., scanning printed documentation) of graphical content from the design activities to create the extended task model. In particular, without using a suitable abstraction on the conceptual level, subsequent task model transformations (e.g., into a presentation/dialog model) would otherwise need to be aware of the intricacies of different data formats. This would severely hinder the integration of smart visual interface components into the overall UI design process.

The designation for units of content as PAGES was chosen because it reflects the navigation metaphor *Scrolling and Paging* nicely. During a single working step the page is "scrolled," i.e., interacted with, and between tasks pages are flipped over to show the next initial representation.

Thus, PAGES are an abstract *container for graphical content* of different visual data types. In addition, we stipulate a PAGE contains all supplemental information required to generate a *default representation* for the contained data.

The raison d'être for this stipulation is that this allows to associate the graphical content encapsulated by the PAGE with any task without the need to specify further details of the task context. In doing so, the task context must be enriched with additional information only for those tasks where it is appropriate from the workflow to further deviate from the default representation.

Concrete instances of PAGES differ depending on the underlying visual data type. Besides technical aspects like the internal data formats, this primarily regards what additional information must be provided to generate a default visual representation of the encapsulated data.

To give an example, a raster image file is in itself sufficient to describe a PAGE, as the image can be directly used as default representation. On the other hand, content

comprising 3D graphics requires additional parameters beyond the plain triangle mesh data to derive its visual representation. This includes a default view point definition and lighting information.

However, these intricacies are irrelevant for the purpose of extending the general task model. Details of PAGE internals will thus be discussed only in the respective chapter 4–7 for each of the four data types.

The next question is how PAGES are used to extend the task model. Generally, graphical content is associated with tasks by *annotating* (linking) the corresponding task nodes in the model with the PAGE definition. In doing so, different strategies are viable regarding the position of the PAGE annotation in the task node hierarchy:

- only at the level of basic tasks, i.e., each leaf node is annotated individually, or
- also at different levels of composite tasks, i.e., the PAGE is annotated once at the root of a sub-hierarchy comprising multiple sub-tasks.

The first strategy amounts to associating the graphical content of a given PAGE with every working step that references it by individually annotating all corresponding basic task nodes. The task model is thus extended locally with regards to the WHAT aspects of the task context. This strategy therefore is equivalent to an approach based on low-level (basic) tasks only: it allows to contemplate basic tasks in isolation (i.e., detached from the task model) because all WHAT-related information are specified in the local task context. This does, however, introduce redundancies in the extended task model as the PAGE annotation must be explicitly specified for all corresponding basic task nodes.

The second strategy utilizes the hierarchical relationships between task nodes in the task model. Here, instead of associating graphical content with multiple working steps individually, it is associated with the higher-level composite tasks that comprises these working steps. Thus, the PAGE is annotated only once in the task model, at the task node that is the root of the sub-hierarchy representing the composite task. To access the PAGE definition from a basic task node, the chain of ancestor nodes is traversed until the node containing the actual annotation is found.

This second strategy allows to take advantage of the extended task context (WHEN) provided by the task model. The PAGE annotation in the root node of the composite task defines the default representation that is inherited by all of the composite task's working steps. Sub-tasks then specialize the task context, and thus, that task's visual representation, at a lower decomposition level only as needed according to task-specific requirements (i.e., WHY, cf. next subsection 3.3.3). In addition, annotating a PAGE only once, in the lowest common ancestor of all basic task nodes that reference it, avoids redundancies in the extended task model. Figure 3.2 illustrates how the extended task context with PAGE annotations is utilized to provide default representations for several working steps.

The approach discussed here uses the second strategy. It therefore presents an improvement compared to contemporary approaches based on the specification of individual low-level tasks. The latter necessitate that the content to be adapted is specified for each task repeatedly.

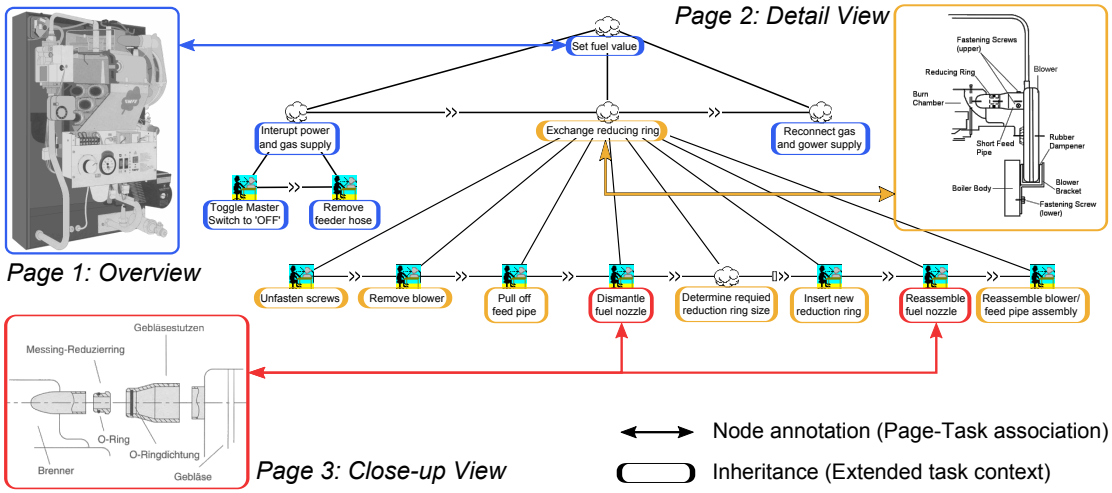


Figure 3.2: Illustration of our annotation strategy that utilizes the extended task context. PAGES shared between multiple working steps are annotated at the least common ancestor node. Child nodes inherit the PAGE's default representation. Only for working steps where the default representation is insufficient, additional task context information is provided for adaptation in the form of FEATURES.

3.3.3 WHY-oriented aspect – FEATURES

PAGES provide a description of the graphical content, constituting the WHAT aspect of the task context, as well as a default visual representation of that content. However, to be able to effect task-specific visual representations, the task context must be further detailed to include information regarding two categorical issues:

- *Which* aspects or elements of the PAGE contents are actually of general relevance in the context of the task at hand?
- *How* relevant are these elements with respect to each other as well as the task's communicative goal?

To this end, we employ the concept of FEATURES as a means to specify this information in the extended task model.

The main idea behind FEATURES is to organize the graphical content of a PAGE into conceptually distinguishable content elements, i.e., separate objects whose appearance in the content's visual representation can be adapted individually, according to task requirements. To be useful for the adaptation of a PAGE's visual representation, this information must be provided at a finer granularity than PAGES. The term FEATURE is borrowed from the image processing domain where it is typically used to describe abstractions of image information relevant to a certain application.

Moreover, FEATURES allow to express relationships between content elements on a semantic level regardless of the way the visual data type is organized internally. What relations between content elements are meaningful depends on the application domain. These are typically specified using a domain ontology description like OWL, see for

example [MMS04, Kli09]. Examples from the maintenance scenario include hierarchies of assembly parts and connectivity (topology) information of electrical circuit schematics.

FEATURE relationships can be utilized for the generation of initial views, and especially for providing FEATURE-based interaction with the adapted visual representation, as discussed in Section 3.3.5. In this thesis in particular, we discuss how *is-attached-to* relations are utilized to create so-called "exploded views" of assembly drawings (Section 5.3.1), as well as how to utilize topology information (*is-connected-to* relations) for the adaptive layout of circuit schematics (Section 5.3.2).

The purpose of FEATURE definitions thus is to introduce a notion of distinguishable, structured content elements of a PAGE on a semantic level. This allows to utilize per-element adaptation of PAGES even if the underlying visual data type itself

- does not provide any information on content structure at all (e.g., raster images), or,
- although it does have a notion of content elements (e.g., vector graphics), elements are structured at a granularity unsuited for adaptation.

FEATURES as defined here therefore are content elements with a *semantic meaning* in the context of the (domain) task at hand, rather than artifacts of the technical organization of the graphical content.

In doing so, FEATURES abstract from the way content elements are actually described and organized in the data format of underlying visual data type. Such an abstraction can be provided for any type of graphical content. In particular, defining FEATURES for unstructured content can be effected by means of external meta-data.

For example, raster images are but a 'pixel soup' with no notion of content structure. Here, defining FEATURES entails defining regions in image space. To this end, regions can be specified e.g., by a boundary representation. Vector graphics are comprised of graphical primitives as their basic content elements. However for the purpose of adaptation, usually several primitives are grouped together into a single FEATURE that describes some more complex object with a semantic meaning in the application domain.

Because the type-specific details of what comprises FEATURE are irrelevant for the discussion of our basic adaptation approach, these will be discussed in more depth in Chapters 4–7, respectively, for each of the four types of graphical content.

Irregardless of the type-specific realization, from the perspective of the adaptation process each FEATURE is treated as an object with defined properties:

- Geometry, which determines the region the content element occupies in the visual representation, i.e., in image space. FEATURE geometry is either defined directly in image space (e.g., regions of interest in a raster image) or derived from higher-order geometry (e.g., the projection to view space of a triangle mesh with size and position given in $\mathbb{R}^3$).
- Visual attributes, which determine the appearance of the element in the visual representation. Examples include line and fill colors (2D graphics), lighting and material properties (3D renderings), and color scales (abstract data visualization).

- Supplemental information, such as human-readable names used as labels (cf. Section 3.3.4) or textual information for text-to-speech output mode, as well as relationships to other FEATURES.

These properties can be adapted independently from each other to obtain a visual representation that adequately reflects the FEATURE's relevance with respect to the task at hand. This will be discussed in Section 3.3.4.

FEATURES are associated with tasks to detail what elements or aspects of the content are of relevance in the context of the task at hand. FEATURES are associated with tasks by annotating the corresponding task model node with the FEATURE definition.

In doing so, we employ the same annotation strategy for FEATURES as for PAGES, for the same reasons. A FEATURES may be of relevance in the context of multiple working steps. In this case, the common ancestor node is annotated.

This allows to utilize the task hierarchy of the extended task context (WHEN). Conceptually, content FEATURES common to multiple tasks are defined on the broader scope of their superordinate composite task. FEATURES specific to a sub-task of that superordinate task (i.e., a sub-hierarchy) are defined on a correspondingly narrower scope of that sub-hierarchy's root. FEATURES specific to an individual working step, finally, are associated with the corresponding basic task[2]. Because FEATURES are defined on the basis of the content described by a PAGE, no FEATURE is reasonably annotated at a task node that is an ancestor of the task node associated with the corresponding PAGE.

With the annotation of FEATURES, the task model now contains information on what content elements are of general relevance in the context of the task at hand. As stated at the outset of this section, this constitutes the first half of the task context's WHY aspect.

Section 3.1 argued how task-driven adaptation relates to the selection of appropriate information and visual scales of these content elements as appropriate for the task's communicative goal. Therefore, it is necessary to further augment the description of the task context's WHY aspect with a means to derive appropriate scales for the associated content's visual representation.

In the same way that PAGES and FEATURES provide an abstracted description of the graphical content and its semantically relevant elements, the data type-dependent realization of different visual and information scales is not relevant at the conceptual level of the task model. Therefore, *relevance values* are used in our approach as an abstracted measure for the FEATURE's scale requirements for the task at hand. Only during the adaptation process are the relevance values evaluated to control the visual representation of FEATURES in a data type-specific way, see Section 3.3.4.

To this end, FEATURES ϕ_i are associated with a numerical value $r_i \in R$ from a specific range of possible relevance values $R = [r_{min}, r_{max}]$. Using such a mapping of FEATURES to relevance values allows to capture the following information in the task context:

[2] In an actual implementation, it will often be more efficient to coalesce all FEATURES into a single set of definitions that is an integral part of the PAGE annotation. This can be easily automated, however, so that in designing the smart visual interface one would still follow the concept of hierarchical specification on different task scopes.

- The relative importance of FEATURES $\phi_1, \phi_2, \ldots, \phi_n$ with respect to each other and to the task's communicative goal, according to the relative magnitude of their relevances $r_{min} \leq r_1 \leq r_2 \leq \ldots \leq r_n \leq r_{max}$. This can be utilized during adaptation to select an appropriate level of detail and visual style in a way so that more relevant FEATURES are visually emphasized over less relevant ones. In particular, it allows to map from the range of relevance values to the (usually bounded) parameter domain of concrete display techniques, as discussed in Section 3.3.4.
- A FEATURE ϕ_i with relevance $r_i = r_{min}$ is defined to not contribute any information to the task at hand's communicative goal. In particular, this definition provides an expedient default value for FEATURE relevance: unless a relevance value $r_i > r_{min}$ is explicitly specified in the task context, a FEATURE is implied to be irrelevant for the task at hand, and thus should retain its default visual representation defined by the PAGE. Therefore, task contexts need to specify FEATURE relevance values as part of their respective WHY aspect only if that task's communicative goal requires to locally deviate from the PAGE's default visual representation.

In this thesis in particular, the specification of FEATURE relevance is based on the normalized continuous value range $r_{min} = 0, r_{max} = 1$.

In doing so, there are several approaches to arrive at concrete relevance values for a particular task at hand. Depending on the visual data type and application context, algorithmic methods are often viable to automatically derive FEATURES and their corresponding relevance values, i.e., by automatic content analysis. For example, in 3D graphics relevance values can be based on the distance to the viewer; in the case of raster images relevance can be based for example on a pixel energy or entropy measure calculated by an image analysis algortihm (cf. Section 2.3.2).

Lacking suitable automatic methods, it is always possible to rely on the manual specification of FEATURES and their relevance values by a content author. This necessitates tool support, as discussed in Chapters 4–7. From the perspective of the enriching the task model, however, it is irrelevant if relevance values are derived automatically or if they are defined manually.

Based on the definition of relevance values, the task model is enriched further. In the same way as PAGES and FEATURES are associated with tasks by annotating the corresponding task nodes in the model, FEATURES are assigned relevance values by annotating task nodes with the corresponding value mappings $\phi_i \rightarrow R$. Similar to PAGE and FEATURE annotations, our approach allows to effect this assignment not only at the granularity of individual basic tasks (i.e., , annotating leaf task nodes), but also at different levels of composite tasks.

In doing so, the extended task context provided by the task hierarchy is utilized. A FEATURE that is equally relevant in the context of multiple sub-tasks is assigned a relevance value at the scope of the superordinate composite tasks by annotating the common ancestor node. This relevance value is inherited down the task hierarchy to the level of basic tasks, unless a task node at a lower decomposition level explicitly re-assigns a relevance value. This allows to locally increase or decrease the respective FEATURE's relevance if it deviates from the inherited value.

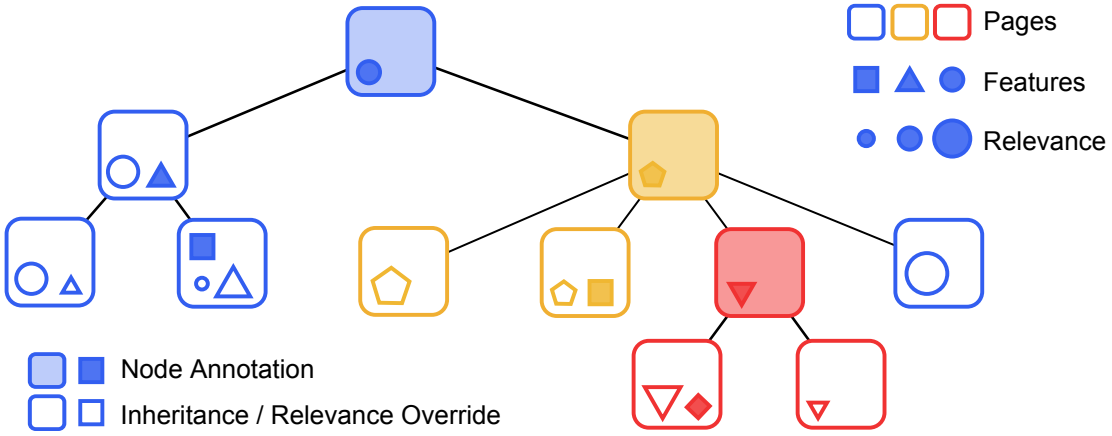


Figure 3.3: Schematic illustration of the distribution of PAGE, FEATURE and FEATURE relevance annotations in the hierarchical task model in our approach. The extended task context given by the task model is thereby utilized to provide more detailed information only where required by the specific task requirements.

Thus the combined annotations of FEATURES and FEATURE relevance values allows a hierarchical specification of the task context's WHY aspect: the information what content elements are generally of relevance, and on the relative importance of these elements regarding the task's communicative goal are both specified on a broad scope of composite tasks. Only for tasks with diverging requirements to the visual representation the task contexts is further detailed locally. Figure 3.3 summarizes the inheritance strategy that facilitates this hierarchical task context specification.

To recap briefly, the starting point for our approach is the general task model resulting from a domain task analysis. Its hierarchical decomposition of the task at hand describes the WHEN aspect of extended task context for visual representations. The PAGE/FEATURE concept introduced here provides the basic building blocks for the necessary extension of this general task model through task node annotations. PAGES describe self-contained units of graphical content of arbitrary data type, thus constituting the WHAT aspect of the task context. FEATURES define, on the level of domain semantics, distinguishable content elements of PAGES. In addition, FEATURES are associated on a per-task basis with relevance values as an abstraction of the information scale required for the task at hand. The combination of FEATURES and FEATURE relevances constitutes the WHY aspect of the task context. The task model is thereby annotated in a way to utilized the task hierarchy so that the specification of the task context becomes more detailed with reduced task scope, i.e., from the root task down towards basic tasks. Figure 3.3 schematically summarizes this concept.

The following section will explicate how the concrete task-driven adaptation of visual representations is effected on the basis of the conceptual task model enriched with PAGE/FEATURE annotations.

3.3.4 How– Generation of Initial Views

The first step in generating the task-specific adapted visual representation for the task at hand is to determine what comprises the basis for the adaptation process. Two principal modes of adaptation can be distinguished to this end:

- exhaustive adaptation with respect to the task at hand, as well as
- incremental adaptation across multiple basic tasks.

The difference between these two modes is in the way relations between sub-tasks of the task at hand are accounted for.

In what we termed *exhaustive adaptation*, causal or temporal dependencies between task nodes are not considered. Information from the extended task context are used to determine the PAGE that provides the graphical content associated with the current working step. The PAGE's default visual representation comprises the starting point for the adaptation process, regardless of the visual representation displayed for the previous working step. Thereby relations to previous working steps are regarded only implicitly, by virtue of shared FEATURES and FEATURE relevance values defined in the task context of common ancestor task nodes.

On the other hand, utilizing the explicit information on dependencies between tasks from the task model allows to perform *incremental adaptation*. Often, a sequence of multiple working steps refers to the same depicted object(s). Two examples are step-by-step assembly instructions, which are common in maintenance task [APH+03], or "tour-through-the-data" presentations that support incremental buildup of knowledge about visualized data by showing a series of progressive previews [RS09].

In these cases, instead of deriving the adapted visual representation for the current working step from the associated PAGE's default representation, the adaptation process incrementally adds detail to the representation of each previous working step. Formally, this condition is specified in the extended task model by the 'enabling with information exchange' operator (A []> B in CTT notation) defined on two consecutive basic tasks A, B. Completing task A will automatically start task B, whereby the information passed from A to B comprises A's visual representation in its current state, including interactive manipulation of the initial view by the user.

Thus for exhaustive adaptation FEATURES and relevance values *comprehensively* define the levels of detail of the *default* visual representation specified by the PAGE; whereas during incremental adaptation, FEATURES defined in the current working step's task context define *additional* detail to the *current* visual representation.

In subsequent steps of the adaptation process, the base visual representation determined in the first step is further modified according to the FEATURES and FEATURE relevance values that are defined in the current task context. In doing so, each feature is considered as a distinct object whose properties are manipulated according to its relevance value.

Adaptation pipeline

To this end, we here introduce the concept of an *adaptation pipeline*. This pipeline is composed of five subsequent stages. On each stage, operators are applied to the result of the previous stage. The adaptation pipeline therefore defines a conceptual framework for the integration of different *adaptation operations*. The different pipeline stages are, in order of traversal:

View selection determines what portion of the PAGE is visible. The result of view selection is a two-dimensional region on screen that comprises the visual representation of the selected graphical content. The actual operation to carry out this selection depends on the visual data type. For example, view selection on 2D raster and vector graphics is limited to the spatial domain. For 3D graphics, it constitutes the set-up of the virtual camera's view frustum. In particular, view selection may be an explicit operation (e.g., according to predetermined region boundaries), or implicit depending on defined FEATURES and relevances (e.g., such that the most relevant FEATURES are initially visible).

Geometry adaptation follows the general idea of the Focus & Context approach [Kea98] to reserve more screen space for more important FEATURES at the expense of less important ones. This comprises two operations: relevance-dependent scaling of FEATURE geometry, and selecting an appropriate geometric level of detail to account for the FEATURE's presentation size. At the extreme end is information hiding, where irrelevant features are removed entirely from the adapted output.

Visual attribute adaptation manipulates the visual appearance of content elements such as colors, fill patterns/textures and stroke styles. In doing so, this pipeline stage offers support for two principal operations to support the task's communicative goal: visual *accentuation* of important information (i.e., FEATURES), and an appropriate *de-accentuation* (subduing) of less relevant information. This creates a *visual hierarchy* [Den99] of FEATURES according to their relevance to the task at hand. As a consequence, the viewers' attention is automatically guided to the most important parts of the adapted visual representation.

Image space manipulation involves filter operations that modify color values the pixel matrix of the task's visual representation, such as color saturation or contrast adjustments. This pipeline stage thus comprises post-processing operators on the output of previous stages, and as such includes the manipulation of contextual information provided by the visual representation's non-FEATURE periphery. In doing so, FEATURE accentuation (on the previous pipeline stage) and context de-accentuation are complementary operations, as reducing the visual saliency of the image background causes the relevant foreground objects stand out at the same time.

Labeling further augments a task's visual representation by inserting additional information for important FEATURES, such as textual labels, icon overlays, or additional

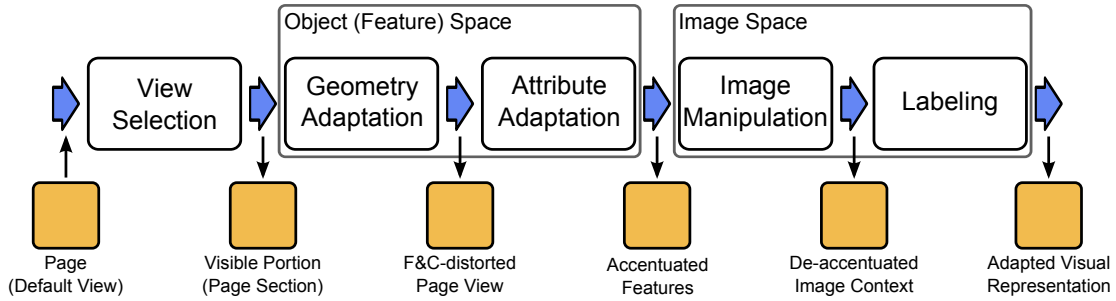


Figure 3.4: Schematic view of the adaptation pipeline.

visual elements like guides. Why this is a distinct and important adaptation step is discussed below.

The first three pipeline stages are dependent on the visual data type of the input PAGE. This means the realization of the respective adaptation operations depends on concrete display techniques developed for the particular visual data type.

By contrast the fourth stage – image space manipulation – operates on the output image of the previous stage, and is therefore independent from the actual display technique that generated that image. Thus, this stage is independent from the underlying visual data type.

From the perspective of the adaptation pipeline's fifth stage, labeling approaches can be distinguished in object-space approaches (i.e., those that determine label positions based on object geometry) and image-based approaches that operate on the pixel matrix of the image being labeled. For our approach we have deliberately chosen to integrate an image-based approach.[3] This choice provides a visual data type-independent solution for the labeling stage of the adaptation pipeline as well, since it allows to operate on the result from the image manipulation stage.

The adaptation pipeline thereby provides two levels of adaptation: on the level of content object described by FEATURES (geometry and visual attribute adaptation), as well as on the image level (image space manipulation and labeling). Figure 3.4 summarized the conceptual set-up of our proposed adaptation pipeline.

For the concrete realization of the different pipeline operations a large number of display techniques has already been proposed for the different visual data types, see Section 2.3.2. However, it was also shown that the majority of these approaches supports only a limited set of adaptation operations each. Distinguishing the adaptation process into five subsequent principal pipeline stages – that are equally applicable to *all* of the four visual data types – makes it possible to selectively apply a combination of concrete techniques as appropriate for the underlying data type, adaptation operation, and task requirements at the granularity of individual content elements defined as FEATURES. This greatly increases flexibility of the PAGE-/FEATURE-based adaptation.

[3] Note that any graphical content can be labeled in image space, not just raster images – labeling is then performed on the framebuffer that holds the rasterized result of the rendering process.

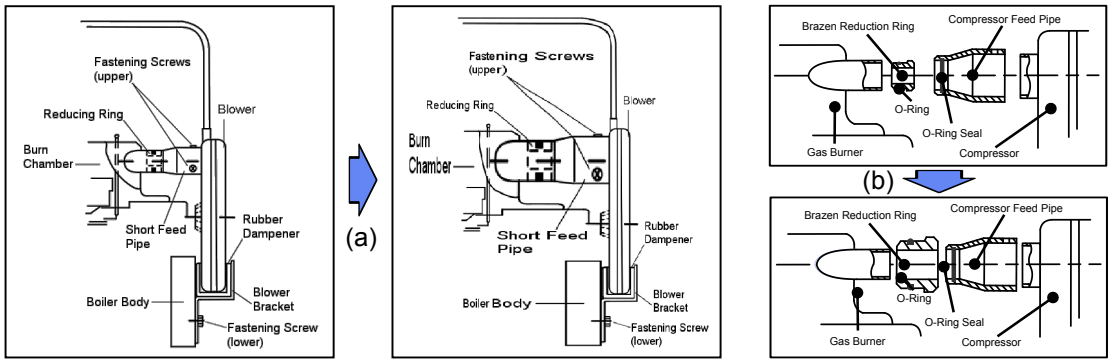


Figure 3.5: Static labels are negatively affected by distortions introduced during geometry adaptation that enlarges important FEATURES. Pre-labeled raster images suffer from scaling artifacts (a), whereas SVG text elements are no longer positioned correctly (b).

Labeling as a distinct adaptation step. A smart visual interface primarily relies on visual representations. However, domain-specific concepts or technical terms may require explicit textual annotations in an image. Therefore, labels are an important element of effective illustrations. However, these labels must be placed dynamically during the adaptation process of the visual representation. Static labels, i.e., text as integral, fixed image content, would be subjected to distortions introduced by the previous geometry adaptation step, which is unsatisfactory both from an aesthetic point of view and for legibility issues (cf. Figure 3.5).

Furthermore, labels serve two additional purposes in multi-modal interfaces in particular:

Coordination of the visual and audio modes. Textual annotations link elements of the visual representation with information conveyed via speech, e.g., by providing corresponding keywords or captions. If speech input is available, the presentation of keyword labels further establishes an intuitive access/control mechanism for voice commands [BDF+06].

Redundant information on the visual and auditive channels. Labels convey additional information in situations when the ambient noise level e.g., at the maintenance site, overpowers the speakers. Moreover, labels may remain on the interface permanently without introducing a comparative "auditive clutter" by constant speech repetitions.

Thus, labels are an integral part of effective, adequate visual representations in multi-modal interfaces.

With regards to dynamic labeling, a large number of approaches has been proposed in literature. For a good recent overview, see [LSC08] and [Ali09]. Common to these approaches are two questions that need to be answered:

- *what* objects should be labeled (assuming that not all labels can be placed due to space constraints), and

- *where* to place them (i.e., as close as possible to the labeled object without introducing overlaps). In case of image-based labeling approaches, this step requires to first segment the image into foreground objects which are being labeled and empty background regions that are available for placing labels [Ali09].

Our PAGE/FEATURE-based adaptation approach readily provides the information that is needed to address both these aspects of the labeling problem:

- On the one hand, the FEATURE relevance values define what labels should be placed in order of priority. Relevance is also applicable to influence stylistic label attributes like font size and weight to visually accentuate labels of important FEATURES. Task node annotations are further amended with a label text for each FEATURE (e.g., its name) to provide all necessary data for the labeling process.
- On the other hand, FEATURES define a region in image space (cf. Section 3.3.3). FEATURES therefore allow to dynamically generate a description of the size and position of objects to be labeled. Moreover, because our general strategy for the geometric adaptation is to reserve more image space for important FEATURES, generally more space is available for label placement in their immediate vicinity.

This makes our PAGE/FEATURE-based adaptation approach compatible with a wide range of existing approaches. Selection of a specific labeling technique thus is primarily a matter of application requirements, i.e., with respect to an appropriate trade-off between labeling quality and computational complexity. This choice will also be driven based on which required features are supported by the various labeling algorithms, e.g., internal vs. external (peripheral) labels [AHS05] or specific aesthetic aspects such as label layout styles [GHS06].

However, the choice of a particular labeling approach is *not* independent of the the target device, i.e., the WHERE aspect. Labeling is a notoriously difficult problem as even the simplest form – finding an optimal layout for the point-feature labeling problem – has been proven to be NP-hard [MS91]. Approaches like Simulated Annealing [KGJV83] or particle-based labeling [LSC08] therefore employ different strategies and heuristics to achieve acceptable solutions in interactive times. Still, dynamic labeling at interactive rates may still be too computationally demanding for small mobile devices despite heuristic solutions. To address this issue, we have developed a client-/server-based remote labeling approach utilizing the content information provided by FEATURES.

For the sake of brevity in discussing our general adaptation approach, details regarding the integration of a general-purpose image-based labeling approach [LSC08] as well as our own solution to FEATURE-based remote labeling are deferred until Section 4.2.5 and Section 4.3.2, respectively.

Adaptation control

In the course of the general model-based UI design process the initial task model is at some point mapped onto a presentation model (e.g., a dialog model, cf. Section 2.3.1). The latter describes the abstract user interface for the modeled domain workflow, i.e.,

the mapping of basic tasks from the task model onto different GUI elements, without however stipulating a specific implementation platform.

Our approach proceeds in an analogous manner: the task contexts (defining the WHAT and the WHY aspects) of basic tasks from the enriched task model are associated with suitable display techniques to arrive at the specification of a smart visual interface that supports the task at hand by providing good initial views for each working step (basic tasks). In doing so, the adaptation pipeline provides the framework for this association.

Thus in order to obtain these initial adapted representations, operations in the conceptual adaptation pipeline must be mapped onto concrete, data-type dependent techniques that comprise the interface components of the smart visual interface. We collectively refer to these display techniques as "Smart-X" techniques because they constitute the building blocks of smart visual interfaces.

For this purpose, a mapping must be determined per FEATURE that assigns the relevance values to concrete parameter values. The thereby generated parametrization of the corresponding "Smart-X" techniques then controls the resulting appearance of the visual representation. In doing so, FEATURE relevance values can be mapped either to a continuous range of parameter values, or to a set of discrete parameter settings.

To give an example for the first mapping type, the belt-based distortion technique discussed in Section 4.3.1 takes as an input parameter a magnification factor s for each FEATURE that controls the amount of image distortion in and around that FEATURE region. A mapping $M : r \in [0.0, 1.0] \rightarrow s \in [0.5, 2.0]$ would thus result in regions of irrelevant FEATURES ($r = 0$) being scaled to half of the original size, whereas the most important FEATURES being scaled up to twice their original size.

An example of the second mapping type is the assignment of distinct rendering styles to parts of a 3D model according to

$$M : r \in \begin{cases} [0.0, 0.5[& \rightarrow & \text{'always cull'} \\ [0.5, 0.8[& \rightarrow & \text{'translucent'} \\ [0.8, 1.0] & \rightarrow & \text{'textured'} \end{cases}$$

This results in an initial representation of the 3D model where the least relevant FEATURES ($r < 0.5$) are removed entirely and important ones ($r \geq 0.8$) are shown in full detail. All FEATURES in between are rendered half-transparent (so as to not occlude or distract from the important ones, cf. Section 6.2). Note that in this example, the domain of parameter values comprises labels instead of numerical values. These are used to select specific functionality/parameter presents of the "Smart-X" display technique for the rendering process.

Clearly, the "Smart-X" technique used for adaptation must, on principle, be suited for the visual data type in question. For example, applying different 3D rendering styles to regions of a raster image simply is not viable. In consequence, as illustrated by the two examples above, the set of parameters and their respective domains depend on the specific display technique.

The following type-specific Chapters 4–5 will discuss several novel "Smart-X"-techniques developed specifically for task-driven adaptation in smart visual interfaces. This discussion includes details on how these techniques are integrated into the general adaptation

pipeline and how FEATURE relevance values are mapped to their input parameters. Moreover, Chapter 8 will review the implementation of an e-manual prototype, whose smart visual interface incorporates several existing techniques for adaptation.

To recap, our approach for the generation of adapted visual representations from an enriched task model is based on two concepts: partition of the adaptation process into an general adaptation pipeline that is independent of the visual data type, and integration of different type-dependent "Smart-X" display techniques for the concrete generation of adapted visual representations on the basis of this pipeline. Therefore, the adaptation pipeline provides the link between the enriched task model that captures a visual representation's context of use on a conceptual level on the one hand, and the concrete smart visual interface providing the corresponding adapted initial view to the user.

3.3.5 Manipulation of Initial Views

As argued in Section 3.1, the primary objective of the adaptation process is to provide a good initial view that supports the user in her task at hand. Ideally, the automatically generated visual representation is immediately useful to the user by communicating all required information without necessitating further user interaction.

However, attaining this ideal through an automatic adaptation process can not be guaranteed in every case. Nor is it *desirable* to strictly stipulate one single visual representation as the 'ideal' view, lest the smart visual interface patronizes the user. Actual requirements to a task-specific visual representation differ depending on e.g., the user's momentary situation, as well as her specific preferences and levels of expertise regarding the task at hand. It is thus necessary that the smart visual interface allows the user to interactively manipulate the automatically generated initial views to fine-tune the visual representation to her requirements.

In particular, the user will interact with visual representations by means of the "Smart-X" display technique that comprise the interface components of the smart visual interface. Concrete realizations of "Smart-X" techniques offer different ways to manipulate views and to adjust parameters, depending primarily on what manipulations or operations are viable on the visual data type. Different realizations will also vary in their general design and the interaction facilities provided by the target platform.

These variances, however, all pertain to details how a particular display technique is used as a tool in accomplishing the task at hand. These details are not relevant to the specification of domain tasks. Rather, these comprise functional properties (of a concrete "Smart-X" technique) below the conceptual level of basic domain tasks.

As such, in the approach presented here the extended task model does *not* specify as distinct basic tasks what manipulations of the initial visual representation are available, or how these would be carried out using concrete "Smart-X" display techniques.

Rather, we here incorporate the concept of task actions similar to several other task modeling approaches (cf. Section 2.1.6): an action is an atomic operation that is executed

upon an artifact by an entity that is involved in the completion of the task (e.g., the user or computer), but represents activities on a level below basic (domain) tasks. Translated to our approach, the associated PAGE marks the corresponding task's artifact or task object, whereas interactive manipulations constitute task actions defined on this task object's visual representation.

For the same reason it is not feasible to model possible interactions of a particular "Smart-X" display technique as explicit tasks in the task model, it is not expedient nor necessary to comprehensively capture this set of technique- and data type-dependent interactions as task actions.

On the other hand, our approach provides a common denominator for actions executed on any visual representation regardless of the visual data type of the associated PAGE: manipulation of FEATURES. To this end, one fundamental task action that can be reasonably executed on every visual representation is the interactive manipulation of FEATURE relevance values. By interactively selecting a particular FEATURE, the user can express her particular interest in the respective content element.

Designating a FEATURE of interest (FoI) ϕ_i causes a change of its relevance value to the maximum relevance value $r_i = r_{max}$, thereby reflecting the user's request to locally deviate from the information and visual scales predetermined by the task context. Moreover, the increased relevance for a FEATURE of interest ϕ_i may be recursively propagated to further FEATURES ϕ_j of the PAGE according to a suitable Degree-of-Interest (DoI) function. This DoI function thereby either

- operates in view space, i.e., the degree of interest for a FEATURE ϕ_j is a function of the distance in view space between the representations of FEATURES ϕ_i and ϕ_j, or
- utilizes the FEATURE relations, i.e., the degree of interest is propagated according to semantic criteria, cf. Section 3.3.3.

Inversely, deselecting a FEATURE will revert its relevance to the predetermined value specified in the task context. The adaptation pipeline is then traversed again with the modified relevance values. This results in a visual representation showing more detail for the FEATURE of interest, and possibly other FEATURES depending on the DoI function.

Because this change in representation is effected based on the abstract relevance value rather than specific parameters of a particular display technique, all adaptation operations defined in the corresponding adaptation pipeline will adequately reflect the user's focus of interest. In particular, the re-adaptation may entail a change in view selection (first pipeline stage) as well as the display of additional detail information for the FEATURE of interest during labeling (fifth stage). Figure 3.6 shows an example for the visual representation of a 3D model. Note how in this particular example the DoI function is defined to assign all FEATURES that are *not* selected as FoI the lowest relevance value, regardless of the relevance value specified in the task model.

The strategy described here thus provides a general means to implement Shneiderman's well-known information seeking mantra "Overview first, zoom and filter, details on demand" [Shn96] – regardless of both the PAGE's visual data type and the "Smart-X"

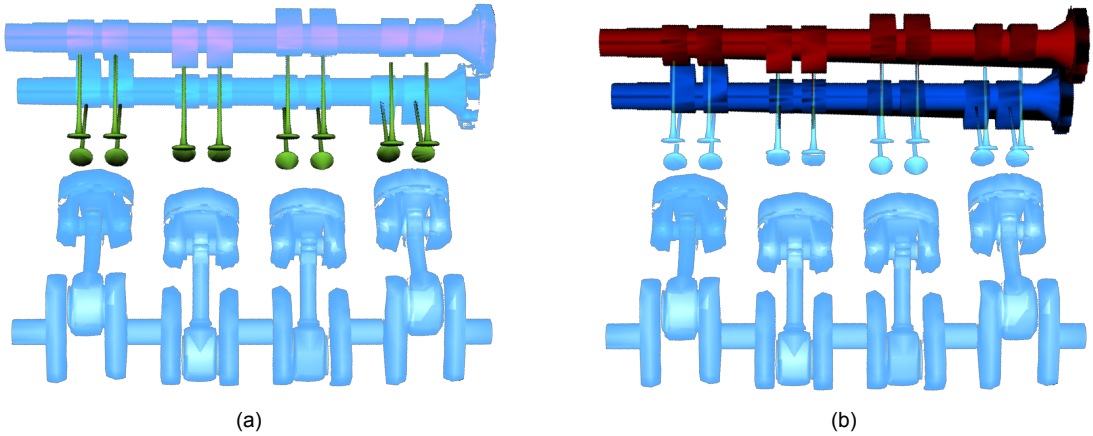


Figure 3.6: Example for the effect of interactive FEATURE relevance manipulation. Different rendering styles are used for FEATURE accentuation. (a) The valves are defined as the most important component for the current task, other components are visually subdued. (b) shows the result of manually selecting another FEATURE (the camshafts) as the focus, temporarily overriding the importance values from the task model for **all** FEATURES.

techniques employed for its representation. The initial view thereby provides an overview on the information relevant to the task at hand. Then, the user can interactively select FEATURE(S) of interest (zoom and filter) that result in a re-adapted visual representation that adequately reflects the manual updates to information relevance (details on demand).

Furthermore, individual "Smart-X" realizations may supply additional possibilities for interaction with the PAGE's visual representation, such as fine-grained view control and manipulation of parameter settings specific to the display technique, to further support the general FEATURE/relevance-driven interaction.

3.4 Summary

In this chapter we presented our basic approach to the *task-driven adaptation* of scalable visual representations within smart visual interfaces. The primary objective is to present good initial views with regard to the task at hand, thereby providing "smart" user support by reducing the need for manual selection and navigation of content.

Section 3.1 detailed the scope of the task context that must be considered to this end. Any task-driven approach will require to address aspects pertaining to the underlying graphical content (WHAT) and the information requirements with respect to the task's communicative goal (WHY).

However we argue that in order to be effective, task-specific representations additionally require to contemplate the *extended task context* (WHEN) of the task's superordinate workflow. In particular, utilizing a *hierarchical* workflow description allows to *propagate* information from higher-level tasks down to basic tasks, which are therefore also utilizable for subsequent steps. This requires the integration of a task model – as a hierarchical

description of the domain-level workflow – with means to describe scalable visual representations with respect to the WHAT and WHY aspects.

A review of related work reveals that there currently is a "modeling gap" regarding this important requirement to the design of smart visual interface. On the one hand, model-based UI design approaches do not adequately consider the integration of scalable graphical content that is the foundation of task-specific visual representations. On the other hand, existing display techniques that provide scalable and interactive representations for the different visual data type insufficiently contemplate task-related influencing factors of the visual representations' context of use.

The approach presented here addresses this modeling gap. It comprises two concepts. The main contribution is the **PAGE/FEATURE concept** that represents a *general, data type-independent description of the graphical content* as well as its task-related aspects. It allows to enrich the general task model with a description of the task context with respect to scalable visual representations. Furthermore, the proposed concept of an **adaptation pipeline** provides the bridge between the conceptual task context specification in the model and concrete display techniques used to generate the adapted representations accordingly.

PAGES represent *self-contained units of graphical content*, thereby abstracting from the technical peculiarities of the underlying visual data type. Each PAGE defines a *default representation* of the graphical content it represents.

FEATURES describe *content elements* of a PAGE *on a semantic level*, i.e., represents a piece of information that is of relevance to a task's communicative goal, while *abstracting from the technical structure/organization* of the different visual data type. FEATURES thus facilitate better scalability of graphical content as the information and visual scale of adapted visual representations can be selected at the granularity of FEATURES, rather than entire PAGES.

Further, we introduce the notion of **FEATURE relevance**. Relevance is an *abstract measure to express the information and visual scale* that is appropriate for the given FEATURE in the context of the task at hand. FEATURE relevance values therefore constitute the basis on which the adaptation of the graphical content is effected.

Together, PAGES, FEATURES and FEATURE relevance values are used as the **building blocks for enriching the domain task model** with a description of the task context for visual representations. To this end, specific task nodes in the model are *annotated* with corresponding definitions of PAGES, FEATURES and relevance values.

In doing so, our approach is to specify aspects of the task context hierarchically on different task scopes, i.e., on different levels of the task decomposition. Context information relevant to multiple basic tasks is specified at the scope of their superordinate composite task, by annotating the common ancestor task node with the corresponding PAGE, FEATURE, or FEATURE relevance value. The task context becomes more detailed as additional information is specified on the subsequent levels of sub-task, until the task context is fully specified on the level of individual basic tasks (leaf nodes). This strategy of a hierarchical definition of a visual representations task context thereby facilitates to specify additional context detail *only* for those sub-tasks that exhibit diverging information requirements with respect to their communicative goal. This sets our approach

apart from contemporary solutions that are based solely on low-level tasks. The latter require the full task context be specified for each working step (basic task) individually and thus provide no means to exploit causal or temporal relations between sub-tasks of a composite task.

The task model enriched with PAGES, FEATURES and FEATURE relevances describes the task contexts for each basic task of a composite domain task on a conceptual level. At runtime, the smart visual interface uses this context information to generate the initial visual representations for the current working step. For this purpose, we propose a **general adaptation pipeline** that facilitates the integration of several concrete display techniques.

The adaptation pipeline comprises five stages. The first stage selects a view of the PAGE, i.e., performs selection operations on its content. The result of view selection in a two-dimensional region that comprises the visual representation. The second and third stages operate on individual FEATURES. In doing so, FEATURES are treated as objects with geometric properties and visual attributes. Adaptation operations on geometry thus manipulate the size and position of a FEATURE, operations on attributes its appearance. The forth stage operates on the entire visual representation in image space, whereas the fifth stage adds additional elements (labels) using an image-based labeling approach. The adaptation process can thus be conceptually distinguished into a visual data type-specific part (stages 1–3) as well as a type-independent part (stages 4–5).

Adaptation operators on the pipeline stages are realized by suitable display techniques. To this end, FEATURE relevance values are mapped onto the parameter domains of the respective technique. This mapping is thereby done as part of the design process of the smart visual interface. Therefore, the adaptation pipeline serves as a **framework for the integration of existing display techniques** for the different visual data types. Because these display techniques thus define the functional building blocks of the smart visual interface, we refer to them as "Smart-X"-techniques.

The PAGE-/FEATURE abstraction concept and the adaptation pipeline represent a general approach to the generation of task-specific adapted visual representations for all visual data types. However, the principles and means by which the PAGE and FEATURE abstraction are derived from graphical content are, on principle, specific to the respective visual data type. Similarly, the adaptation parameters on which the FEATURE relevance values are mapped varies with both the data type and the concrete display technique employed.

Therefore, we will discuss the three conceptual aspects PAGE definition, FEATURE specification, and adaptation control (i.e., , integration of concrete "Smart-X" techniques) for each of the four visual data types in the following chapters. In addition, each chapter will review authoring tools for the preparation of graphical content to provide manual means to define PAGES and FEATURES for the respective visual data type.

And even though the PAGE/FEATURE annotation concept proposed here allows to incorporate a wide range of existing techniques into the design process of smart visual interfaces without significant modification, some task-specific aspects of representation scalability were found to be still open research questions. Thus as a further contribution

of this thesis, a number of novel "Smart-X" techniques covering several different aspects of task-driven adaptation for the four visual data types will be described in detail in the corresponding chapters as well.

To this end, Chapter 4 discusses raster graphics first, which is the most limited of the four visual data type. Chapter 5 details solutions for vector graphics, specifically the Scalable Vector Graphics (SVG) format [FFJ03]. 3D graphics are dealt with in Chapter 6. Finally, Chapter 7 discusses representations of abstract data, which is the most general but also most flexible visual data type regarding adaptation.

4 Smart Raster Images – Adaptation of Raster Graphics

Raster images are common graphical content in visual interfaces. In this thesis in particular, adaptation of raster images plays an important role in the maintenance scenario. Often, graphical content associated with maintenance tasks such as two-dimensional technical illustrations and schematics are available only as plain raster images. This is because typical content sources are the extraction of raster images from PDF files, or by digitalization (scanning) of printed manuals.

However, raster images are also the most constrained visual data type. The primary challenge from the viewpoint of task-based adaptation is the absence of any structure or semantic information about the depicted content: the image is but a 'pixel soup'. Only image space operations that modify the pixel matrix are possible.

4.1 Raster Image Content Preparation: Concepts and Authoring Tools

Thus plain raster images must be enriched with supplemental information to facilitate scalable, task-specif visual representations. To this end, raster image content preparation yields the definition of raster PAGES with associated FEATURES.

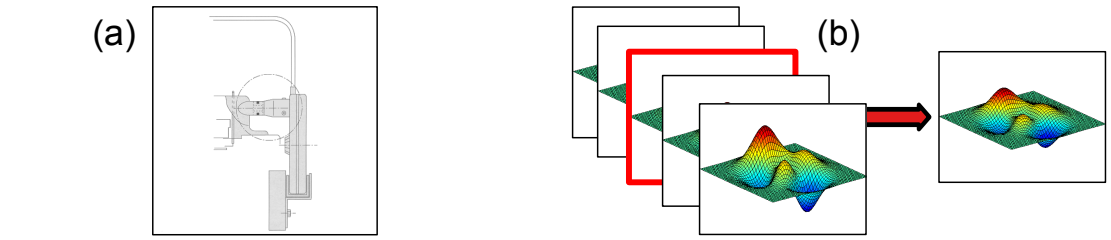
4.1.1 Definition of Pages

Section 3.3.2 defined two properties of PAGES with respect to task-based adaptation of visual representations:

- A PAGE definition encapsulates a self-contained unit of graphical content, whereby it abstracts from the internal data format.
- In addition, the PAGE includes all information required to generate a default visual representation so it can be used as the visual representation for a given task without stipulating further details in the task context.

Thus conceptually, a raster graphics PAGE is the source for a pixel matrix that defines a two-dimensional view of the depicted content. There are two principal sources for this kind of content:

Static image files. Both PAGE properties are immediately supported by a wide variety of raster image formats such as PNG [Mia00] or JPEG2000 [TM01]. Here, a single image file completely describes the pixel matrix comprising the image. A raster




```
<?xml version="1.0" encoding="utf-8"?>
<pageMetadata symbolicName="Abb29L_noLabels"
              xmlns="http://www.m6c.de/">
    <imageFilename>
        Abb29L_noLabels.png
    </imageFilename>
    ...
</pageMetadata>
```

```
<?xml version="1.0" encoding="utf-8"?>
<pageMetadata symbolicName="MathIllustration"
              xmlns="http://www.m6c.de/">
    <multiframeImage defaultFrame="2">
        Animated2DFunction.gif
    </multiframeImage>
    ...
</pageMetadata>
```

Figure 4.1: XML definition of raster graphics PAGES. (a) Static image implicitly define a default representation. (b) Multi-frame images provide a sequence of views over time so the default representation is defined by selecting a point in time, or frame number.

image file therefore is in itself sufficient to describe a PAGE, as the image also describes its default representation.

Multi-frame images and video streams. Some image formats, such as Animated GIF or Motion-JPEG [JPE01b], as well as video streams provide a sequence of images with a temporal dependency. Each individual frame comprises a pixel matrix that represents a two-dimensional view for a specific point of time. The PAGE content is therefore described by a single file encoding a multi-frame image or video stream. The page's default representation is defined by selecting a specific point of time (i.e., a particular frame).

Note that in the scope of this thesis adaptation is deliberately limited to single frames rather than image sequences in their entirety. For this reason, for sources of the latter type a single frame for a particular point of time is assumed as the default representation. This notwithstanding a specific video may be played for a given task which however is not adapted further.[1] Adaptation of time-variable image sequences is a possible extension to the basic approach discussed in Section 3.3 but presents challenges of its own beyond the scope of this thesis, see Chapter 9.

Figure 4.1 illustrates how both types of sources for pixel matrices are encapsulated as a PAGE to obtain a self-contained description of graphical content with a default representation.

4.1.2 Definition of FEATURES

Because raster images by themselves have no notion of content structure beyond individual pixels, content elements can only be defined with respect to regions in image space.

[1] This, by extension, also applies to animated 2D vector (Chapter 5) and 3D graphics (Chapter 6).

Therefore for raster graphics PAGES, FEATURES are equivalent to Region of Interest (RoI) within the image. Their definition involves two aspects:

- determining what image regions comprise FEATURES, and
- means to encode the location and shape of these regions to facilitate the annotation of task nodes with FEATURE definitions.

Determining FEATURE regions

On a conceptual level, FEATURES represent content elements with a semantic meaning within the application domain. Thus the location, size and shape of corresponding image regions representing these objects will vary significantly depending on the depicted image contents. Therefore, RoI definitions over raster images must satisfy the following properties to support arbitrary FEATURES:

- the RoI boundaries of FEATURES are of arbitrary shape,
- FEATURE regions can overlap, i.e., pixels can contribute to more than one FEATURE,
- complex FEATURES may include several related content elements, i.e., be composed of sub-features defining disjunct regions, and
- and FEATURE regions can contain holes, i.e., pixels within the outer boundary that are not considered part of the FEATURE's RoI.

Figure 4.2 exemplifies these requirements for a technical illustration from the maintenance scenario. The FEATURES representing connected machinery components can partially (dark blue and red polygons) or completely overlap (orange rectangle in blue polygon). In addition, a FEATURE may comprise of several elements depicted in non-connected image regions (e.g., yellow rectangles marking a pair of fastening screws). Certain illustrative techniques like cutaway views result in FEATURE regions that have holes with respect to their representation in the raster image.

On a technical level, determining feature RoIs is effectively an image segmentation problem: the goal is to classify each pixel according to whether it contributes to particular FEATURE region(s), or no to RoI at all (i.e., is a background pixel providing contextual information). In doing so, a given raster image can be segmented in two ways: *automatic segmentation* using algorithmic image segmentation approaches, and an *interactive specification* of segments by a content author.

Automatic image segmentation. A number of segmentation algorithms for raster graphics have been developed that partition raster images into foreground objects and background regions, cf. Section 2.3.2. Using an algorithm-based approach to FEATURE detection offers two advantages:

- Because image segmentation classifies individual pixels, very accurate FEATURE regions are defined at the granularity of pixels.
- Using a segmentation algorithm allows to perform FEATURE detection automatically. This reduces the amount of manual labor required to prepare raster graphics-based scalable content for the use in smart visual interface.

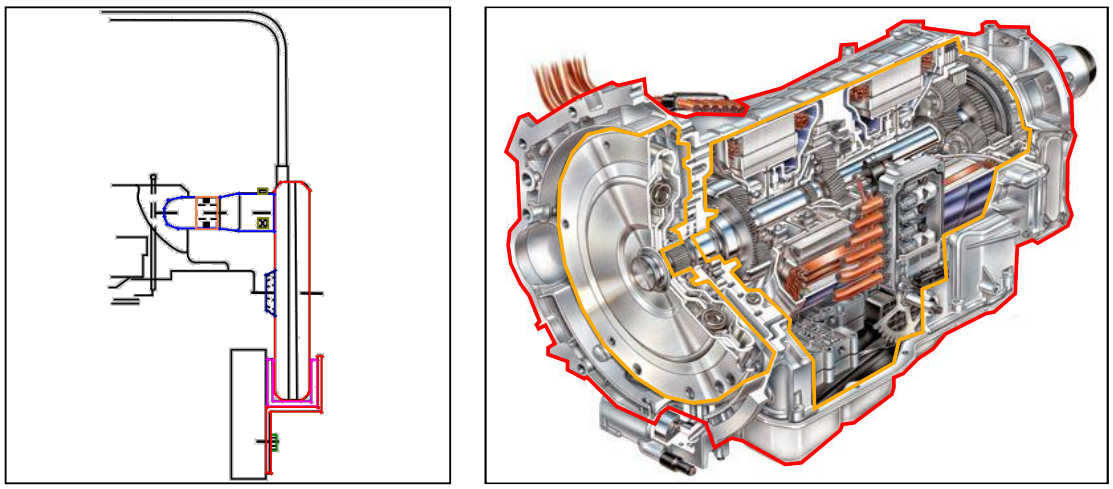


Figure 4.2: Division of exemplary raster image PAGES for maintenance tasks into FEATURE regions with domain-level semantics. (a) Region boundaries can partially overlap or contain other regions, FEATURES may comprise several disjunct sub-regions (colored polygons). (b) Cutaway views result in FEATURE regions (red polygon) with holes (orange polygons) (Image source: [She09]).

However, automatic image segmentation also faces several problems with regard to detecting FEATURES that are relevant to the specific task at hand. Contemporary segmentation approaches work either *bottom-up* or *top-down*.

- Bottom-up approaches use saliency maps based on visual attention models to determine "visually interesting" image regions. Visually salient regions, however, often will not correspond to FEATURES with a semantic meaning in the context of the task at hand. In these cases detected low-level image features still need manual refinement by a human content author. This severely limits the utility of bottom-up approaches as an automatic content preparation tool.

- Top-down approaches try to segment an image into high-level features (e.g., human faces) and background pixels. For top-down approaches to be useful, classifiers for semantic features of the particular application domain must be available. While this is principally feasible for highly formalized drawings like technical circuit schematics, it is not viable to support arbitrary illustrations with different (artistic) styles, perspectives etc.

- Moreover, almost all of the automatic approaches assume a disjunctive segmentation, i.e., each pixel belongs to at most one segment. As a result, features detected by automatic segmentation may at most be adjacent to each other. Specifying FEATURES with overlapping regions thus requires multiple segmentation passes and merging their results, which is difficult to automate.

For these reasons, the specification of FEATURE regions in raster images that are relevant in the context of a task in the application domain is a largely manual task.

Automatic segmentation algorithms can support, but not replace, interactive definition of the location and shape of FEATURE regions by a content author. In particular, image segmentation algorithms can be used to arrive at a provisional segmentation, which is then refined manually to represent the desired image regions of domain-level FEATURES.

Interactive specification of FEATURE regions. Here, a human content author manually defines regions in the PAGE's raster image and associates them with FEATURES. For this purpose the author views the image in a suitable software tool that allows her to trace FEATURE regions depending on the depicted contents. The principal advantage of interactive specification is thus that it is applicable to arbitrary images regardless of the application context.

In doing so, there are two principal ways how FEATURE regions are designated. These represent different trade-offs between the accuracy at which regions are defined, and the effort required in doing so.

Regions are traced with pixel accuracy. The procedure is equivalent to the method employed by most raster image manipulation programs. A region is interactively manipulating by iteratively adding or removing pixels to the current selection of pixels. When the author is satisfied with the shape of the region, the current selection is designated as FEATURE region.

Thus designating all FEATURE regions results in an *ID buffer* with the same pixel resolution as the PAGE's image, whereby each ID buffer 'pixel' holds a tuple of FEATURE IDs indicating that pixel's contribution to the regions of the respective FEATURES.

The advantage of pixel-accurate tracing is that a high accuracy up to pixel granularity is achieved. The drawback is that it requires comparatively elaborate interaction, although this level of region accuracy is usually unnecessary for task-based adaptation.

Regions are defined by their boundaries. To this end, a content author defines polygonal shapes in image space to demarcate FEATURE regions. Simple FEATURES RoIs are quickly defined by simple geometric shapes, e.g., rectangles and circles. More complex regions are defined by tracing a polygon around the outline of the depicted objects.

This allows to interactively strike a balance between authoring effort and the accuracy of FEATURE regions that is appropriate for the given application domain. This is illustrated by Figure 4.2: small details like the fastening screws (yellow rectangles) in sub-figure (a) are designated by an approximate bounding shape, whereas large and complex shaped objects like the cut-away engine casing in sub-figure (b) are traced by polygons.

Therefore, in this thesis the author-controlled, polygonal representation-based approach to the specification of FEATURE regions has been pursued. This necessitates proper tool support for the preparation of raster graphics content, as discussed in Section 4.1.3.

Encoding raster graphics FEATURES for task model annotation

The defined FEATURE regions represent an external description of distinct content elements of the PAGE, thereby augmenting the unstructured pixel matrix provided by the PAGE's raster image. To facilitate task-based adaptation of the raster image's representation, the task model must be enriched with these descriptions by annotating the corresponding task nodes (cf. Section 3.3.3). To this end, the region boundary definitions and their association to FEATURES must be encoded.

According to the requirements to raster graphics FEATURES as defined on page 63, the following FEATURE properties must be encoded:

- A FEATURE may comprise of multiple distinct content elements, each defined by a disjunct Region of Interest. To express this *is-part-of* relation between high-level FEATURES and content elements, FEATURES are hierarchically organized. Thus, a FEATURE has either two or more sub-FEATURES as children, or a single region definition as its only child. FEATURE boundary definitions thereby represent the leaf level of a FEATURE hierarchy. Figure 4.3a shows this hierarchical organization schematically.

- The boundaries of FEATURE regions are defined as geometric shapes in the pixel coordinate system of the raster image (Figure 4.3b). Polygon boundaries are represented compactly by listing polygon vertices as pixel coordinate pairs. Simple shapes like rectangles are defined by a corner vertex and the width and height dimensions in pixels. Every FEATURE region boundary comprises exactly one shape defining its *outer ring*.

  To further allow for FEATURE regions with holes, a FEATURE's boundary description optionally includes a number of *inner rings*.[2] Inner rings are themselves geometric shapes that specify the boundary of local areas excluded from the FEATURE region. Thus, a pixel is considered inside a region if it is inside (or on) the region's outer ring, but outside all inner rings (cf. Figure 4.2b).

Appendix A lists the XML Schema that formally specifies the encoding of raster image FEATURES used in this thesis.

4.1.3 Feature Relevance Values

To effect task-specific adaptation of FEATURES that have been annotated in the task model, their relevance values have to be specified. While there exist approaches to automatically derive image regions corresponding to FEATURES themselves, in contrast task-specific FEATURE relevances can *only* be determined manually; just as establishing a suitable breakdown of a specific domain work flow (the task model) is a manual design task by necessity. For this reason, appropriate *authoring tool*s are required that allow a human content author to interactively define task-specific FEATURE relevance and further properties per task, and to annotate the task model accordingly.

[2] This nomenclature follows the naming convention used by the Geographic Markup Language (GML) to describe boundaries of complex area (map) features [Por07]. It is not to be confused with the mathematical term for an algebraic structure.

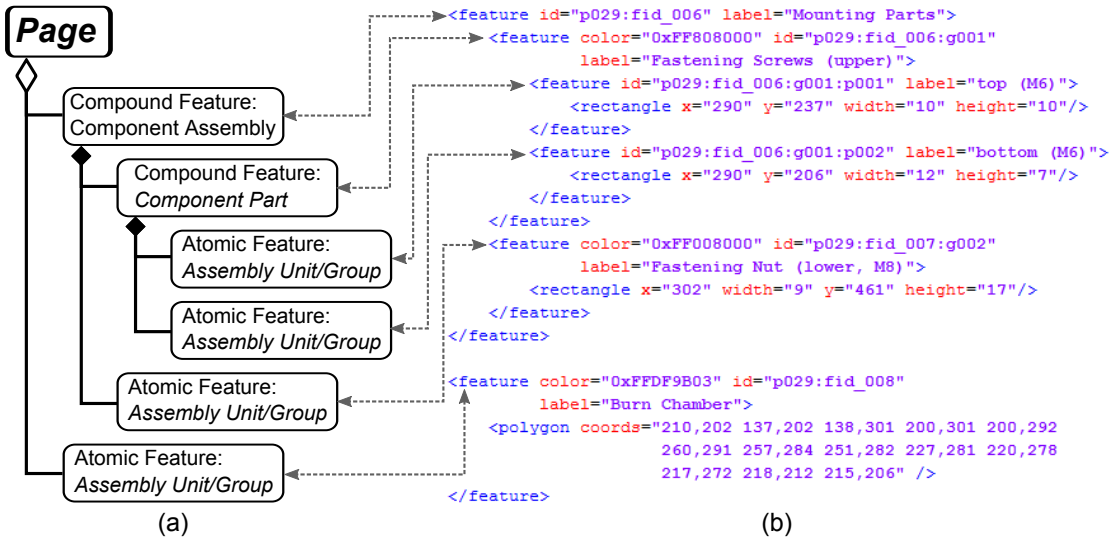


Figure 4.3: (a) Schematic view of the hierarchical structure of FEATURE annotations for raster images to e.g., reflect the composition of machinery assemblies. (b) Corresponding XML notation for some of the FEATURES defined in Figure 4.2a: the boundary of FEATURE regions is described by a geometric shape in pixel coordinates.

Raster Authoring Tool

For raster graphics, in this thesis an authoring solution was pursued that supports all steps of content preparation within a single tool. This facilitates seamless and interactive enrichment of the task model by avoiding the need to switch between several different software tools. In particular, the following authoring steps are covered:

- Creation and editing of PAGE definitions,
- interactive definition of Regions of Interest and their association with FEATURES of the given PAGE,
- annotation of nodes from the CTT task model with PAGE/FEATURE definitions, and
- interactive specification of task-specific FEATURE relevance values as well as additional FEATURE attributes utilized during adaption (see Section 4.2).

Because the primary functionality is the interactive definition of FEATURE regions – or "masking" the raster image – as well as the enrichment of task nodes with visual representation-related annotations, the authoring tool has been dubbed "Mask and Task Editor", or *MaTE* for short. Figure 4.4 shows a screen shot of this authoring tool.

Creating and editing of PAGE definitions. To create a new PAGE, the software automatically edifies a new PAGE manifest. A PAGE manifest is an XML-encoded structure that associates a raster image with adaptation-related information, cf. Appendix A. In particular, the PAGE manifest for raster image pages includes:

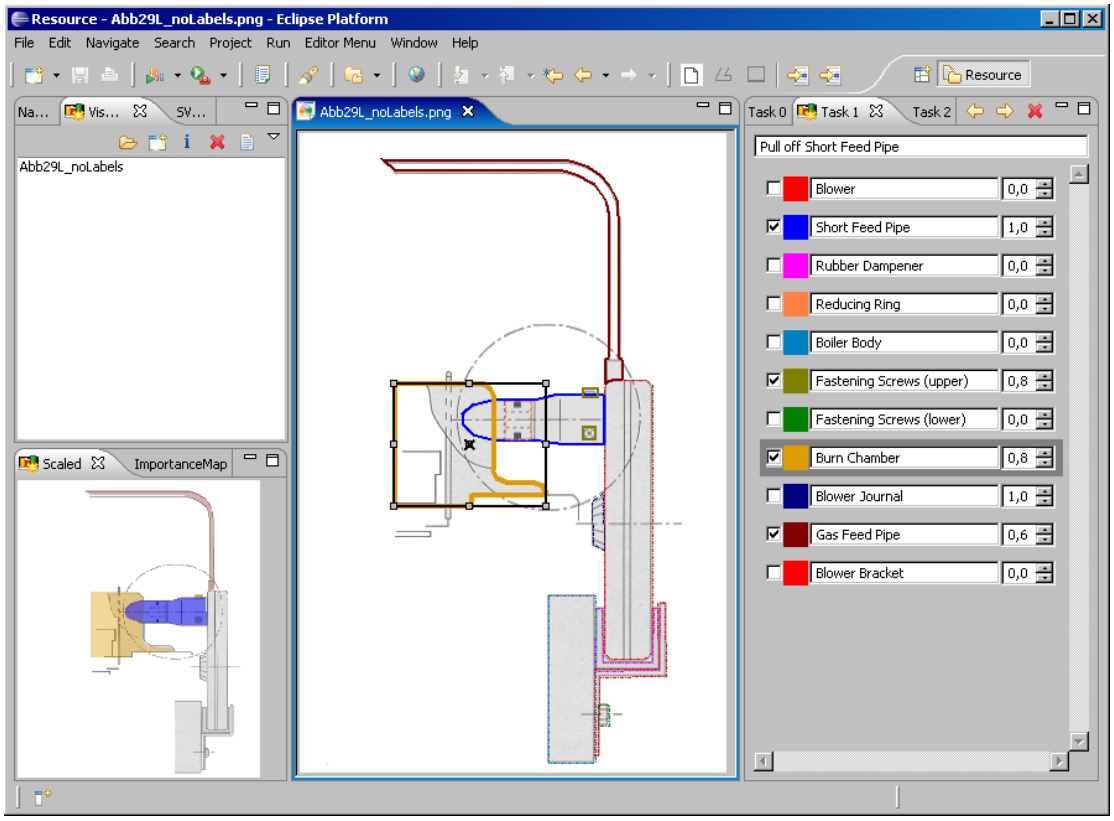


Figure 4.4: MaTE authoring tool used to prepare raster image PAGES. The author can interactively outline FEATURE regions in the image (center pane). FEATURES can be associated with task nodes, assigned task-specific relevance values and attributes for the adaptation process (tabbed pane on right). Previews allow to check the effect of these task-specific settings (tabbed pane on lower left).

- A PAGE identifier. The identifier is used to reference a particular PAGE manifest from the annotated task model.
- A human-readable title, that is e.g., displayed as the window title within the smart visual interface.
- The reference to a raster image file storing the image that constitutes the content of this PAGE.
- The list of FEATURES defined on this PAGE.[3]

All these properties can be edited for existing PAGES, including the image file reference. The latter is useful in cases where the initial raster image has been replaced by an enhanced version, e.g., by the removal of noise, moire and similar digitalization artifacts.

[3] Conceptually, FEATURES are defined at the scope of the task that they are generally relevant to, which may be at a narrower scope (lower task hierarchy level) than that of the PAGE definition, cf. Section 3.3.2. However, for practical reasons of data consistency all FEATURE definitions are coalesced into the PAGE manifest.

Interactive definition of Regions of Interest. The basic procedure involved in the interactive definition of FEATURE regions has already been described in Section 4.1.2. A new FEATURE is created by placing a boundary shape over the image and associating it with an identifier. The current implementation of the authoring tool to this end supports the boundary shapes polygon, axis-aligned rectangle, and ellipses. The identifier is used to reference the corresponding FEATURE RoI from task node annotations. Newly created FEATURES are automatically added to the PAGE manifest.

Already defined region boundaries can be selected and manipulated by moving and resizing the boundary shape in its entirety, see Figure 4.4. Polygon boundaries can further be manipulated by dragging individual vertices, thus allowing to incrementally refine the tracing of object outlines.

Moreover, defined boundaries and FEATURES can be grouped to form composite FEATURES. By repeatedly grouping a hierarchical composition of FEATURES (image regions) can be obtained to express an is-part-of relation, cf. Section 3.3.3.

PAGE/FEATURE association with tasks. While the content authoring tool itself is not designed to create or structurally modify CTT task models, it can read existing models stored in XML format.[4] This results in a list of tasks that can then be associated with appropriate PAGE manifests. Because FEATURES are an integral part of the PAGE manifest, FEATURE-task association is effected by selecting the subset of principally relevant FEATURES for the task at hand. Figure 4.4 shows the corresponding check boxes in the list of PAGE features (right window pane).

Specification of task-specific FEATURE properties. This working step primarily comprises the interactive definition of FEATURE relevance values. In addition, the authoring tool currently supports the addition of two types of auxiliary FEATURE attributes: a short text that is used for labeling, as well as a specific highlighting color that is used during the visual attribute adaptation phase (cf. Section 4.2).

After the task nodes of the imported CTT model have been annotated with these data, it can be saved back in XML format. This enriched task model is then further processed in the course of the superordinate UI design process, see e.g., [BDF+06, FRW08].

4.2 Adaptation Control

After the graphical content preparation using the authoring tool has been concluded, the task model enriched with PAGE/FEATURE/relevance annotations contains all information necessary to control the adaptation process.

In order to tailor task-specific initial representations, adaptation operations of the general adaptation pipeline are mapped onto concrete, data-type dependent techniques. In this section, we discuss what concrete operations are viable on the different pipeline

[4] The MaTE authoring tool is implemented as a plug-in for the Eclipse Rich Client Platform (RCP). The model editing suite developed in the LFS project (cf. Sect. 2.2) is likewise based on Eclipse RCP (see [FWDR06, RDFW08]). Therefore, MaTE can interoperate with other CTT editing tools within a single RCP application.

stages specifically for raster graphics, and what information must be annotated to the leaf task nodes of the enriched task model to effect these operations.

4.2.1 View selection

The goal of view selection is to determine a rectangular section of the raster image that constitutes the visible portion of the PAGE for the task at hand. By default, this section is identical to the entire image. However, precisely because a PAGE is associated with a subtree representing a composite task, the visible portion of the PAGE must be adjusted for each basic task (leaf node) individually. In particular, the visible section may be smaller than the size of the original image, e.g., if the current working step requires a zoomed-in detail view. The PAGE's visible portion thus becomes the basis for adaptation in subsequent pipeline stages. To this end, two principal questions need to be answered:

- What portion of the PAGE is to be displayed for the task at hand? This means the position and size of a *source region* with respect to the original raster image constituting the PAGE needs to be selected.
- How is this source region fitted into the *target viewport*, i.e., the display area that is available for the visual representation on screen?

With regard to the first question there are two principal approaches: the source region can either be *derived automatically*, or *specified explicitly*.

Our strategy to automatic derivation utilizes the FEATURE definitions. On the basis of a relevance threshold t all FEATURE regions with relevance $r \geq t$ are selected that contribute relevant information to the task at hand. Then, the bounding rectangle that contains the regions of all thusly selected FEATURES regions determines the source region of the PAGE for the given task at hand. In particular, using a threshold of $t = 0$ the source region is defined so as to select all FEATURE content while discarding the image's non-FEATURE periphery, see Figure 4.5a.

This automatic determination of the task-specific source region is deployed as the standard view selection operation on raster image PAGES. However, some tasks may require initial views that show additional image context that extends beyond the bounding rectangle of relevant FEATURES. In these cases, we allow to manually override the view selection by the content author. This necessitates an explicit specification of the source region by a rectangle $(x_0, y_0, \mathit{width}, \mathit{height})$ given in pixel coordinates, with (x_0, y_0) the upper left corner and *width* and *height* the source region's extend to the right and down, respectively.

The second question is directly related to the WHERE aspect (display scalability) of the visual representation's context of use. Depending on the output device the sizes (pixel resolution) and aspect ratios of source region and target viewport may not match. Here, it must be decided on how the source region contents are scaled to fit into the target viewport. In some cases it is desirable that the graphics stretch to fit non-uniformly to take up the entire viewport (in other words, fill the entire available display area). In other cases, it is desirable that uniform scaling be used for the purposes of preserving the aspect ratio of the graphics.

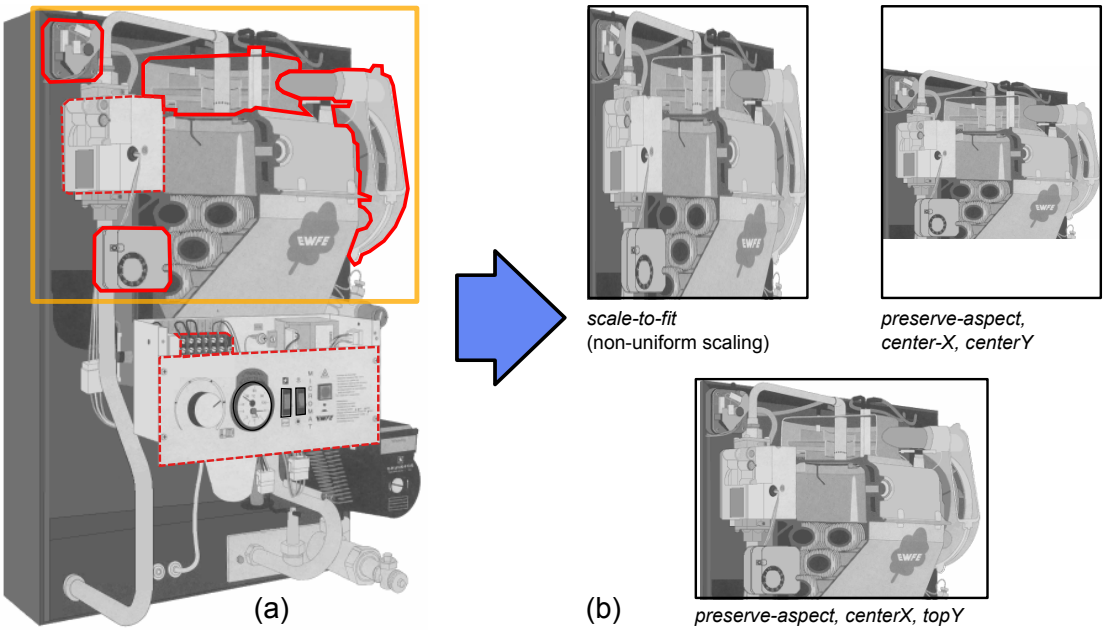


Figure 4.5: (a) The automatic relevance-based view selection (orange) crops the source image to contain all FEATURE regions above a threshold t (solid lines), discarding FEATURES of lesser relevance (dashed) and image periphery. (b) Target viewport fitting is necessary if mismatching the source region's aspect ratio.

To this end, different fitting strategies can be expressed by a combination of two parameters (cf Figure 4.5b):

- A flag indicating whether or not non-uniform scaling of the source region is allowed, and
- an alignment parameter that determines how the source region is placed inside the target viewport.

If non-uniform scaling is allowed, the graphic content of the source region is scaled such that its bounding box exactly matches the viewport rectangle. The alignment parameter is ignored in this case.

If uniform scaling is enforced, the source region is scaled such that it completely fits in the target viewport. The alignment parameter then is a tuple $(align_{horizontal}, align_{vertical})$ that determines, respectively , if the source region is left-aligned, centered or right-aligned horizontally, and top-aligned, centered, or bottom-aligned vertically. See Figure 4.5b for an illustration. Note that unused space in the target viewport is filled with image content from outside the specified source region in order to utilize as much of the available display space as possible.

4.2.2 Geometry adaptation

The core idea behind geometry adaptation is to enlarge important FEATURES at the expense of less relevant ones. With regards to raster images, this amounts to adjusting

the size of individual image regions according to FEATURE relevance. In our adaptation approach two strategies are employed to attain this goal:

Utilizing Focus & Context distortion. The general approach to Focus & Context-based distortion of raster images is to enlarge a focus Region of Interest in such a way that a continuous transition to the downscaled image context is achieved. To this end, a Degree of Interest (DoI) function is used that defines how scaling factors are propagated from the focus to the context regions.

However, the majority of Focus & Context distortion approaches assume but a *single focus* point or region *with maximum DoI* and a *uniform DoI distribution* across the image. For raster images, the DoI function is typically based on distance to the focus (region) in image space.

Contrary to this, the approach pursued in this thesis is to allow for *multiple focus regions* with *varying DoI* (i.e., relevance values), and a *non-uniform DoI distribution* which is determined by the size and distribution of focus regions in image space. For this purpose, the FEATURE information provided by the extended task model are utilized. The FEATURE regions constitute the focus Regions of Interest. The FEATURE relevance values are the input parameter to the DoI function that determines the distortion of FEATURE and non-FEATURE regions accordingly.

In doing so the WHERE aspect must be considered as well. Evaluating complex distortion functions per-pixel is computation-intensive. On compact mobile devices this will negatively impact both response times and battery life. Therefore, a good trade-off between computational complexity and visual quality must be found.

To meet these requirements, a novel image distortion technique called belt-based Focus & Context has been developed that can handle multiple FEATURE-based RoI with varying relevance values. It was initially published in [FRSF06]; a detailed discussion of the proposed approach is given in Section 4.3.1.

Adoption of the Seam Carving approach is our second strategy to achieve FEATURE-based distortion of raster images. The Seam Carving technique developed by Avidan and Shamir [AS07] introduce a non-linear, content-dependent scaling of raster images. It works by removing (for downscaling) or duplicating (for upscaling) 8-connected seams of low-energy pixels based on an energy map of the image, until the desired image size is reached. Its primary application is image *size reduction* by removing pixels in "uninteresting" (low-energy) background regions of photo images such as Figure 2.7 on page 22.

Pixel energies can be locally biased within author-defined regions to affect the routing of seams. Regions biased towards lower energy attract more seams, thus removing comparatively more content from those regions. Conversely, regions biased towards higher energy levels will not be affected by seam cutting until most low-energy background regions have already been reduced.

Utilizing Seam Carving as a means for adaptation has been reviewed in parallel with Thiede [Thi10]. There, the base approach has been extended regarding two perspectives: multiple heterogeneous output devices, and its application to not only photographies

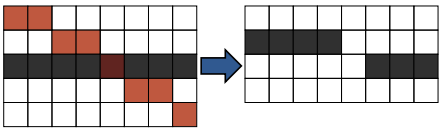


Figure 4.6: Effect of a horizontal seam cutting a horizontal line. The seam pixels (orange) are removed while pixels below the seam are shifted up, resulting in a disrupted line.

but also information visualizations. That means [Thi10] examined the *evaluation* of the energy function with respect to concrete representations (HOW) and devices (WHERE aspect), rather than adapting pixel energy determination itself. By contrast, the idea pursued in the present thesis is to further incorporate the WHY aspect to influence *determination* of image energy in a task-specific way.

The basic Seam Carving approach thereby interfaces with our FEATURE-based distortion strategy in that task-specific FEATURES provide regions of pixel energy bias. FEATURE relevance values determine the magnitude of energy bias: the higher the relevance value, the stronger the energy bias. The bias value range that relevance is mapped to depends on the concrete pixel energy metric used.[5] Generally, the minimum relevance $r = 0$ is mapped to zero bias, whereby the most important FEATURE regions ($r = 1$) are biased with the maximum possible pixel energy value. The effect of applying Seam Carving to the raster image of a PAGE therefore is a non-linear down-scaling of non-FEATURE context regions while important FEATURES retain their original size.

One challenge to the application of Seam Carving are technical illustrations with vertical and horizontal edges between FEATURES (e.g., connecting lines between elements of an electrical circuit). These might become disrupted as connected seams of pixels are removed from the image (Figure 4.6).

This problem has been addressed in our adaptation approach by extending the basic seam carving approach. Unlike the the original technique, our extended solution uses separate energy maps for horizontal and vertical seams. For horizontal seams, the energy of horizontal lines is increased by applying a corresponding edge detection filter, thereby preventing seams from cutting these lines at acute angles. Analogous, the energy of vertical edges is increased in the energy maps for vertical seams. Figure 4.7 gives an example for the results obtained with the extended Seam Carving approach: circuit elements constitute FEATURES of the page; seams are removed only in background regions in between, cutting connecting lines only at right angles.

4.2.3 Visual Attribute Adaptation

For raster images, adaptation of FEATURES' visual attribute equates to a manipulation of FEATURE regions in image space. For this reason, the visual attribute adaptation and image space manipulation stages of the adaptation pipeline comprise the same operations – those that modify values in the pixel matrix by applying image filter operations.

[5] See [AS07] for a detailed discussion of viable pixel energy metrics.

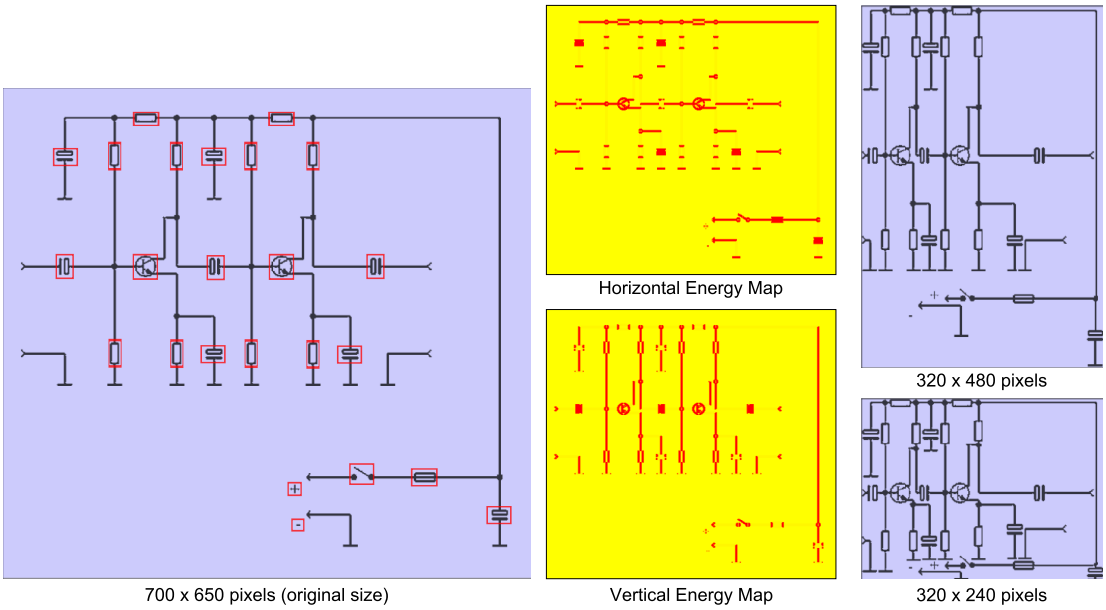


Figure 4.7: Extended Seam Carving applied to a raster image of a circuit schematic to obtain a visual representation of reduced size. FEATURE bounds (overlaid in red) define protected regions (positive energy bias). Using separate energy maps for horizontal and vertical seams prevents disruption of connecting lines.

The distinction between both stages is that conceptually, FEATURE attribute adaptation modifies the pixel matrix in an image section constrained to the FEATURE region, whereas the subsequent pipeline stage also includes modification of non-FEATURE image regions. We therefore review viable operations for both pipeline stages together in the following section.

4.2.4 Image Space Manipulation

The purpose of pixel matrix manipulation is to visually accentuate relevant FEATURES and to visually de-emphasize less relevant context regions, thus creating a visual hierarchy of FEATURES according to their relevance to the task at hand (cf. Section 3.3.4). The objective is to guide the viewers' attention to the most important parts of the adapted visual representation. There are two primary ways to create the desired visual hierarchy of FEATURES regions on raster images considered for our approach in particular:

- the local modification of color saturation, brightness, or image contrast [Den99] to alter the visual saliency of image regions, and
- blurring (unsharpen) the image in regions with lesser relevance to achieve a *Depth of Field* (DOF) effect [KMH01].

Color adjustment is the principal and most frequently used method for raster image manipulation. It is applicable for both the accentuation of relevant FEATURES as well as for de-emphasizing less relevant picture context.

Increasing contrast and/or brightness cause accentuation of FEATURES, whereas reducing contrast and darkening are employed to de-accentuate image regions, cf. [Hop99]. These adjustments are applicable to any raster image regardless of whether the pixel raster contains color information or brightness values only (i.e., it is a gray-scale image). To this end, FEATURE relevance values $r_i \in [0,1]$ are mapped to an adjustment factor $a \in [-100\%, +100\%]$ for the respective color parameter. Brightness and contrast can be adjusted independently by mapping to two independent adjustment factors $a_{brightness}$, $a_{contrast}$. Different mapping functions are possible, such as linear, exponential or discrete (stepwise) adjustment, see Figure 4.8. Using these functions image space adaptation can be parameterized in two ways:

- the relevance threshold value r_t that determines what FEATURE regions are visually accentuated ($r_i > r_t$) and which are de-accentuated ($r_i < r_t$). Thus, FEATURE regions with relevance $r_i = r_t$ are left unmodified.
- The cut-off limits for the maximum and the minimum adjustment factors. Especially when reducing contrast and brightness, the minimum relevance value $r = 0$ is usually mapped to an adjustment factor well above -100% lest image details become indistinguishable.

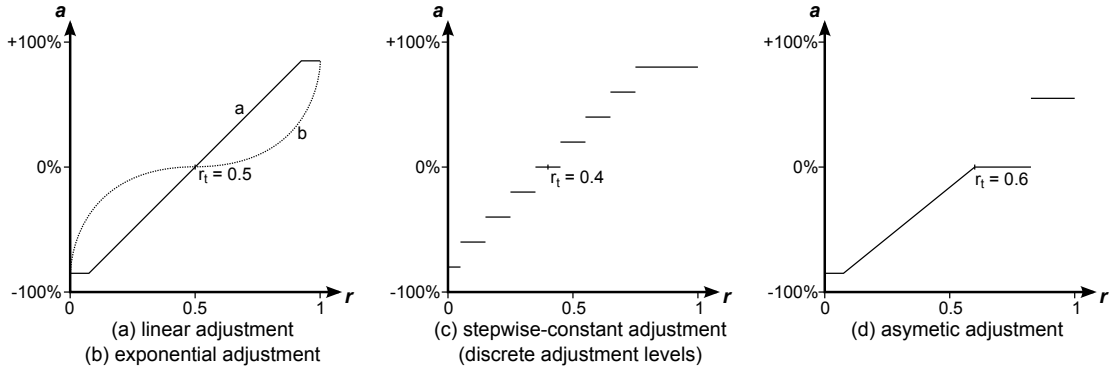


Figure 4.8: Examples for different mappings of FEATURE relevance r to color adjustment factors a.

On the other hand, adjusting the color saturation as a means for accentuation obviously requires color information. Therefore in this thesis, we distinguish between color images versus gray-scale images, including binary black-and-white illustrations. We here propose two different strategies that allow to employ color saturation adaptation to both image classes regardless:

- Color images are adapted by de-emphasizing non-FEATURE regions, i.e., by desaturating colors in regions of the image context according to the mapping function assigning FEATURE relevance values to an adjustment factor $a_{saturation}$. Here again, non-FEATURE regions are assigned the minimum relevance $r = 0$.
- Gray-scale images, by contrast, are adapted by visually accentuating FEATURE regions. Because gray-scale images lack color information, an effective means for

Figure 4.9: Examples for adapted visual representation of raster images. (a) Irrelevant regions in color images are de-accentuated to visually promote important FEATURES that retain the full color saturation. (b) FEATURES in black-and-white/gray-scale images (cf. Fig. 4.5) are accentuated by applying a tinting effect with a FEATURE-specific color. Blurring background regions to simulate Depth-of-Field is applicable to both image classes.

accentuation is to apply a tinting effect. This effect is achieved by compositing the FEATURE's boundary polygon filled with a "highlight" color over the source image according to Porter-Duff's *source-over* composition rule [PD84]. This color thereby constitutes a visual attribute of the corresponding FEATURE. Here, the adjustment factor $a_{saturation} \in [0\%, 100\%]$ is used as the alpha value in the blending equations.

Figure 4.9 illustrates the visual effects that are achieved by applying relevance-driven color adjustment to examples for both image classes. The most relevant FEATURES are accentuated the strongest, whereas irrelevant FEATURES and the image background are visually subdued.

Semantic Depth of Field (SDOF) is another method for guiding the user's attention. Less relevant parts of the display are de-emphasized by blurring, while the relevant information is displayed sharply [KMH+02b, Hau06]. This method has long been used in photography, where the camera lens' depth of field (DOF) determines which depth range of a captured scene is depicted sharply.

The main idea behind the semantic Depth of Field approach proposed by Kosara et al.[KMH01] is to control the sharpness of objects by their current relevance rather than their distance. The original SDOF approach has been developed as a Focus & Context technique in computer graphics, visualization applications in particular. As such, it takes as input the spatial arrangement (in 2D or 3D) of distinct objects and a relevance criterion for each object. Object relevance is mapped by a blur function onto a *blur diameter* that emulates the so-called Circle of Confusion of a real lens camera and thus, the amount of blur. From the desired blur diameter a Z-coordinate (depth values) is derived, whereby objects are placed further away from the focal plane as their

relevance diminishes. Based on these values a blurred representation of the objects is rendered using a photo-realistic camera model.

The general approach to apply SDOF to raster image PAGES is as follows. The FEATURE regions (defined by their boundary shapes) constitute the objects subjected to SDOF blurring. FEATURE relevance values directly provide the input to the blur function. Most important FEATURES (relevance $r = 1$) are placed directly in the focal plane, i.e., they are depicted perfectly sharp. Regions of FEATURES with lower relevance values are subjected to progressively stronger blur depending on the blur function used. See [KMH01] for a list of suggested functions mapping relevance to blur factors. Image background (non-FEATURE regions) is assigned a relevance $r = 0$ resulting in the maximum blur diameter.

However, when using SDOF as a means to de-emphasize less relevant PAGE regions the output device (WHERE aspect) also must be considered. The camera models for blurred rendering proposed in [KMH01] are not viable on most compact mobile devices lacking hardware-accelerated graphics [KMH02a]. Therefore on low-end mobile devices, instead of rendering a blurred representation of FEATURE regions with a complex camera model, we use simple Gaussian blur filters.[6] Filters are applied constrained to the regions of individual FEATURES. Here, the feature relevance is used to select the filter's kernel size and thus, the amount of blur applied to the respective FEATURE's region. This provides an acceptable approximation of a "real" DOF effect that is computationally less complex, see Figure 4.9.

4.2.5 Labeling

The final stage of the adaptation pipeline comprises dynamic labeling of the adapted visual representation. Section 3.3.4 argued why labeling is an important step of the adaptation process and how dynamic labeling is effected as a visual data type-independent adaptation step in image space.

This section discusses in detail how the particle-based labeling approach by Luboschik et al.[LSC08] is integrated with our FEATURE-based adaptation approach. This particular approach has been chosen as an example for the integration of external image-space labeling algorithms because it is one of most efficient solution for labeling point features to date [SMA+09, Che10].

Being an operation in (raster) image space the labeling procedure is identical for all four visual data type contemplated in this thesis. The discussion in this section therefore equally applies to the last adaptation pipeline stage for the data types reviewed in the following Chapters 5–7.

Integration of particle-based labeling. The approach by Luboschik et al.[LSC08] is based on two core concepts: a *labeling pipeline* as well as *conflict particles*.

The basic idea of the labeling pipeline is to label as many point-features as possible within a fast labeling step, to save CPU power and processing time for subsequently more sophisticated approaches applied to fewer remaining points in later pipeline stages.

[6] For very low-end devices, an even simpler average filter (box filter) might be substituted.

Labels placed during early stages also tend to be placed in better positions (i.e., closer to the labeled feature) than those placed later, when close-by positions are no longer available.

To this end, the approach relies on a defined *labeling order* based on label importance. In their paper, the authors assume importance values are given, but do not address how these values might be derived. In combination with our FEATURE-based adaptation approach, this ordering is readily available: the FEATURE relevance values define a task-specific labeling order of FEATURE labels.

The approach's second core concept is that of conflict particles. These represent a method of dynamically discretizing the search space for label-label and label-object overlaps – that both render a potential label position invalid – by placing particles in the 2D search space (image space). Particles are distributed according to requirements derived from the size of the labels as well as the size, shape and distribution of the labeled objects. In particular, Luboschik et al.propose a method to derive conflict particle configurations for arbitrarily shaped object based on the discretization of object boundaries in vector format.

Exactly this information is provided by the boundary shapes of FEATURES in our approach. Since the boundary shapes have been subjected to the relevance-driven geometry adaptation on the second pipeline stage (cf. Section 3.3.4), labels placed with respect to FEATURE boundaries adequately reflect task-specific distortions of the PAGE's visual representation.

Thus, the task context described by the extended task model provides all necessary information to apply particle-based labeling as the fifth adaptation pipeline stage independent of the visual data type: (i) the shape and position of labeled objects in (distorted) image space by means of FEATURE boundaries, (ii) a list of labels to place by means of the corresponding FEATURE attribute, and (iii) a task-specific labeling order by means of FEATURE relevance values.

However, the approach presented in [LSC08] also has one drawback. It is optimized toward placing as many labels as fast as possible. Options to control the style and layout of labels are very limited. In Section 4.3.2 we therefore discuss our own approach to image-based labeling that precedes the particle-based approach, published in [FLH+06]. Here, the focus has been on two aspects: the integration of space-efficient labeling styles with the basic FEATURE-based adaptation approach particularly for small screens; as well as support for remote labeling to provide dynamic labeling even on low-end mobile devices.

4.3 Smart Raster Image Techniques

Up to this point, the general procedure for task-based adaptation of raster image PAGES has been discussed. On this basis two concrete techniques are proposed in this section that address some of the specific challenges that have been outlined in Sections 4.2.2–4.2.5.

4.3.1 Belt-based Focus & Context

Section 4.2 discussed the general idea of using FEATURE regions as Regions of Interest and relevance values to control the geometric distortion of raster images. However, this poses several challenges:

- There is more than one FEATURE region, and each FEATURE is associated with a distinct relevance value that represents this region's Degree of Interest (DoI). It is therefore insufficient to contemplate only a single focus region.
- Multiple FEATURES may overlap in image space. This affects the shape and position of both the resulting focus as well as context regions. It must also be considered when calculating the propagation of scaling factors for image regions, which are derived from relevance (DoI) values, across image space.

Our belt-based Focus & Context approach that is proposed in the following[7] addresses these issues. The basic idea is to use piecewise-constant scaling of adjacent, rectangular image regions (*tiles*) to achieve image distortion. This avoids computationally intensive continuous, non-linear distortion functions that must be evaluated per pixel, as it is for example done in [CCF95].

The scaling factors for the image regions are determined as a function of FEATURE relevance values. In order to ensure that tiles always maintain adjacency, i.e., no gaps or overlaps between neighboring tiles occur, all tiles on the same row must always be scaled uniformly in height (vertical dimension). Likewise, a column of tiles must always be scaled uniformly in width (horizontal dimension). These rows and columns of image tiles with a uniform scaling factor in vertical, respectively horizontal dimension are called magnification *belts*. In particular, each image tile is the intersection of a vertical and a horizontal belt. To this end, our belt-based distortion approach comprises two steps:

1. Calculation of a the grid of adjacent tiles in image space comprising the horizontal and vertical magnification belts. Distribution and width of belts in both horizontal and vertical direction is determined by the intersections of all FEATURE regions.
2. Size adjustment of individual tiles in horizontal and vertical direction according to the relevance values of FEATURES contained in the corresponding belts.

Calculation of belts is done by examining the distribution of FEATURE regions along the X- and Y-coordinates of image space separately. In the following, the procedure is explained for the horizontal projection onto the X-axis. The vertical belt boundaries are obtained analogous by projecting FEATURE boundaries onto the Y-axis.

First, the boundary for each FEATURE ϕ_i is projected onto the X-axis to obtain a minimum coordinate x^i_{min} and a maximum coordinate x^i_{max} (cf. Figure 4.10a). These coordinates represent interval bounds in which ϕ_i is said to *influence* scaling along the X-axis. Then, the projected intervals of all FEATURES are nested according to the order of FEATURE relevance, so that the higher relevance value prevails. This comprises a change to the set of intervals in the following cases (cf. Figure 4.10b):

[7] It has first been published in [FRSF06].

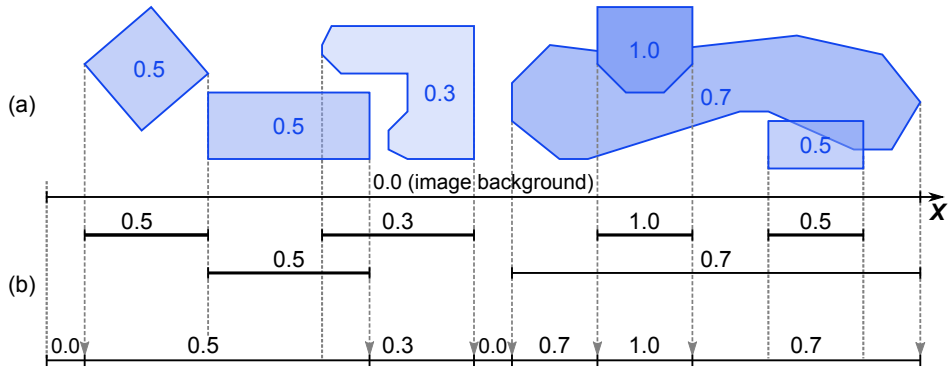


Figure 4.10: (a) Horizontal interval nesting according to maximum feature relevance by projecting the FEATURES' boundaries onto the X-axis. (b) In cases where the projected bounding boxes of multiple FEATURES overlap, the highest relevance FEATURE prevails.

- Intervals of equal relevance are adjacent, overlap partially or completely: both intervals are merged.
- An interval of higher relevance partially overlaps an interval of lesser relevance: the lower-relevance interval is truncated so it is adjacent to the higher-relevance interval.
- An interval of higher relevance is embedded in an interval of lesser relevance: the lower-relevance interval is split into two neighboring intervals adjacent to the higher-relevance interval.
- An interval of lesser relevance is embedded in an interval of higher relevance: the lower-relevance interval is removed from the set.

The result of the interval nesting step is a representation of *belts of maximum relevance* in both horizontal (intervals along the Y-axis) and vertical (along the X-axis) direction. Intersecting horizontal and vertical tiles yields the partition of the raster image into tiles, cf. Figure 4.11a, b.

Size adjustment of tiles is then controlled by mapping the relevance values associated with belts onto magnification factors. This is a two-step process:

1. Relevance values r are mapped onto scaling factors $s \in \mathbb{R} \geq 0$ according to a scaling function S. These scaling factors determine the *initial distortion* of tiles in both X- and Y-direction.
2. To match the size of the target viewport determined during the view selection step of the adaptation pipeline (cf. Section 4.2.1) scaling factors are further multiplied by a correction factor k_x, k_y for the respective axis to arrive at the *final distortion*.

The scaling function $S : r \to s$ can be any monotonous function that maps the normalized range of relevance values $r \in [0, 1]$ to a range of scaling factors $s \in [s_{min}, s_{max}]$, e.g., by a linear, exponential, or stepwise-constant function. Thereby scaling factors s determine the relative size of FEATURE regions depending on the values of s_{min}, s_{max}.

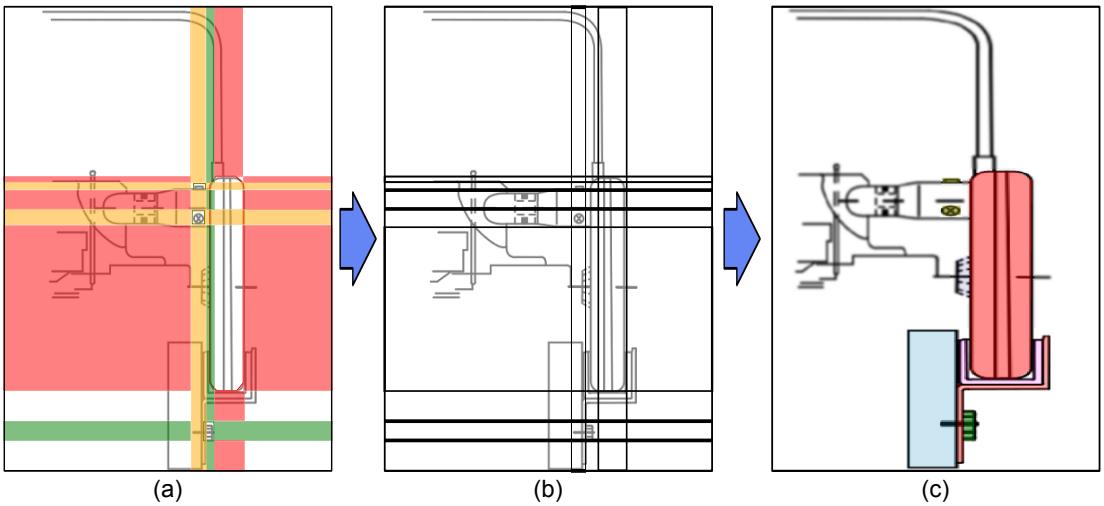


Figure 4.11: Intersection of the horizontal and vertical *belts* of maximum relevance (a) yields the boundaries of image *tiles* (b). Tiles of one belt (horizontal or vertical) are uniformly scaled along the corresponding axis to obtain a piecewise-constant distortion of the image without gaps (c).

For example, mapping relevance values r to the range of scaling factors $[0.5, 2.0]$ results in the most important FEATURE regions ($r = 1$) to be magnified to quadruple the size of context regions ($r = 0$). The smallest possible value for $s_{min} = 0$, effectively removing belts of relevance $r = 0$ from the image.

The initial distortion results in an image with X- and Y-extents determined by the distribution and relevance of FEATURES as well as the selected range s_{min}, s_{max} of scaling factors. Therefore to obtain the final distorted image, correction factors k_x, k_y are applied. This per-axis correction factor is the ratio between the desired target viewport size and the total extent of all initially distorted tiles along the respective axis. Thus for a viewport width of w_{vp} pixels and n_x belts (tiles) along the X-axis with undistorted pixel widths w_i, scaling factors s_i the horizontal scale correction

$$k_x = \frac{w_{vp}}{\sum_{i=1}^{n_x} w_i s_i}.$$

The vertical scale correction factor k_y is determined from the desired viewport height h_{vp} accordingly.

The final distorted size of tiles is thus obtained by scaling the undistorted belts by $s_i k_x$ horizontally and $s_j k_y$ vertically. The final distorted visual representation is then constructed by assembling the resized tiles into the target image (Figure 4.11c). This approach generates satisfactory results on low-end devices like PDAs.

Due to the nature of interval nesting in the first step, so-called ghost foci can be produced: FEATURES that are enclosed in a belt with higher relevance will be over-magnified because each belt is scaled according to the *maximum* relevance of all features contained therein. However, the impact of such ghost foci is significantly mitigated by the subsequent relevance-based adaptation of visual attributes, as discussed in Section 4.2.3, cf. Figure 4.11c.

4.3.2 Space-efficient Remote Labeling

An effective and aesthetic label layout must meet various constraints: the layout must guarantee that the viewer can extract the correct co-referential relations between graphical and textual elements (*unambiguity*), and labels should be laid out so as to ease their *readability*. For interactive applications, the computation further has to be *efficient*. Note that these requirements might conflict with one another. Therefore, most labeling approaches seek a layout that balances these requirements, cf. [FLH+06, Ali09].

Mobile applications such as the mobile maintenance scenario contemplated here pose two additional challenges. The restricted display size of compact mobile devices requires the layout to be *space-efficient* to fit within the limited available space while compromising neither the number of labels placed nor the overall layout quality too much.

Moreover, despite heuristic solutions dynamic labeling at interactive rates might still be too computationally demanding for compact mobile devices due to the inherent problem complexity [MS91]. This means either the quality and number of placed labels must be reduced significantly, or the computational burden must be off-loaded from the mobile device to a more potent remote machine.

To address these particular challenges, we propose a novel *space-efficient labeling approach* that is employable as a *remote service* in client/server environments, as published in [FLH+06].

Our approach is based on an existing labeling algorithm originally proposed in [AHS05]. Before discussing the extensions that have been developed to facilitate task-specific and remote labeling of visual representations, the principles of the underlying annotation module are briefly recapped here. For a description in acute detail refer to [Ali09].

Underlying Approach

The base algorithm supports different layout styles for both internal and external labels. Objects with a large enough area can accommodate internal labels that overlay their reference objects. External labels are placed in the empty space which is not covered by primary elements (objects). While internal labels tightly integrate textual and visual elements, *connecting lines* have to link labeled objects and their annotations, whereby *anchor points* ease the identification of the visual reference object. The approach is based on a set of three *buffers* that store at each pixel, color information (*frame-buffer*), segmentation information (*ID-buffer*) and distance-to-objects information (*distance-buffer*). The frame-buffer is simply the rasterized visual representation as displayed on screen. The ID-buffer, also known as *color-coded image*, represents a segmented view of all visible graphical objects, assigning each pixel a unique color-ID value according to the object that pixel contributes to. Moreover, negative space[8] that is available for placing labels is assigned a unique *background color* in the ID-buffer. The distance-buffer stores for every pixel a value corresponding to the minimum distance to the nearest boundary pixel in the ID-buffer, which either marks the boundary of a feature object, or the edge of the image.

[8] Negative space, in graphic design [Whi02], is all space that is not primary subject. For 2D images in particular, it comprises the background regions not occupied by the depicted object(s).

This distance-buffer is calculated using a fast and efficient *Pseudo-Euclidean* metric. It is derived by applying to the ID-buffer a two-pass distance transform algorithm that propagates minimum distances of pixels to the boundary of segments [SW04]. By analyzing the ID- and distance-buffers, the annotation module decides for each object whether to use internal or external annotation for it and then computes the placement of internal labels, respectively external labels and associated anchor points, in the frame-buffer. To this end an *annotation table* links color-IDs in the ID-buffer to label texts.

Proposed Extensions

For our space-efficient remote labeling approach the following extensions to the basic annotation framework were developed:

- A procedure to dynamically generate the ID-buffer for adapted visual representations of arbitrary visual data type,
- a space-efficient label layout particularly suited for small display sizes, and
- client-server communication based on FEATURE properties to facilitate remote labeling.

ID-buffer Generation. In theory, the use of rasterized buffers (frame-, ID- and distance-buffer) to describe the representation being labeled removes the distinction between annotating 2D raster or vector images and 3D graphics – the annotation module always operates in 2D image space. However, the catch is the required segmentation of the ID-buffer. If the ID-buffer is rendered from a segmented 3D model (cf. Section 6.2.5), it is easily obtained by flat-shading model components using unique color-ID values. However, in [Ali09] it is stipulated that each input 2D image that needs to be annotated comes bundled with the corresponding, manually pre-segmented ID-buffer. This poses a significant limitation if visual representations are subjected to geometric adaptation (i.e., Focus & Context distortion) prior to being annotated.

Instead, with our extended approach the ID-buffer is derived *dynamically* from the adapted visual representation *after* geometric adaptation has been performed. This comprises three steps (see Figure 4.12):

1. Binary segmentation of the visual representation into empty image background (negative space) and foreground pixels to obtain a *content mask*.
2. Assignment of FEATURE to color-IDs and generation of an *ID mask*.
3. Composition of the content mask and the ID mask to obtain the final ID-buffer.

Binary segmentation: Background pixels are determined by the specification of a background color in the PAGE definition. For typical schematics and other technical illustration, this will for example be pure white (cf. Figure 4.2) or fully transparent pixels for image formats that support transparency e.g., PNG. Empty pixels are assigned full black color in the content mask, while non-empty pixels are assigned full white (cf. Figure 4.12a). The result is a binary map that indicates empty background vs. non-empty pixels. Note that non-empty pixels do not necessarily

belong to a FEATURE region – *all* pixels *not* having the designated background color are considered non-empty.

ID mask generation: First, each FEATURE is assigned a unique color-ID. Typically, the highlight color (cf. Section 4.2.4) can be reused for this purpose. The FEATURES' boundaries are then rendered to the ID mask as filled polygons (Figure 4.12b) whereby the polygon vertices are transformed as to match the representation's current Focus & Context distortion (cf. Section 4.3.1). The ID mask is initialized with full white color. If two FEATURES overlap, the smaller one is rendered on top of the larger, as determined by the areas of their respective (distorted) boundary polygons. This prevents large FEATURE from blanketing out smaller ones that could potentially be completely contained with the former's boundaries.

Mask composition: The final ID-buffer is obtained by *multiplicative blending* of the content and the ID mask. That is, each pixel component's (red, green, blue) intensity from the content mask is multiplied with the corresponding component's intensity in the ID mask. Multiplying any color with a black pixel (intensity value 0.0 for all components) thus results in a black pixel. White pixels (intensity 1.0) have no effect and are thus effectively transparent. After composition, the ID-buffer contains a pixel-accurate view of the distorted representation with three types of pixels: empty (i.e., fit for placing labels), non-empty non-FEATURE pixels (i.e., contextual information that should not be occluded by labels), and color-coded FEATURE pixels. Note that unlike the rendition of the FEATURE polygons (Figure 4.12b), the ID-buffer captures every empty space even within FEATURE regions at pixel granularity (Figure 4.12c).

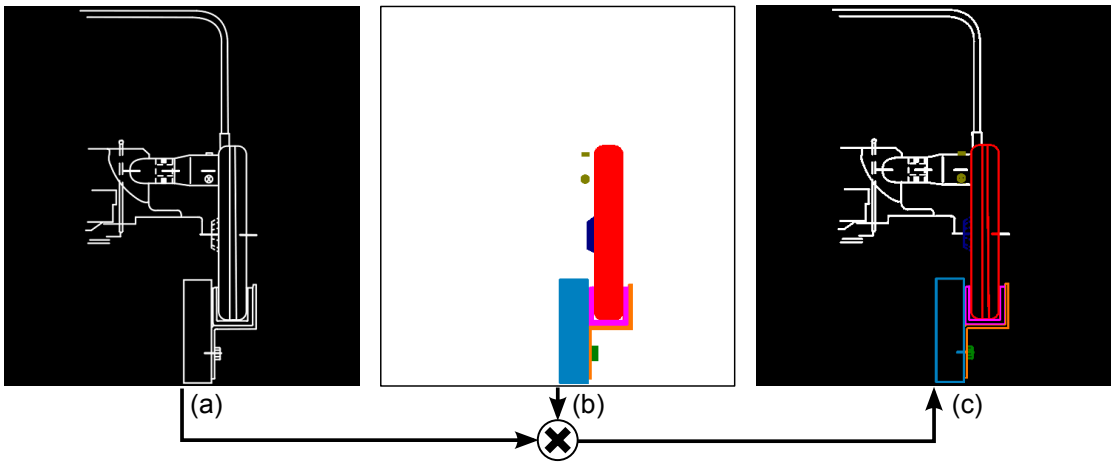


Figure 4.12: Creation of the ID-buffer for arbitrary PAGE content. A binary segmentation first distinguishes between all background pixels of a defined color and foreground (a). This mask is composed with a distorted color-coded rendition of the feature polygons (b) to obtain the final ID-buffer (c).

Space-efficient Label Layout. The basic idea behind the space-efficient layout is to use the distance transform encoded in the distance-buffer not only to compute anchor points and internal labels, but also to locate label positions the empty regions in the negative (free) space as close as possible to the labeled FEATURE regions. Labels are placed in a greedy manner, immediately updating the distance-buffer after each placement to reflect the consumption of free space. The labeling procedure thus comprises the following steps:

1. Select the most important task-related label not yet placed, in order of FEATURE relevance.
2. Determine the anchor point within the FEATURE region.
3. Find a rectangular region of free space closest to the anchor point that can accommodate the label.
4. Render the label into *both* the frame-buffer and the ID-buffer.
5. Update the distance information in the distance-buffer from the ID-buffer.
6. Repeat until all *task-related* labels have been placed (or the current label can not be placed due to space constraints).

For the first step, contrary to the original approach the static annotation table is replaced by a *task-specific table* that is dynamically generated from the task context information provided by the enriched task model. The set of task-related labels, their respective texts and importance (relevance) ordering may vary between basic tasks. In particular, the stop criterion (step 6) is determined by a relevance threshold t. Labels of FEATURES with $r < t$ are omitted from the annotation table.

A key factor of this approach is to efficiently update the distance information after each label has been placed. To facilitate this, the standard distance transformation algorithm is modified to use two distance-buffers, one for horizontal boundary-label distances (D_X-buffer) and one for vertical distances (D_Y-buffer), see Figure 4.13. Both fields are initialized from the ID-buffer (cf. Figure 4.12c).

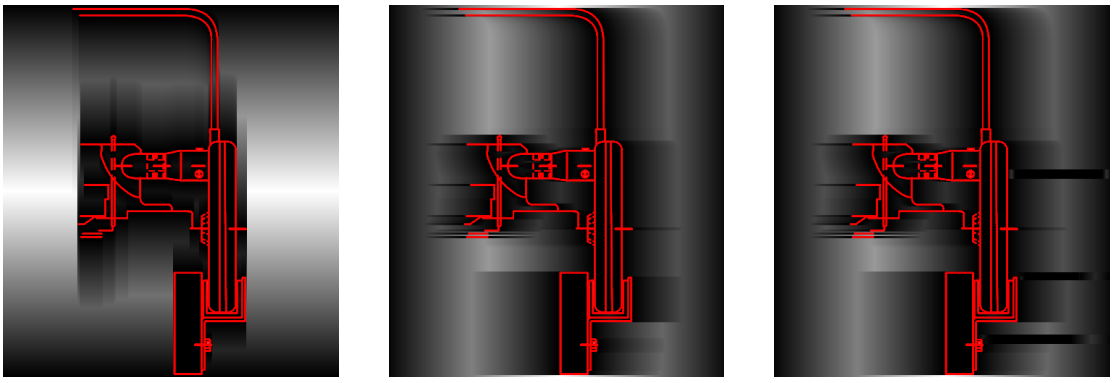


Figure 4.13: Two distance buffers are used for fast determination of space availability in (a) vertical and (b) horizontal direction. After each label placement, the distance buffers are updated to reflect the reduction of free space in the vicinity of placed labels (c – D_X after first three labels).

The benefit of this approach is two-fold. First, the search for a rectangular region of empty space that can accommodate a label with a bounding box of width w and height h can be separated into lookups for the closest region of empty space that is wide enough (using D_X) and high enough (using D_Y). Second, updating the distance information after a label has been placed can use smaller, thus faster kernels for distance propagation in the respective distance-buffer.

Figure 4.14 compares the labeling result form our proposed approach to a layout done using the *Left-and-Right* layout style proposed by [AHS05]. The latter relied on first placing the labeling strictly on the left and right sides of the graphical object and then bringing them close to the silhouette boundaries without taking into account the space that is available in the vicinity. Our new approach explicitly looks for locations closest to the objects where the corresponding labels could be placed. It thus makes efficient use of existing image space without increasing the illustration size, even if empty space is separated into small and irregular regions.

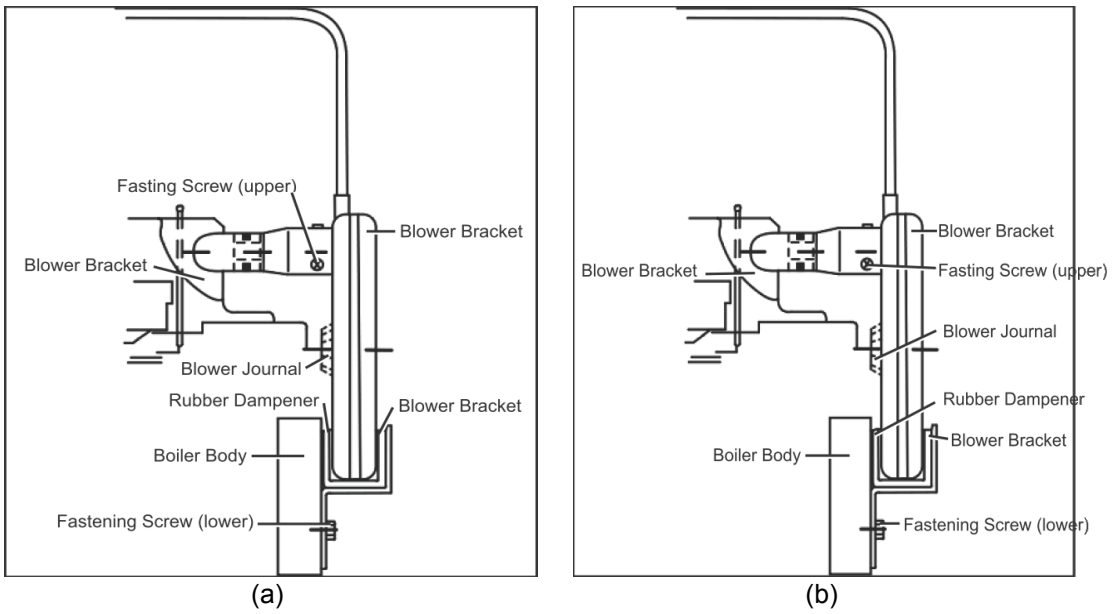


Figure 4.14: Comparison between space-efficient adaptive labeling (a) and the *Left-and-Right* layout from [AHS05] (b).

Remote Labeling. To support remote labeling for adapted visual representations using the extended labeling module, the concept of *layers* was adopted. The client first creates an adapted representation that does not contain any labels, by executing the adaptation pipeline up to and including the forth stage (image space manipulation). In particular, the resulting image includes relevance-driven distortion of FEATURE geometry that affects the amount of image space available within and in the vicinity of FEATURE regions. This image constitutes the *base layer* for the annotated visual representation.

The remote labeling service then generates a matching *annotation layer* that is com-

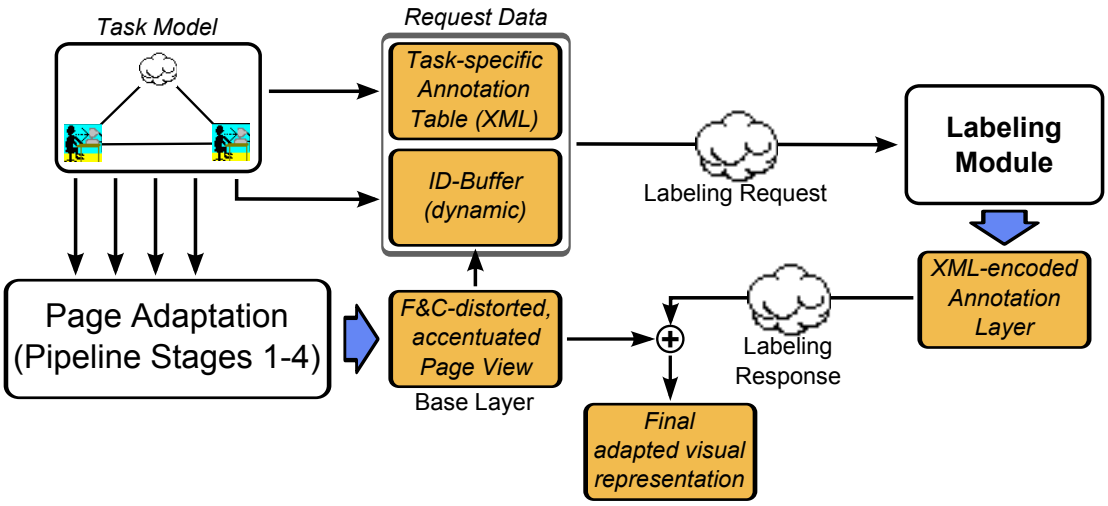


Figure 4.15: Schematic illustration of the remote labeling process.

posited over the base layer to obtain the final, annotated representation. To this end, the mobile client generates a *labeling request* that contains all information required to perform space-efficient labeling at server-side. The response comprises a description of the annotation layer containing all labels, their positions, as well as the layout of secondary elements (i.e., connecting lines). Figure 4.15 illustrates the general procedure.

Labeling Request: To generate a remote labeling request, the client first creates an ID-buffer as described above. In addition, it generates an annotation table from the information provided by the task context. The annotation table is an ordered list of FEATURES, whereas each list entry associates a FEATURE with its color-ID and the label text. List entries are ordered by FEATURE relevance values. In addition, the relevance value is used during labeling to adjust label attributes such as font face, size and weight to further emphasize the most important labels. This annotation table is encoded in XML format and transmitted to the labeling service together with the generated ID-buffer. Note the base layer is *not* transmitted as part of the request – since the labeling service is generating a separate annotation layer, it does not require a frame-buffer to actually render labels into.

Labeling Response: Rather, while placing labels using the space-efficient layout algorithm, only the distance buffers D_X, D_Y are updated. Instead of writing placed labels directly to the frame-buffer, the selected position for the label's bounding rectangle is recorded. After all labels have been placed the service returns an XML-encoded description of the label layout to the client. It contains positions of all label bounding boxes, label font information and the coordinates of anchor points and connecting lines This information is sufficient for the client to render an annotation layer that is superimposed over the adapted visual representation's base layer.

4.4 Summary

This chapter proposed several general methods for the task-based adaptation of raster images based on the general adaptation pipeline introduced in Section 3.3.4. It has been shown that the PAGE/FEATURE approach facilitates a wide range of **automatic, relevance-driven adaptation operations** even though the source raster image in itself does not contain enough information on content structure for fine-grained information or visual scalability. In particular, it has been shown how **existing display techniques can be integrated** to generate task-specific adapted representations utilizing the geometry information afforded by FEATURE boundaries.

Because the definition of FEATURES over raster images is highly dependent on the image contents, as well as the parametrization of techniques to achieve a specific communicative goal, the process of *content preparation* can be automated only partially. Thus the approach proposed here is to take the route of **manual content preparation, augmented by automatisms** where these are viable. To this end, suitable authoring tools are required. In this chapter, we briefly reviewed the **authoring tool for raster image preparation** that has been developed in the scope of this thesis. Once content preparation has been completed, however, task-specific initial views within the smart visual interface are generated automatically by Smart Raster Image techniques, as motivated in Chapter 3.

In addition, two novel "Smart-X" display techniques have been proposed that specifically address challenges related to compact mobile devices encountered e.g., in the maintenance scenario: a **belt-based Focus & Context scheme** for relevance-driven distortion of FEATURE regions [FRSF06], as well as a **space-efficient remote labeling** approach [FLH+06].

Since the general adaptation pipeline on its forth and fifth stage operates in image space regardless of the visual data type of the PAGE being adapted, the majority of approaches and techniques contemplated here for raster graphics are equally applicable to the fourth adaptation pipeline stage of the further data types discussed in the following chapters.

5 Smart Vector Graphics – Adaptation of Vector Graphics

Unlike raster images examined in the previous chapter, vector graphics build on 2D geometric primitives to represent image contents. Therefore vector formats do exhibit some inherent notion of content structure that can be utilized for adaptation. As primitives are defined by means of geometric coordinates that are independent of actual pixel resolutions, vector graphics can be scaled without loss of quality. This makes vector graphics especially attractive in mobile applications where display scalability is a primary concern (cf. Section 2.2).

Scalable Vector Graphics (SVG) is a modularized language for describing 2D vector graphics in XML by the WWW consortium [FFJ03]. SVG uses three types of graphic objects or *primitives*: geometric shapes (e.g., rectangles, circles, or paths), raster images, and text. Graphic primitives can be transformed, logically grouped, and styled by assigning presentation attributes like fill and stroke colors. SVG has a well-defined *rendering model*, which is mandatory for compliant user agents (UAs). To this end, groups and individual objects are organized in an *object hierarchy* controlling what styling attributes are applied and how graphical elements are transformed and composed during rendering.

SVG drawings can be interactive and dynamic. Animations can be defined and triggered either declaratively, i.e., by embedding SVG animation elements in SVG content, or via scripting. A rich set of event handlers can be assigned to any SVG graphic primitive. A supplemental scripting language further allows to modify the SVG Document Object Model (DOM),[1] which provides complete access to all elements, attributes and properties. This facilitates sophisticated applications based on SVG.

SVG has been explicitly designed to offer broad interoperability with other web technologies and standards. For example, Cascading Stylesheets (CSS) can be used to encode styling information. In particular, it offers extensive support for integration with domain-specific languages (DSLs) through XML namespaces [BHLT06] and linked resources, like stylesheets or embedded images (raster or SVG), based on the XML Linking Language (XLink) [DMO01].

Specifically in this thesis, the SVG standard is used exclusively. Therefore, in the following 'vector graphics' is used synonymously with SVG-encoded vector graphics.

[1] The Document Object Model of an XML file (like SVG) represents the XML markup elements as a hierarchical structure of *element nodes*. For SVG documents in particular, the DOM node tree reflects the SVG object hierarchy. For the purpose of the following discussion, the two terms are used synonymously.

5.1 Vector Graphics Content Preparation: Concepts and Authoring Tools

Despite its capabilities, SVG still is primarily a presentation format for (interactive) graphical content that offers good display scalability. As such, SVG groups are used to organize primitives that are subjected to the same transformations (e.g., moved together during animation) or that share visual attributes. Hierarchical grouping is rarely used to express containment with respect to application semantics e.g., *is-part-of* relations between depicted objects. This is due to the fact that containment is far less relevant in graphical design than composition and superposition of visual elements such as backgrounds, shadows and highlights.

An SVG graphics typically does not, therefore, describe semantic properties of the depicted content in a way useful for task-driven adaptation of the SVG's visual representation. For this reason, preparation of vector graphics is necessary to obtain PAGES with defined FEATURES that are relevant to the task at hand.

5.1.1 Definition of PAGES

Starting point for the definition of vector graphics PAGES is an SVG file describing a 2D image. The contained object hierarchy comprising graphical primitives and primitive groups is organized according to graphical criteria e.g., all background gradients or all lines with the same style.

The aim of PAGE preparation based on a given input file is then to re-organize the SVG object hierarchy to facilitate the definition of FEATURES in a way that

- allows to individually modify FEATURES both in terms of their geometry and their visual attributes – corresponding to the second and third stage of the adaptation pipeline (cf. Section 3.3.4) – and
- that the re-organized hierarchy results in a correct *default* visual representation (cf. Section 3.3.2) of the PAGE.

In the approach proposed here, the first aspect is addressed by comprising groups of primitives according to semantic rather than graphical criteria. This is discussed in detail in the following Section 5.1.2.

The second aspect must be considered because a hierarchy node's child order does carry semantics under the SVG "painter's model" of rendering. Paint[2] is applied to the output device in successive drawing operations. When the currently drawn area overlaps a previously painted area the new paint partially or completely obscures the old, depending on the paint's opacity (alpha value). The order in which drawing operations within child nodes are applied is defined implicitly by the order of the corresponding XML elements in the SVG DOM. In particular, if a node is itself a group, its child nodes

[2] A paint represents a way of putting color values onto the 'canvas' (i.e., the screen). A paint consists of both color values and associated alpha values which control the blending of colors against already existing color values on the canvas.

are recursively rendered onto a temporary canvas, which is then composed on top of previously painted elements.

In effect, this introduces an implicit Z-ordering (depth ordering) of graphic primitives depending on their position in the SVG hierarchy. This means that re-structuring the object hierarchy of the input SVG file must consider alterations of the implicit drawing order of primitives that can potentially introduce visual artifacts in the SVG's rendering. This constraint may conflict with a grouping of primitives solely according to their containment within FEATURES.

To address these issues, this thesis proposes a specific decomposition scheme for the SVG object hierarchy as follows. The main idea is to decompose the SVG content into *content fragments* according to adaptation requirements that provide the building blocks to define SVG-based PAGES. Furthermore, by composing fragments in a specific order defined by *content layers*, conflicting requirements regarding semantic and rendering aspects of the primitive hierarchy organization are resolved.

SVG Content Fragmentation distinguishes SVG elements according to their function and contribution to the PAGE's visual representation.

We here make a distinction between SVG groups and fragments. Groups are a means to *logically* organize graphic primitives for purposes of visual element composition, as detailed above. SVG fragments, on the other hand, are sub-trees of the DOM hierarchy constituting an SVG's internal structure. Just like groups can contain sub-groups, fragments can be composed of sub-fragments. In particular, a sub-fragment can also comprise another, self-contained SVG file that is referenced from the enclosing fragment (e.g., the primary SVG file) as an external resource. Fragments therefore represent a means to *structurally* organize graphic content for purposes of FEATURE-based adaptation.

To this end, the approach discussed here distinguishes SVG elements into three principal fragment types:

- The basic scaffolding that describes the structure of a minimal valid SVG file. This minimal base file is referred to as the SVG *skeleton*.
- Content elements that describe the image background, i.e., parts of the object hierarchy that does not contribute to any task-specific FEATURE.
- Content elements that constitute FEATURES.

The **SVG skeleton** forms the core of the PAGE definition. Each PAGE comprises a single fragment of this type. It includes the XML header, the <`svg`> element presenting the object hierarchy root, SVG meta-data elements as well as the root element of the section for file-global definitions (see [FFJ03] for details). All other content fragments are embedded into the skeleton to form the SVG object hierarchy. Thus in itself, the SVG skeleton simply describes an empty image without further graphical content.

Background content fragments comprise of graphic primitives that describe contextual or auxiliary visual elements such as image backdrops or reference grid lines.

Background fragments do not contribute to any task-relevant FEATURES. Thus conceptually, all non-FEATURE primitives constitute a single background fragment per PAGE. However, because of rendering order issues stated above, background primitives might be split into multiple fragments that are distributed across several content layers, as explained in the following paragraph.

FEATURE fragments are the last type of content fragment comprising those primitives that describe graphical content associated with task-specific FEATURES. Each FEATURE thereby corresponds to a content fragment that assorts primitives according to semantic criteria of the application domain. Within FEATURE fragments, graphic objects are further organized to facilitate task-specific adaptation of the respective FEATURE. This is reviewed in more detail in Section 5.1.2.

Figure 5.1 gives an illustrative example of how an SVG file is broken down into the principal content fragments constituting a vector graphics PAGE.

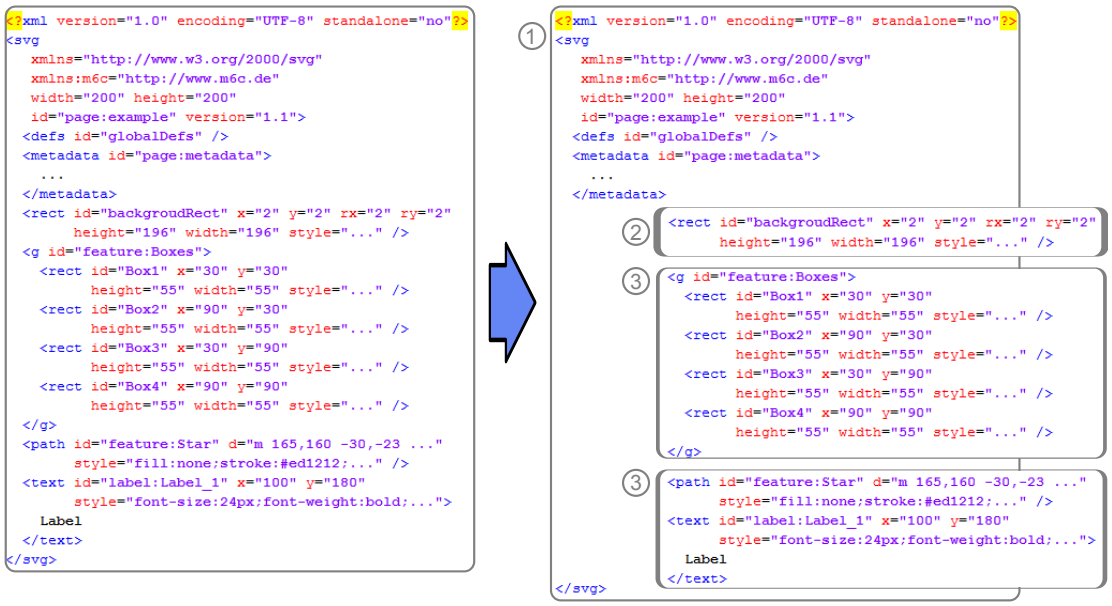


Figure 5.1: An input SVG file is conceptually distinguished into three principal fragment types according to their function/contribution to image contents: the basic scaffolding or skeleton (1), contextual background information (2), and FEATURES (3). Shown here is the SVG DOM for the example PAGE shown in Fig. 5.2.

Content Layers are used in the approach proposed here to address the necessity of maintaining the rendering order of graphical primitives for their correct composition. Layers are simply regular SVG group elements <g> that are inserted as direct child nodes of the SVG's hierarchy root (the <svg> element, cf. Figure 5.1). Because child nodes are rendered in the order they are defined in the XML DOM, an implicit depth ordering of content elements that are children of these top-level layer nodes is achieved.

Thus, content fragments are not added directly to the hierarchy root, but instead under the group element representing the respective layer the content is assigned to.

Figure 5.2 illustrates this concept. Layer nodes are rendered in reading order (i.e., from top to bottom row), therefore, the background layer is rendered first whereas subsequent layers (FEATURES) are composed on top of background elements. Labels are rendered last.

Figure 5.2a also shows our proposal for a general partition of content fragments into layers. The lowermost layer comprises background content. FEATURES are distinguished into 'opaque' and 'transparent' elements. A FEATURE is considered opaque if it contains area primitives that have a fill attribute (e.g., a solid color or pattern) assigned. When these primitives are composed on top of previously rendered elements they occlude parts of the current image and thus should be draw first. Contrary to this, transparent FEATURES contain only primitives with an assigned stroke style (line color, thickness). Composing these FEATURES over others does not incur noteworthy occlusion of content (cf. Figure 5.2b). Finally, a top-most labeling layer ensures that labels placed during the last adaptation pipeline stage are always composed over any other image content.

Within each layer, implicit depth ordering of FEATURES is controlled by adjusting the child order of FEATURE fragments. In particular, because layers are ordinary SVG group elements, the basic layer concept is trivially extended to include sub-layers, in case the layout of overlapping FEATURES requires it.

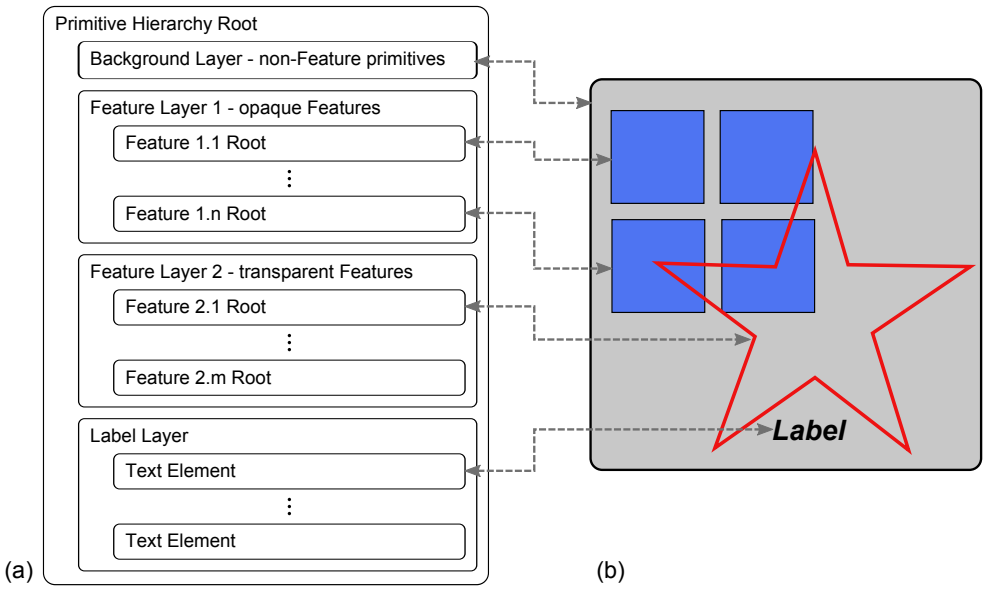


Figure 5.2: To avoid rendering order issues, SVG's hierarchical structure is utilized to define a layered description of image content by inserting top-level group elements (a). Background and FEATURE fragments are then assigned (inserted as child nodes) to corresponding layers (b).

The vector PAGE's SVG skeleton, extended to contain the top-level content layer groups, provides the basis for the integration of the content fragments defining individual FEATURES.

5.1.2 Definition of Features

Since vector graphics do have the notion of distinct graphic primitives, FEATURE definition can be carried out in object space. The previous section already stated the basic idea underlying FEATURE specification of this thesis' approach in particular: conceptually, each FEATURE comprises of a self-contained *content fragment* that is treated as a single coherent object for purposes of adaptation. Defining FEATURES on (SVG) vector graphics PAGES therefore comprises the following aspects:

- Identification of what parts of the SVG object hierarchy contribute to what FEATURES, i.e., how the object hierarchy of the input SVG file is partitioned into content fragments.
- Organization of graphic primitives within an identified FEATURE fragment.
- Determination how FEATURE fragments are integrated into the SVG skeleton.
- Annotation of task nodes in the task model with corresponding FEATURE associations.

Fragment Identification

It has been reasoned in Section 3.3.3 that FEATURES represent content elements of a PAGE that have an application-specific semantic meaning in the context of the task at hand. Therefore, what primitives in a vector graphic constitute a given FEATURE is dependent on the application domain, the task at hand and the image contents. The question thus is how the affiliation of SVG elements to FEATURE fragments is determined.

The identification of fragments is rarely automatable. It might be feasible in case the input SVG file has been created according to application-related criteria. For example, some CAD/CAM programs allow to export technical drawings in SVG format. Assuming the SVG's object hierarchy represents an application-specific breakdown of vector primitives into depicted objects, FEATURE fragments can be automatically identified by matching SVG group node identifiers with known object names used in the generating program. However, this presupposes that both the list of object names is available, as well as that matching identifiers are created in the exported SVG file.

Therefore, the normal case requires a manual organization of graphic primitives into hierarchical groups that represent FEATURE fragments. This necessitates tool support allowing a content author to interactively re-arrange (groups of) primitives in the SVG DOM tree, and to manually designate specific group nodes as the root of FEATURE fragments. An authoring tool that has been developed to this end in this thesis is introduced in Section 5.1.3.

Organization of feature fragments

The central point of FEATURE definition on vector graphics PAGES is the organization of graphic primitives in such a way that these represent application- and task-specific FEATURES. Besides the proper fragment identification, this comprises a secondary aspect: how the primitives are organized within the FEATURE fragment, i.e., the structure of the object sub-hierarchy below the fragment's root node. In this regard there are two principal approaches:

- the existing element structure of the input SVG file is retained, i.e., sub-hierarchies are adopted without modification as FEATURE fragments, or
- below the fragment root a further re-organization of the primitive hierarchy is conducted.

The latter can in particular be used to support additional FEATURE-specific adaptation operations. To this end, in the present thesis the concept of *sub-fragmentation* has been developed. The basic concept is discussed in the following. Section 5.2 presents how this approach is utilized during the geometry and visual attribute stages of the adaptation pipeline.

FEATURE sub-fragmentation. Core idea of sub-fragmentation is to further sub-divide the object hierarchy within a FEATURE fragment according to the function of objects with respect to the description of a vector graphics. To this end, the following principal types of sub-fragments can be distinguished:

- Geometry, i.e., SVG elements that define graphic primitives,
- visual attributes, i.e., SVG elements describing styling information that are applied to geometry, and
- additional elements that define the dynamic and interaction behavior of the vector graphics.

Individual graphic primitives (geometric shapes, raster images and text) and object groups – composed of multiple primitives, subgroups and transformations – make up the first type of sub-fragments defining a FEATURE's geometry data.

Visual attributes such as fill colors, line strokes and colors gradients constitute the second sub-fragment type. The approach presented here allows for two sub-fragments of visual attributes: styling properties such as line strokes and solid colors that are encoded in CSS notation, as well as complex SVG paints (i.e., gradient and fill pattern definitions) that must be specified using SVG syntax.

The last fragment type comprises all scripts that enhance the SVG image with interaction functionality (e.g., in reaction to mouse input) and dynamic animations. Because these elements are executed by the SVG user agent (rendering agent), they are not considered during adaptation.

This conceptual primary fragment decomposition is illustrated in Figure 5.3, left. Its purpose is to facilitate the adaptation of FEATURES in two separate steps – geometry as well as visual attributes – according to the general adaptation pipeline introduced in Section 3.3.4. This primary decomposition into sub-fragments is effected solely according to the element types defined by SVG. It can therefore be performed automatically.

In addition, the approach presented here utilizes a further, fine-granular decomposition of a FEATURE fragment into multiple sub-fragments per fragment type. This secondary decomposition allows to select between different Levels of Detail (LoD) during the geometry adaptation stage of the pipeline, respectively of different quality levels of texturing (visual appearance) during the visual attribute adaptation stage, cf. center column of Figure 5.3

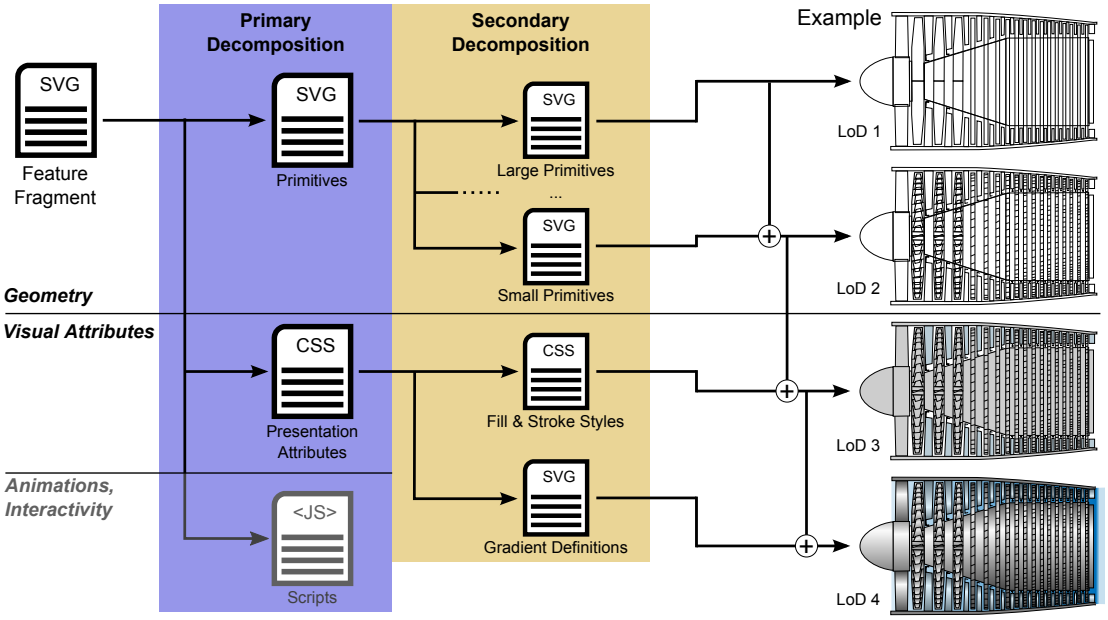


Figure 5.3: General scheme for the division of a FEATURE fragment into sub-fragments to facilitate LoD on FEATURES. The right column shows an example of a jet engine compressor divided into four sub-fragments: two geometry LoD, two texturing levels.

The right column in Figure 5.3 illustrates this for a technical illustration where two geometric LoD have been defined: shapes defining the outlines of major components were grouped into a sub-fragment constituting the lowest LoD; smaller primitives in a second geometry sub-fragment provide structural detail as the next LoD. For other PAGES and tasks a breakdown into even more fragments for a corresponding number of geometry LoD may be appropriate.

The proposed two-way splitting of styling information then provides three further quality levels regarding visual appearance: no visual styling at all for unimportant or background objects; flat colors and line width for a basic accentuation of relevant FEATURES; as well as supplemental fill gradients and patterns which provide additional information on the depicted object. Especially gradients are often used to create a 'pseudo-3D' appearance of objects e.g., curvature, and as a means to hint at surface material (Figure 5.3, bottom right).

The secondary decomposition of geometry data into sub-fragments can be performed automatically according to preassigned criteria. The strategy employed in the present thesis in particular is to first sort primitives according to their size. The ordered list is then partitioned evenly into a number of sub-fragments corresponding to the desired number of geometry LoD. The reasoning behind this is that primitives with a smaller extend usually contribute local details compared to large primitives constituting major image features.

This notwithstanding, it is nonetheless expedient to provide content authors a means to manually perform the partition of graphic primitives into LoD with the help of a suitable tool. This ensures appropriate detail levels can be defined corresponding to the communicative requirements of the task at hand, which may not always be obtained by automatic means.

How the concept of sub-fragmentation of FEATURES introduced here is utilized during the adaptation process is subject of Section 5.2.

Skeleton-Fragment Integration

The SVG skeleton provides the base node hierarchy into which the identified FEATURE fragments are embedded. More specifically, FEATURES are integrated by appending the corresponding content fragment's root node as either

- a child of the group node corresponding to a content layer, thus defining a top-level FEATURE (cf. Figure 5.2), or
- as the child of another FEATURE fragment, thereby creating a hierarchy of sub-features. This can be utilized to express *is-part-of* relations between FEATURES that are common e.g., in technical applications such as machinery assemblies, cf. Figure 5.4.

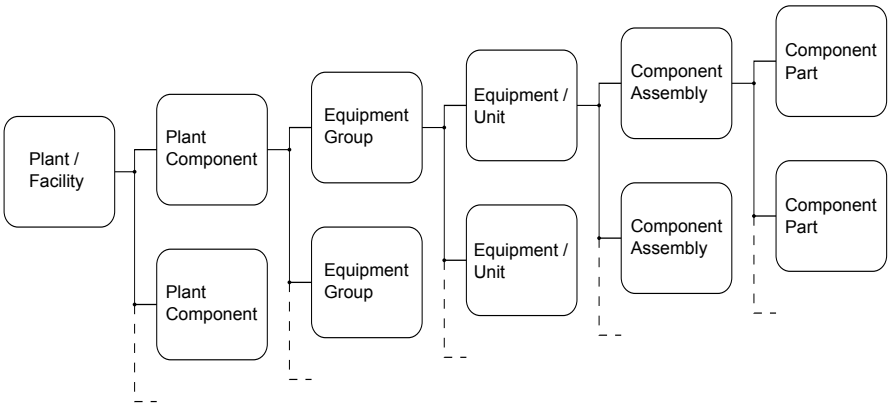


Figure 5.4: Within each content layer (cf. Fig. 5.2) a hierarchical organization of FEATURE fragments can be used to express *is-part-of* relations e.g., between components of a mechanical assembly.

To this end, SVG provides two ways to append a FEATURE fragment (i.e., a sub-hierarchy of nodes) as a child node:

- *directly*, i.e., the FEATURE root node is a regular group node (with sub-nodes) in the object tree of the SVG skeleton, as well as
- *indirectly* as a reference to a linked resource.

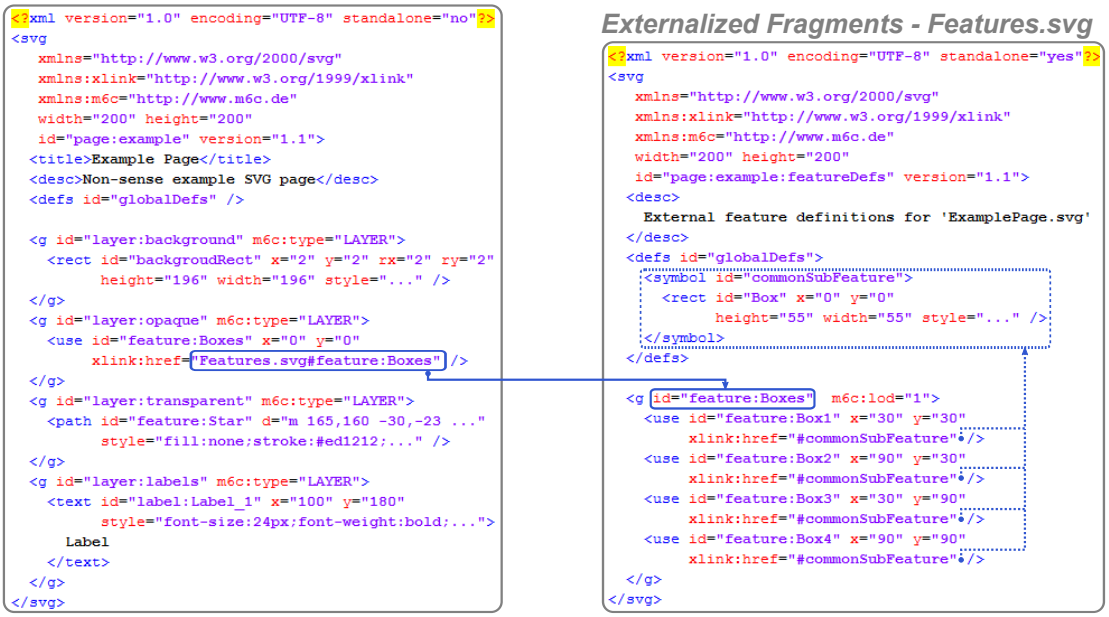


Figure 5.5: Example of a fully specified PAGE with one background fragment and two FEATURES based on the SVG file shown in Figs. 5.1, 5.2. Note the addition of top-level groups functioning as content layers. FEATURE fragments are identified by a special attribute. Moreover, fragments may be externalized to facilitate task-driven exchange of FEATURE representations.

Direct embedding is the standard approach in order to obtain a self-contained description of a PAGE that comprises only a single SVG file.

For the purpose of indirect embedding of content fragments, SVG provides a reference element (<use>) that contains an XLink pointer [DMO01] to the target resource. There are two types of linked resources: so-called symbols and external resources. Symbols are object groups that are defined at at global scope in the SVG file (specifically, the definition section in the skeleton, cf. Figure 5.5). External resources reference content fragments in external files.

Using indirect embedding of FEATURE fragments offers a distinct advantage that the target of the reference element may be modified as part of the adaptation process. In particular, it facilitates task-specific switching between independent visual representations of the same FEATURE, for example, a detailed versus an abstracted symbolic representation. Section 5.2 discusses how this mechanism is utilized in combination with the sub-fragmentation concept for a Level of Detail approach on vector graphics FEATURES. Furthermore, Section 5.3.2 introduces a novel "Smart-X" technique that exploits this mechanism for the task-driven adaptation of circuit schematics.

Task Node Annotation

A PAGE provides a structured description of the image content by means of the SVG object hierarchy. To associate FEATURES with specific tasks the respective FEATURE fragment in the SVG object tree must be linked with the corresponding task node.

SVG mandates that every DOM element has an element identifier, including primitives and group nodes. This ID attribute is a string value that must be unique across the SVG DOM tree, but is not constrained otherwise with respect to how it is formated. This allows for application-dependent formating schemes conveying auxiliary information.

In this thesis' approach in particular, the element identifier of the FEATURE fragment's root node is used to identify the respective FEATURE. For this purpose, FEATURE root elements are assigned an identifier attribute according to a specific pattern – here, by prepending the prefix "`feature:<ID>`", cf. Figure 5.5. Then, the FEATURE association with a task node is established by referring to `<ID>` from the task node annotation as `<feature id='<ID>'/>`.

5.1.3 Feature Relevance Values

The FEATURES that have been defined and associated with task model nodes in the previous content preparation step are then assigned relevance values for each relevant basic task. This, too, is by necessity a manual authoring step: as argued in Section 4.1.3, task-specific FEATURE relevances can *only* be determined manually for the same reasons that establishing the domain-specific task model itself is a manual design task.

SVG Authoring Tool

Thus, similar to the MaTE authoring tool developed for raster image PAGES (cf. p. 68) a corresponding tool MaTE-SVG has been developed to aid the creation of SVG-based vector graphics PAGES. It shares many common functionality with its raster counterpart. In particular, MaTE-SVG allows a human content author to interactively

- create and edit PAGE definitions,
- interactively define content fragments that constitute FEATURES of the given PAGE,
- annotation of CTT task model nodes with created PAGE/FEATURE definitions, and
- interactive specification of task-specific FEATURE relevance values as well as additional FEATURE attributes utilized during adaption (see Section 5.2).

Note that MaTE-SVG is not a fully-fledged design tool for vector graphics. There are several comprehensive SVG design software suites already available, such as Inkscape.[3] Duplicating the functional range of such programs would neither have been reasonable nor feasible within the scope of this thesis. Instead, the aim of our vector authoring tool concentrates on two aspects: (i) to enrich existing SVG files with the necessary information for FEATURE-based adaptation, and (ii) to provide means for enriching the general task model with SVG-based PAGE definitions. Sophisticated editing of SVG content beyond simple re-ordering of DOM nodes to define FEATURE fragments (see below) is carried out using one of the available design suites.

Like its raster image counterpart, the SVG authoring tool is an Eclipse RCP plug-in, which facilitates interoperability with other tools from the model-driven design tool chain, the CTT editing tool in particular. Figure 5.6 shows a screen shot of the MaTE-SVG authoring tool.

[3] `www.inkscape.org`

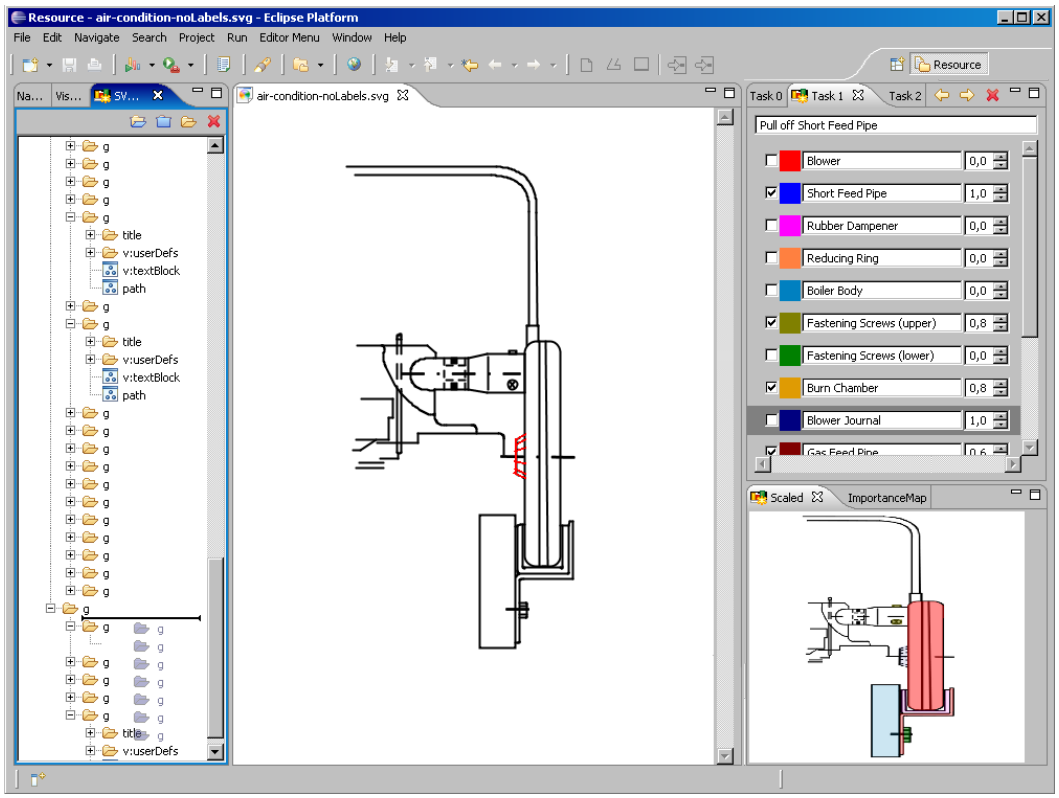


Figure 5.6: The MaTE authoring tool for vector graphics PAGE preparation. The author can interactively select (center pane) and re-arrange (left pane) primitive groups in the hierarchy to create content fragments that define individual FEATURES, associate them with task nodes and assigned task-specific relevance values and attributes for the adaptation process with previews (right panes).

Creating and editing of PAGE definitions. Creation of PAGE definitions utilizes the inherent extensibility of the SVG format. A new PAGE is created simply by opening an SVG file. PAGE meta-data is then embedded directly into the SVG skeleton as custom XML elements.[4] In particular, SVG already allows to specify an ID attribute – which is used as the PAGE identifier –, a human-readable title, as well as a short description of the depicted graphics. These properties are utilized directly.

Because additional information pertaining to individual FEATURES is appended directly to the corresponding SVG element that constitutes the FEATURE root (cf. Section 5.1.2), a PAGE definition itself does not require to provide further information.

Interactive definition of FEATURES. As described in Section 5.1.2, the concept behind FEATURES in SVG PAGES is to re-structure content elements into content according to semantic criteria. The authoring tool supports this in the following ways:

[4] XML – and therefore, SVG – uses the concept of *namespaces* [BHLT06] (i.e., unique name qualifiers) to identify XML elements. Extension works by defining extra elements in an own namespace, which are simply ignored by SVG user agents that do not know how to interpret them.

- New SVG group nodes can be created in the object hierarchy. This allows to define new top-level groups i.e., layers (cf. Section 5.1.1) as well as groups that form the root element of a FEATURE fragment.

- Specific elements can be designated as the root of a FEATURE fragment. To this end, it is selected from either the node hierarchy or by selecting a contained primitive from the visual representation. Internally, a designated group element is identified as FEATURE root by a custom element attribute (cf. . Figure 5.5). Designating a FEATURE root automatically adds it to the list of FEATURES available for the PAGE.

- Nodes in the SVG object hierarchy can be interactively re-organized. For this, MaTE allows to assign nodes to a new parent by drag-and-drop gestures (see Figure 5.6, left). In particular, this enables an author to assign primitives and primitive groups to FEATURE roots, and to assign FEATURES to the appropriate layers. Moreover, composite FEATURES can be created by assigning sub-FEATURES. This allows to express hierarchical is-part-of relations, cf. Section 3.3.3.

PAGE/FEATURE association with tasks. Like its raster image counterpart, the SVG content authoring tool itself is not designed to create or structurally modify CTT task models. Instead, it is used to import existing models stored in XML format to obtain a a list of tasks that can then be associated with SVG PAGES.

Thus to annotate a task node with a PAGE definition, it is sufficient to provide a link to the enriched SVG file. Association of FEATURE fragments with task nodes utilizes the fact that every SVG element has a mandatory, unique identifier attribute: a FEATURE is associated with a task node by referring to the identifier of the fragment root element.

Specification of task-specific FEATURE properties. This working step primarily comprises the interactive definition of FEATURE relevance values. In addition, for each FEATURE two auxiliary attributes are currently supported. The first is a short text used during labeling. A specific highlighting color can also be assigned that is used during the visual attribute adaptation phase (cf. Section 5.2). In addition, Section 5.2 discusses how FEATURES can further be associated with transformation data to facilitate task-specific complex geometric adaption.

After annotation of task nodes of the imported CTT model with these data, it is saved back in XML format for further processing in the course of the superordinate UI design process.

5.2 Adaptation Control

The graphical content preparation results in an enriched task model that contains all information necessary to control the adaptation process of vector graphics PAGES. In order to tailor task-specific initial representations, adaptation operations of the general adaptation pipeline are mapped onto concrete techniques for vector graphics.

Because SVG vector graphics are a declarative language for describing visual representations, the fundamental approach to adaptation of vector graphics is to directly modify the graphic elements declarations (i.e., the SVG DOM, its elements and attributes) accordingly. The interpretation of the modified SVG by the rendering agent then results in a partially adapted visual representation. The subsequent image space pipeline stages then operate on the raster image that results from the SVG rendering process, cf. Figure 3.4 on p. 50.

This section discusses the operations that are thus viable on the different pipeline stages specifically for SVG. This includes a review of additional information that is annotated to leaf task nodes of the enriched task model to effect these operations.

5.2.1 View selection

The goal of view selection is to determine a rectangular section of of the vector graphics image that represents the portion of the PAGE initially visible. This selection defaults to the entire image. As reasoned in Section 4.2.1, this section has to be adjusted: a PAGE associated with a compound task (i.e., the root of task node subtree) is shown as the initial view of several working steps, each with different view requirements. Thus, adaptation for the respective working step's visual representation must be based on an appropriate sub-region of the vector graphics.

To this end, the exact same considerations apply to two-dimensional vector graphics as do to two-dimensional raster images. To recap briefly, the task is to (i) determine the visible rectangular *source region* from the PAGE's base graphics, and (ii) to specify a strategy for fitting the source region into the *target viewport* that is defined by the available display space on the output device. The latter is of importance when the aspect ratios of the source region and target viewport do not match. For a detailed discussion on both aspects of the view selection problem refer to Section 4.2.1.

The only difference between raster images and vector graphics is the coordinate system used for the specification of the source region. For raster images, it is always specified in pixel coordinates in image space. Contrary to this, vector graphics specify image content by geometric primitives with coordinates in $\mathbb{R}^2$. Therefore, source region selection is relative to the SVG skeleton's coordinate system.[5]

SVG directly supports the selection of a rectangular source region by means of the `viewBox` attribute that can be specified on the <`svg`> root element element. The attribute value is a quadruple $x_{min}, y_{min}, width, height$ given in user coordinate space, with (x_{min}, y_{min}) the upper left corner and $width$ and $height$ the source region's extend to the right and down, respectively. In addition, the `preserveAspectRatio` attribute controls how this source region is scaled to fit the target viewport – in particular, whether or not non-uniform scaling is permissible, and how to align the source region in the viewport. See [FFJ03] for a list of supported fitting strategies.

Therefore to effect view selection on SVG vector graphics PAGES, the corresponding information from the task node annotation, as introduced in Section 4.2.1, are applied as

[5] SVG supports a number of unit identifiers for its coordinate systems, such as absolute "user units" (dimensionless), pixels (px), centimeters (cm) and points (pt); as well as relative units such as percent (%) and font size-relative (em).

the attribute values of the `viewBox` and `preserveAspectRatio` attribute, respectively.

It is worth mentioning that this attribute pair is also supported on the <use> element that allows to indirectly (via XLink) reference child fragments. In particular, this facilitates the selection of a source region on FEATURE fragments that are embedded indirectly via <use>.

5.2.2 Geometry adaptation

With regards to the adaptation of vector graphics, the geometry adaptation step serves addresses three aspects:

- (I) The main purpose is to enlarge important FEATURES at the expense of less relevant ones that are scaled down, as reasoned in Section 3.3.4.

Furthermore, because vector graphics are comprised of distinct primitive objects that can be individually transformed geometrically, two additional adaptation operations are feasible:

- (II) FEATURES are *repositioned* (translated) from their original coordinates, and
- (III) the *selection of a geometric level of detail* (LoD) to match the FEATURE's modified presentation size.

(I) Relevance-driven Size Adjustment

Adjusting FEATURE sizes amounts to uniformly scaling all graphic primitives contained in the FEATURE fragment. This is effected by providing a corresponding transformation attribute on the root element of the fragment hierarchy that describes a transformation matrix $\left(\begin{smallmatrix} s & 0 \\ 0 & s \end{smallmatrix}\right)$. In doing so, the scaling operation is parameterized by the FEATURE's relevance value according to a mapping function $r_i \rightarrow s$. See Figure 5.7a for examples of mapping functions that are typically useful. Thus for $s < 1$ a given FEATURE is enlarged, and scaled down for $s < 1$. At the extreme end is assigning a scaling factor of $s = 0$, effectively removing the FEATURE from the adapted representation altogether. Thereby information hiding of irrelevant FEATURES is achieved.

(II) Feature Repositioning

Translation of FEATURES serves two purposes as a geometry adaptation operation. It is either

- employed to displace adjacent FEATURES to *maintain relative positions* of FEATURES after resizing – especially, enlargement – operations, or
- an *explicit translation* used to spatially separate important FEATURES from surrounding objects.

Maintaining relative positions. Resizing individual FEATURES according to their respective relevance values may necessitate to simultaneously alter their relative positions. Enlarging FEATURES may cause overlaps, while downsizing may result in gaps

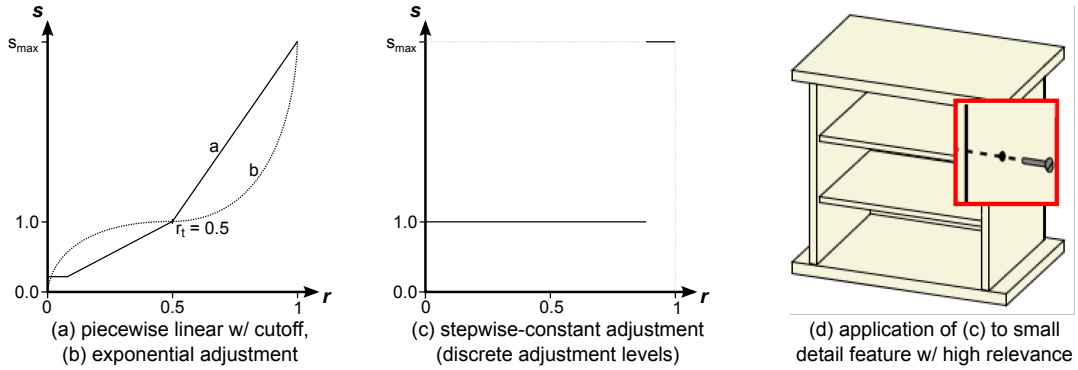


Figure 5.7: Examples for continuous mapping functions that assign SVG scale parameter values according to FEATURE relevance (a). Specific FEATURES are assigned stepwise functions (b). This is especially useful to highlight important but very small details that require large scaling factors (c).

between primitives that were adjacent in the undistorted image. This may be acceptable e.g., if the enlargement is employed to generate some form of inset detail view (cf. Figure 5.7c). Usually, however, these effect are undesirable since they distort the original spatial relations between depicted object(s). Thus to avoid or at least mitigate these artifacts, in the face of size changes adjacent FEATURE geometry must also be translated. Specifically, enlarging an important FEATURE necessitates displacing ('pushing outwards') other objects in its immediate neighborhood. To this end, two cases can be distinguished:

Resizing FEATURES comprising sub-features: SVG content fragments are defined in the coordinate system of their parent node. This means that if a composite FEATURE is geometrically transformed, the local coordinate systems of contained sub-FEATURES are implicitly transformed accordingly. Therefore, sub-FEATURES utilize the FEATURE hierarchy to automatically retain their correct relative positions and sizes.

Resizing FEATURES with siblings on the same hierarchy level: here, the SVG rendering model makes no assumptions on the spatial relationship of SVG fragments. This means when resizing a particular FEATURE, the relative positioning of sibling FEATURES on the same hierarchy level must be considered explicitly.

Because the direction and magnitude of the required translations in the second case depends on the relative position, relevance values and mapped scaling factors of the affected FEATURES, their manual specification is only feasible for simple vector graphics. Instead, the present approach derives the appropriate transformation automatically by utilizing connectivity information between (groups of) primitives.

To this end, task nodes are annotated with a set of point constraints (called *anchor points*) that reflect which FEATURES and primitive groups should maintain their relative positions.[6] Here, a point constraint is a one-way constraint comprising an *originating*

[6] Typically, these constraints defined once in the same scope (task node) as the PAGE itself and inherited to all sub-tasks. On principle, however, a task-specific set of constraints may be defined for any task node.

and *destination* FEATURE as well as the shared *anchor point* in the coordinate system of the originating FEATURE. The shared point can be arbitrarily chosen e.g., a shared primitive vertex, a point on a shared line, or a corner of the FEATURE's bounding box (Figure 5.8a).

The set of constraints define a directed graph that determines how scaling and translations of one FEATURE transitively affects connected objects (Figure 5.8b): as the originating FEATURE is scaled and translated by a compound transformation T, the coordinates of the shared point $\mathbf{p} = \binom{x}{y}$ is transformed to $\mathbf{p}' = T \cdot \mathbf{p}$. The resulting displacement vector $\mathbf{d} = \mathbf{p}' - \mathbf{p}$ thus determines the translation that has to be applied to the destination FEATURE (Figure 5.8c).

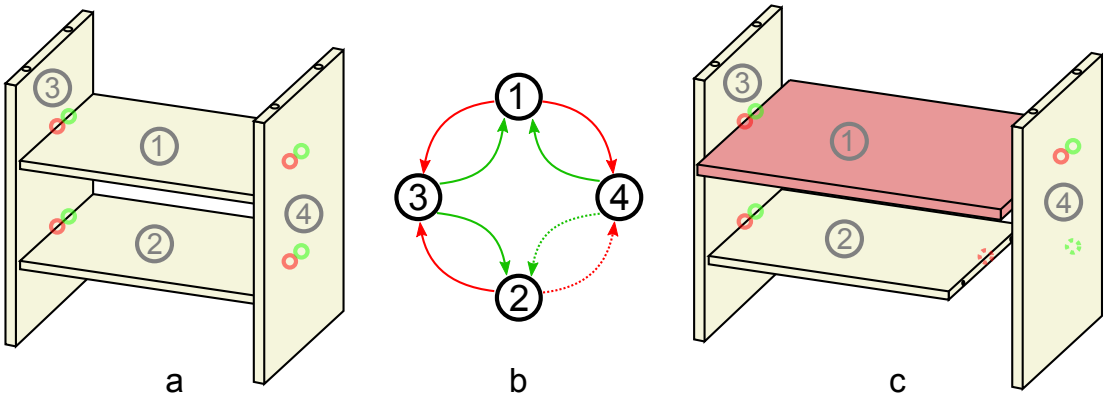


Figure 5.8: Illustration of transitive adjustment of FEATURE positions in reaction to scaling. One-way point constraints (a) define a connectivity graph (b). Enlarging a FEATURE implicitly translates connected FEATURES, whereby conflicting constraints may need to be removed (c).

Two constraints may contradict each other if the connectivity graph contains cycles. To resolve these, graph edges are weighted with the *minimum* relevance of the two FEATURES it connects. This allows to remove edges from the connectivity graph (i.e., ignore conflicting constraints) in order of increasing FEATURE relevance starting from the least relevant one. Figure 5.8c shows an example where two constraints (indicated as dashed in sub-figures b, c) have been ignored to achieve conflict-free translations of FEATURES transitively connected to an enlarged FEATURE.
The approach presented here is similar in objective to Constraint SVG (CSVG) [BTM+01] that provides a set of layout constraints for SVG-encoded diagrams using the Cassowary constraint solver [BB98]. CSVG is however designed primarily towards the dynamic *presentation layout* of diagram representations on different networked client output devices, i.e., primarily addresses the WHERE aspect. This is reflected in the set of available constraint rules (see [BTM+01]). By contrast, the constraints of the present approach capture *semantic aspects* of the task at hand based on FEATURE relevance; i.e., according to WHY aspects which are not represented in CSVG. Thus, the approach discussed here can be understood as a task-driven extension on top of CSVG as a potential concrete "Smart-X" technique.

Explicit translations. The above approach to automatically maintain relative FEATURE positions during relevance-based resizing covers a wide range of situations. However, some tasks may require deliberate positioning of specific FEATURES as an integral part of their visual representation's message subject. It is therefore reasonable to allow content authors to manually specify the translation or position of FEATURES *explicitly*.

For this purpose, task nodes are annotated with an absolute displacement vector $\mathbf{d_i} = \binom{d_x}{d_y}$ that is applied as a transform operation to the root element of the respective FEATURE ϕ_i. Thereby an explicit translation takes precedence over automatic repositioning. That is, an explicitly moved FEATURE affects constraints where it is the source (i.e., outgoing constraints), but itself ignores constraints where it is the target.

The definition of explicit FEATURE translation vectors offers a simple means to alter the composition of depicted objects in the visual representation for individual working steps. This can be used to spatially separate specific FEATURES as a means of emphasis.

Spatial separation is also commonly used in so-called 'exploded view diagrams' to communicate structural relationships between object components (e.g., mechanical assemblies). To this end, in Section 5.3.1 a novel approach to the semi-automatic generation of task-specific exploded view diagrams that utilizes hierarchical FEATURES and relevance information from the task model.

(III) Selection of Geometric LoD

The core idea of relevance-driven scaling of FEATURES is to assign more display space to important FEATURES than to less relevant ones. To fully utilize this change in presentation size (i.e., the visual scale) the level of detail of the respective FEATURE must also be adjusted (i.e., its information scale).

With respect to vector graphics, this adjustment can be effected on the level of geometric detail. In particular, important FEATURES can be rendered with full geometric detail including primitives with small extend that contribute local detail. Less relevant FEATURES, on the other hand, are represented by few large primitives i.e., those constituting the FEATURE's outline and general shape, and omitting small detail elements. This can be achieved in one of two ways:

- *selection* of an appropriate FEATURE representations from a set of *alternate* content fragments, or
- *composing* the FEATURE's geometry from a number of *progressive* sub-fragments.

The first approach works as follows: for a given FEATURE, several independent representations are generated at different levels of (geometric) abstraction. For example, curved paths may be replaced by simpler polylines in a simplified version. Then during adaptation, the default FEATURE fragment is *swapped* in its entirety with an alternative instance depending on the current FEATURE relevance. Swapping thereby works by modifying the target attribute of the reference element constituting the FEATURE root, as explained in Section 5.1.2. To this end, a stepwise-constant relevance mapping function is defined that maps relevance values to labels that identify the respective alternative fragment.

A considerable drawback to this approach is that it requires to have available or to create a suitable number of alternate representations for each FEATURE. It is thus feasible mainly for applications that use highly formalized visual encodings based on a fixed set of symbols, such as technical diagrams (cf. Section 5.3.2). This requirement does, however, significantly increase the amount of manual labor necessary during vector content preparation for general illustrations of any complexity.

For this reason, the present thesis pursues a novel approach to compose a FEATURE's geometric LoD from a *combination of progressive sub-fragments* instead. The key component to this approach has already been introduced in Section 5.1.2: a FEATURE's primitive hierarchy is partitioned into multiple primitive groups according to their relative sizes. The sub-fragment containing the largest primitives constitutes the lowest LoD and is always rendered. Including subsequent sub-fragments progressively amends this coarse base representation with further geometric detail (see Figure 5.3).

To this end, a stepwise-constant mapping function is defined that maps relevance values r to discrete LoD n_{lod}. Sub-fragments are assigned an attribute value $contrib_{lod}$ that identifies the LoD the respective sub-fragments contributes to. Then, depending on the effective detail level n_{lod} all sub-fragments that contribute to LoD $contrib_{lod} > n_{lod}$ are removed[7] from the PAGE's visual representation.

Because the sub-fragmentation of FEATURE fragments is automatable as explained in Section 5.1.2, the geometric LoD proposed here can be applied to arbitrary illustrations with minimal content preparation overhead.

5.2.3 Visual attribute modification

The aim of this pipeline stage is to accentuate relevant aspects of the visual representation by creating a visual hierarchy of FEATURES. To this end, the following adaptation operations are feasible on vector graphics:

- Adjustment of line width and stroke styles (e.g., solid vs. dashed lines), to indicate the relevance of objects (Figure 5.9a).
- Changing the color hue of strokes (outlines) and fills (internal areas) of primitives to specifically highlight the respective FEATURE (Figure 5.9b).
- Modification of color saturation or brightness of either stroke and fill of primitives to alter the general visual saliency of objects (Figure 5.9b).
- Modification of primitive transparency to both alter its visual saliency as well as a means to create 'ghosted views' [Vio05] that expose occluded FEATURE geometry (Figure 5.9c).

To parameterize these operations, FEATURE relevance values r_i a mapped to corresponding adjustment factors for the respective visual attribute $a_{attribute}$ using a suitable

[7] This can be effected by means of setting the fragment root node's `visible` attribute to '`hidden`'. This will prompt the SVG user agent to skip rendering the respective fragment.

mapping function. Depending on the attribute type functions either map relevance values to a range of values e.g., saturation, brightness, and line width; or assign labels corresponding to a discrete parameter setting e.g., stoke styles. See Section 4.2.4 (Figure 4.8) for a discussion of suitable mapping function for different attributes.

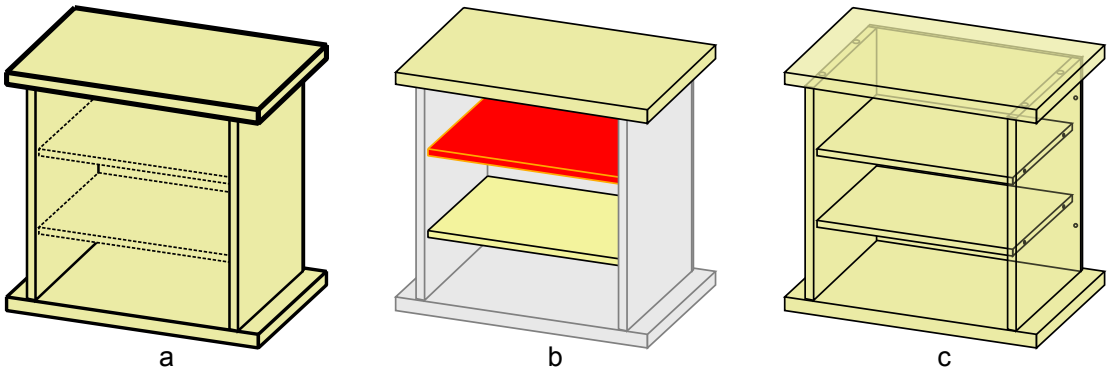


Figure 5.9: Illustrative examples of visual attribute adaptations: stroke style and thickness (a) as well as fill and line color adjustment (b) to accentuate FEATURES; transparency to expose occluded geometry (c).

These attribute modifications are effected by adjusting the corresponding presentation parameters of the corresponding SVG elements in the DOM. In particular, because presentation parameters are inherited to nested child nodes unless defined locally, visual attributes of FEATURES are defined and modified in the FEATURE's root element.

Besides these general attribute modifications, the present adaptation approach further utilizes the concept of FEATURE sub-fragments for visual attribute modification. Recall from Section 5.1.2 that SVG presentation information for each FEATURE fragment are cumulated into two sub-fragments: simple styles, as well as gradients and patterns. This allows to select from three distinct 'texturing quality' levels for each FEATURE according to its current relevance value:

- The lowest level is to apply no styling at all i.e., the FEATURE geometry is rendered only as outlines with a default color, usually black.
- On the second level, specifications of solid stroke and fill colors as well as line styles and thickness are associated with FEATURE primitives.
- On the last level, additional gradient and pattern definitions (e.g., hatching) are associated with FEATURE geometry.

Thereby these quality levels successively build on top of each other. The are selected in the same way as described for the geometric LoD selection above: FEATURE relevance is mapped to a desired quality level that determines which styling sub-fragments need to be assigned to the FEATURE root.

The presented approach to distinct texturing quality levels can on principle be combined with the general visual attribute modification operations: the selected quality

level determines what styling information is available, whereas further style parameter adjustment fine-tunes the applied styles according to FEATURE relevance.

A point to note is that following the conceptual anatomy of a vector graphics PAGE, image background constitutes a content fragment on its own (cf. Section 5.1.1). Therefore, visual attributes of background objects can be modified following the same procedures just outlined for FEATURE fragments. In particular, visually accentuating FEATURES while simultaneously de-emphasizing contextual background objects are complementary operations, as reducing the visual saliency of the latter causes foreground objects stand out at the same time.

5.2.4 View Space Manipulation

This stage of the adaptation pipeline is the first that operates in image space (cf. Figure 3.4 on p. 50). As such, it is conceptually independent of the underlying PAGE's visual data type.

For SVG vector graphics, this stage thus operates on the raster graphics image that is the result of the SVG rendering process. Therefore, the discussion on view space manipulation of raster image PAGES from Section 4.2.4 equally applies to vector graphics PAGES. In particular, operations on this stage may be applied to Regions of Interest defined by task node annotations in *image space* independently of vector graphic primitives.

A point to note in this regard is that SVG itself provides facilities for so-called filter effects that operate on the rasterized result of SVG rendering. In particular, it offers a Gaussian blur filter that can be employed to emulate the semantic Depth-of-Field (sDoF) de-accentuation operation, cf. Section 4.2.4. Under the SVG render model each fragments is rendered in isolation on a temporary canvas, then defined filter effects are applied. Only then is the rasterized rendering result composed over previously rendered content (see [FFJ03]). This restricts filter influence on primitives of the given fragment. Therefore, applying the SVG blur filter to individual FEATURES according to relevance values offers an efficient way to emulate per-FEATURE sDoF on vector graphics PAGES.

5.2.5 Labeling

The final stage of the adaptation pipeline comprises dynamic labeling of the adapted visual representation. To this end, Section 4.2.5 discussed the integration of particle-based labeling [LSC08] as a general-purpose labeling approach. Being an image-based algorithm, its application to 2D vector graphics is completely analogous to the procedure described in Section 4.2.5. The only minor difference is that instead of annotated FEATURE boundary polygons (cf. Section 4.1.2), the primitive geometry is used directly to derive the conflict particle configuration as described in citeLuboschikEtAl2008. The algorithm then places labels on the raster image that is the result of the SVG rendering process in exactly the same way as for raster PAGES.

The space-efficient remote labeling approach introduced in Section 4.3.2 also applies to vector graphics PAGES in a similar way as it does to raster images but for two minor

adjustments. These adjustments utilize advantages inherent to the SVG format that increase the efficiency of (i) the creation of the required ID-buffers, and (ii) of labeling response handling.

For raster images, the ID-buffer generation required the creation of two intermediate buffers: the binary segmentation as well as the color-coded FEATURE mask, cf. Figure 4.12. These intermediate buffers are composed to obtain the final ID-buffer that classifies the image space into negative space, non-empty and foreground (FEATURE) pixels. For SVG images, the ID-buffer can be created in a single rendering pass. To this end, the canvas is initialized with the color code for negative space. Then, primitives from the background content layer are rendered with the non-empty color code assigned. Finally, all FEATURES are rendered in order using their respective ID color.

Furthermore the returned labeling layer is an XML representation that comprises a *geometric* description of the labeling result, in particular the position of labels and the geometry of label-object guides. Therefore, the logical approach is to integrate the labeling module's response directly into the SVG DOM as corresponding <`text`> child elements of the top-most labeling layer (cf. Section 5.1.1). This way, labels are rendered automatically by the SVG user agent. Note this still makes remote labeling of SVG PAGES an image-based approach: the labeling module itself operates on the rasterized ID- and Distance-Buffers (cf. Section 4.3.2).

Finally, the general content layer concept proposed in Section 5.1.1 facilitates to add additional auxiliary geometry to the adapted visual representation. To this end, task nodes are annotated with references to SVG content fragments describing the auxiliary geometry. During labeling, the corresponding fragments are embedded (via <`use`>) under the label layer, ensuring that they are painted on top of any image content.

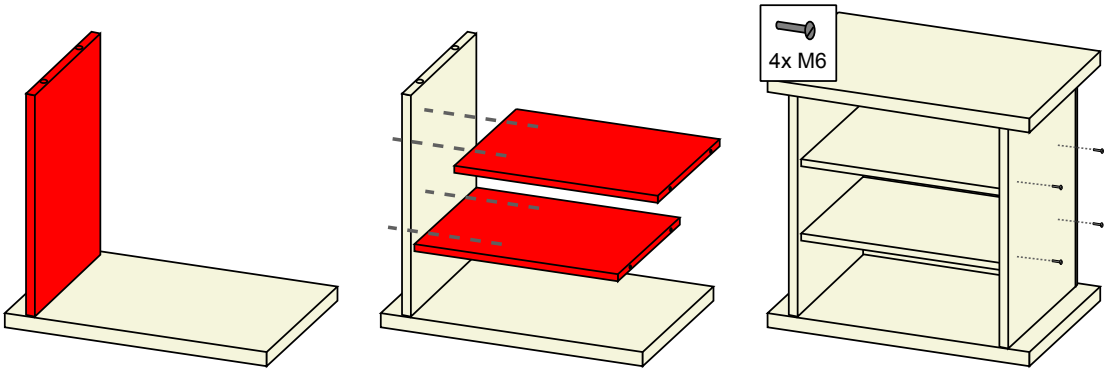


Figure 5.10: Example for the combined application of the adaptation operations discussed in this Section. Shown are the initial visual representations for three consecutive working steps of assembly instructions.

Figure 5.10 shows a comprehensive example how individual adaptation operations introduced above are concerted to achieve complex task-specific adaptations of the base PAGE. Shown here are the resulting visual representations for three consecutive working steps that employ a combination of information hiding (removal of irrelevant FEATURES),

accentuation through adaptation of visual attributes (assigning a highlight fill color), explicit FEATURE translation, as well as the use of auxiliary geometry (guiding lines and inset detail views). This results in three visually consecutive illustrations, i.e., one initial view for each basic task node, for (partial) assembly instructions of the depicted cupboard.

5.3 Smart Vector Graphic Techniques

So far this chapter described the present thesis' fundamental concept how to structure and organize vector graphics PAGES in a way that facilitates FEATURE-based, relevance-driven adaptation of task-specific visualizations. Further, it has been discussed how this conceptual PAGE structure is utilized on the different stages of the general adaptation pipeline introduced in Chapter 3. In the following, two novel display techniques are introduced that build on these adaptation operations.

First, a method is described for the semi-automatic generation of so-called exploded view diagrams that is commonly used to convey information on complex assemblies.

The second technique has been developed specifically to address the challenges arising from displaying large diagrammatic representations on compact mobile devices, such as electric circuit diagrams.

5.3.1 Smart Exploded View Diagrams

A specific application of FEATURE translation (cf. Section 5.2.2) is a presentation technique referred to as 'exploded view diagrams.' These are an established method to elucidate the structure of complex objects that are composed of many subparts, such as mechanical assemblies, architectural environments and biological organisms [LAS04]. Exploded view diagrams simultaneously convey the global structure of the depicted object, the details of individual components, and the local relationships among them.

In particular, we here propose an approach that follows an idea presented by Li et al. [LAS04]. Unlike the vast majority of exploded view algorithms that rely on detailed 3D models of the objects to explode, Li et al.use a '2.5D' solution based on two core concepts: (i) a *stack of* (raster) *image fragments* that is sorted according to the depth order of the fully collapsed ('assembled') object, as well as an *explosion axis* along which object fragments are translated ('exploded') in 2D image space. To this end, their approach requires several preparatory, semi-automatic working steps to create the depth-ordered stack raster fragments representing object parts [LAS04].

We here present an alternative approach that utilizes the hierarchy of SVG FEATURE fragments instead. As a second contribution, it is discussed how exploded views can be generated from the PAGE description automatically based on FEATURE relevances. We initially published this idea in [FSS07].

With our approach, a given FEATURE constitutes the object subjected to 'explosion', whereas its sub-FEATURES represent the individual components that comprise the object (see also Figure 5.4). The child order of the respective fragment root nodes implicitly defines the required depth ordering of fragments. This child order thus also implicitly

determines the stacking order of fragments, in other words, the 'previous' and 'next neighbor' relations between sub-FEATURES along the explosion axis (see Figure 5.11a). Therefore, the only additional information that must be provided as task node annotation is the direction of the explosion axis in image space, which can be specified simply as a direction vector $\mathbf{d} = \binom{d_x}{d_y}$.

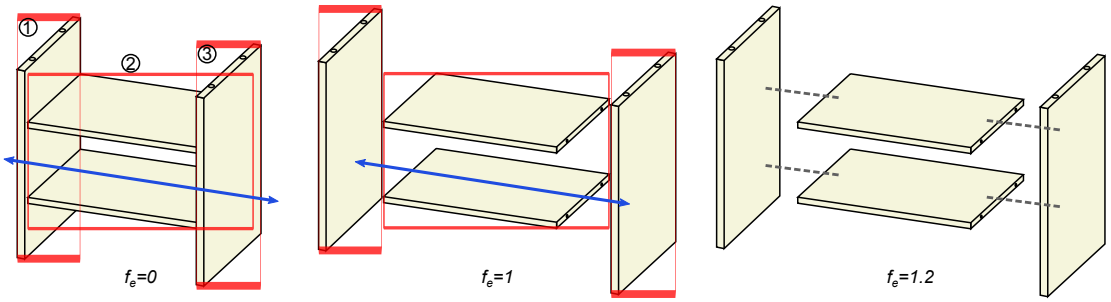


Figure 5.11: Illustration of FEATURE-based, relevance-driven 'exploded view' generation. The implicit depth order (child order) of sub-FEATURES (numbered) defines the stacking order in the initial, collapsed configuration (a). De-conflicting of bounding boxes (red) along a specified explosion axis (blue) yields the fully exposed configuration (b). Over-exposing further separates components (c).

With the approach described in [LAS04], the 'explosion magnitude' – i.e., the amount by which individual fragments are displaced along the explosion axis – is interactively controlled to vary between zero (fully collapsed object) and a predetermined maximum value. Contrary to this, we here propose to control the 'explosion magnitude' automatically according to the relevance values of the sub-FEATURES of the exploded FEATURE as follows.

The core idea is to map each sub-FEATURE's relevance value r_i to an *exposition factor* $e_i \in [0, e_{max}]$ that represents the *degree of exposition* for that sub-FEATURE ϕ_i; whereby e_{max} is a pre-defined upper limit. In the simplest case, this is a linear mapping so that a relevance of $r_i = 0 \rightarrow e_i = 0$ and $r_i = 1 \rightarrow e_i = e_{max} = 1$. However, other mapping function are viable, for example exponential or stepwise-constant functions similar to those shown in Figure 5.7.

The second aspect to consider is then how exposition factors correspond to translation distances along the specified explosion axis.

- An exposition factor $e_i = 0$ represents a completely collapsed representation, i.e., the component is placed in the original, 'assembled' position relative to its neighboring sub-FEATURES. This corresponds to zero translation distance.

- At the degree of exposition $e_i = 1$, the component is exposed so that no overlap occurs between itself and its previous and next neighbor along the explosion axis. This corresponds to a translation distance sufficiently large to resolve overlaps between the respective FEATURES' geometries or suitable proxy geometry, such as the FEATURES bounding rectangle shown in Figure 5.11.

- Separating exploded components even further (i.e., introducing gaps between them) thus corresponds to exposition factors $e > 1$.

Figure 5.11b illustrates this concept. Obviously, if two subsequent sub-FEATURES ϕ_i, ϕ_j on the explosion stack are assigned differing relevance values r_i, r_j, the translation distance between them will correspond to the exposition factor mapped to $max(r_i, r_j)$.

Figure 5.11c shows the resulting visual representation for the cupboard example used throughout this chapter. The shelves and the sides have been arranged on a slanted explosion axis with an exposition factor of $e = 1.2$, introducing visual separation between components. To further emphasize the spatial relations, (manually) predefined auxiliary guides have been superimposed during the labeling step (cf. Section 5.2.5).

Because the translation distances are determined exclusively from FEATURE relevance values, the general approach presented here also facilitates *interactive* exploded view diagrams beyond the initial visual representation for the task at hand. Recall from Section 3.3.5 that our concept for the interactive manipulation of initial views is based on user selection of FEATURE(s) of Interest (FoI). In particular, selected FoI are assigned the maximum relevance value $r_{max} = 1$, whereas non-FoI FEATURES are assigned lower values according to some Degree-of-Interest (DoI) function (often, the minimum relevance $r_{min} = 0$). After a change to the set of selected FoI, the adaptation pipeline is re-executed to reflect the change to the user's focus of interest. Thus, by selecting FEATURE(s) in the exploded view diagram, the user can interactively control which object components are exposed (by selecting them) and which are reverted to the fully assembled representation (deselected, $r = 0 \rightarrow f_e = 0$). More complex DoI functions can take into account the neighbor relations defined by the stacking order to e.g., achieve a gradual drop-off in FEATURE relevance (and thus, degree of exposition) in an n-neighborhood around a selected FEATURE. This effectively creates an 'explosion fisheye view' Focus & Context display of the exploded FEATURE.

Moreover, the general concept of hierarchical FEATURES introduced in Section 5.1.2 facilitates the creation of complex exploded view diagrams along multiple nested explosion axes: an explosion axis can be defined for each decomposition level of a hierarchically composed FEATURE. This allows to 'explode' sub-FEATURES along one axis, while its nested sub-sub-FEATURES are translated along a second axis. Figure 5.12 gives an example: here, components of the top-level FEATURE (the jet engine) are exploded vertically; one of the contained sub-FEATURE (the compressor-turbine assembly) is exploded horizontally.

5.3.2 Smart Technical Diagrams

The discussion so far in this chapter concentrated on task-related WHY and WHAT aspects of adapted visual representations. However, especially in mobile application scenarios, the WHERE aspect of a smart visual interface's context of use must not be neglected. Here, due to the small size of typical handhelds available screen space is very limited.

This is a challenge in particular when presenting diagrammatic representations of complex (technical) structures like electric circuit diagrams or process flows. Such rep-

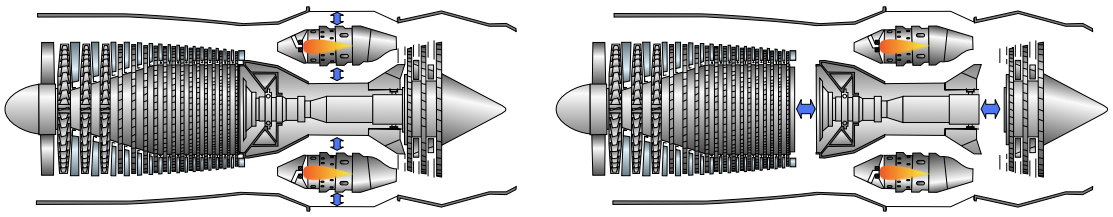


Figure 5.12: Example for a complex FEATURE-based exploded view diagram using two nested explosion axes on two different levels of the FEATURE hierarchy: first, the top-level FEATURE is exploded vertically (left), then one contained sub-FEATURE is exploded horizontally in the gained space (right).

resentations usually focus upon displaying the functional dependencies between different parts or modules. While the individual parts themselves can be adapted using the operations described in Section 5.2, the depiction of their relationships to one another adds to the difficulty of bringing such structure representations to the limited screen of a handheld device. Figure 5.13 illustrates this by comparing the resolution of a typical PDA to that of a rasterized representation of a medium-sized circuit schematic.

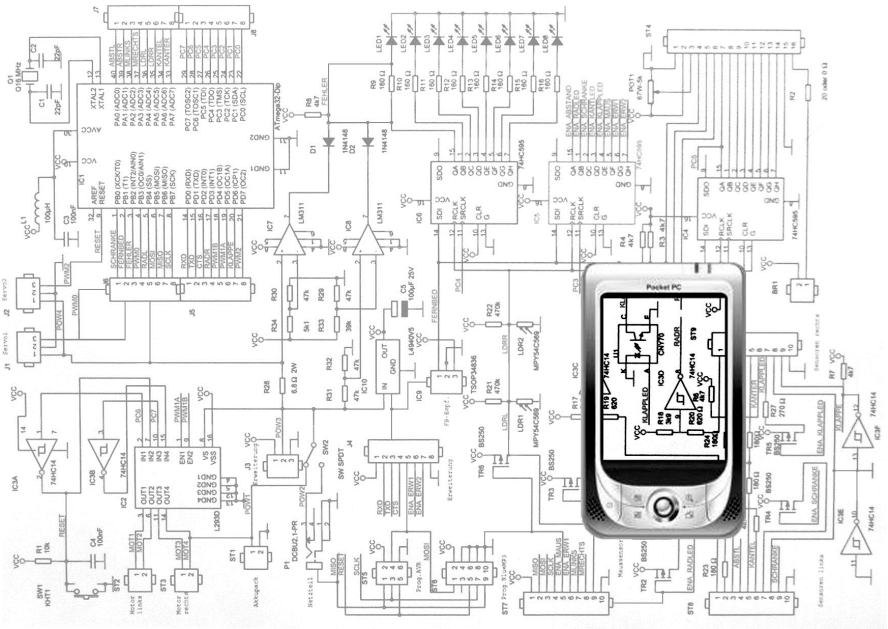


Figure 5.13: Comparison of the screen resolution of a typical PDA (320x240 pixels) to the resolution of a medium-sized circuit schematic (rendered at 1742x1275 pixels).

Even though encoding the shown circuit's representation as vector graphics allows to render it at arbitrary resolutions, this naïve approach to display scalability of downscaling to the size of the PDA screen would lead to indiscernible tiny or cluttered details. Therefore, the adaptation must also affect the information and visual scales of the technical drawing by selecting a subset of information to be displayed. With regard to structure schematics there are two principal strategies to this (cf. Figure 5.14):

Segmentation: the visual representation is limited to show only a certain part of the entire technical drawing that fits the output device, thus sacrificing the completeness of the representation but preserving its local levels of detail.

Depth reduction: Groups of diagram elements are merged into aggregate representations that take up less screen space, thus sacrificing the in-depth-depiction of all objects but preserving the completeness of the overall structure.

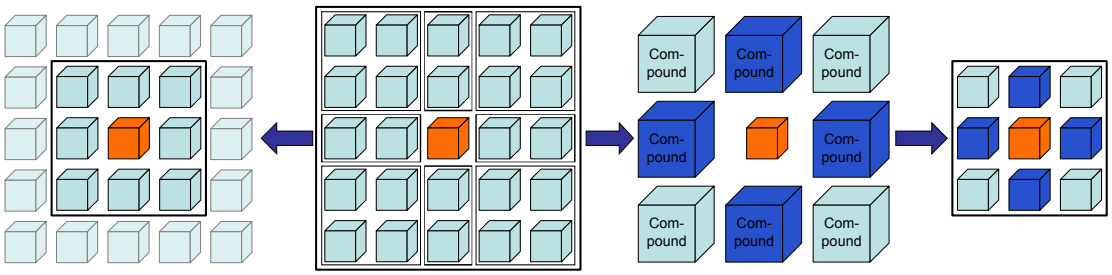


Figure 5.14: Conceptual view of adaptation strategies for diagrammatic representations to small displays: segmentation (show element subset at full local detail, left) versus aggregation (composite element representation with reduced detail).

A fundamental approach to effect information scalability of circuit representations is to model them as a graph (see e.g., [HO95, WKM+09]): circuit elements represent graph nodes, while line interconnections constitute edges between nodes. In particular, an electric network diagram defines a hierarchical graph:[8] circuit elements in the wiring diagram (the lowest-level graph) are aggregated into compound circuitry, which are further abstracted into electric components in the schematic (function block) diagram [WKM+09]. This *connectivity graph* thus facilitates local changes to the information scale on a *structural* level by pruning subgraphs (i.e., segmentation) or collapsing node groups into meta-nodes (depth reduction). The diagram's *visual* representation is then tailored to match the graph state. To this end, contemporary approaches (e.g., [LAS04, FD06, WKM+09]) rely almost exclusively on interactive manipulation of the graph state through its visual representation.

Contrary to these, in this section an approach to the *task-driven* adaptation of schematic diagrams is proposed that utilizes the general definition of vector graphics PAGES with hierarchical FEATURE as its basis. The core idea is to derive the connectivity graph's state for the task at hand from relevance values in the enriched task model, which is then reflected by the adaptation of corresponding FEATURES. As with all other approaches presented in this thesis, the primary goal is to provide good initial views of schematic diagrams to the task at hand, which can then be interactively refined further. A first version of this approach was published in [FSS07].

[8] A hierarchical graph is a graph where groups of nodes on one hierarchy level are aggregated (clustered) into a single *meta-node* on the next higher level, see e.g., [Sch10].

Starting point is the PAGE definition of the diagram representation. Its FEATURES describe the diagram elements at different levels of abstraction. To this end, the hierarchical organization of FEATURES described in Section 5.1.2 is utilized as follows:

- Top-level FEATURES represent diagram elements at the highest level of abstraction. It therefore comprises an aggregation of several elements at the next lower abstraction level of the diagram.
- At each FEATURE hierarchy level but the leaf level two sub-fragments are defined: one describing the abstracted representation (i.e., geometry) for that level of abstraction, and a sub-FEATURE that describes an *optional* decomposition into sub-elements on the next lower level.
- The leaf level of the FEATURE hierarchy describes the most detailed representation of the diagram. FEATURES on this level constitute nodes in the connectivity graph.

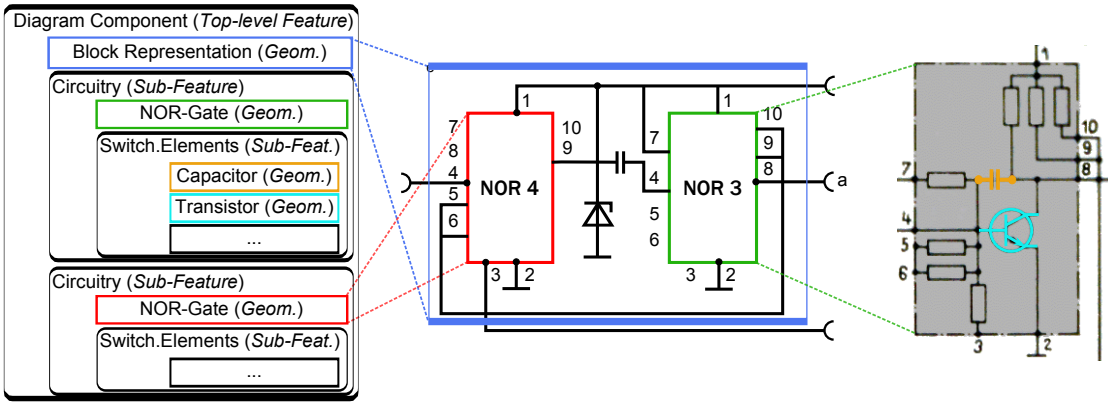


Figure 5.15: A hierarchical FEATURE describes a group of diagram elements at different levels of detail. Each FEATURE hierarchy level contains two content fragments: the geometry describing an abstracted visual representation for that level, as well as a sub-FEATURE describing its *optional* decomposition into further sub-elements.

Figure 5.15 illustrates this for a section of an electrical circuit with three levels of abstraction: On the lowest level of abstraction (the leaf level) are FEATURES that define individual switching elements e.g., capacitors and transistors. These are represented by standardized symbols.[9]

The next higher level of abstraction models the circuit as a number of circuitry (in this case, NOR gates) that are wired together via connecting pins. These are represented by rectangles indicating the pin layout. The top-most level groups gates into functional units (e.g., registers). These are represented by simplified block representations (not shown in Figure 5.15).

[9] This facilitates the re-use by reference of symbol geometry defined once (cf. Section 5.1.2). In fact, SVG's `<symbol>` element has been introduced with this type of application in mind [FFJ03].

This hierarchical encoding of diagram elements into FEATURES already captures two aspects of adapted diagram representation:

- It readily describes groups of diagram elements (nodes) that are aggregated into meta-nodes on the next higher level of abstraction.
- On each level, two *alternative* visual representations are encoded: an abstracted representation appropriate for the respective level of abstraction, as well as a FEATURE that itself defines an optional, further sub-divided and thus more detailed representation.

At this point the PAGE description misses, however, the connectivity information (e.g., wiring) between FEATURES/nodes (diagram elements).[10] To this end, the basic PAGE is further associated with a corresponding graph structure. This graph may be directly available e.g., as export data from dedicated CAD programs like EAGLE.[11] For smaller circuits, a manual creation of the connectivity graph is also a viable option. For example, Figure 5.16 shows a screenshot of a simple tool [Woy06] used during prototype development in the LFS project (cf. Section 8.1).

Using an XML encoding for the connectivity graph facilitates direct integration with the SVG file storing the PAGE skeleton. GraphXML [HM01] is especially well-suited because it explicitly supports the distinction of graph structure (nodes and edges) and their visual representations (FEATURES and line connection geometry, respectively). In particular, this allows to account for the spatial extend of FEATURE geometry during graph layout (see below).

Based on this extended PAGE description, the task-specific adaptation of the diagram representation is effected. This comprises the following steps:

1. Determine a graph state representing which diagram elements are visible at which level of abstraction. This amounts to a selection of a sub-section of the complete diagram. Thus conceptually, this step represents the view selection stage of the adaptation pipeline.
2. Selection of a suitable visual representation for each diagram element (FEATURE) corresponding to the abstraction level and FEATURE relevance. This is a geometry adaptation operation, in particular, a combination of display size and LoD selection.
3. Layout of the diagram elements and their line interconnections using a graph layout algorithm. This is also an operation on the geometry adaptation stage and amounts to an explicit re-positioning of FEATURES.
4. Further adaptation of visual attributes is applied according to FEATURE relevance on the subsequent pipeline stage.

[10] The FEATURE hierarchy in itself only expresses containment e.g., 'transistor A is contained in (*is-part-of*) circuitry C'. It does not express connectivity such as 'transistor A is connected (by a wire) to resistor B'.

[11] `http://www.cadsoft.de`

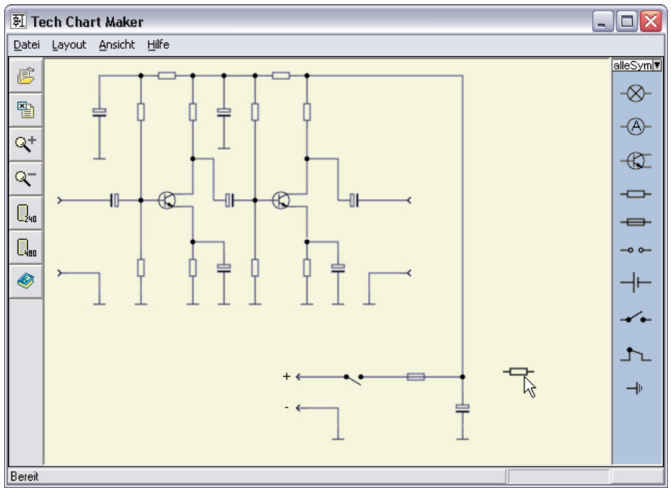


Figure 5.16: Simple authoring tool for creating small electric circuit representations including the connectivity graph by tracing a circuit template (from [Woy06]).

Determine a task-specific graph state. The present approach utilizes the FEATURE relevance values annotated in the task node to determine what FEATURES (diagram elements) should be displayed at which level of abstraction. To this end, FEATURE relevance r_i is mapped to an integer value $d_a \in [0..d_{max}]$, with d_{max} the maximum number of node aggregations into meta-nodes. Figure 5.17 shows this mapping function schematically. In particular, FEATURE relevance below a certain threshold r_t is mapped to $d_a = 0$ signifying the corresponding sub-tree is pruned i.e., the respective diagram element is removed from the representation.

In the example from Figure 5.15 $d_{max} = 3$, i.e., the leaf nodes (switching elements) and two subsequent levels of aggregation (into circuitry and functional units) are available as abstraction levels.

In doing so, r_t is either defined statically in the task node; or it is employed as a dynamic filter operation that successively includes additional elements by expanding sub-trees in order of FEATURE relevance. Expanding continues until a "presentation budget" is reached with respect to the number of diagram elements displayable on the output device.

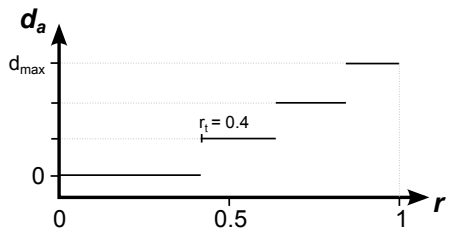


Figure 5.17: Mapping function to derive the connectivity graph state from FEATURE relevance values: meta-nodes are expanded with increasing relevance up to the leaf decomposition level d_{max}. Nodes representing irrelevant FEATURES below a given threshold r_t are pruned ($d_a = 0$).

Selection of visual representations. The selected aggregation level also determines which level of the FEATURE hierarchy is used as the visual representation. In the example provided in Figure 5.15, for $d_a = 1$ the top-level geometry is used, at $d_a = 2$ the second-level NOR gate representation, and individual switch elements for $d_a = 3$. Selecting a FEATURE fragment according to d_a thus selects the geometry that is used for the diagram element's visual representation. This selection is effected using the mechanism of *alternative content fragments selection* that has been introduced in Section 5.2.2 (III) as a means for geometric LoD control.

Graph Layout. The previous two steps resulted in the selection of a subset of diagram elements (nodes) and the geometry to represent them. In order to utilize this segmentation and depth-reduction of the diagram structure for size reduction of the visual representation, the describing graph must be laid out. This entails (i) positioning the graph nodes (diagram elements) and (ii) routing of the graph edges (line connections). This is a problem in the graph drawing domain, which is not in the focus of this thesis. Refer to [Sch10] for an exhaustive discussion on this topic.

There exist, however, a number of software libraries that do provide graph layout functionality, such a jGraph[12]. The latter in particular supports styled layouts e.g., an orthogonal layout where edges are routed at right angles to maintain the illustrative conventions of electric circuit schematics.

The graph layout module takes as input the pruned and folded connectivity graph created in step 1 as well as information on available display space (the target viewport, cf. Section 5.2.1). From these data, it calculates a layout of the circuit elements (i.e., FEATURE positions), and generates the polyline geometry representing routed edges. The layout thereby accommodates for different sizes of node representations (FEATURE geometry). The result of the graph layout step thus represents a representation of the diagram geometry that has been adapted to the task at hand by way of FEATURE relevances.

Visual attribute adaptation. Since diagram elements are regular FEATURES with respect to their visual attributes, operations that modify the visual appearance of elements are effected based on FEATURE (element) relevance as explained in Section 5.2.3. This also applies to polyline geometry generated during the graph layout phase. Visual attribute adaptation is employed, for example, to highlight the most relevant circuit elements and their interconnections (see Figure 5.18).
A prototypical implementation of the proposed approach has been presented in [FSS07], see Figure 5.18.

5.4 Summary

This chapter discussed how the proposed concept of a general adaptation pipeline is applied to the task-based adaptation of vector graphics. A particular focus in this thesis

[12] `http://www.jgraph.com/jgraph.html`

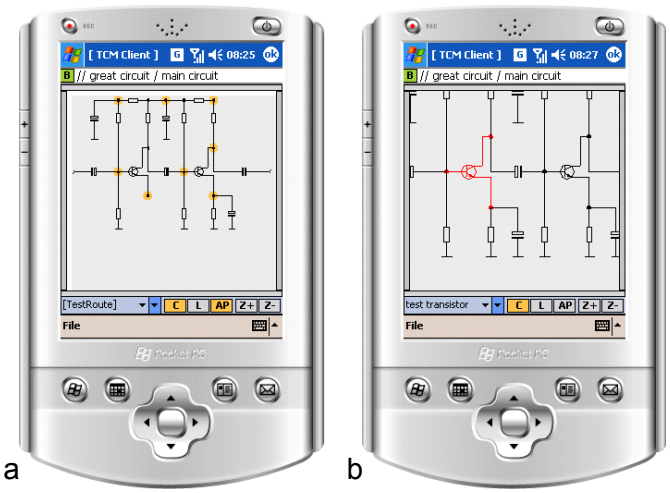


Figure 5.18: Screenshot examples from the prototypical implementation of Smart diagram layout. Irrelevant parts of the circuit are pruned (a). Important features such as testing contacts are accentuated (b). Yellow markers in (a) indicate 'articulation points' were sub-graphs can be interactively folded/expanded to manipulate the view.

has been on vector graphics encoded in the declarative XML-based SVG standard format.

As the core of vector graphics adaption Section 5.1 proposed a concept for the partition of PAGES into specific **content fragments** on multiple **layers**. Fragmentation utilizes SVG's object hierarchy to define FEATURES according to *semantic criteria*, as determined by the application and task at hand. The described approach allows to create nested FEATURES (expressing *is-part-of* relations). This concept moreover enables to define several **alternative visual representations** for individual FEATURES that can be referenced from different tasks.

While several aspects of vector graphics content preparation can be carried out automatically, the identification of relevant FEATURES is highly dependent on the application task at hand and the depicted image contents, as is the parametrization of techniques to achieve a specific communicative goal. SVG content preparation is therefore, on principle, a predominantly manual task necessitating tool support. For this, an **authoring tool** has been described that augments vector graphic design tools by functionality related to PAGE/FEATURE specification and task model annotation.

After content preparation is concluded, the enriched task model provides all information necessary to conduct an automatic task-driven adaptation of vector graphic content. To this end, Section 5.2 discussed viable operations on the different stages of the adaptation pipeline based on the proposed PAGE/FEATURE concept. The conceptual separation between content geometry and its visual attribute thereby offers great flexibility during the first three pipeline stages. In particular, **sub-fragmentation** is employed to facilitate selection of different *geometric Levels of Detail* as well as *texturing quality levels* according to FEATURE relevance values.

Furthermore, Section 5.3 discussed two novel "Smart-X" techniques that exploit the semantically structured description of content afforded by the general PAGE/FEATURE concept for vector graphics: an approach to the **FEATURE-relevance driven generation of exploded view diagrams** [FSS07, FS09]; as well as a **display technique for complex diagrammatic representations** on compact mobile devices [FSS07] based on the task-specific relevance of individual diagram elements (i.e., FEATURES).

As a final remark it shall be pointed out that the concept of FEATURES, FEATURE sub-fragmentation proposed here is not only applicable to task-driven adaptation of 2D representations. Rather, the fine-grained control over content elements make it gainful for other applications as well.

In particular, we showed in a recent publication [FRS11] how the idea of content partitioning into FEATURE fragments and sub-fragments can be applied to cover the requirements for *progressive vector imagery* in mobile image communication scenarios (see [RS09]). Specifically, the geometric LoD and texturing quality levels afforded by our approach are utilized for progressive refinement of image content in user-selected Region of Interest: (sub-) fragments are linked in the SVG skeleton as external resources; these are then transmitted to the output device in progression order (lowest LoD $\rightarrow$ highest LoD $\rightarrow$ texture quality levels), whereby fragments intersecting the user RoI are prioritized. Transmission stops after either the entire SVG image has been transmitted, or a user-defined refinement level has been reached.

Note that progressive imagery also bears relation to the topic of adaptation: In [Thi10] the idea of adaptive progressive visualization as a means for *device-driven adaptation* (WHERE-aspect) has been proposed. Therefore, because it enables *progressive SVG imagery* the FEATURE, sub-fragmentation concept proposed here allows to *combine task-based adaptation* with this particular form of device-driven adaptation.

6 Smart Meshes – Adaptation of 3D Meshes

3D models are a content type that is becoming increasingly abundant: today, more and more processes related to product development, production and servicing are implemented almost exclusively by means of CAD/CAM software. This has resulted in a fast growth of both the number as well as the complexity and fidelity of digital 3D models that are used for a variety of purposes, such as system simulation, product presentations, or specialist training.

This trend results in a number of research challenges [May07, ABF+06]. Besides questions of how to create and (re-)use models or how to preserve semantics across the entire modeling process, the efficient presentation of, and interaction with, such complex models is essential.

In particular, this chapter discusses how 3D polygonal models are integrated with our basic approach to facilitate the adaptation of task-specific visual representations. This includes support of *Mesh Exploration* through the smart visual interface, i.e., supporting the user in the process of understanding or verifying the structure and functionality of an object (modeled as a polygonal mesh) within the context of the task at hand. We have previously published the ideas and concepts described in this chapter in [FHS08].

6.1 3D Content Preparation: Concepts and Authoring Tools

This imposes two additional requirements to the modeling process: (i) enriching the 3D content with information necessary to define a PAGE; and (ii) identify distinct components of the model with semantic meaning in the application domain, so that these can be associated as FEATURES with the task at hand.

6.1.1 Definition of PAGES

Starting point in the creation of a 3D graphics PAGE is a description of the contained mesh geometry and its visual appearance i.e., its surface material. There exist a number of data representations that facilitate rendering of 3D graphics such as polygonal meshes, surface point sampling methods (so-called surfels, cf. e.g. [Hol07]) or volumetric representations (voxel, [KCY93]).

In the present thesis in particular, triangle meshes – as the most common representation form of 3D graphics – are considered. Thus, a PAGE's content is provided by ways of a surface description (vertices and faces defining triangle meshes) of the depicted object(s) with attributed surface material properties.

As explained in Section 3.3.2 each PAGE additionally includes all information to derive a default visual representation from the graphical content without requiring further specification in the enriched task model. The rationale behind this is that the task context only needs to be refined where necessary. In particular, graphical content comprising 3D graphics requires additional parameters beyond the plain triangle mesh data to derive a default visual representation:

- A view point definition i.e., parametrization of the virtual camera, as well as
- lighting information that determines the resulting colors of surfaces in the rendering which allows to ultimately discern shape and characteristics of the virtual object(s).

This additional information is either integral to the model description format e.g., X3D [X3D04] that describes complete virtual scenes; otherwise it is specified as part of the PAGE—task node annotation.

As a further content preparation step, a number of distinct Levels of Detail (LoD) for each of the PAGE's object meshes is built. This later allows a relevance-driven selection of different LoD per FEATURE to adapt the visual representation. Here, we use the multi-resolution hierarchy approach proposed by Holst [HS06], which is an extension of Floriani et al.'s MT-Hierarchy approach [FMP97].

The core idea is to iteratively build a hierarchy of simplified mesh representations bottom-up from the original mesh by a series of edge collapse operations [HDD+93], whereas all possible edge-collapse operations are ordered in a priority queue according to their simplification error. The resulting hierarchy is a directed acyclic graph. Its nodes represent local simplification operations, hierarchy arcs[1] are labeled by triangle sets or *triangle patches*. A specific LoD of the mesh is selected by a cut through the hierarchy, crossing a number of arcs and thus selecting a set of simplified triangle patches. Thereby the set of arcs is selected as to ensure no overlaps or holes between patches occur. See [HS06] and [Hol07] for a more exhaustive discussion of the used approach.

In particular, the approach allows a mesh representation with locally adapted LoD for defined triangle patches. This facilitates interfacing with our FEATURE-based adaptation approach, whereby triangle patches are associated with task-specific FEATURES.

6.1.2 Definition of Features

To further associate specific aspects of the PAGE's content with the task at hand meaningful object components have to be identified. These components are then further attributed with representation information to comprise task-specific FEATURES of the 3D PAGE.

In the ideal case, an object model is already decomposed into appropriate sub-meshes e.g., a piece of machinery where attachment parts are each represented by individual meshes. To this end, 3D modeling software usually employ hierarchical structures (so-called scene graphs) that represent how the composite object is composed from its individual parts.

[1] The term hierarchy arc is used by the authors instead of hierarchy/graph edge since the latter already has a meaning with respect to the triangle mesh.

However, such hierarchies and associated data may not be available due to limitations of the data format, intellectual property protection considerations, or simply because no meta data was generated for the model components in the first place. Analogous to raster graphics, such 3D models are but a set of triangles without semantics, a 'triangle soup'. In this case defining FEATURES becomes a task of first *segmenting* the triangle mesh into triangle patches representing meaningful object components.

Section 2.3.2 already reviewed several approaches to mesh segmentation, cf. p. 27. To recap briefly, mesh segmentation is carried out either according to geometric aspects – it is decomposed into patches that are equal with respect to a certain property (curvature, distance to reference plane) – or semantic-oriented aspects whereas object parts are identified that correspond to relevant features of the object's shape. As further pointed out in Section 2.3.2, the performance of automatic algorithms generally depends strongly on the model geometry and its prevalent features. Using multi-segmentation [ARSF07b] is one way to ensure meaningful results.

However, the approach taken in the present thesis is that of a semi-automatic content authoring process. In doing so, the FEATURE definition is broken down into two phases as follows:

1. Perform an automatic mesh segmentation to arrive at an coarse pre-segmentation of the input mesh, then

2. interactively refine the automatic segmentation result in a *model inspection* step supported by an authoring tool.

This semi-automatic approach to content authoring of 3D graphics PAGES allows to obtain the desired FEATURES of arbitrary objects (meshes) while balancing FEATURE accuracy and required manual effort. This ensures that besides geometric also application-specific FEATURES can be taken into account. An technical domain example are weld seams in the concavities between object components that would hardly be detected as independent segments by automatic methods.

Our approach to the automatic mesh (pre-)segmentation as well as the subsequent interactive model inspection step are detailed in the next two paragraphs, respectively.

Automatic Mesh Segmentation. The choice of an automatic segmentation algorithm will largely be determined by the type of models encountered in a given application domain. For example, Plumber [MPS+04] is a specialized shape classification method for detecting tubular features of 3D objects that are either body parts or elongated, handle- or protrusion-like features. Hierarchical fitting primitives [AFS06] is a hierarchical face clustering algorithm for triangle meshes based on fitting primitives from an arbitrary set e.g., boxes and cylinders.

For our prototypical implementation (see [Goe07]) in the scope of the LFS project in particular (cf. Section 2.2), we opted to use a slightly modified feature point & core extraction algorithm [KLT05]. The basic idea is to classify mesh vertices according to if they belong to the object's core or to a protruding feature. This makes the method

suitable for a wide range of both natural and technical models with a pronounced core; e.g., bodies with attached limbs and machinery with attachment parts, respectively.

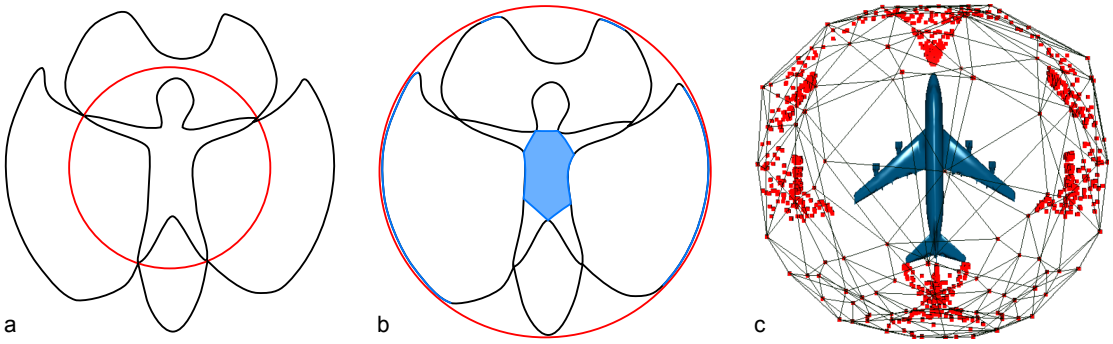


Figure 6.1: Basic idea behind the feature point & core extraction algorithm: mirroring vertices on the object's bounding sphere (a) allows their classification into core and protruding 'limb' vertices (b). Classification example for a complex model (c). (Image source: [Goe07])

In the original approach [KLT05], the object core was extracted by mirroring the mesh vertices on the object's bounding sphere. Any vertex that maps to the convex hull of the mirrored vertex set is classified as belonging to the core. All other vertices belong to protruding features or 'limbs'. This is illustrated in Figure 6.1. In order to improve the segmentation results for arbitrary objects that may exhibit bumpy surfaces or scanning artifacts, we introduced an additional parameter to classify vertices with a maximal hull distance $d \geq 0$ as core vertices as well. Also, unlike proposed in [KLT05], we do not apply a multi-dimensional scaling (MDS) transformation of the mesh vertices before feature point detection. This transformation was applied in the original approach to achieve pose-invariant segmentation results. However, for rigid technical models that are not posed the results are quite acceptable without MDS transform.

Interactive Segmentation Refinement. During subsequent model inspection, the author can verify whether the automatically detected segments match the desired result. To this end, the model is rendered in the authoring tool, where the author can manipulate the 3D view (zoom, rotate) to inspect each segment patch. Segmentation refinement thereby comprises these operations:

- recursive sub-segmentation of mesh segments using the segmentation algorithm,
- merging of small segments into larger ones, and
- fully manual specification of new segments without algorithmic pre-segmentation.

Details on how these operations are supported by the authoring tool are given in Section 6.1.3 when reviewing the particular tool that has been developed to support smart visual interface design for 3D mesh content.

Finally, after the segmentation of the object mesh into FEATURES has been completed, the result is mapped onto all other levels of the LoD hierarchy. The surface mesh can therefore be presented with variable LoD for each FEATURE in the smart visual interface.

6.1.3 Feature Relevance Values

After mesh segments have been identified as FEATURES, the last step of task model enrichment is to associated mesh FEATURES with the appropriate task nodes, as well as to assign task-specific relevance values to effect the adaptation process for each basic task.

For the same reasons already articulated in Sections 4.1.3 and 5.1.3, this constitutes a manual authoring step. Because relevance values depend on the communicative goal for the task at hand, it would be difficult if not impossible to derive these automatically for domain tasks in arbitrary application domains.

Therefore, the following section reviews the functional requirements to a suitable authoring tool that allows a human content author to manually execute portions of the content preparation and visual interface design which can not be automated.

Mesh Authoring Tool

Authoring support must thus include the following operations:

- Creation and editing of PAGE definitions,
- FEATURE designation: automatic pre-segmentation, as well as support for interactive model inspection and manual segmentation refinement,
- annotation of CTT task model nodes with the created PAGE/FEATURE definitions, and
- the specification of FEATURE relevance values per basic task node as well as an interactive design of the corresponding parametrization of the PAGE's adapted visual representation.

A screenshot of a prototypical implementation of a corresponding authoring tool is shown in Figure 6.2.

Creating and editing of PAGE definitions. A 3D mesh PAGE comprises to parts, the mesh data itself, as well as the PAGE manifest. A new PAGE is created by loading a triangle mesh from file, which is then automatically transformed into a LoD hierarchy representation [Goe07, Hol07].

The associated PAGE manifest is an XML-encoded structure that associates the mesh data with adaptation-related information. In particular, the manifest for 3D mesh PAGES comprises:

- A PAGE identifier. The identifier is used to reference a particular PAGE from the annotated task model.
- A human-readable title, that is e.g., displayed as the window title within the smart visual interface.
- The reference to the file that stores the LoD hierarchy representation constituting this PAGE's the contents.
- The list of FEATURE fragments defined on this PAGE.

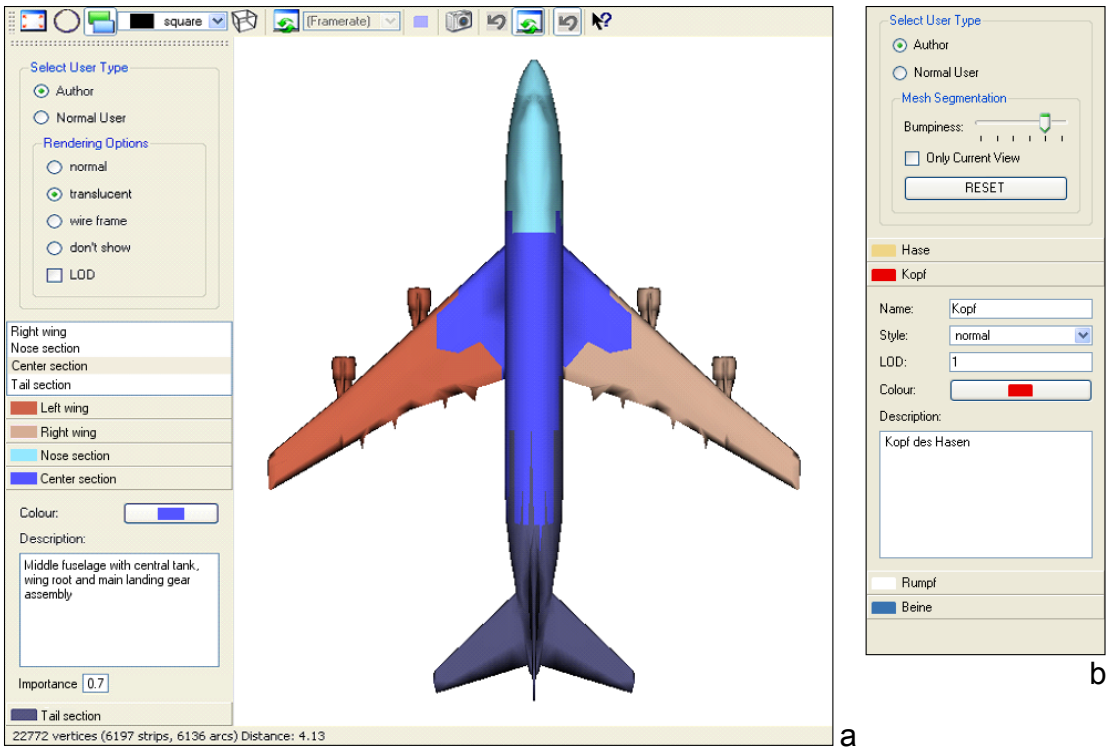


Figure 6.2: Screenshot of the authoring tool prototype. The 3D inspection view shows the current segmentation of the model (a). The control panel on the left is used to set the FEATURE (segment) relevance value and other task-specific properties. (b) shows panel for automatic segmentation and explicit LoD selection.

All these properties can be edited for existing PAGES, including the mesh data reference. The latter is useful if the initial 3D model has been replaced by an updated version.

FEATURE designation. This core step of the content preparation process comprises the automatic (pre-)segmentation of the PAGE mesh, followed by the interactive inspection with manual segmentation refinement.

To this end, the authoring tool provides means to execute a selected segmentation algorithm. To facilitate the integration of different algorithms an implementations, these are integrated as self-contained plug-ins (cf. [Goe07]). The prototype implementation presented here currently supports the modified feature point & core extraction algorithm described in Section 6.1.2.

During subsequent *interactive model inspection*, the author can verify whether the automatically detected segments match the desired result. To this end, the model is rendered in the authoring tool with the segment patches assigned unique colors. The author can manipulate the 3D view (zoom, rotate) to inspect each segment patch. Individual segments can be selected, either by picking them in the 3D view or from a list control (cf. Figure 6.2), and sub-segmented further by iteratively applying the segmentation algorithm to the associated triangle patch.

Another option during the inspection phase is fully manual segmentation of model parts. Manual segmentation offers great flexibility to overcome algorithm limitations as it does not rely on its suitability for a given geometry. By interactively placing simple shapes, such as spheres or cubes, in the 3D scene the author can quickly define Regions of Interest (Figure 6.3a). Parts of the mesh intersecting these volumes can be assigned to a new segment, or added to an existing one.

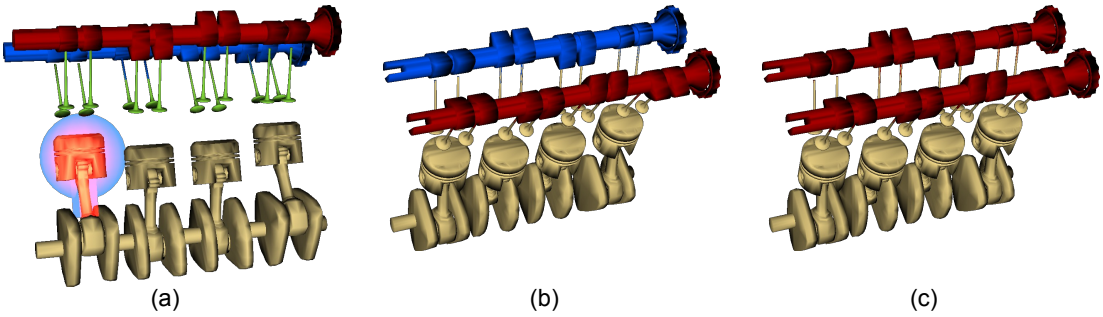


Figure 6.3: Definition of 3D RoIs to obtain a first coarse segmentation (a). It is refined by selecting and further subdividing segments, or by merging multiple segments (b, c).

The resulting segmentation of the current inspection iteration can be refined even further by three additional operations:

- expanding a segment by adding a triangle strip at its boundary,
- shrinking a segment by removing a triangle strip from its boundary,
- merging two segments into one (Figure 6.3b-c).

Expanding and shrinking segments can also be applied to obtain a disjunctive, exhaustive segmentation of the entire model, if so desired. This is done by expanding patches into mesh areas yet unassigned to any segment, and shrinking overlapping segments in the respective areas. Note that the final segmentation does neither need to be disjunctive – triangles may be associated to more than one overlapping FEATURE– nor exhaustive if some regions of the surface do not belong to any FEATURE.

PAGE/FEATURE association with tasks. Similar to the authoring tools created for raster and vector graphics, the 3D mesh authoring tool itself is not designed to create or structurally modify CTT task models. Rather, it operates on existing models stored in XML format. This results in a list of tasks that can then be associated with appropriate PAGE manifests. Because FEATURES are an integral part of the PAGE manifest, FEATURE-task association is effected via relevance values – recall from Section 3.3.3 that the default relevance value $r = 0$ indicates the given FEATURE is irrelevant to the task at hand.

Specification of task-specific FEATURE properties. This working step primarily comprises the interactive definition of FEATURE relevance values. In addition, the au-

thoring tool currently supports the addition of the following types of auxiliary FEATURE attributes to tailor the visual representation:

- Definition of a good initial viewpoint on the model (by saving the current viewpoint in the authoring tool),
- an explicit LoD selection in case relevance-driven LoD selection should not be used on the geometry adaptation stage,
- an explicit rendering style (hidden, wire frame, translucent, normal),
- a specific highlighting color that are used during the visual attribute adaptation phase, as well as
- a short text used as label during the last adaptation pipeline stage.

Fig. 6.2b shows a screenshot of the task pane for explicit setting of task-specific parameters. How these options are generally effected in an automatic fashion based on FEATURE relevance is discussed in the following Section 6.2.

6.2 Adaptation Control

The visual representation of 3D models is generated by a rendering process of the describing triangle mesh. For this reason the task model enriched with the description of semantic FEATURES and task-specific FEATURE relevance values allows extensive automatic adaptation of the PAGE's initial view based on relevance values alone. The following subsections will discuss the operations viable on each adaptation pipeline stage to thusly effect smart adaptation of 3D graphical content.

6.2.1 View selection

For a good initial view on the 3D model for a given task, the virtual camera must be positioned so that as many relevant model FEATURES as possible – at the very least, the *most relevant* one – are initially visible. Unless the content author explicitly provided a viewpoint selection as a task node annotation, the initial viewpoint must be calculated automatically. For this, the relevance values from the task model can be used to control automatic viewfinding algorithms.

We here adopt the idea presented by Götzelmann et al. [GHS06] that itself is based on the viewpoint entropy developed by Vázquez [VÓ3]: from a set of view candidates, the best-rated view is selected. View candidates are for example the eight canonical views (top, front, side etc.) or the eight corners of the 3D model's bounding box, and looking at its center.

Each view candidate is evaluated by calculating the weighted sum of the FEATURES' projected areas' visible sizes. This evaluation is carried out on a color-coded projection of the 3D scene. Thus, for each FEATURE we determine the number of visible pixels, weighted by its relevance value. The color-coded projection is thereby identical to the scene's *ID-buffer* used for labeling (cf. Section 4.3.2). See the 'Labeling' subsection below for an explanation on how the ID-buffer is generated for 3D scenes.

6.2.2 Geometry adaptation

When applied to a 3D model there are two options for the relevance-driven adaptation of geometry: (i) a change in relative sizes of model FEATURES so that more relevant objects are represented larger, and (ii) adjusting the respective meshes' LoD.

With respect to size adjustment two cases need to be distinguished. If a model FEATURE is described by an isolated mesh it is simply resized by specification of a corresponding scaling transform during rendering. The corresponding transformation matrix is parameterized according to FEATURE relevance. See Section 5.2.2 on p. 103 for a discussion on mapping of relevance values to scaling factors.

However, if a FEATURE comprises a triangle patch that is part of a closed object surface (e.g., the wingtips in Figure 6.2) resizing is most often not feasible as it will result in 'cracks' and discontinuities in the model surface. Patch-based FEATURES are therefore not resized by default in our approach.

Normally, a mesh's LoD is selected according to surface-viewpoint distance. This results in the visual representation of objects farther away omitting details which would be imperceivable anyway. By contrast, for the task-driven adaptation the LoD selection is utilized as a means to de-emphasize less relevant FEATURES. For this, FEATURE relevance is mapped onto discrete Levels of Detail using a stepwise-constant mapping function (cf. Figure 5.17, p. 118). The mapped detail level designates the *maximum LoD* that the respective FEATURE is rendered at. Thus, highly relevant FEATURES' LoD are determined by the view distance, whereas less important ones are rendered at low detail even if close to the viewpoint. Irrelevant FEATURES ($r = 0$) are rendered at the lowest LoD, or are even not rendered at all.

This basic approach can be combined with more sophisticated techniques for the creation of *cutaway views* [DWE03]. Here, geometry is removed to reveal otherwise occluded FEATURES above a certain threshold. Viola [Vio05] has developed a corresponding illustration framework using object relevances without however addressing the origin of illustration-specific values. The contribution of our approach is therefore to deliver *task-specific* relevance values in the *extended task context* of a composite workflow (WHY and WHEN aspects), making presentation techniques like [Vio05] "Smart-X" techniques.

6.2.3 Visual attribute modification

To efficiently convey the important aspects of the task at hand, the attention of the user should be further guided to the relevant FEATURES by means beyond initial viewpoint selection. This is achieved by accentuating important model components and visually subduing (de-accentuating) unimportant ones according to the relevance values from the task model.

There is a vast number of methods for accentuation that can be effected automatically based on FEATURE relevance, cf. Sect. 2.3.2. Simple adjustments include a change in surface material color (i.e., saturation, brightness) and properties (e.g., shininess, transparency). In particular, this includes the assignment of specific highlight colors as material color.

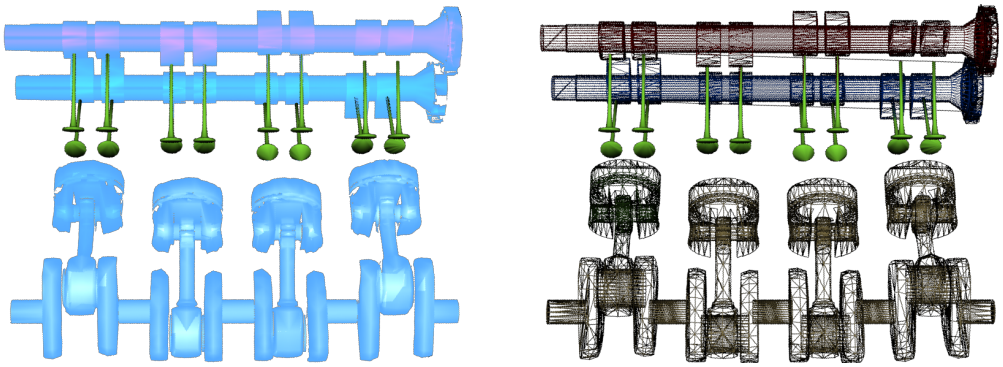


Figure 6.4: Example for the application of different rendering styles for segment accentuation. The valves are defined as the most important component for the current task, other components are visually subdued.

Alternatively, relevance is used to select a distinct *rendering style*. To this end, a defined set of styles is ordered according to their visual saliency, for example 'hidden (invisible) – wire frame – translucent – normal (colored/ textured)'. Each style is a label for a particular set of rendering parameters (e.g., a corresponding shader program). FEATURE relevance is then mapped onto these labels in order of increasing visual saliency to facilitate automatic style selection. Figure 6.4 shows an example.

This basic approach is easily extended to include more complex rendering styles, in particular non-photorealistic (NPR) techniques [SS02] such as Gooch shading [GGSC98].

6.2.4 View Space Manipulation

This pipeline stage is applied to the rasterized result of the rendering process i.e., the graphics frame-buffer. Conceptually, it is therefore completely independent of the rendering process and the underlying 3D model description. Viable operations on this stage have already been discussed in Section 4.2.4.

6.2.5 Labeling

Likewise, since we use an image-based labeling approach (cf. Section 3.3.4), the visual representation of 3D model PAGES is labeled using the exact same techniques detailed in Chapter 4 for the integration of particle-based labeling (4.2.5) and space-efficient remote labeling (4.3.2), respectively.

Image-based labeling techniques rely on segmentation information about the labeled raster image in the form of a color-coded *ID-buffer* (cf. Section 4.3.2). For the visual representation of 3D models in particular, this ID-buffer is generated by the following steps:

1. Assign each FEATURE segment a new material with an unique emissive color. Non-feature mesh segments are assigned a material color identifying them as 'context object',

2. disable lighting calculations and activate flat shading in the rendering process (i.e., objects are rendered as flat shapes with constant color),

3. render the 3D scene to the frame-buffer, and

4. read back the frame-buffer and store as the ID-buffer.

Because the rendering step uses the exact same settings with regard to FEATURE geometry adaptation – in particular, size adjustment and FEATURE removal – the generated ID-buffer matches the visual representation of the 3D model rendered regularly. This ensures label position found by the labeling algorithm in the ID-buffer integrate with the visual representation.

6.3 Smart Mesh Exploration Viewer

The adaptation operations defined in the previous section allow to generate task-specific adapted visual representations of 3D models from the enriched task model. Their rendering within the smart visual interface requires a suitable viewer that can interpret the information provided by the task model in order to display these initial views. To this end, we here propose a *mesh exploration viewer* as a 3D mesh-related "Smart-X" technique. A demonstrator implementation has been realized as a 'user mode' inside the authoring tool, see Figure 6.2.

However, while the above measures generate appropriate initial views, it is not appropriate to restrict the user to this predetermined view only. Rather, she should be able to adjust the visual representation to her personal preferences and requirements with regards to the task at hand. Therefore, the ability to interactively *explore* the model via the smart visual interface is nonetheless important.

The exploration viewer supports mesh exploration in a number of ways. Besides the obvious view manipulation, it is always possible to select and thus focus on an arbitrary model component. This overrides the importance value for the associated FEATURE and the visual representation is updated accordingly, as discussed in Section 3.3.5. Component selection is realized either through picking in the 3D view or through selecting the associated list entry. Figure 3.6 on p. 56 already illustrated how this looks for the engine example shown in Figure 6.4.

Another tool for exploring the model in a flexible way is the use of interactive 3D lenses that can be positioned freely in 3D space. In doing so, the user defines a dynamic RoI in which rendering parameters are changed independently of FEATURE segment boundaries. In our prototype, we so far employ only a single lens that changes the geometric LoD of the mesh, see Figure 6.5. This allows a user to view those parts of the model in full fidelity even if their predefined relevance for the task at hand is minimal. However, other lens functions such as switching the rendering style or an 'X-ray' effect [VCWP96] to interactively reduce occlusion could be realized as well.

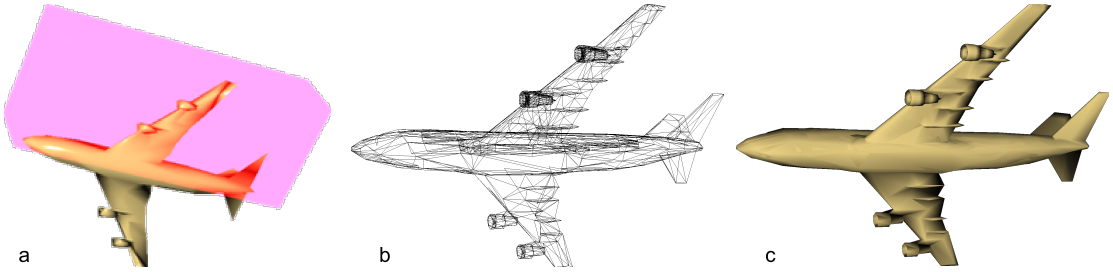


Figure 6.5: Applying a 3D lens to manipulate the Level of Detail in selected object regions. (a) Lens volume, (b) resulting hybrid LoD mesh, (c) visual result.

6.4 Summary

This chapter discussed how the proposed concept of a general adaptation pipeline is applied to the task-based adaptation of 3D models described as triangle meshes.

PAGES are comprised of a *LoD hierarchy* that describes the triangle mesh as a hierarchy of progressively simplified triangle patches. A PAGE further provides a default viewpoint definition.

FEATURES of a 3D PAGE are defined by *segmenting* the triangle mesh into triangle patches representing meaningful object components. Depending on the type of modeled object, application domain and task at hand this operation usually can not be fully automated. For this reason, a semi-automatic approach has been discussed here that combines algorithmic pre-segmentation with an interactive *mesh inspection* and *segmentation refinement* step. A corresponding authoring tool has been introduced. This tool also facilitates the enrichment of the task model with PAGE/FEATURE annotations as well as task-specific FEATURE relevance values, which is usually a manual task too (cf. Section 3.3.3).

The task model thus enriched with 3D content information then allows diverse adaptation operations for a fully automatic generation of task-specific initial views. This includes an automatic determination of a good viewpoint, mesh LoD, and composition of rendering styles based solely on FEATURE relevance values. A large variety of sophisticated 3D rendering techniques have been developed over the last years to effect such visual representations. Sometimes these already have the notion of object importance (e.g., [Vio05]). Thus by integrating these techniques with our proposed enriched task model providing task-specific FEATURE relevance, they become "Smart-X" display techniques.

This notwithstanding it is still important to enable the user to fine-tune the initial view to her personal requirements. A smart *mesh exploration viewer* thus allows the user to select individual FEATURES to override predetermined relevance values, and to interactively manipulate the visual representation locally by employing *3D lenses*. A demonstrator implementation of such a viewer has been introduced in this chapter as well.

7 Smart Visualization – Adaptation of Abstract Data

The previously examined visual data type are immediate descriptions of graphical content: adaptation is effected on the level of graphic primitives in view space. By contrast, abstract data has no inherent visual representation, rather it must be transformed into an image through a visualization process. This makes abstract data the most flexible type with respect to adaptation – it additionally enables operations on the input data, as well as modification of the subsequent visual encoding of data values in graphic primitives and their visual attributes.

At the same time, this flexibility presents significant challenges with respect to task-driven adaptation of data visualizations. There exist innumerable visualization techniques in different implementations for a broad range of data sets and associated data analysis problems. Visualizations are used for the visual communication of known facts, but even more often as tool for the interactive exploration of unknown data sets (exploratory visualization). Here, subsequent analysis steps depend on findings made by the user up to that point, as well as her mental map of the problem domain built up so far. This makes it very difficult to provide an adequate workflow description capturing the task context for adaptation. For this reason, as examined in Section 2.3.2 the vast majority of visualization techniques do not incorporate task-specific (WHY) aspects of the context of use beyond the level of isolated low-level analytic activities, if at all.

In this chapter, we present first ideas on how to integrate our basic adaptation approach with the visualization process of abstract data. Key contributions thereby are mainly in the sense of a problem analysis as well as a systematic view on the different levels at which task-driven adaptation can be effected, in line with the adaptation pipeline concept proposed in Section 3.3.4. However, this chapter also discusses two concrete examples regarding implementation of several aspects thus addressed.

7.1 Preliminary Considerations

At the core of the approach proposed in this thesis are PAGES with FEATURES as the building blocks of adapted visual representations. A PAGE functions as a graphical content container providing a default representation that is associated with a – possibly composite – task described by the general task model. PAGES for the visual data type previously examined are *presentation–oriented*. FEATURES over these PAGES are defined in presentation space, with adaptation effected based on their relevance with respect to the visual representation's communicative goal. This orientation is expedient: the visual data types considered so far are presentation formats, comprising graphic primitives (2D vector and 3D graphics) or even only an unstructured pixel raster (raster images). These

formats thus provide little or no structural and semantic information on the depicted content.

By contrast, for abstract data content as considered in this chapter a visual representation must be created from the input data in the first place. Thereby all visualization techniques can be seen as realizations of a common conceptual model of this image generation process: the *visualization pipeline*, a data flow model that describes how input data is transformed into a visual representation, see e.g. [HM90, CM97]. Its quintessential steps can be summarized as data preparation (*Filtering*), generation of a geometry model comprising graphic primitives from prepared data (visual encoding or *Mapping*) and finally, image generation from the geometry model (*Rendering*), cf. [SM00], Figure 7.1.

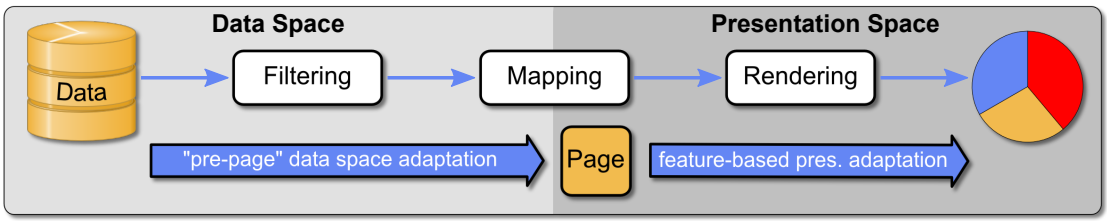


Figure 7.1: Schematic view of the principal visualization pipeline for abstract data (top, cf. [SM00]). Conceptually, PAGES are the result of the filtering and mapping steps. This affords both pre-page and PAGE-based adaptation of visual representations (bottom).

For the benefit of subsequent discussion a brief definition of what comprises the data space indicated in Figure 7.1 shall be given. See e.g., [SM00] for an exhaustive discussion on this topic. The data space is comprised of data elements. Data elements correspond to concrete points of measurements in the domain under observation.[1] The dimensions that span the observation domain are also known as independent variables. Each data element is defined by concrete values for the set of data attributes. Collected attributes are also referred to as dependent variables. Attributes and values further determine *data characteristics* such as data type, spatial frequency, and distribution of data values.

For abstract data, independent and dependent variables do not immediately define the presentation space (i.e., 2D or 3D geometry). Hence both data filtering and mapping steps of the image generation process are necessary to arrive at a default representation that can be encapsulated as a PAGE for association with domain tasks from the general task model.

It is of course viable to influence this process in a task-related manner. Thus for abstract data there are two forms of task-driven adaptation:

- towards the PAGE i.e., *pre-page* modification of the image generation process in *data space*, and
- *based on* the PAGE's default view i.e., modification of the image generation process in *presentation space*.

[1] Note this notion includes both actual measurements in the physical world as well as parameter values obtained e.g., through simulation.

The interrelation between the principal visualization pipeline stages and both forms of adaptation is illustrated by Figure 7.1.

Both filtering and visual mapping in the visualization pipeline operate on data characteristics. This in turn raises two main questions with respect to pre-page adaptation:

- WHAT and WHY is adapted in data space?
- HOW is this adaptation effected in data space?

WHAT and WHY

The first questions primarily relates to the relation between data characteristics and the definition of FEATURES in presentation space. Generally, PAGES and FEATURES provide a *presentation-oriented* perspective on task-driven adaptation according to the communicative goal. By contrast, data value manipulation affords *data-driven* aspects[2] of task-driven adaptation. In particular, the visual representation's *information scale* is influenced by a selection of relevant data elements and characteristics. Manipulation of mapping from data characteristics to graphical attributes influences its *visual scale*.

There are three principal approaches how to relate data-oriented pre-PAGE adaptation with FEATURES of that PAGE:

Aligned Adaptation: Only those data characteristics mapped to FEATURES and their properties are contemplated. Thus, filtering and mapping are effected to *create* task-specific FEATURES. Examples for the former include task-specific value thresholds to filter irrelevant data elements, the selection of the number of clusters for k-means clustering; and task-specific color codings for the latter (see Section 7.4.2).

Separate Adaptation: Data characteristics are adapted independently from the task at hand. Pre-PAGE adaptation is carried out primarily based on data-driven considerations.[2] Presentation-space FEATURES are thus only *influenced* indirectly, by changing data characteristics that serve as input to visual mapping operations. Examples include the selection of a cluster algorithm suitable for the given data, and the selection of default color codings according to established conventions in the application domain.

A combination of both: Data-space adaptation addresses both data-driven and task-driven aspects as outlined in the previous two headwords. This approach is preferable as it not only allows task-driven adaptation of visual representations but also to incorporate further data space manipulations that nonetheless affect FEATURE-based adaptation in presentation space.

For this reason in the present thesis the combination of (data space) pre-PAGE and presentation space FEATURE-based adaptation is pursued. In the following Section 7.2, we propose a concept how to integrate pre-PAGE adaptation with the basic adaptation approach introduced in Chapter 3.

[2] We here deliberately neglect further aspects of the Context of Use that are not in the context of this thesis. In particular, data space adaptation may also include user-related aspects (FOR WHOM) e.g., visual mappings according to personal preferences; and device-driven aspects (WHERE) e.g., data size reduction, see [Thi10].

How
The second question relates to how this adaptation is effected in data space within concrete smart visual interfaces. In previous chapters, it has been discussed how adaptation control is achieved using the enriched task model to parameterized an adaptation pipeline operating on content elements. This corresponds to the second half of the visualization pipeline. For pre-PAGE adaptation of abstract data visualizations it must therefore be extended to allow input data filtering and mapping operations as well. This is discussed in Section 7.3.

Section 7.4 further presents two examples for "Smart-X" techniques utilizing this concept. These examples have been intentionally chosen to highlight combined adaptation.

7.2 Abstract Data Preparation

Chi's Data State Reference Model (DSRM) [CR98] is one variant of the general visualization pipeline model. The idea behind the DSRM is to describe the visualization process as a pipeline of four *stages* – representing *data states* – and *operators* that manipulate and transform data between states (Figure 7.2).

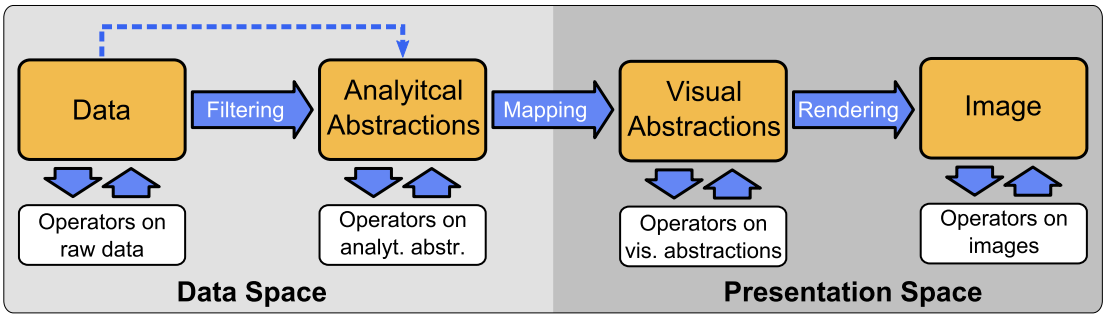


Figure 7.2: Schematic of the Data State Reference Model of the visualization pipeline, adopted from [CR98, Chi00]. Arrows indicate operators, yellow boxes represent stages.

The DSRM thereby comprises four data states or stages. These are, in order of traversal:

Data state represents the given input data elements for which a visual representation shall be generated. To emphasize this state comprises unprocessed data, it is sometimes referred to as *raw data*.

Analytical abstractions are derived from the input data through *filtering*. Data elements in this state constitute meta data or calculated properties of the visualized data. Examples are statistical moments (average, minimum, maximum, ...), and cluster properties like representative objects or means.

Visual Abstractions are created from the previous stages through *mapping* operations. This state comprises graphic primitives that encode information in geometric properties such as shape, orientation and size, as well as visual attributes like color and texture [Ber82].

Image data finally is the result of the rendering process i.e., a matrix of pixel values. This data state represents the data's visual representation.

As indicated in Figure 7.2 data states are understood to define both a data space (data and analytical abstraction stages) as well as the presentation or view space (visual abstraction and image stages). By comparison, visual data types discussed in previous Chapters 4–6 are defined exclusively in view space.

Data is transformed and propagated through the pipeline by operators of two different types, *stage operators* and *transformation operators* (cf. Figure 7.2):

Stage operators work within a single data stage, thus manipulating data of a given state – data, analytical and visual abstraction, image data.

Transformation operators convert data from one state to the next. Operators of this type thus comprise the principal transitions between visualization pipeline stages: filtering, mapping, and rendering.

The original design goal behind DSRM has been to integrate a description of the visualization pipeline's data flow with specifications of operators that actually realize an interactive display technique [Chi00].

Its basic concept, however, does make the DSRM well-suited as basis for task-driven adaptation of visual representations as well: both the DSRM and our adaptation pipeline approach incorporate the notion of data elements and (graphical) content elements, respectively, that are manipulated through a pipeline comprising distinct operators. In particular, describing the visualization process by a series of operators on individual data elements yields sufficiently fine-grained control of the image generation process to facilitate per-FEATURE adaptation of visual representations.

We first proposed the core idea of utilizing fine-grained operators as a means for adaptation based on a data state model in [FTS07], with a substantiated discussion published in [TFS08a]. This core idea has subsequently been further developed into a conceptual framework in a joint effort with Thiede [Thi10] within the MuSAMA project [MuS09]. It was designed to address two interrelated aspects of the context of use of visualizations within smart environments (cf. Section 2.2.2): the task-driven adaptation of the visual representation, which is the focus of the present thesis; as well as the device-driven adaptation with in the environments heterogeneous, dynamic device ensemble, which is the focus of [Thi10].

In the latter thesis, the modularity afforded by the DSRM has been utilized for the distributed generation of visual representations on available devices in the ensemble. For this, pipeline operators are executed as independent software services (service layer) marshaled by a control layer. Service composition is thereby specified on the functionality/interface level by so-called *pipeline templates* describing the visualization process. Depending on available resources and output device capabilities, appropriate service implementations are selected. For details, see [TTS09, Thi10].

However, Thiede discussed *only* challenges arising on the service and control layers. While he acknowledged the need for a superordinate model layer to provide context-specific – in particular, task-specific – pipeline templates (see [Thi10], p. 100–103), his thesis deliberately excluded this layer.

To this end, the following Sections detail how the DSRM-based approach is utilized for the specification of WHAT and WHY aspects (PAGES and FEATURES, respectively) of task-specific visual representations of abstract data.

7.2.1 Definition of Pages

A PAGE functions as a container for graphical content that provides a default representation. This allows to associate PAGES with task nodes in the general task model without strictly requiring further specification of its task context (cf. Section 3.3). Thus for abstract data, a PAGE definition comprises two main parts:

- the input ('raw') *data source*, e.g., a binary or character data file or a database
- information on *how this input data is transformed* into graphic primitives, in other words, which visualization technique is employed for the data and task at hand.

The first part is pre-determined by the problem domain – it is the data the user is currently working with through the smart visual interface. On the other hand, the choice of a suitable visualization technique is either

- made explicitly by a visualization expert during the design of the smart visual interface, or
- effected automatically based on the data type and the current (low-level) abstract task. Section 2.3.2 reviewed several task-by-data type taxonomies that are utilized for this purpose.

The disadvantage in using low-level task taxonomies exclusively is that it does not allow to utilize the extended task context provided by the task model: visual encoding are selected based on the visualization task type of the current working step in isolation. Therefore, we propose to combine a default visualization specified as PAGE on the level of the domain task model with automatic, task-driven adaptation within the smart visual interface instead. Definition of PAGES is part of the interface design performed by a visualization expert (author). Adaptation of particular aspects of the default visualization for individual working steps is then automatized based on the low-level task type.

For purposes of PAGE definition, the chosen visualization technique's default representation comprises at least the operators on first three data stages of the DSRM (i.e., up to visual abstractions) as well as the filtering and visual mapping transformation operators (cf. Figure 7.2).

Since the visual abstraction stage comprise graphic primitives, the PAGE definition may additionally require an initial viewpoint definition. In this regard, 2D visualizations are akin to 2D vector graphics where graphic primitives are given as geometry in $\mathbb{R}^2$ (see Section 5.2). 3D visualizations are comparable to 3D graphics, where rendering of 3D geometry further requires set-up of a virtual camera and lighting information (see Section 6.1.1). From the perspective of the DSRM, these information are parameters of rendering transformation operators.

Operator configuration and their default parametrization constitute a DSRM *pipeline template* in much the same way as proposed by Thiede [Thi10]. Pipeline templates

therefore assume the role of the PAGE manifest employed for the previous visual data type.

In line with the approach taken for the previous visual data type, these information are associated with the task by annotating task model nodes. Recall from Section 3.3.2 the main idea behind the annotation strategy is to annotate the least common ancestor node of basic tasks that share the same graphical content to minimize the amount of additional information that needs to be provided for each of the respective working steps.

Because the PAGE of a data visualization comprises not just plain graphical content but two parts, this strategy is extended for abstract data content:

- The PAGE's data source is associated with the common ancestor node of all composite tasks that work with that data set. If the entire modeled workflow operates on the same data, this will be the task decomposition's root node.
- Several composite task may operate on the same data set but require different visualization techniques (i.e., visual mappings of data characteristics). Thus, the second part of the PAGE manifest – the specification of operators transforming input data elements into graphic primitives – is annotated at the corresponding composite task's root node.
- Individual working steps of the composite task then define FEATURES that adapt specific aspects of the base visualization (see next Section).

Figure 7.3 illustrates this concept for a hypothetical analysis scenario where a given data set (a credit card transaction database) is analyzed for cases of fraud. Different visualization techniques (DSRM templates) are associated with specific sub-tasks depending on the task goal.

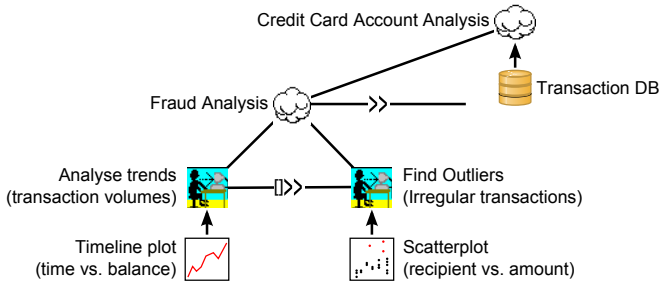


Figure 7.3: Annotation of different visualizations (DSRM templates) per sub-task applied to the same data set. This extends the general annotation strategy outlined in Section 3.3.2: the data set itself is associated with a high-level node in the workflow decomposition; refined by information on more specific sub-task nodes.

7.2.2 Definition of FEATURES

The purpose of FEATURES is to designate specific aspects of the PAGE's content that are relevant in the context of the task at hand. For the previous graphical content types,

FEATURES are defined by a RoI in presentation space (2D vector and 3D graphics) or even directly in image space (raster images). This made definition of FEATURES a largely manual task due to missing semantics about the depicted content; support through automatic methods is only punctually viable.

By contrast, defining FEATURES over the visual representation of abstract data can utilize information inherent in the input data itself, as well as derived meta data i.e., analytical abstractions in the DSRM. By first identifying data characteristics of interest, a subset of data elements with matching attribute values is selected. This effectively creates a RoI in data space. Visual mapping transforms data elements into graphic primitives; a FEATURE thus comprises the graphic primitive(s) that a data elements from the data-space RoI is mapped to.

There are numerous methods to designate data characteristics of interest depending on the data and problem domain. Examples include user-specified value ranges for data attributes; derived or explicit thresholds that select any data elements with attribute values exceeding that threshold; and the application of hierarchical or density-based clustering algorithms to identify outliers (see e.g., [BEPW00, HK00]). Analytical abstractions such as clustering further influence potential FEATURES e.g., by changing the number of clusters formed whereby each cluster is mapped to a visual abstraction corresponding to a FEATURE.

An interesting approach in this regard is presented by Tominski [Tom06] that is based on the notion of data events that identify data elements of interest by predicate logic expressions. It has primarily been incepted as a tool for visualization of dynamic (time variable) data sets. However the general concept is well applicable to task-driven adaptation as well. Here, different sets of event are associated with different tasks.

The preceding discussion notwithstanding, it is of course also viable for a content author to manually define FEATURE regions in presentation space directly i.e., a selection at the level of graphic primitives. Depending on the type of visualization – 2D or 3D – these explicit definitions follow the procedures outlined in Sections 5.1.2 and 6.1.2, respectively.

In Section 7.4.1 we present our own "Smart Lenses" approach that allows arbitrary combinations of implicit, data-driven FEATURE specification with presentation space-only definition of FEATURE RoIs.

7.2.3 Feature Relevance Values

Even more so than the definition of presentation-space FEATURES themselves, specification of task-specific relevance values is, by necessity, a strictly manual task; this has been argued in Section 4.1.3.

This is less strict for FEATURES defined over abstract data visualizations. The same mechanisms to define FEATURES implicitly by first identifying data characteristics of interest are useful to derive their task-specific relevance (semi-)automatically. This is achieved in the following ways.

Relevance is derived as a function of attribute values and a reference value. Examples include element distance to reference objects (e.g., cluster representatives); or proportional to the magnitude a data element's attribute value departs from a given threshold. The threshold is a task-specific value that is typically specified explicitly by the content author. In some cases reference values are even pre-determined by the application domain e.g., allowable limits for temperature, pressure etc. In these cases, FEATURE relevance could be derived fully automatic.

Relevance is derived based on a scoring mechanism. If multiple data attributes are mapped to a FEATURE, relevance is proportional to the number of mapped attributes that match data characteristics of interest (e.g., the number of data events [Tom06] joined on the FEATURE). Relevance therefore conveys the general notion that graphic primitives that encode a greater number of relevant data characteristics are more relevant than those representing only a single characteristic of interest.

Both approaches can be combined: reference value functions are applied for each mapped attribute individually; the resulting FEATURE relevance is then e.g., the maximum (or average) relevance score over all mapped attributes.

Of course, as with FEATURE definition itself, a content author can always provide an explicit relevance value independently of the encoded data values through a corresponding task node annotation.

7.3 Adaptation Control

The DSRM introduced in Section 7.2 represents the framework for task-based adaptation of data visualizations as follows:

- Operators in the base data state model (describe by the PAGE) determine how the unadapted visual representation is generated.
- Operators on different stages of the adaptation pipeline correspond to stage and transformation operators in the DSRM.
- The FEATURE information provided by the enriched task model determines what operators and their parameterizations describe a task-specific visual representation of the data set. Thus, adaptation entails adding, replacing, or re-parameterizing DSRM operators.

The central aspects of how PAGES and FEATURES relate to the DSRM and the data states it comprises were covered in the previous Section. In the following, we will explain how different data states and DSRM operator types relate to the conceptional adaptation pipeline introduced in Section 3.3.4. Section 7.4.1 describes our "Smart Lenses" approach used to realize this DSRM-based adaptation.

7.3.1 View selection

For the visual data type contemplated so far view selection has been limited to the view space because these only comprised view space data (raster images) or visual primitives in presentation space. Information hiding on these types is thus effected by selectively removing FEATURE geometry from the visual representation.

By contrast, during abstract data visualization the generation of visual primitives comprises explicit mapping operations from analytical abstractions to graphic primitives. Therefore, view selection selection is effected by either of two ways:

- by specifying a two-dimensional region in view space, or
- by a selection of value ranges on variables.

The first methods works identical to view selection for 2D vector and 3D graphics for 2D and 3D visualization techniques, respectively. It corresponds to a *graphical zoom* of the representation.

On the other hand, selecting value ranges on variables define a section of the observation domain. Rendering the corresponding subset of visual abstractions to a given display area thus results in a zoomed-in view. This method of of view selection is usually far more effective for data visualizations.

Consider for example a 2D scatterplot that maps data elements to points in a plane where point positions are determined by two attribute values corresponding to one axis each. Selecting a 2D rectangle in view space and stretching it to the available display area will result in a zoomed-in view, but will also cut off the axis representations indicating attribute values encoded in the visible points. By contrast, selecting a reduced value range for both axes allows to keep the axis representations on screen while altering what region of the data space is rendered to the available display area. Therefore, view selection performs *semantic zooming*. In addition, because data elements are filtered out before being mapped and rendered, generation of the visual representation likely is computationally more efficient.

7.3.2 Geometry adaptation

Geometry data is generated as a result of the visual mapping transformation operation. As such, geometry adaptation comprises three options:

- Existing DSRM operators are re-parameterized. This results in a change of position, shape, and size of graphical primitives depending on how data values are mapped to these geometric properties.
- Additional geometry is added by introducing additional operators that map previously non-shown variables to new graphic primitives. This allows to adjust the information and visual scales of the visual representation for zoomed-in detail views [Shn96]. In particular, it allows to encode additional information (variable values) of important FEATURES.
- Existing DSRM mapping operators are removed from the pipeline. This removes the visual encoding of the corresponding variables, thus implementing information hiding.

By applying these modifications to the DSRM only within a FEATURE region, local adaptation according to FEATURE relevance is effected. This is detailed in Section 7.4.1.

7.3.3 Visual attribute modification

Visual attributes are determined in parallel to FEATURE geometry[3] as the result of visual mapping transformation operations by mapping attribute values to fill and line colors, stroke width etc. Therefore, visual attribute modification is effected by the re-parametrization of existing DSRM mapping operators. Note that it is rarely useful to add or remove mapping operators on visual attributes: adding operators will conflict with existing visual mappings (each primitive can have only one stroke color), while removing visual mappings leave aspects of the data's visual representation undefined (a primitive with neither stroke nor fill color is not visible).

7.3.4 View Space Manipulation

By definition, this adaptation pipeline stage operates on the result of the image generation process in 2D view space (cf. Section 3.3.4). This corresponds to DSRM operators on the final image data state. Manipulations are effected by adding respective image operations to the DSRM pipeline on this stage. See Section 4.2.4 for a discussion of viable image-space adaptation operations.

7.3.5 Labeling

Our approach uses image-based labeling. Conceptually, labeling can thus be seen as an operator on the image data state of the DSRM as well. It is therefore subject to the exact same requirements and procedures as those outlined in Section 4.2.5 for raster images using particle-based labeling [LSC08].

However, to facilitate task-specific label sets and labeling styles our space-efficient labeling approach 4.3.2 is a viable alternative for abstract data visualizations as well. The only variance is in the way the required ID-buffer is generated from the geometric primitives. For 2D visualizations, the method described in Section 5.2.5 for 2D vector graphics is appropriate; whereas for 3D visualizations, the generation method outlined in Section 6.2.5 for 3D mesh representations is more suitable.

Note that because the ID-buffer is effectively another raster image, its generation can be seen as a rendering transformation operator in the DSRM that transforms visual abstractions (primitives) into a color-coded image.

[3] Note that the well-known graphics semiology by Bertin [Ber82], instead of distinguishing geometric properties (position, shape, size and orientation) and visual attributes, treats these uniformly as *visual variables*. This classification focuses on possible visual encodings of data values to perceivable graphical properties. By contrast, we here maintain the distinction driven by the presentation-oriented visual data type that comprise graphic primitives i.e., geometry with styling attributes.

7.4 Smart Visualization Techniques

In this section, we provide details on two "Smart-X" techniques for the basic adaptation approach of abstract data proposed in this chapter. Both techniques constitute examples for the combination of data- and task-driven aspects of adaptation.

In Section 7.4.1 we present a concept for adaptation of data visualizations within FEATURE regions based on the the DSRM. The core idea is to utilize so-called lenses to specify FEATURE-based adaptations in both data and presentation space. We proposed the core idea of DSRM-based lenses in [FTS07], with further details discussed in [TFS08a].

Color coding is a fundamental visualization method for representing scalar values, and is therefore widely used in a large variety of application scenarios. To generate expressive and effective visual representations, it is extremely important to carefully design the mapping from data to color. Section 7.4.2 describes a color coding approach that accounts for the different tasks users might pursue when analyzing data. It has initially been published in [TFS08b].

7.4.1 Smart Lenses

So-called lenses are filters that are placed – usually interactively by the user – over the presentation, thus altering the visualization in a locally confined RoI [BSP+93]. Multiple lenses can also be 'stacked' to combine filter effects [BSP+93, FTS07]. This makes lenses a suitable tool for adaptation in task-specific FEATURE regions as well. Smart in this context thus means that rather than depending on the user to select and place lenses manually, the lens' *initial* placement is derived automatically, corresponding to FEATURES relevant to the task at hand. Of course, the user can also re-position, remove or replace lenses interactively to manipulate this initial view.

A lens is generally defined by two parameters: its *region of effect* and the *lens function*.

The **region of effect** determines which data elements are considered inside (are affected by) a particular lens and which elements are outside or unaffected by it. Typically, this region is determined in presentation space by a boundary *shape* and its *position* in the view. For this, often simple shapes such as rectangles, circles are used for 2D representations [BSF+94, BSP97], as well as spheres or cubes comprised of clipping planes in the 3D case [VCWP96].

In our approach, a lens' region of effect is given by the FEATURE region. In Section 7.2.2 it has been argued how FEATURE regions are defined in one of two ways: directly in presentation space, which is identical to the usual definition above; or implicitly by selection of a data space RoI (data elements of interest) which are subsequently mapped to graphical primitives.

In the latter case, the FEATURE (and thus, lens) region is therefore determined by the resulting set of primitives. This facilitates data-driven changes to its shape. An example is given in Figure 7.5 where FEATURES over a map visualization are defined via selection of administrative districts as elements of interest in data space; the resulting lens region in presentation space corresponds to the respective area's geographic boundaries.

The **lens function** determines what operations are applied to elements in the lens' region of effect. In the context of the DSRM, a lens can therefore be understood as an operator within this framework. Depending on the data state(s) the lens operates on, lenses can be broadly classified as semantic and graphical lenses that modify elements in data space and presentation space, respectively. In [GFS05] we proposed a more complex classification scheme for general lenses.

The relevant distinction here is that graphical lenses operate only on graphical primitives and/or image data i.e., on the final two data stages of DSRM pipeline. Graphical lenses are therefore capable of realizing all adaptation operations discussed in Chapters 4–6 for raster images as well as 2D vector and 3D graphics, respectively.

On the other hand, semantic lenses operate on the input data and analytical abstraction data states. This allows data-space adaptation not available in the previous visual data types. In particular, it facilitates adjustment of the information scale in the FEATURE region by adding, removing or altering analytical abstractions; as well as the corresponding visual scale by altering how the set of analytical abstractions is mapped onto visual abstractions (primitives).

Stacking multiple lenses with different lens functions further allows to apply multiple adaptation operations to FEATURES. In particular, by stacking semantic and graphical lenses combined adaptation in both data and presentation space as proposed in Section 7.1 is achieved.

The versatility of this operator-based concept for lens definitions is illustrated by the following examples.

- Lenses as *stage operators* on *data values* can control the ratio of visualized values in the lens region to avoid clutter (sampling e.g., [ED06, ED07]), remove errors or interpolate new data.
- Lenses as operators on *analytical abstractions* can be used to sample the corresponding data elements on this stage, as well as to show additional informations about the data e.g., cluster properties.
- Lenses as operators on *visual abstractions* can for example change the LoD of a given surface mesh [Kea98, FHS08] or generalize (simplify, aggregate) shapes in the lens region [IDE].
- Lenses as operators on *image data* realize standard per-pixel image modifications like hue or contrast adjustment, which e.g. can be used to overcome color vision deficiencies, or pixel-based distortions [Kea99].
- Lenses as *filter operators* offer the possibility to calculate additional data characteristics on demand, like node metrics in a large graph [LH94], and vice versa, to filter out unimportant information like crossing edges to reduce clutter [TAHS06].
- Lens as *mapping operators* modify the mapping process and thereby decide about the FEATURE's visual representation. This can be used to visualize additional or different parameters, cf. e.g., [Kea98, BSP+93, BSP97]; or to adjust existing mappings in a task-specific way. Section 7.4.2 discussed task-driven color coding as one of the most important aspects.

- Lenses as *rendering operators* realize graphical lenses, for example distortion lenses like Rase's Cartographic Lens [Ras97, LA94].

Furthermore there are lens functions that consist of more than one operator on multiple pipeline stages, if for example rendering is modified according to semantic information from the data domain. A representative for this kind of functions is the Semantic Depth-of-Field approach by Kosara et al. [KMH01], cf. Section 4.2.4.

As the examples show, lenses as FEATURE adaptation operators can be applied on every stage of the model in a useful way. In doing so, there are two principal ways to integrate lenses into the visualization process [ED06]:

- Two separate pipelines are used, one for creating the unadapted visualization and a second one for the lens region. The result of lens manipulation is composed over the base visual representation in image space.
- The entire visual representation including FEATURE/lens regions is produced using a single pipeline. The operators constituting the lens function alter data elements on the different stages directly.

Although with the first approach is easier to modularize – each additional lens spawns an new visualization pipeline – it can not be used for stacking lenses, as each processes the data independently. In the present thesis therefore the approach of a single integrated visualization pipeline has been pursued, cf. [FTS07, TFS08a].

This integration poses the following aspects must be addressed in an implementation of the proposed framework:

- A means to ensure *consistency* between FEATURE/lens regions in data and presentation space.
- Handling of overlapping FEATURES. In these cases, lenses operate on the same set of data elements that contribute to the intersection region. This may cause *conflicts* between operators regarding data element access. This also applies if multiple lenses are applied to a single FEATURE.
- A means to describe *lens definitions* and parametrization of its operators suitable for task model enrichment. This description should abstract from specific data source formats and implementation specifics so it can be defined on the conceptual level of the task model (cf. Section 3.2).

Lens region Consistency

The challenge is to map FEATURE/lens regions from one pipeline stage onto the other and vice versa. If the RoI has been specified at the first stage (raw data) this is not a problem. The transformation operators between data states map the lens region onto the following stages.

However, this becomes an issue when the FEATURE region is defined in presentation space (i.e., as a shape in image space, or by a selection of primitives) but the lens

function operates in data space. In this case we additionally need an inverse mechanism, or back-projection, to collect the data elements associated with this region on the earlier stages.

There are two approaches to the inverse mapping required for the second case:

- If an inverse operator to the lens function exists, it can be used to back-project the FEATURE region on earlier stages of the DSRM. Sometimes, however, inverse operators are not viable or its results would be ambiguous. Finding an inverse operator (automatically) is often also not trivial.

- Another approach is to use multiple passes of pipeline execution. During the first pipeline pass, lookup tables are generated. These contain, for each data element on a given stage, the associated input elements for the transformation operators from the respective previous stage. During the second pass, data elements contributing to the lens regions on the different stages are processed according to the lens function.

A disadvantage of the second approach is the potential memory requirements for the lookup tables, especially for large data sets. Two passes also increase the amount of computational resources required. However, we still consider it the better choice because we can not assume that all inverse operators are given for arbitrary data sets and visualization techniques.

Lens conflict detection and resolution

Combining operators of multiple lens functions contributing to a single FEATURE's region yields three types of potential conflicts (cf. [TFS08a]):

- The results of two lenses overwrite attributes of the same data elements on the same data stage, referred to as a *write conflict*.

- One lens L_1 operates on data elements which attributes are modified by another lens L_2's output. In this case lens L_1 *depends* on L_2.

- If there are two or more lenses which have a potential *write conflict* between their results and a third lens uses these contested data elements as input, a *read conflict* occurs i.e., an ambiguity about which of the two possible result sets the third lens operator should use.

Based on the assumption that the non-adapted visualization pipeline itself (i.e., without applied lenses) is void of conflicts, a single lens will never result in a conflict as lens functions on principle always take precedence over regular pipeline operators. Furthermore, read conflicts can only occur if there are unresolved write conflicts. Solving the triggering write conflict automatically cancels the read conflict as well. Dependency conflicts between operators on different pipeline stages are trivially avoided by executing lens operator in stage order; for operators on the same stage the operator that writes to a contested attribute must execute before the one reading it (see below). Only write conflicts can not be solved by intrinsic means.

Before they can be resolved conflicts need to be detected first. To this end, formal description of the output elements of the lens operators is required. We use a mathematical set notation as follows (cf. [FTS07, TFS08a]).

Each of the four stages in the DSRM is represented by a set A_i of attributes a_{ik}; $1 \leq i \leq 4; 1 \leq k \leq |A_i|$ that describe data elements at the corresponding stage i.e., A_1 correspond to *Raw data* attributes, whereas elements from A_2 represent attributes of *analytical abstractions*, and so on.

A lens operator f transforms data elements with associated attributes from a source set S_n^f (n^{th} stage) to a target set T_m^f (m^{th} stage):

$$S_n^f \subseteq \bigcup_{1 \leq i \leq n} A_i, \quad T_m^f \subseteq \bigcup_{m \leq i \leq 4} A_i$$

This formalism allows us to detect, and in some cases to automatically resolve, the aforementioned conflict types. Let

$$f : S_n^f \rightarrow T_m^f; 1 \leq n \leq m \leq 4, \text{ and}$$
$$g : S_p^g \rightarrow T_q^g; 1 \leq p \leq q \leq 4.$$

A potential *dependency conflict* occurs if $T_m^f \cap S_p^g \neq \emptyset$. In this case g depends on f – f writes attribute values that are used as input to g – and therefore to avoid dependency conflicts f must be executed before g. This guarantees attributes S_p^g are only processed by g set after all modifying operations have completed by execution of f.

A *write conflict*, on the other hand, occurs if $T_m^f \cap T_q^g \neq \emptyset$ i.e., f and g both modify one or more contested attributes. As mentioned above, although this type of conflict can be detected, it does not lend itself to an inherent solution.

However, smart lenses must have the ability to resolve write conflicts automatically. We here addressed this problem by using the corresponding FEATURES relevance values to determine which lens operator takes precedence.

Lens Description for Task Model Enrichment

To describe the parametrization of smart lenses in a form suitable for task node annotation we employ declarative *lens scripts*. A lens declaration thus comprises the following information (cf. Figure 7.4):

- the FEATURE/lens region, as well as
- the operators comprising lens function for FEATURE adaptation.

The lens region is either a fixed geometric shape defined in view space (Figure 7.4a); or a RoI in data space, cf. Section 7.2.2. In the latter case the actual FEATURE region in presentation space is data-driven. For this, *data characteristics of interest* are specified. This can be expressed by means of suitable filter conditions. We here use a notation inspired by the OGC Filter Specification [Ope05]. Its XML syntax allows to express complex nested conditions including spatial (presentation space), arithmetic and logic constraints, cf. Figure 7.4b.

The *lens function* can be selected from a list of predefined functions (i.e., available as plug-in classes within the concrete smart visual interface implementation. Depending

```
...
<shape type="fixed">
    <circle>
        <center>0 0</center>
        <radius>50</radius>
    </circle>
</shape>
<!-- data-driven position: lens position updated internally by operator class -->
<position type="data-driven"/>

<operator class="lvis.smartLenses.MagicEyeLens">
    <paramDefaults>
        <magnify>2.0</magnify>
        <maxdist>50</maxdist>
    </paramDefaults>
</operator>
<operator class="lvis.smartLenses.RiverIcon"/>
...
```

```
...
<shape type="data-driven"/>
<position type="user-driven"/>
<operator class="lvis.smartLenses.TextureLens">
    <Filter>
        <And>
        <PropertyIsEqualTo>
            <PropertyName>krankheit_nr</PropertyName>
            <Literal>5</Literal>
        </PropertyIsEqualTo>
        <PropertyIsBetween>
            <PropertyName>datum</PropertyName>
            <LowerBoundary>2000-01-01</LowerBoundary>
            <UpperBoundary>2000-12-31</UpperBoundary>
        </PropertyIsBetween>
        </And>
    </Filter>
</operator>
...
```

Figure 7.4: Examples of lens declaration scripts used for task node annotation. On the left is the definition of the lens from Fig. 7.5c. The script on the right defines the Texture lens from Fig. 7.5d, including a composite filter defining a RoI in data space.

on the type of operator offered by a plug-in, the declaration may be further attributed with parameter settings, see Figure 7.4a.

Thus our Smart Lenses approach to FEATURE-based adaptation can be summarized as follows: on the conceptual level of the task model FEATURES/lenses are specified by annotating corresponding task nodes with declarative scripts. Scripts therefore abstract from the actual implementation of the visualization process, and can be reused or modified for other data sets and visualization settings. At runtime of a concrete smart visual interface corresponding DSRM operators are provided as plug-ins. Subsequently, the adapted visual representation is generated By processing the operator pipeline of the DSRM augmented with task-specific smart lenses, an adapted initial visual representation is generated for the task at hand.

Figure 7.5 shows smart lens examples from our prototypical implementation of a visualization system *LandVis-Lens* for health care data based on the DSRM operator framework [FTS07, TFS08a]. The base visualization shows a map divided into areas correlating to administrative districts on various levels (federal states, districts, ZIP code areas). Different choropleth representations and icon techniques are available for the incident analysis of several diseases (e.g., flu or respiratory problems). The images show different combinations of lens regions with FEATURE adaptations on different stages of the DSRM.

7.4.2 Smart Color Coding

Designing appropriate color scales is an intricate step that influences the expressiveness and effectiveness of visual representations significantly. *Appropriate* in this context means that the color-coded visualization really supports analysts in deriving valid statements about the underlying data. Appropriateness can only be achieved if characteristics

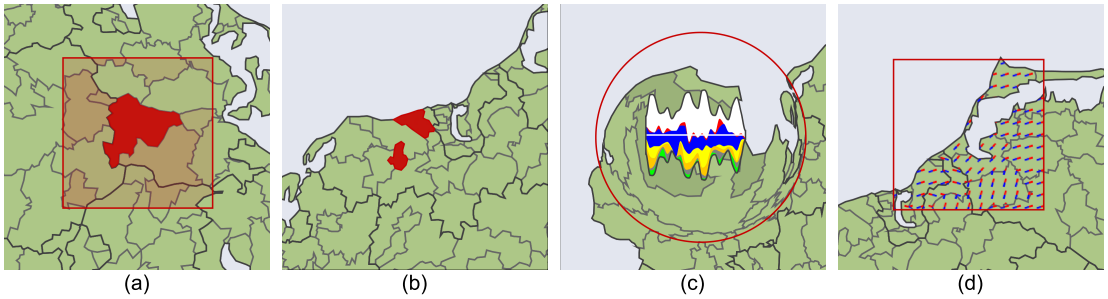


Figure 7.5: Examples for Smart Lens adaptation. (a) Lens modifying the fill color of map areas according to a variable of task-specific interest, lens shape set to specific geometric shape in view space; (b) same lens, but data-driven shape derived from map area boundary. (c) geometric distortion of primitives to locally explore a task-specific map region of interest. (d) generation of additional graphic primitives (oriented icon texture similar to [PG88] encoding extreme values).

of the data and the task at hand are taken into account. Therefore, it is necessary to provide flexible color schemes that can be adapted to the data and task at hand.

Generally, a color coding scheme can be characterized by a color mapping function $f : D \rightarrow C$ that maps data values D to colors from the color scale C.

From literature we know several approaches that address adequate definition of the color mapping function, or give guidelines for the use of color scales. In this context, Brewer's *ColorBrewer* has to be mentioned as a pioneer work [Bre94, Bre99, HB03]. In Schulze-Wollgast [SWTS05], the influence of data characteristics (e.g., the distribution of data values) on color coding is investigated in more detail. General overviews on the color coding problem are given in [Sto03] and [Sto07], including aspects of basic vision and psychophysics, color reproduction, and color design. For a more in-depth discussion of related work refer to our publication [TFS08b].

In this Section, we will discuss color coding with regard to the task at hand. Previous publications, like [BRT95] or [SWTS05], only address the design of color scales for specific tasks (e.g., isomorphic, segmentation, comparison) while others, like [Rhe99] or [Tel07], just point out that the goal of the user has to be considered when using color coding.

Requirements for Color Coding

A fundamental requirement for effective color coding is that the color mapping function f has to be invertible. This means that every data value (or every well-defined group of data values) is associated with exactly one color, and vice versa, every color represents a fixed range of data values. In other words, colors encoded from two different data values should be visually distinguishable. On the other hand, visually similar colors imply that they represent data values that are close to each other.

Besides these basic requirements, further aspects decide about effectiveness of color coding. Telea identifies the following factors to be relevant for color coding [Tel07]:[4]

Characteristics of the data: Statistical features, overall distribution of data values, as well as data variation speeds and domain sampling frequencies are data characteristics should be considered when designing color coding schemes.

[4]For a more detailed description including illustrative examples for each aspect see [TFS08b].

Characteristics of tasks: Different tasks require different color coding schemes. A main distinction here is whether the task requires the comparison of exact values (i.e., quantitative analysis) or the judgment of qualitative differences. Furthermore, task-specific RoIs in the data domain should be accentuated, for instance by using bright, warm, and fully saturated colors.

Characteristics of the user: Individual color perception varies from user to user due to capabilities and the cultural as well as professional background. This obviously includes color blindness but also application domain conventions such as placing colors in spectrum scales in order of increasing wavelength vs. increasing frequency.

Characteristics of the output device: Different output devices use different systems to define and display colors. Thus, a color coding scheme being appropriate for displaying data on a computer display might be inappropriate to show the same data on printed materials.

This is in line with our categorization of context of use (cf. Section 2.1.2) and addresses four aspects: WHAT is visualized (data), WHY and FOR WHOM is it visualized (task respectively user), and WHERE is the output displayed (device). True to the focus of the present thesis, we here investigate the WHY aspect in more detail in particular.

Typification of the Task at Hand

Task-driven color coding requires the specification of type and goal of the task at hand as a basis. Most early task descriptions are given as verbal lists of visualization goals, including identification, correlation, comparison, and others (cf. Section 2.3.2). These descriptions lack formal description, and hence, concrete understanding of the tasks can vary. Recently, Andrienko & Andrienko proposed a task model that is based on formal definitions [AA05]. This allows for a precise typification of visualization tasks.

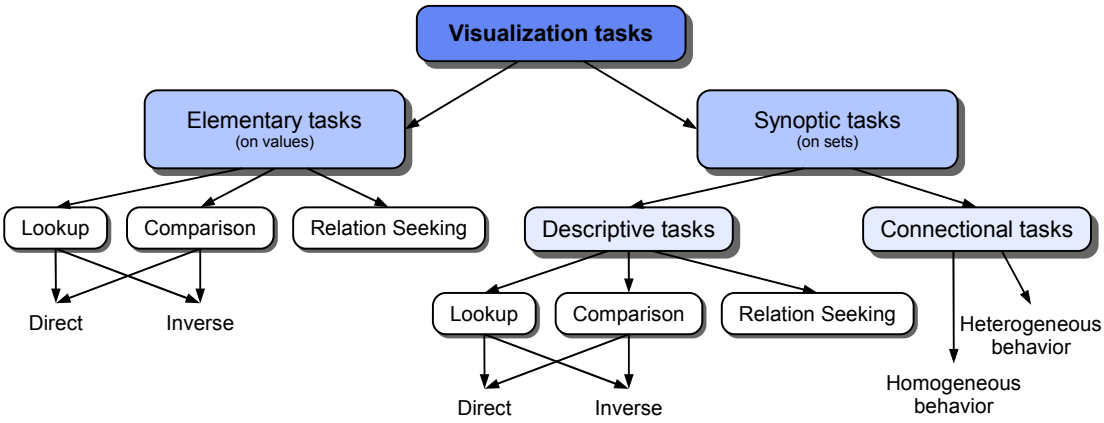


Figure 7.6: Task typology used for our task-driven color coding approach (adapted from [AA05]).

The formalism behind the typology of Andrienko & Andrienko uses two basic notions: *references*, the space where data values have been collected, and *characteristics*, the space of variables that were collected (cf. Section 7.1).

On its first level, two classes of tasks are distinguished: *elementary* and *synoptic* tasks. Elementary tasks address individual data elements. This may include groups of data,

but the main point is that data values are taken into account separately and are not considered as a whole. Synoptic tasks, on the other hand, involve a general view and consider sets of values in their entirety.

Elementary tasks are further divided into *lookup*, *comparison*, and *relation seeking*. The lookup task defines a search for data characteristics. This includes both direct and inverse lookup, depending on if references are given and corresponding data values are sought (also referred to as *identification*), or data values are given and associated references are of interest (also called *localization*). Relation seeking tasks search for occurrences of relations specified between data characteristics or references. In a broader sense, comparison can also be seen as relation seeking, but the relations to be determined are not specified beforehand. Direct comparison tasks relate characteristics, whereas inverse comparison tasks search for relations between references.

Synoptic tasks are divided into *descriptive* and *connectional* tasks. Descriptive tasks specify the properties of either a set of references or a set of characteristics. The first case belongs to the group of *identification* tasks. Here, a set of references is given, and the task is to find a pattern that describes the behavior of the given reference points. The second case belongs to the group of *localization* tasks. Here, a concrete pattern is given, and the task is to search for those reference points that exhibit the pattern. Besides specifying the properties of a set of characteristics or references, the *comparison* of those sets is of high relevance. As in the case of elementary tasks, we have to distinguish between direct and inverse comparison tasks, depending on whether a set of references or a set of characteristics is compared. Moreover, the synoptic task *relation seeking* considers two sets of characteristics or references to come up with relationships between these sets. In contrast to descriptive tasks, connectional tasks establish connections between at least two sets taking into account the relational behavior of two or more variables. Depending on the set of underlying references – either variables are considered over the same set or over different sets of references – *homogeneous* and *heterogeneous* behavior tasks are distinguished.

To achieve task-driven color coding based on this typology, we have to identify color maps that are appropriate for the different tasks. The first-level categorization of tasks draws a distinction between individual data values and sets of data values. This can be reflected by applying either continuous/discrete color scales, where each data value is encoded by a separate color, or segmented color scales, where each color stands for a set of data values. At the second level, the tasks lookup and comparison are of particular interest with regard to the color coding problem (see [SWTS05]). The lookup task requires color scales that support differentiation of sought data values. In order to accomplish comparison tasks, all variables involved in the comparison must be represented by a unified color coding scheme; perceptual separation of colors for the individual variables is not the primary goal here. Usually, the same holds true for relation seeking and connectional tasks. The third level addresses identification and localization tasks. Both problems demand for different color scales: identification (direct lookup or direct comparison) requires recognizing characteristics as precisely as possible, whereas localization (inverse lookup or inverse comparison) requires easy recognition of those spatial references that exhibit certain characteristics of interest. In the latter case, color coding

schemes that support accentuation and de-accentuation are suitable. The bottom line is that we have to consider three alternatives to come up with an initial approach to task-driven color coding:

- Individual values vs. sets of values,
- Lookup vs. comparison, and
- Identification vs. localization.

In the following section, we will discuss color coding schemes for these alternatives in more detail.

Color Coding for Specific Tasks

Color coding individual data values requires unsegmented color scales, i.e., every color $c \in C$ represents exactly one value in $d \in D$. Contrary to that, segmented color scales should be used to encode groups of data values, i.e., every color in $c \in C$ represents a subset $D' \subset D$. The specification of both of these types of color scales is a well investigated problem [BRT95, HB03, SMS07]. Figure 7.7 shows examples of unsegmented and segmented color scales for the identification and the localization tasks. The segmented color scale for identification represents five different value sets. The color scales for localization are designed in such a way that they support pre-attentive recognition of reference areas (i.e., FEATURES) of interest. The segmented color scale for localization supports a binary decision: Areas drawn yellowish match the selection criteria, other areas do not. The unsegmented color scale communicates a smooth selection of reference areas comparable to smooth brushing [DH02].

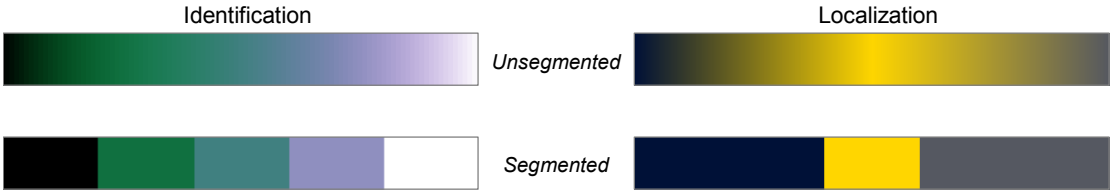


Figure 7.7: Unsegmented and segmented color scales for identification and localization tasks.

Designing appropriate color coding for lookup and comparison tasks requires additional effort. We will introduce two new concepts for this purpose next.

Color Scales for the Lookup Task

Lookup tasks are basically a search for concrete characteristics or references. This search can be facilitated by applying appropriate color coding schemes. While the inverse lookup task is relatively easy to handle, the design of adequate color scales for direct lookup is intricate. Schulze-Wolgast et al. propose the extraction of statistical meta data to adapt a given color scale accordingly [SWTS05]. The adaptation process includes three steps: (i) Expansion of the value range to be mapped onto the color scale such that the lower and upper bounds are intuitive to interpret, (ii) adjustment of control points of the color mapping function to improve the color coding for data-dependent segmentation or highlighting, and (iii) skewing of the color mapping function (e.g., applying logarithmic

or exponential mapping functions, rather than linear ones) to handle data ranges with special value distributions.

Here, we use two approaches to improve color coding for the lookup task: *histogram equalization* and *Box-Whisker plot adaptation*. Both methods address the problem that certain value distributions can lead to situations where the majority of data values is represented by only a narrow range of colors. This is unfavorable for lookup tasks. By histogram equalization and Box-Whisker plot adaptation, we spread the colors according to the data's value distribution and achieve that more colors are distinguishable in dense parts of the data.

Histogram equalization: Histogram equalization was originally introduced to improve the contrast of gray-scale images. The same concept can be used to adapt a given color scale according to the value distribution of the data at hand, and thus, to improve the perceptibility of the color-coded visualization. The procedure can be described as follows. First, the value range is subdivided into uniform bins, and the number of data values falling into the bins is computed. Second, the color scale is sampled according to the same uniform subdivision. The corresponding sample points, which represent specific colors, are then shifted based on the computed cumulative frequencies. Finally, a linear color mapping function is applied to establish a continuous color scale. As a result, more colors are provided for those segments that contain a higher number of data values, making values in high density regions easier to distinguish.

Box-Whisker plot adaptation: An alternative procedure to adapt the color coding to the characteristics of the value distribution is to utilize Box-Whisker plots. Box-Whisker-Plot adaptation subdivides the data range based on quartiles and inter quartile range (IQR), which are commonly accepted features to describe value distribution. Quartiles and IQR are more robust against outliers than other statistical indicators. The Box-Whisker plot segmentation reflects the underlying data distribution closely, and hence, also leads to improved color coding.

Figure 7.8 compares the segmentation strategies of histogram equalization and Box-Whisker plot adaptation. Visualization examples generated without and with the proposed methods are given in Figure 7.9. It can be seen that colors are hard to distinguish in dense parts of the data (a). By applying the proposed adaptation methods, more colors are assigned to these dense parts, and hence, colors can be discerned more easily (b) + (c). It is important to mention that we deliberately relinquish linearity of the color scale in favor of separability of colors in dense regions.

Color Scales for the Comparison Task

The comparison of two or more attributes requires a global color coding scheme that guarantees that equal colors stand for equal values. This leads to problems, in particular, if the value ranges of the attributes to be compared are quite different. In such cases, an attribute with a smaller value range would be represented by only a very small region of the global color scale. The goal now is to improve the differentiability of colors for these small value ranges.

We handle this problem by merging overlapping value ranges. The result of this

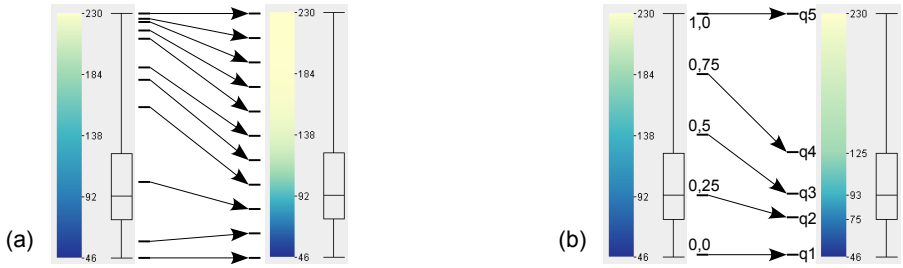


Figure 7.8: Color scale adaptation. (a) Histogram equalization; (b) Box-Whisker plot adaptation; Complementary Box-Whisker plots visualize data distribution.

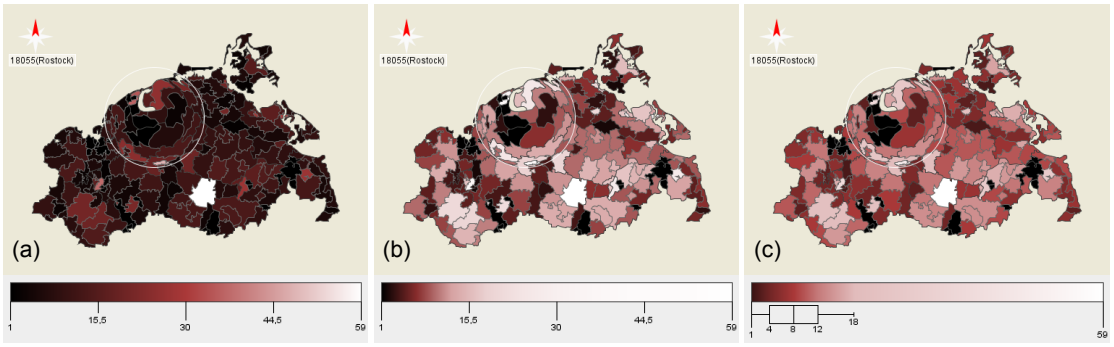


Figure 7.9: Visualization of quantitative data on a map. (a) Classic linear color coding; (b) histogram-equalized color coding; (c) color coding adapted based on Box-Whisker plot.

process are fewer distinct value ranges that do not share common intervals. Now, for each distinct value range a separate color scale is designed. Since the newly defined ranges do not overlap, it is possible to assign a separate hue to each, while varying only brightness and saturation to define the color coding. These separately specified color scales are integrated into one global scale (see Figure 7.10(a)). To avoid inconsistencies, it must be guaranteed that the brightness and saturation values of the boundary of one color scale correspond to the respective values of the neighboring scale. In other words, for one value range the hue is constant while brightness and saturation vary, whereas at the boundary from one value range to the next the hue varies while brightness and saturation are equal (see Figure 7.10(b)). This way, even small value intervals will be represented by their own brightness-varying subrange of the global color scale and the differentiation of data values is improved.

Figure 7.11 shows how different color coding schemes influence the task of comparing three attributes of a data set. Figure 7.11(a) uses individual color scales for each attribute. Visual comparison is hardly possible because one and the same color represents three different data values (one in each value range). A global color scale as shown in Figure 7.11(b) allows for visual comparison, but data values of the first and third attribute are no longer distinguishable because their value ranges are rather small compared to the one of the second attribute. Applying Box-Whisker plot adaptation helps to improve perception, but the results presented in Figure7.11(c) are not really

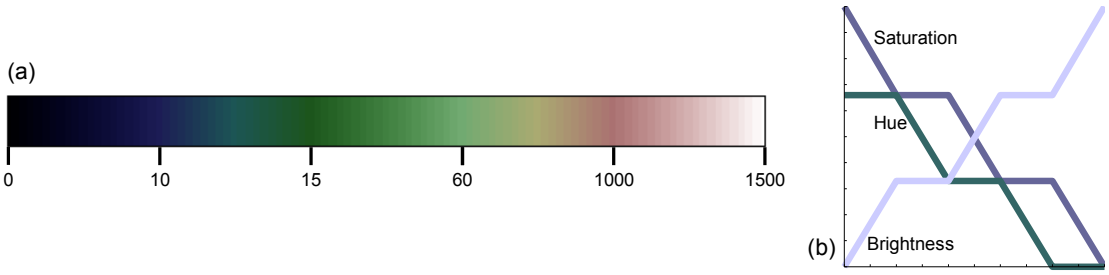


Figure 7.10: Color scale for comparison tasks. The five value ranges [0,10], [20,50], [15,30], [40,60], and [1000,1500] were merged resulting in the three distinct ranges [0,10], [15,60], and [1000,1500]. Each distinct ranges is assigned with a distinct hue blue, green, and red, respectively. (a) Color scale for comparison; (b) its definition in the HSB color space.

convincing. The novel concept we propsed in the previous paragraph delivers better results (see Figure 7.11(d)). Values of the first and second attribute are easier to identify and to compare. However, the third variable is still hard to perceive.

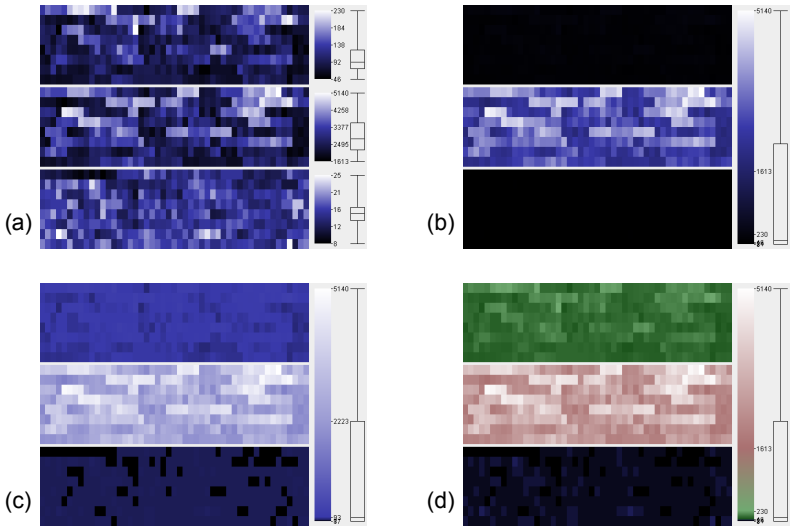


Figure 7.11: Visual comparison of three attributes. (a) Individual color scales; (b) one global color scale; (c) adapted color scale based on Box-Whisker plot; (d) optimized color scale for comparison tasks (also see Figure 7.10). (Images from prototype software [Sch06].)

Figure 7.11 shows the potential, but also the limits of our poposed approach to support comparison tasks. The problem is that we cannot yet guarantee that adequate color scales will be generated in all cases. In particular, if the merging process generates too many or too few distinct value ranges, the problem becomes apparent. In the former case, it would be difficult to assign distinguishable hues for each value range. In the latter case, it could be possible that small value ranges are merged into huge ranges nonetheless, and thus are still represented by only a few colors.

We conclude that achieving appropriate color coding for comparison tasks is a challenging problem and requires further investigation. Our approach can be seen as a first step towards a general solution.

The Task-Color-Cube

As a summary of the previous discussions, Figure 7.12 shows the task-color-cube. The three task alternatives introduced at the end of Section 7.4.2 span a three-dimensional space with eight discrete points. Each of these points represents a specific task and requires a specific color coding scheme to achieve expressive visualization. We assigned color coding schemes to each corner of the task-color-cube based on the following guidelines:

1. Individual data values vs. sets of values: unsegmented vs. segmented color scales.
2. Lookup vs. comparison: several separate color scales vs. one global color scale.
3. Identification vs. localization: color scale that match data characteristics vs. color scale that accentuates ranges of interest.

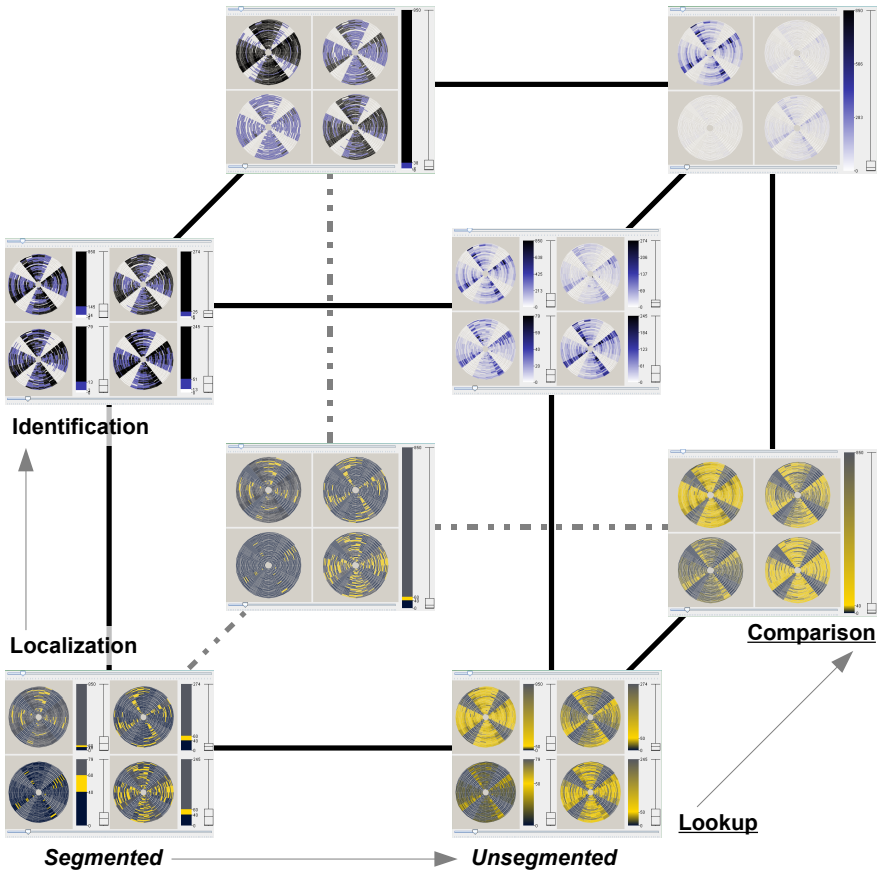


Figure 7.12: The task-color-cube. The cube spans along the axes segmented vs. unsegmented, identification vs. localization, and lookup vs. comparison. Each corner of the task-color-cube represents a specific task instance and is associated with a particular color coding scheme.

Since color perception is always dependent on the user, all our methods are subject to interactive refinement. Interactively adjusting the color coding is particularly useful when exploring unknown data.

7.5 Summary

Abstract data lacks an intrinsic visual representation and must be transformed into an image through a visualization process. With respect to task-driven adaptation, this offers significant flexibility – it principally allows to effect adaptation not only based on FEATURES in presentation space, but also pre-PAGE **adaptation in data space**. It also poses challenges due to the vast variety of visualization techniques, data sets, and associated data analysis problems.

In this chapter, we presented first ideas on how to integrate our basic adaptation approach with a suitable model of the visualization process. The main idea is to use Chi's **Data State Reference Model** comprised of *operators on data elements* on different *data states* as a **general framework** for this purpose.

Data elements (graphics primitives) on the DSRM's visual abstraction stage provide the basis for the specification of PAGES. FEATURES over such a PAGE can then be defined in one of two ways: as *RoIs in presentation space* similar to previously discussed visual data types; or by defining *data characteristic of interest* in data space to select a subset of data elements that contribute to FEATURE geometry.

The information afforded by access to the input data as well as derived meta data (analytical abstractions) allows to incorporate **data- and task-driven automatisms** in the process of determining task-related FEATURES and FEATURE relevance values. Section 7.2 suggested several possibilities to this end.

Task-driven adaptation within concrete smart visual interfaces is *effected on the level of individual DSRM operators*. We here proposed the concept of **Smart Lenses** as a general "Smart-X" technique for abstract data visualizations. The core idea is to utilize so-called lenses to specify FEATURE-based adaptations in both data and presentation space. Single *lenses represent operators in the DSRM* and are employed to **adapt specific aspects** of the visualization **within FEATURE/lens regions**. Combination of multiple lenses allows to adapt different aspects of each FEATURE depending on the data states lenses operate on. Using *declarative scripts* further allows to **specify lenses on the conceptual level of the task model** while abstracting from concrete operator implementations in specific smart visual interfaces.

Color coding is a fundamental and widely used visualization method. Numerous studies on this subject identified key aspects of color scale design and provided guidelines for developers to make appropriate choices of color scales for expressive and effective visualizations. However, even though it is commonly accepted that different tasks require different visualizations, so far no systematic approach to task-driven color coding had been developed. For this reason, we presented here **task-driven color coding concepts** that address different visualization tasks derived from a formal task typology. Our concepts are to choose from different color scales and adapt color mapping functions according to tasks and data characteristics.

The key contribution of this chapter should be understood mainly as a problem analysis as well as a systematic view on the different levels at which task-driven adaptation

can be effected. Although some open questions remain as indicated in the text, implementations of Smart Lenses and Smart Color coding substantiate the feasibility of our proposed approaches. A further investigation into the remaining issues is definitely worthwhile as necessary steps towards for smart, task-driven visualizations.

8 Implementations

Chapters 4–7 examined how task-based adaptation of the different visual data types is effected based on the adaptation pipeline approach proposed in Chapter 3. This included a discussion on how existing and several novel "Smart-X" display techniques – as components of concrete smart visual interfaces– are utilized and parameterized by means of an enriched task model.

These concepts and "Smart-X" techniques have been developed against the background of the applied research project Landesforschungsschwerpunkt (federal state research focus, LFS) and research training school Multimodal Smart Appliance ensembles for Mobile Applications (MuSAMA). These provided two use cases exposing challenges and research questions with regard to system design in general and smart visual interface design in particular: mobile maintenance support and Smart Meeting Room scenarios. They have been introduced in Sections 2.2.1 and 2.2.2, respectively.

In this Chapter both application scenarios are revisited with respect to the contributions presented in this thesis as well as software implementations supporting them.

8.1 E-manual demonstrator

The scenario that has been considered in the LFS project [LFSa, LFSb] for a demonstrator implementation is as follows. A public utility provider (i.e., gas, electricity and water services) administers and monitors its assets using a centralized technical facility management software. The mobile e-manual application serves as an interface to this centralized system for maintenance technicians in the field. A fault report is submitted to the provider, which triggers the generation of a corresponding job assignment (work package). Job assignments comprise relevant information on the affected unit (type, status, service records etc.) as well as navigational information to the work site itself and the location of the unit at the site. The technician downloads the job assignment to his mobile device (e.g., a PDA). Using the navigational data he navigates to the work site and locates the defective unit on site. To support the actual repair or maintenance task, the mobile e-manual application provides corresponding instructions in different formats e.g., step-by-step illustrated assembly instructions or short tutorial videos. After the repair/maintenance has been completed, the technician uses the e-manual application to file a job report. This report is uploaded to the central management software updating the respective unit's service record.

This scenario was chosen for two reasons. First, because the maintenance of machinery and facilities is a requirement faced by many enterprises e.g., public utility companies. Second, the design of an interactive, mobile maintenance support software that can interoperate with existing management systems is interdisciplinary nature. It involves

aspects from database and wireless networking technologies, software and user interface design, as well as visualization and natural language interfaces. It therefore caters to many of the challenges and research topics relevant to the groups involved.

With regard to smart visual interfaces and task-specific visual representations in particular, this scenario calls for a wide variety of representations from different sources and of different visual data types: navigational maps in raster or vector format e.g., provided by a GIS; technical illustrations and schematics as the crucial part of repair instructions; as well as images helping technicians to identify and troubleshoot affected units.

General architecture

The demonstrator's software architecture is shown in Figure 8.1. It is based around a central Content and Knowledge Management (CKM) system that serves as the interface to the existing facility management software and its databases. With regard to task-specific visual representations in particular, it is responsible for locating and serving graphical content and task information to mobile clients according to the task at hand. CKM also provides communication interfaces with back-end services that are required by the user interface but that are too computationally intensive to run on the client device itself, such as the remote labeling module (see Section 4.3.2). Prototypes of the demonstrator have been shown on the 2005–2007 CeBIT yearly international IT fairs, among others.

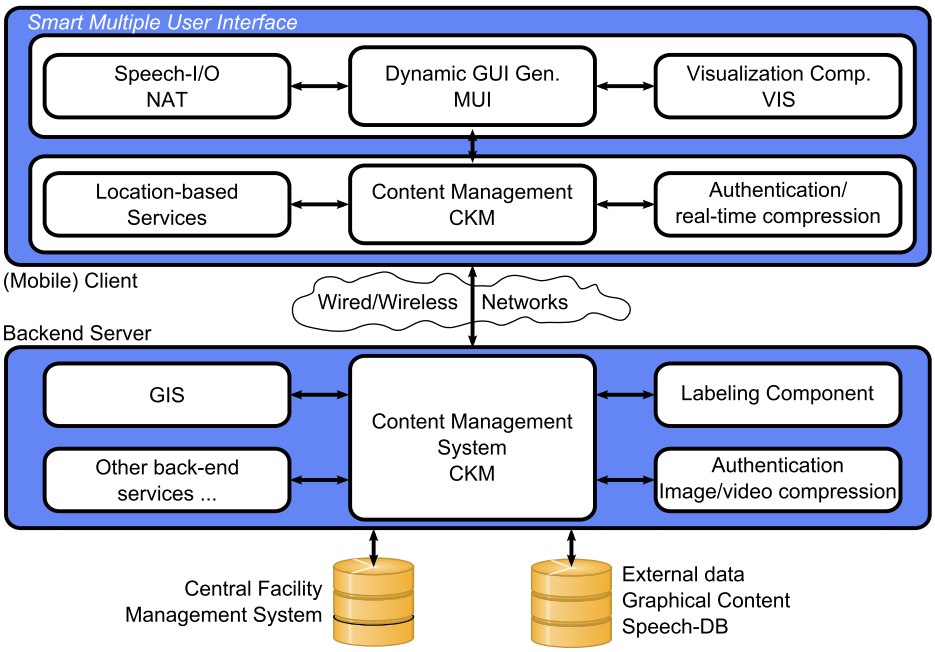


Figure 8.1: Schematic view of the principal software components comprising the e-manual demonstrator implementation for the LFS use case "mobile maintenance support" (cf. Section 2.2.1).

This section concentrates on the mobile client software since it incorporates smart visual interfaces. For information on the CKM component, see Bruder et al. [BZM+04]. The client has been implemented using Microsoft's .NET Compact Framework on PDAs

running Windows Mobile operating systems. The .NET framework was chosen because of portability issues, and because its runtime can be operated royalty-free on any compatible hardware. This was a important argument because of the declared goal of multi-platform support. Native (i.e., WinCE-API) libraries are used where the managed .NET environment is too restricting.

The client software comprises three main components (Figure 8.1): the Multiple User Interface (MUI) generator, the visualization component (VIS), and an optional natural language interface component (NAT) depending on hardware capabilities. This corresponds to the three principal components of a smart multi-modal UI as discussed in Section 2.2.1, cf. Figure 2.3 on page 13.

The MUI component is responsible for creating the client user interface dynamically according to the device's screen resolution. This includes dialogs handling login to the central management system, reviewing, selecting and downloading job assignments. It further provides means to navigate information contained in job assignments. These functionality is offered via conventional WIMP interfaces, see Figure 8.2. The MUI component thereby uses a general application task model, from which a corresponding dialog model is derived. This dialog model is encoded in XML-based XUL descriptions and instantiated at runtime using a XUL interpreter.

Most importantly, the MUI component allows the user to select and execute instructions for maintenance procedures (i.e., a composite task) that are supported through a smart visual interface. This functionality is provided by the visualization component, which is described in more detail below.

Both the MUI and VIS components are optionally supported through a natural language interface by means of voice commands (e.g., "next step", "back") and text-to-speech output of non-visual information. For the sake of brevity this component is not discussed further here, for details refer to [FRSF06, BDF+06].

Visualization component

The visualization component has been developed in the scope of the present thesis. It is used by the MUI component to provide task-specific visual representations for the selected composite (maintenance) task. Maintenance task are described by enriched task models in CTT format. Recall from Section 3.3 that the enriched task model captures both the graphical content associated with tasks (WHAT, via PAGE annotations) as well as the WHY aspects (FEATURES and relevance value annotations). Furthermore, adaptation (HOW) is controlled on the level of basic tasks through adaptation parameter annotations.

The design thereby utilizes that CTT models may be composed by linking several smaller task trees. This was primarily intended for modeling of collaborative work flows comprising of cooperative as well as role-exclusive parts which are quite common in smart environments. Here, it facilitates modularization of the application task model: activities supported through smart visual interfaces (which are in the focus of this thesis) reside in individual, succinct CTT models. The maintenance task from Figure 2.6 (page 19) is an example: all its basic tasks are associated with visual representations, whereas the UI allowing the user to bring up this particular maintenance procedure is derived from the superordinate application task model.

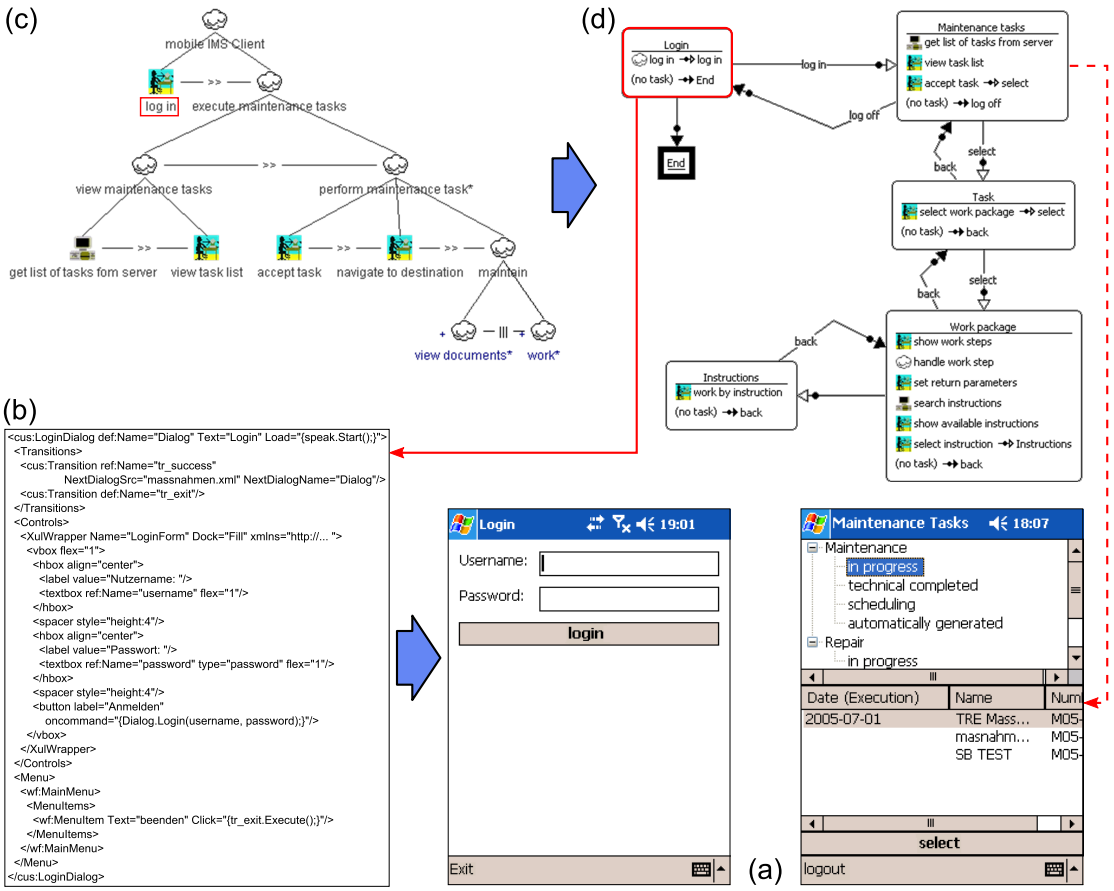


Figure 8.2: The demonstrators interface (a) is generated dynamically from an abstract UI description (b) depending on the client device. The UI description itself is derived from the application task model (c) that is transformed into a dialog model (d) as an UI design step.

To this end, communication between the MUI and VIS components comprises provision of the enriched task model as well as signaling which task node corresponds to the current working step. That is, *task navigation* is handled by the dynamic UI (or the NAT component, via voice commands); whereby the VIS component evaluates the extended task context information provided by the enriched task model to *automatically generate an initial view* for the respective task at hand. Moreover, the VIS component reports high-level user interactions (i.e., FEATURE selection) back to the MUI component. These may trigger application state changes not related to the visual representation itself, as well as task transitions. An example for the latter is the request for a specific detail view depending on the overview FEATURE selected.

The VIS component itself comprises several sub-components, as shown in Figure 8.3. The `VisualizationDialogue` constitutes the interface to the MUI component – it provides a generic container that can be placed on a GUI dialog by the interface generator/XUL interpreter. Within this container, the `VisualizationControl` manages

display of initial visual representations as well as re-adaptation in reaction to user interaction (FEATURE selection, cf. Section 3.3.5).

Processing and adaptation of graphical content is handled by *image providers*. This allows to maintain independence of the `VisualizationControl` from peculiarities of the different visual data types. Currently, image provider implementations for raster and 2D vector graphics have been incorporated into the demonstrator. An abstract base class `ImageProvider` provides commonly required functionality, in particular, loading and parsing of XML-encoded CTT task nodes and annotations.

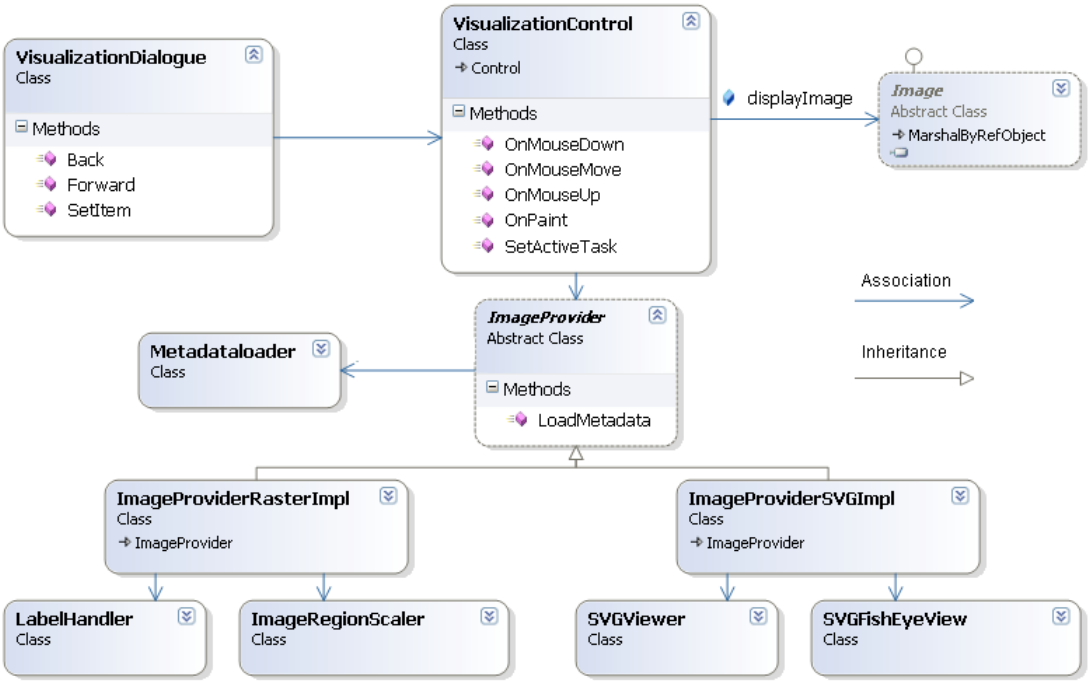


Figure 8.3: (Simplified) class diagram of the e-manual demonstrator's visualization component supporting task-driven adaptation of raster images and SVG vector graphics. A modular design of visual data type-specific image providers allows easy extension of the prototype to other content formats.

Concrete image provider implementations in turn make use of specialized handler classes and libraries that realize concrete, visual data type-specific "Smart-X" display techniques. Indicated in Figure 8.3 are:

- A combined implementation of the Belt-based Focus & Context technique and visual attribute manipulation (`ImageRegionScaler`, cf. Sections 4.3.1 and 4.2.4, respectively).
- Integration of image-based labeling including remote labeling (`LabelHandler`, cf. Section 4.3.2).
- Geometry adaptation of SVG 2D vector graphics (`SVGFishEyeView`, Section 5.2.2).
- Visual attribute manipulation of SVG graphics (`SVGViewer`, cf. Section 5.2.3).

The visual output generated by these implementations have been presented by figures in the indicated sections already.

Example maintenance task

In order to illustrate the aspects of the e-manual design, a routine maintenance task has been fleshed out as an example for the demonstrator. It was taken from the printed reference manual of an LPG/natural gas burner that is part of the actual HVAC system installed in one of the faculty's buildings. The case study comprises a routine maintenance task, which is setting the fuel value of the burn chamber according to local gas quality. To accomplish this, the technician has to perform the following steps:

1. Shut down the unit
2. Disconnect electronic power, water and gas supply
3. Unfasten and remove the blower unit
4. Pull off the short feed pipe to the blower
5. Dismantle the fuel nozzle and remove the old reducing ring
6. Set the burn value by inserting a new reducing ring into the nozzle, depending on gas quality
7. Re-assemble and re-connect unit in reverse order
8. Start up unit and perform function tests

The graphical CTT notation of this maintenance task is depicted in Figure 2.6 (page 19). Some of these steps require the lookup of values in tables (e.g., the selection of the correct reducing ring), while others should provide the user with assembly drawings or schematics to illustrate the task at hand (e.g., what electric contacts have to be disconnected). Thus, this brief example is well-suited to illustrate the requirements to the client application.


```
<?xml version="1.0" encoding="UTF-8"?>
<taskmodel:TaskModel id="id11096072294040" xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:taskmodel="http://wwwswt.informatik.uni-rostock.de/uris/m6c/taskmodel">
  <root id="id11096072327322" name="Leistungsklasse einstellen"
        childorder="1>>2>>3">
    <subtask id="id11096074641292 81" name="Gas-Odorieranlage ausschalten"
             childorder="1>>2">
      <subtask id="id114975231258247"
               name="Unterbrechung Strom- und Gasversorgung"
               desc="Strom- und Gasversorgung des Heizkessels unterbrechen">
        <property key="keyword" value="ausschalten"/>
        <VisualizationPage resourcename="Abb07"/>
        <ActiveFeature id="feature_3" importance="1.0"/>
        <ActiveFeature id="feature_4" importance="0.7"/>
        <ActiveFeature id="feature_5" importance="0.3"/>
        <Effect type="ColorSaturate">
          <Mapping type="linear">0.0,0.5 1.0,1.0</Mapping>
        </Effect>
      </subtask>
      ...
```

Figure 8.4: Initial visual representations for two working steps of an example maintenance task. On the right an excerpt of the enriched task model is shown that provides the task context information for the left view.

Figure 8.4 shows the initial visual representation for working steps 2. and 5. of the above maintenance task, as well as an excerpt of the enriched task model providing the corresponding task context information. Note how PAGE/FEATURE definitions are annotated by reference; FEATURE relevance values as well as adaptation parameters are

annotated directly. In this example, several consecutive tasks share the same default representation. Some of these also refer to common subsets of depicted assembly parts (i.e., FEATURES), albeit with different emphasis (i.e., FEATURE relevance). Here, the least common ancestor node is annotated with the information accordingly, thus utilizing the WHEN aspect captured by the hierarchical task model. Appendix B lists the complete XML-encoded CTT model as well as a selected PAGE manifest for this maintenance task.

8.2 Smart Document displays in Smart Meeting Rooms

Like in the LFS project, in MuSAMA [MuS09] the principal challenges encountered within smart environments (cf. Section 2.2.2) have been exposed by an example scenario. Here, the use case comprised a group of domain experts utilizing a smart meeting room to jointly explore and analyze data with visual means to arrive at a solution for a specific problem. This includes individual activities (e.g., an expert exploring data on his personal machine), work group scenarios (e.g., a small sub-group discussing intermediate results) as well as presentation scenarios (a single presenter imparts information or findings to an audience). As stated in Section 2.2.2 this primarily requires adaptation of visual representations acoording to each user's task at hand. However, a major additional challenge in regard to smart visual interface utilization in such environments is how to take advantage of available display areas.

In [RLS11], Radloff et al. propose a Smart View Management that addresses these challenges. Instead of attempting to directly integrate different software systems employed by the users on the device ensemble, their approach focuses on the visual output of these systems i.e., the views. Their concept comprises a three-tier architecture to dynamically distribute views on the available displays in the smart meeting room according to the users' current work situation:

In a first step, views are grouped into so-called *view packages*. View packages represent semantically related visual output that should be shown on a single display or at least close-by displays. Radloff et al. propose an interactive view package generation according to three semantic aspects: mode of view usage (presentation to many viewers vs. comparison with other views by a single user), robustness of view content (i.e., whether the view may be subjected to distortion), as well as viewer stance (all looking at a single presenter, or towards their own displays). For typical combinations of these three aspects, their approach provides customizable presets.

The second tier is smart display mapping that comprises the automatic assignment of view packages to display surfaces. This is realized by the *display mapper*, a software component that picks up the basic idea of display mapping presented in [Hei09] but improves upon the approach described therein in regard to spatial and temporal quality of the obtained mapping solution. This is achieved by an extended geometric model of the environment as well as optimized quality tests that incorporate the semantic view relations implied by the view packages.

The final tier is smart view layout. Dynamic mapping of multiple views onto available displays inevitably affects presentation sizes and aspect ratios of individual views as

well. Therefore, view contents need to be laid out dynamically to accommodate for changes in display area. Radloff et al. describe a simple force-based approach that is able to arrange multiple views on a single display according to constraints. This includes objects or persons occluding parts of the display, presuming they are recognized by the environment's sensors. It does not, however, address adaptation of contributing visual representations within the aggregate view temselves.

The proposed framework for smart view management is summarized in Figure 8.5. For details on operations on each tier, refer to [RLS11].

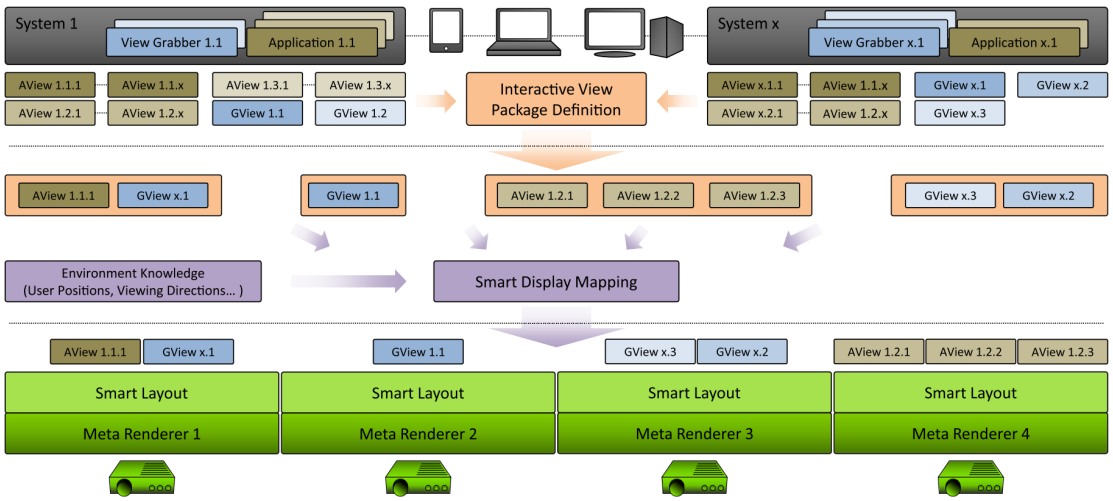


Figure 8.5: Smart view management as proposed by Radloff et al. Views from applications (A) and view grabbers (G) are grouped interactively into view packages to encode semantic relations. View package configuration and current environment state are considered in mapping views to display surfaces. If several views are aggregated on one display (by a meta-renderer), the views are laid out according to semantics and display characteristics. (Image from [RLS11])

In the following, we will discuss how concepts and techniques developed in the present thesis contribute to this Smart View Management framework. As of this writing, the framework design description [RLS11] represents submitted work. Likewise, integration of concepts and techniques into the framework as proposed here represents ongoing and planned work.

Task-driven (smart) document layout

On the third tier the smart view management framework only addresses arrangement of multiple individual views into aggregate views. Contrary to this, we previously published an approach to dynamically layout individual content elements within compound, interactive documents [AHFS08]. The basic idea can be summarized as follows.

A generic document layout is provided by a document template. A template governs what content element types (text, 2D images, 3D graphics, and video) are part of the document, and how elements are arranged with respect to each other. Thus a template captures semantic (selection of content elements and their relations) as well as aesthetic

aspects. Document templates are created by a designer/content author and collected into a template library.

At runtime, documents are laid out depending on available display space and user interactions. For this, first an initial layout is created using constraint-based layout techniques. Constraints are derived from the document template. The layout engine matches the template items to the given content and computes how well this content fits within the template. If display space is insufficient, some content elements may be removed from the document. In a second phase, the initial layout is refined using a force-based approach. Attractive and repulsive forces arrange content elements, whereby internal and external pressure regulate how content elements are enlarged or shrunk depending on their relevance.

Details on template design as well as constraint- and force-based layout phases are subject of Ali's thesis [Ali09]. With respect to contributions of the present thesis, [AHFS08] discussed an extension to the basic layout approach — *task-driven document layout*.

Task-driven document layout is a specific realization of the basic PAGE/FEATURE concept proposed in Section 3.3: document templates specify a default representation (layout) of a dynamic document. A PAGE annotation thus provides a template selection. FEATURES determine what concrete content elements should be used to populate the template during the initial layout. FEATURE relevance values then serve as parameters to the layout engine to determine (i) what content elements to drop, assuming not all can be placed due to space constraints; and (ii) the inner pressures of content elements, affecting their relative sizes in the final document layout. Figure 8.6 illustrates the relations between the enriched task model, document layout engine and template library.

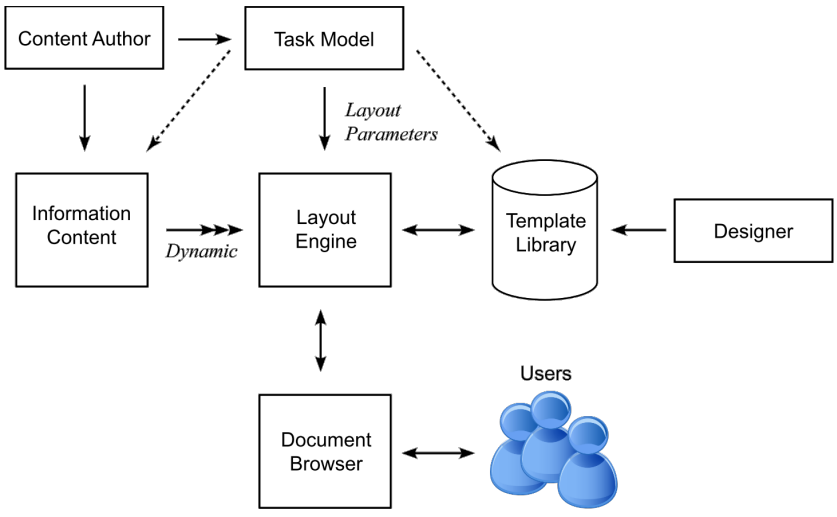


Figure 8.6: Task-driven adaptive document layout. The enriched task model provides a document template (PAGE) and content elements (FEATURES) selection for initial layout. FEATURES relevance values influence the final, force-based layout.

In addition to facilitating task-specific document layouts, task-based adaptation can further be applied to individual content elements to account for their relevance and final display size. To this end, adaptation methods are applied according to the elements' visual data type, cf. Chapters 4–7.

To summarize, our proposed approach to task-driven Smart Document Layout is based on author-defined layout templates and a provision of document content hierarchically organized according to the enriched task model, cf. blocks "information content" and "template libraries" in Figure 8.6. The base approach is based on the works of Ali [Ali09], for which we integrated a task-based extension, see [AHFS08].

Task-driven Document Layout as supplement to Smart View Management

Radloff et al. [RLS11] in their concept so far deliberately abstain from using templates and hierarchical content management because these can not be assumed universally available in ad-hoc environments like Smart Meeting Rooms (cf. Section 2.2.2).

Thus, the core idea here is to use the Smart Document Layout approach to present a preliminary layout as an initial view, and to iteratively refine it whereby dynamic changes are incorporated. Here, the 'document' is an aggregate view produced by a meta-renderer. Individual contributing views represent content elements. In particular, views generated by task-driven applications such as LandVis-Lens (see Section 7.4.1) can further contribute task-specific view properties defined by the extended task context. Dynamic changes then comprise both changes of the user's task at hand as well as updates to the environment's configuration e.g., the set of available displays in the device ensemble and users moving about the room.

In order to contribute to Smart View Management in this manner, our present approach needs to be modified. This affects the following aspects of Smart View Management as proposed in [RLS11]:

Task-based grouping of of views: This relates to the first tier of Smart View Management, view package definition, which is currently a manual step. Categorizing the task at hand (cf. Section 7.4.2) a given view is associated with allows to infer the overall mode of view (presentation vs. comparison, see above). Information from the extended task context can here be utilized to (partially) automate selection and customization of view package presents: views associated with tasks executed in parallel should be grouped into the same package, whereas consecutive working steps indicate a view currently displayed should be replaced (rather than juxtaposed) with its successor.

Task-based view layout (I): This relates to the third tier of view management. So far, Radloff et al. use a simple force-based layout of views that does not account for semantics of the view contents. By contrast, information provided by the task context from which a particular view was generated can be used to influence positioning of individual views on the display area. To this end, the spring-force models applied for element layout are extended to initially position views with high task relevance values closer to the center of the display.

Task-based view layout (II): This, too, relates to the third view management tier. Available display space can be utilized more effectively by adjusting individual view sizes according to their importance. Here, an approach similar to the proposed for belt-based image distortion (see Section 4.3.1) is applicable: The total size of the assigned display area as well as the (rectangular) views that must be positioned and displayed within are given. Based on the relevance values associated with views their relative sizes are evaluated. This can be combined with relevance-driven selection of initial view positions.

Task-based adaptation of view contents: Each view displayed in the smart environment constitutes a visual representation. Depending on the visual data type of the underlying content, this representation in and of itself may be adapted in a task-specific way according to the methods and techniques outlined in Chapters 4–7. For data visualizations – as the most common type of content in the Smart Meeting Room scenario – in particular, task-driven adaptation can be effected using our proposed DSRM-based framework introduced in Section 7.4.1.

Concrete realizations of these proposed extensions are currently the subject of active investigation.

8.3 Summary

This chapter briefly discussed how the concepts and techniques for the task-driven adaptation of graphical content have been integrated into concrete software prototypes.

The majority of development effort went into a concept demonstrator implementation for the "e-manual" mobile maintenance support scenario within the LFS project. It incorporates adaptation methods and "Smart-X" techniques for raster images and SVG 2D vector graphics as introduced in Chapters 4 and 5, respectively.

By contrast, with regard to the MuSAMA project stand-alone proof-of-concept implementations have been developed, in particular, LandVis-Lens as a realization of the DSRM-based adaptation framework (cf. Section 7.4.1) and a task-based extension to Smart Document layout. Nonetheless, we here discussed ideas and current works on the integration of these implementations with a smart view management framework facilitating smart visual interfaces in smart environments.

9 Conclusion and Future Work

9.1 Summary

In the face of today's ever larger data sets and more complex models, the research in the field of Smart Graphics – seeking means to effectively and efficiently communicate these information – is becoming increasingly important. Development of appropriate, context-sensitive smart visual interfaces to access and explore graphical content plays an important role in these efforts.

The aim of this thesis has been the development of concepts for the automatic generation of *good initial views* with respect to the user's task at hand, thereby providing "smart" user support by reducing the need for manual selection and navigation of content. A focus has been on applications where visual representations are consumed as a source of supplemental information in the course of complex work flows. This requires the adequate consideration of the visual representation's Context of Use (CoU) as well as scalable description of graphical content so that adaptation can be suitably effected. A secondary goal of this thesis has been to provide first steps toward closing the modeling gap between Model-based User Interface Development methods and scalable display techniques as a contribution to the model-based design of smart visual interfaces.

The results of the present thesis are an basic approach and supporting techniques for the task-driven adaptation of graphical content within smart visual interfaces. These shall be briefly summarized in the following.

As a first contribution we proposed a **general categorizations of influencing factors** into five high-level aspect categories. These **'5Ws'** – WHAT, WHY, WHEN, WHERE and FOR WHOM– define the context of use of visual representations within Smart Visual Interfaces.

This categorization is used to derive a classification of graphical content that constitutes the basis for task-specific visual representation within smart visual interfaces. In this thesis, a distinction into **four principal visual data types** has been proposed according to the degree of inherent scalability: raster graphics, 2D vector graphics, 3D graphics, and information visualizations.

The main contribution of this thesis is the development of an approach for automatic generation of task-specific visual representations based on an enriched general (domain) task model. It comprises two concepts.

The **PAGE/FEATURE concept** represents a *general, data type-independent description of graphical content* as well as *task-related aspects of the CoU*. It allows to enrich the general task model with a description of relevant task context information with respect to scalable visual representations. PAGES represent *self-contained units of graphical content*,

thereby abstracting from the technical peculiarities of the underlying visual data type. FEATURES describe *content elements* of a PAGE *on a semantic level* i.e., represent units of information that are relevant to a task's communicative goal, while *abstracting from the technical structure/organization* of the different visual data type. FEATURE relevance is an *abstract measure to express the information and visual scale* that is appropriate for the given FEATURE in the context of the task at hand. Relevance values therefore constitute the basis on which the adaptation of the graphical content is effected.

Together, PAGES, FEATURES and FEATURE relevance values are used as the **building blocks for enriching the domain task model** with a description of the task context for visual representations. To this end, specific task nodes in the model are *annotated* with corresponding definitions of PAGES, FEATURES and relevance values. In doing so, our approach is to *specify aspects of the task context hierarchically* on different task scopes, i.e., on different levels of the task decomposition. This sets the approach presented in this thesis apart from contemporary solutions that are based solely on low-level tasks. The latter require the full task context be specified for each working step individually and thus provide no means to exploit causal or temporal relations between sub-tasks of a composite task.

The second proposed concept of an **adaptation pipeline** provides the bridge between the conceptual task context specification in the model and concrete display techniques used to generate adapted visual representations accordingly. The adaptation pipeline comprises five stages – view selection, geometry, visual attributes, view space and annotations – that allow to effect different aspects of visual representations by corresponding *adaption operations*. The adaptation process can thus be conceptually distinguished into a visual data type-specific part (stages 1–3) as well as a type-independent part (stages 4–5). Adaptation operators on the pipeline stages are realized by suitable display techniques. To this end, FEATURE relevance values are mapped onto the parameter domains of the respective technique. Therefore, the adaptation pipeline serves as a **general framework for the integration of existing display techniques** for the different visual data types. Because these display techniques thus define the functional building blocks of the smart visual interface, we refer to them as "Smart-X"-techniques.

The PAGE-/FEATURE abstraction concept and the adaptation pipeline represent a general approach to the generation of task-specific adapted visual representations for all visual data types. However, the principles and means by which the PAGE and FEATURE abstraction are derived from graphical content are, on principle, specific to the respective visual data type. Similarly, how FEATURE relevance values are mapped to adaptation parameters varies with the concrete display technique employed.

It has been shown in Chapters 4 to 7, however, that the three conceptual aspects PAGE definition, FEATURE specification, and adaptation control (i.e., integration of concrete "Smart-X" techniques) is indeed feasible for each of the four visual data types. To this end, adaptation examples were given for each type on every pipeline stage. It has also been shown how powerful existing display techniques – image-based labeling, different forms of Focus & Context distortions and importance-driven NPR rendering, to name but a few – can be integrated into the framework provided by the adaptation pipeline.

In the same way that the specification of the general task model (as a result of domain

task analysis) is by necessity a manual task, some aspects of graphical content preparation and smart visual interface (adaptation) design must be specified manually as well. It has been argued for each visual data type how this process is supported by suitable authoring tools and which aspects can be supported by automatic methods. Moreover, the respective chapters reviewed concrete implementations of authoring tools that have been developed in the course of this thesis.

Although the proposed PAGE/FEATURE annotation concept in combination with the adaptation pipeline framework allow to incorporate a wide range of existing display techniques into the design process of smart visual interfaces without significant modification, some task-specific aspects of representation scalability were found to be still open research questions. Thus as further contributions of this thesis, a number of **novel "Smart-X" techniques** covering several different aspects of task-driven adaptation for the four visual data types have been developed. All are based on FEATURES and task-specific FEATURE relevance in line with the proposed PAGE/FEATURE concept. These are:

- a belt-based raster image distortion technique,
- an image-based space-efficient remote labeling approach,
- a method to generate exploded view diagrams from vector graphics PAGES,
- an initial approach to smart circuit schematic representation,
- a general smart lens-based approach to the adaptation of information visualizations, and
- a systematic approach to task-driven color coding in information visualizations.

Finally, it has been discussed how the works presented here relate to further approaches pursued in the LFS project [LFSa, LFSb] and MuSAMA research training school flanking this thesis. In the former a joint e-manual demonstrator has been developed, to which this thesis contributes the central component of its smart visual interface.

Regarding Smart Environments as the subject of MuSAMA a number of publications [FTS07, AHFS08, TFS08a, FTSS09] co-authored with school members substantiate the principal utility of the concepts proposed here. However, there still remain several open research questions with regard to smart visual interfaces in these environments which are actively investigated. As of this writing, some continuative concepts and initial results of this ongoing effort have for example been submitted to the Smart Graphics 2011 symposium [RLS11].

Conclusion: The concepts and techniques discussed in this thesis constitute a first, extensible solution for task-driven adaptation of graphical content in smart visual interfaces. Using the presented PAGE/FEATURE approach and adaptation pipeline framework as a basis, several research directions can be pursued from hereon.

9.2 Open Questions for Future Research

One of the most interesting aspects is to consider time-variable content as visual data types. This includes video streams, dynamic and animated 2D vector and 3D graphics

as well as visualizations of dynamic data sets. So far, videos are used only as source for 2D (still) images.

Integration of time-variable content thus requires extension of the PAGE/FEATURE concept. It needs to be determined what temporal range of a video stream or scripted animation comprises the PAGE's content. The default representation of the PAGE then comprises not only a spatial but also a selection in the temporal domain. The latter is either a fixed point in time or a specific time interval. Similarly, so far FEATURES have been considered primarily as regions of interest in presentation space. It would thus be interesting to integrate concepts that allow a specification of FEATURES in the temporal domain as well. For example, event-based visualization proposed by Tominski [Tom06] represents a means to specify time-dependent data characteristics of interest in dynamically changing data sets. Finally, an interesting extension to the basic approach proposed here would be integrate task-specific modification of scripting elements itself in order to provide temporal adaptation i.e., means to generate task-specific animations.

This discussion shows that although temporal dynamics of graphical content has not been at the focus of the present thesis, corresponding extensions can nonetheless be incorporated in line with the proposed approach to hierarchical task context specification outlined in Chapter 3.

Another idea worth pursuing is a corollary to integration of temporal adaptation. So far, it has been implied that the correspondence between PAGES and FEATURES is $1 : n$, i.e., multiple FEATURES may be defined per PAGE for different tasks. This could be extended to allow for $m : 1$ correspondence, i.e., one FEATURE is shared between multiple PAGES assigned to different tasks. An example is video analysis where one could have multiple video feeds but a single semantic region of interest (e.g., doorways observed by surveillance cameras).

Another question that arises is how to track the progress as the user follows the work flow. This relates to a traversal of the task nodes in the CTT hierarchy, which must be detected at runtime. Work so far concentrated mainly on the maintenance scenario where prescribed procedures with tasks comprising largely sequential working steps are prevalent. Task navigation in these types of workflows is easily supported in the smart visual interface by 'wizzard'-style interaction i.e., providing means to select the previous or next task, cf. Section 8.1.

In typical exploratory visualization scenarios – as an important use case in Smart Meeting Rooms – this no longer suffices. This is due to the vastly increased degrees of freedom with respect to the set of enabled tasks (i.e., available for starting execution) as well as the number of possible permutations in which order various analytic tasks are carried out by the user(s). Instead, it must detected which task from the enabled task set the user has actually chosen. This challenge is further compounded by the fact that in ad-hoc environments like Smart Meeting Rooms hierarchical workflow descriptions are often not available in the first place.

This closely correlates to the problem of user intention analysis, a problem known from the domain of pro-active assistance systems. A solution thus requires methods to link the enriched task model with methods used for user intention tracking and inference.

First ideas in this regard have been published in cooperation with co-researchers from the MuSAMA project, see [GFF+07]. However, approaches to address these challenges in combination with requirements to task-driven adaptation of visual representations are far from being conclusive and certainly warrant further research.

Currently, preparation of graphical content as well as selection and parametrization of adaptation operations is a largely manual task. For complex workflows and intricate visual representations, specifying large numbers of FEATURES, task-specific relevance values and relevance–parameter mappings manually would be a tedious task. Although Chapters 4–7 pointed out aspects that can benefit from algorithmic solutions, it would be useful to include yet further automatisms for both content preparation and specification of adaptation operations. Interesting aspects for future research in this regard are:

- Providing means for the derivation of the (extended) task context from a high-level specification of communicative goals. In particular, this includes algorithmic and rule-based approaches to determine FEATURES of a PAGE that exhibit domain-level semantics, as well as their task-specific relevance.
- Support for automatic selection of suitable adaptation operations (e.g., color adjustment, rendering styles) according to perceptional and aesthetic considerations. This requires integration of rule-based expert systems like e.g., [SF91, Krü98].
- Addition of algorithmic methods to adjust task-specific FEATURE attributes based on FEATURE relevance e.g., replacing full FEATURE labels by abbreviations below a certain relevance using automated text summarization methods [MM99].
- Consideration of influencing factors from further CoU aspect categories, characteristics of the user (FOR WHOM) and output devices (WHERE) in particular. The former aspect is at the focus of current research efforts within the MuSAMA project e.g., [LRS10]. Although punctual consideration of WHERE-related aspects has been done in the present thesis, a far more detailed examination is the focus of Thiede [Thi10]. We published first ideas on the integration of task- and device-driven adaptation of visualizations in [FTSS09].

In conclusion, it can be stated that Smart Graphics in general and smart visual interface design in particular is an active but also very broad field of research. It requires cooperation between several disciplines including visualization, graphics design, software engineering and psychology. Due to this broad scope of the problem domain future work will inevitably identify and address new research questions beyond the scope of the present thesis. The concepts and techniques proposed in the present thesis constitute a basis for these future works, as well as providing first concrete steps towards dynamic and smart generation of task-specific visual representations, which represents a key aspect of smart visual interface design.

A XML Schema

Listing I – Basic XML data types

These XML data types are used across all solutions developed in the course of this thesis, including PAGE manifest definition, communication between the `VisualizationControl` smart visual interface component and the superordinate dynamic UI generator (see Section 8.1) and client–labeling component communication, among others.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.m6c.de"
            xmlns="http://www.m6c.de"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">
  <xsd:annotation>
    <xsd:documentation>
      This is the schema definition for all facetted, basic and abstract base
      types that are used throughout the HCI components for inter-component
      communication as well as for resource metadata descriptions.<br/>
      <br/>
      Created:      2005-11-15  Georg Fuchs<br/>
      Last edited: 2011-01-31  Georg Fuchs
    </xsd:documentation>
  </xsd:annotation>

  <!-- ========== Restricted/facetted numeric types ==================== -->

  <xsd:simpleType name="nonNegativeDouble">
    <xsd:restriction base="xsd:double">
      <xsd:minInclusive value="0.0"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="normalizedWeightType">
    <xsd:restriction base="xsd:decimal">
      <xsd:minInclusive value="0.0"/>
      <xsd:maxInclusive value="1.0"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!-- ========== Enumeration/list types ============================== -->

  <xsd:simpleType name="doubleListType">
    <xsd:list itemType="xsd:double"/>
  </xsd:simpleType>

  <xsd:simpleType name="pageContentFormat">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="RASTER"/>
        <xsd:enumeration value="SVG"/>
        <xsd:enumeration value="X3D"/>
    </xsd:restriction>
```

```
</xsd:simpleType>

<!--- ========= Color-related types ============================== -->

<xsd:simpleType name="ColorCodeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="(0x)?[0-9A-F]{6}"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="RGBAColorType">
  <xsd:simpleContent>
    <xsd:extension base="ColorCodeType">
      <xsd:attribute name="alpha" type="normalizedWeightType" use="optional"
                     default="1.0"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<!--- ========= Font system types ================================ -->

<xsd:simpleType name="FontAlignEnumType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="left"/>
    <xsd:enumeration value="right"/>
    <xsd:enumeration value="center"/>
    <xsd:enumeration value="leading"/>
    <xsd:enumeration value="trailing"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="FontDefType">
  <xsd:attribute name="type" type="xsd:string" use="required"/>
  <xsd:attribute name="size" type="xsd:nonNegativeInteger" use="optional"
                 default="10"/>
  <xsd:attribute name="align" type="FontAlignEnumType" use="optional"
                 default="left"/>
  <xsd:attribute name="bold" type="xsd:boolean"
                 use="optional" default="false"/>
  <xsd:attribute name="italic" type="xsd:boolean"
                 use="optional" default="false"/>
</xsd:complexType>

<!--- ========= 2D/3D geometry types & common elements ============== -->

<xsd:element name="_AbstractGeometry" abstract="true"/>

<xsd:element name="_PointGeometry"
             abstract="true" substitutionGroup="_AbstractGeometry"/>
<xsd:element name="_LineGeometry"
             abstract="true" substitutionGroup="_AbstractGeometry"/>
<xsd:element name="_AreaGeometry"
             abstract="true" substitutionGroup="_AbstractGeometry"/>

<xsd:complexType name="IntegerPointType">
  <xsd:attribute name="x" type="xsd:int" use="required"/>
  <xsd:attribute name="y" type="xsd:int" use="required"/>
  <xsd:attribute name="z" type="xsd:int" use="optional" default="0"/>
</xsd:complexType>

<xsd:simpleType name="DecimalPointType">
     <xsd:restriction base="doubleListType">
      <xsd:minLength value="2"/>
```

```
      <xsd:maxLength value="3"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="point" type="IntegerPointType"
             substitutionGroup="_PointGeometry"/>
<xsd:element name="coordinates" type="DecimalPointType"
             substitutionGroup="_PointGeometry"/>
<xsd:element name="vertex" type="DecimalPointType"
             substitutionGroup="_PointGeometry"/>

<xsd:complexType name="LineGeometryType">
  <xsd:sequence>
    <xsd:element ref="_PointGeometry"/>
    <xsd:element ref="_PointGeometry"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="PolylineGeometryType">
  <xsd:sequence>
    <xsd:element ref="_PointGeometry" minOccurs="2" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="line" type="LineGeometryType"
             substitutionGroup="_LineGeometry"/>
<xsd:element name="polyline" type="PolylineGeometryType"
             substitutionGroup="_LineGeometry"/>

<xsd:complexType name="RectangleGeometryType">
  <xsd:attribute name="x" type="xsd:int" use="required"/>
  <xsd:attribute name="y" type="xsd:int" use="required"/>
  <xsd:attribute name="width" type="xsd:nonNegativeInteger"
                 use="optional" default="1"/>
  <xsd:attribute name="height" type="xsd:nonNegativeInteger"
                 use="optional" default="1"/>
</xsd:complexType>

<xsd:complexType name="Rectangle2ptGeometryType">
  <xsd:sequence>
    <xsd:element ref="_PointGeometry"/>
    <xsd:element ref="_PointGeometry"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TriangleGeometryType">
  <xsd:sequence>
    <xsd:element ref="_PointGeometry" minOccurs="3" maxOccurs="3"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="PolygonGeometryType">
  <xsd:sequence>
    <xsd:element ref="_PointGeometry" minOccurs="3" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ComplexPolygonGeometryType">
  <xsd:sequence>
    <xsd:element name="outerRing" type="PolygonGeometryType"/>
    <xsd:element name="innerRing" type="PolygonGeometryType"
                 minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
```

```
  </xsd:complexType>

  <xsd:complexType name="CircleGeometryType">
    <xsd:sequence>
      <xsd:element ref="_PointGeometry"/>
    </xsd:sequence>
    <xsd:attribute name="radius" type="nonNegativeDouble"/>
  </xsd:complexType>

  <xsd:complexType name="EllipseGeometryType">
    <xsd:sequence>
      <xsd:element ref="_PointGeometry"/>
      <xsd:element name="radiusX" type="nonNegativeDouble"/>
      <xsd:element name="radiusY" type="nonNegativeDouble"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

Listing II – XML Schema for raster PAGE manifests

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.m6c.de"
            xmlns="http://www.m6c.de"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">

  <!-- Bring in the components from the basic definition schema -->
  <xsd:include schemaLocation="BasicDatatypes.xsd"/>

  <!-- ========================================================
       The feature mask definition
       ======================================================== -->

  <xsd:element name="feature">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="polygon" type="ComplexPolygonGeometryType"
                     minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="rectangle" type="RectangleGeometryType"
                     minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="triangle" type="TriangleGeometryType"
                     minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="circle" type="CircleGeometryType"
                     minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:ID"/>
      <xsd:attribute name="label" type="xsd:string"/>
      <xsd:attribute name="color" type="ColorCodeType"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="featureList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="feature" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="background" type="ColorCodeType"
                     use="optional" default="0x000000"/>
      <xsd:attribute name="nonFeature" type="ColorCodeType"
```

```
                    use="optional" default="0xFFFFFF"/>
    </xsd:complexType>
  </xsd:element>

  <!-- ========================================================
       The page manifest is a combination of the reference content resource
       (e.g., a raster image), a default viewport (rectangular image region)
       selection, a list of feature definitions
       ======================================================== -->

  <xsd:element name="pageMetadata">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="contentResource" type="xsd:normalizedString"/>
        <xsd:element name="defaultViewport" type="RectangleGeometryType"
                     minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="featureList"/>
        <xsd:element ref="taskList" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="symbolicName" type="xsd:token" use="required"/>
      <xsd:attribute name="type" type="pageContentFormat"
                     use="optional" default="RASTER"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

B Enriched Task Model Example

Listing I – Enriched task model

```
<?xml version="1.0" encoding="UTF-8"?>
<tm:TaskModel id="id11096072294040" xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:tm="http://wwwswt.informatik.uni-rostock.de/uris/m6c/taskmodel"
    xmlns:m6c="http://www.m6c.de">
  <tm:root id="id11096072327322" name="Leistungsklasse einstellen"
       childorder="1>>2>>3">
    <!-- Optional default adaptation operations. Beware that this causes all
         VisPages with at least active feature to be modified accordingly! -->
    <m6c:DefaultEffects>
      <m6c:Effect type="Distort">
        <m6c:Mapping type="linear">0.0,0.5 1.0,2.0</m6c:Mapping>
      </m6c:Effect>
      <m6c:Effect type="Tinting">
        <m6c:Mapping type="linear">0.0,0.0 1.0,0.5</m6c:Mapping>
      </m6c:Effect>
    </m6c:DefaultEffects>
    <tm:subtask id="id1109607464129281" name="Schritt 1"
           desc="Gas-Odorieranlage ausschalten" childorder="1>>2">
      <tm:subtask id="id114975231258247"
            name="Schritt 1.1"
            desc="Strom- und Gasversorgung des Heizkessels unterbrechen">
        <xmi:property key="keyword" value="ausschalten"/>
        <m6c:VisualizationPage resourcename="Abb07"/>
        <m6c:ActiveFeature id="feature_3" importance="1.0"/>
        <m6c:ActiveFeature id="feature_4" importance="0.7"/>
        <m6c:ActiveFeature id="feature_5" importance="0.3"/>
        <m6c:Effect type="Distort">
          <m6c:Mapping type="linear">0.0,0.5 1.0,2.0</m6c:Mapping>
        </m6c:Effect>
        <m6c:Effect type="ColorSaturation">
          <m6c:Mapping type="linear">0.0,0.5 1.0,1.0</m6c:Mapping>
        </m6c:Effect>
      </tm:subtask>

      <tm:subtask id="id114975231923851" name="Schritt 1.2"
            desc="Wasseranschluss unterbrechen, anschliessend Schlauch
            abschrauben">
        <xmi:property key="keyword" value="Bedienschrank"/>
        <m6c:VisualizationPage resourcename="Abb03"/>
        <m6c:ActiveFeature id="feature_1" importance="0.7"/>
        <m6c:ActiveFeature id="feature_2" importance="0.7"/>
        <m6c:ActiveFeature id="feature_3" importance="1.0"/>
        <m6c:ViewSelection x="170" y="140" width="120" height="125"/>
        <m6c:Effect type="Tinting">
          <m6c:Mapping type="linear">0.0,0.0 1.0,0.4</m6c:Mapping>
        </m6c:Effect>
      </tm:subtask>
    </tm:subtask>

    <tm:subtask id="id1109607534528294" name="Schritt 2"
```

```
        desc="Messing-Reduzierring wechseln"
        childorder="1>>2>>3>>4>>5>>6>>7">
<m6c:VisualizationPage resourcename="Abb29L"/>

<tm:subtask id="id1109607591424327" name="Schritt 2.1"
          desc="Die zwei Kreuzschlitzschrauben am Geblaesestutzen
                und die Sechskantschraube am Geblaesehalter loesen"
          category="interaction">
  <xmi:property key="keyword" value="Geblaesestutzen"/>
  <xmi:property key="keyword" value="Geblaesehalter"/>
  <m6c:ActiveFeature id="feature_1" importance="0.8"/>
  <m6c:ActiveFeature id="feature_5" importance="1.0"/>
  <m6c:ActiveFeature id="feature_6" importance="1.0"/>
  <m6c:ActiveFeature id="feature_7" importance="0.2"/>
  <m6c:Effect type="Distort">
    <m6c:Mapping type="linear">0.0,0.5 1.0,2.0</m6c:Mapping>
  </m6c:Effect>
  <m6c:Effect type="Tinting">
    <m6c:Mapping type="linear">0.0,0.0 1.0,0.75</m6c:Mapping>
  </m6c:Effect>
</tm:subtask>

<tm:subtask id="id1109607626895342" name="Schritt 2.2"
          desc="Geblaese abziehen" category="interaction">
  <xmi:property key="keyword" value="Geblaese"/>
  <m6c:ActiveFeature id="feature_1" importance="0.7"/>
  <m6c:ActiveFeature id="feature_3" importance="0.6"/>
  <m6c:ActiveFeature id="feature_5" importance="1.0"/>
  <m6c:ActiveFeature id="feature_6" importance="1.0"/>
  <m6c:ActiveFeature id="feature_7" importance="0.4"/>
  <m6c:ActiveFeature id="feature_9" importance="0.5"/>
  <m6c:ActiveFeature id="feature_11" importance="1.0"/>
  <m6c:Effect type="Distort">
    <m6c:Mapping type="linear">0.0,0.5 1.0,2.0</m6c:Mapping>
  </m6c:Effect>
  <m6c:Effect type="Tinting">
    <m6c:Mapping type="linear">0.0,0.0 1.0,0.75</m6c:Mapping>
  </m6c:Effect>
</tm:subtask>

<tm:subtask id="id1109607626895342" name="Schritt 2.3"
          desc="Geblaesestutzen abziehen" category="interaction">
  <xmi:property key="keyword" value="Geblaesestutzen"/>
  <m6c:VisualizationPage resourcename="Abb29R">
    <m6c:ActiveFeature id="feature_1" importance="1.0"/>
    <m6c:ActiveFeature id="feature_3" importance="0.8"/>
  </m6c:VisualizationPage>
</tm:subtask>
<tm:subtask id="id1109607626895342" name="Schritt 2.4"
          desc="Messing-Reduzierring herausnehmen" category="interaction">
  <xmi:property key="keyword" value="Messing-Reduzierring"/>
  <m6c:VisualizationPage resourcename="Abb29R">
    <m6c:ActiveFeature id="feature_1" importance="0.5">
      <m6c:Effect type="Distort">
        <m6c:Mapping type="stepwise">
          0.0,1.0 0.5,1.5 0.75,2.0
        </m6c:Mapping>
      </m6c:Effect>
      <m6c:Effect type="ColorSaturation">
        <m6c:Mapping type="stepwise">
          0.0,0.0 0.5,0.75 0.75,1.0
        </m6c:Mapping>
      </m6c:Effect>
```

```
        </m6c:ActiveFeature>
        <m6c:ActiveFeature id="feature_2" importance="1.0">
          <m6c:Effect type="ColorSaturation">
            <m6c:Mapping type="stepwise">0.0,0.0 0.5,1.0</m6c:Mapping>
          </m6c:Effect>
        </m6c:ActiveFeature>
      </m6c:VisualizationPage>
    </tm:subtask>

    <tm:subtask id="id1109607626895342" name="Schritt 2.5"
              desc="Neuen Messing-Reduzierring nach Tabelle auswaehlen und
                    mit O-Ring einsetzen. Der O-Ring hat dabei keine
                    Dichtungsfunktion."
              category="interaction">
      <xmi:property key="keyword" value="O-Ring"/>
      <xmi:property key="keyword" value="Messing-Reduzierring"/>
      <m6c:VisualizationPage resourcename="Abb29R">
        <m6c:ActiveFeature id="feature_1" importance="0.5"/>
        <m6c:ActiveFeature id="feature_2" importance="0.8"/>
        <m6c:ActiveFeature id="feature_5" importance="1.0"/>
      </m6c:VisualizationPage>
    </tm:subtask>

    <tm:subtask id="id1109607777164547" name="Schritt 2.6"
              desc="Geblaesestutzen wieder aufsetzen und dabei auf
                    einwandfreien Sitz der Gummidichtung achten"
              category="interaction">
      <xmi:property key="keyword" value="Geblaesestutzen"/>
      <xmi:property key="keyword" value="Gummidichtung"/>
      <m6c:VisualizationPage resourcename="Abb29R">
        <m6c:ActiveFeature id="feature_1" importance="0.9"/>
        <m6c:ActiveFeature id="feature_3" importance="0.8"/>
        <m6c:ActiveFeature id="feature_4" importance="1.0"/>
      </m6c:VisualizationPage>
    </tm:subtask>

    <tm:subtask id="id1149754788082251" name="Schritt 2.7"
              desc="Geblaese wieder aufsetzen und mit den zwei
                    Kreuzschlitzschrauben und der Sechskantschraube
                    verschrauben"
              category="interaction">
      <xmi:property key="keyword" value="Geblaese"/>
      <m6c:ActiveFeature id="feature_1" importance="0.7"/>
      <m6c:ActiveFeature id="feature_3" importance="0.6"/>
      <m6c:ActiveFeature id="feature_5" importance="1.0"/>
      <m6c:ActiveFeature id="feature_6" importance="1.0"/>
      <m6c:ActiveFeature id="feature_7" importance="0.4"/>
      <m6c:ActiveFeature id="feature_9" importance="0.5"/>
      <m6c:ActiveFeature id="feature_11" importance="1.0"/>
      <m6c:Effect type="Distort">
        <m6c:Mapping type="linear">0.0,0.5 1.0,2.0</m6c:Mapping>
      </m6c:Effect>
      <m6c:Effect type="Tinting">
        <m6c:Mapping type="linear">0.0,0.0 1.0,0.75</m6c:Mapping>
      </m6c:Effect>
    </tm:subtask>
  </tm:subtask>

  <tm:subtask id="id1109607552639306"  name="Schritt 3"
            desc="Strom- und Gasversorgung wieder einschalten"
            category="interaction">
    <xmi:property key="keyword" value="Inbetriebnahme"/>
    <xmi:property key="keyword" value="Reglerart"/>
```

```
      <m6c:VisualizationPage resourcename="Abb07">
        <m6c:ActiveFeature id="feature_3" importance="1.0"/>
        <m6c:ActiveFeature id="feature_4" importance="0.7"/>
        <m6c:ActiveFeature id="feature_5" importance="0.3"/>
      </m6c:VisualizationPage>
    </tm:subtask>
  </tm:root>
</tm:TaskModel>
```

Listing II – Example raster PAGE manifest

```
<?xml version="1.0" encoding="utf-8"?>
<pageMetadata symbolicName="Abb29L_noLabels"
              type="RASTER"
              xmlns="http://www.m6c.de">
  <imageFilename>Abb29L_noLabels.png</imageFilename>
  <featureList background="0x000000" nonFeature="0xFFFFFF">
    <feature color="0xFFFF0000" id="feature_1" label="Blower">
      <polygon>
        <points>
          338,199 312,199 307,204 307,419 312,424 339,424 346,419 346,203
          341,199
        </points>
      </polygon>
    </feature>
    <feature color="0xFF0000FF" id="feature_2" label="Short Feed Pipe">
      <polygon>
        <points>
          306,211 270,210 249,215 217,215 200,219 192,226 193,235 201,243
          216,247 227,248 249,247 270,250 307,251
        </points>
      </polygon>
    </feature>
    <feature color="0xFFFF00FF" id="feature_3" label="Rubber Dampener">
      <polygon>
        <points>
          300,383 299,434 351,433 352,383 347,383 346,427 306,427 305,383
        </points>
      </polygon>
    </feature>
    <feature color="0xFFFF8040" id="feature_4" label="Reducing Ring">
      <rectangle x="227" width="21" y="216" height="31"/>
    </feature>
    <feature color="0xFF0080C0" id="feature_5" label="Boiler Body">
      <rectangle x="262" width="38" y="373" height="135"/>
    </feature>
    <feature color="0xFF808000" id="feature_6" label="Fastening Screws (upper)">
      <rectangle x="290" width="10" y="237" height="10"/>
      <rectangle x="290" width="12" y="206" height="7"/>
    </feature>
    <feature color="0xFF008000" id="feature_7" label="Fastening Screws (lower)">
      <rectangle x="302" width="9" y="461" height="17"/>
    </feature>
    <feature color="0xFFDF9B03" id="feature_8" label="Burn Chamber">
      <polygon>
        <points>
          210,202 137,202 138,301 200,301 200,292 260,291 257,284 251,282
          227,281 220,278 217,272 218,212 215,206
        </points>
      </polygon>
    </feature>
    <feature color="0xFF000080" id="feature_9" label="Blower Journal">
      <polygon>
```

```
        <points>
          306,290 297,297 298,332 307,336</points>
      </polygon>
    </feature>
    <feature color="0xFF800000" id="feature_10" label="Gas Feed Pipe">
      <polygon>
        <points>
          310,182 311,68 308,55 299,42 284,34 274,31 138,31 128,22 279,23
          292,27 303,34 311,43 317,54 318,67 319,182 321,183 322,199 313,200
          308,202 308,183
        </points>
      </polygon>
    </feature>
    <feature color="0xFFFF0001" id="feature_11" label="Blower Bracket">
      <polygon>
        <points>
          350,379 351,432 299,432 299,502 302,502 303,439 357,438 356,379
        </points>
      </polygon>
    </feature>
  </featureList>
</pageMetadata>
```

Bibliography

[AA05] Natalia Andrienko and Gennady Andrienko: *Exploratory Analysis of Spatial and Temporal Data – A Systematic Approach*. Springer-Verlag, December 2005. ISBN 3-540-25994-5.

[AASRS03] R. Abi-Aad, D. Sinnig, T. Radhakrishnan and A. Seffah: CoU: Context of Use Model for User Interface Design. In *Proceedings of HCI International 2003*, volume 4, (pp. 8–12), 2003.

[ABF+06] Marco Attene, Silvia Biasotti, Bianca Falcidieno, Michela Mortara, Giuseppe Patanè and Michela Spagnuolo: Topological, Geometric and Structural Approaches to Enhance Shape Information. In *Proceedings of Eurographics Italian Chapter*, (pp. 7–13). Catania (I), 22-24 Feb 2006.

[AD67] J. Annett and K.D. Duncan: Task Analysis and Training Design. *Journal of Occupational Psychology*, volume 41:211–221, 1967.

[AE06] E.H.L. Aarts and J.L. Encarnação: *True Visions: The Emergence of Ambient Intelligence*. Springer, 2006.

[AES05] R. Amar, J. Eagan and J. Stasko: Low-level components of analytic activity in information visualization. In *Proc. IEEE Symposium on Information Visualization (InfoVis)*, (pp. 111–117), 2005.

[AFS06] Marco Attene, Bianca Falcidieno and Michela Spagnuolo: Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput.*, volume 22(3):181–193, 2006. ISSN 0178-2789.

[AHFS08] Kamran Ali, Knut Hartmann, Georg Fuchs and Heidrun Schumann: Adaptive Layout for Interactive Documents. In *Proc. 9th International Symposium on Smart Graphics (SG 2008)* (edited by Andreas Butz, Brian Fischer, Antonio Krüger, Patrick Oliver and Marc Christie), volume LNCS 5166, (pp. 247–254). Springer Verlag, Rennes, France, August 27th - 29th 2008. ISBN 978-3-540-85410-4. ISSN 0302-9743. Received Best Short Paper Award.

[AHS05] K. Ali, K. Hartmann and Th. Strothotte: Label Layout for Interactive 3D Illustrations. *Journal of WSCG*, volume 13(1–3):1–8, 2005.

[AKM+06] M. Attene, S. Katz, M. Mortara, G. Patanè, M. Spagnuolo and A. Tal: Mesh Segmentation - A Comparative Study. *IEEE International Conference on Shape Modeling and Applications (SMI'06)*, volume 0:7, 2006.

[Ali09] Kamran Ali: *Adaptive Layout for interactive Documents*. PhD Thesis, University of Rostock, 2009.

[Alm06] João Paulo Andrade Almeida: *Model-Driven Design of Distributed Applications*. CTIT Ph.D.-Thesis Series, No. 06-85 Telematica Instituut Fundamental Research Series , No. 018 (TI/FRS/018), Telematica Instituut, University of Twente, Enschede, The Netherlands, 2006.

[APH+03] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan and Barbara Tversky: Designing Effective Step-By-Step Assembly Instructions. In *SIGGRAPH 2003*, (pp. 828–837), July 2003.

[ARSF07a] Marco Attene, Francesco Robbiano, Michela Spagnuolo and Bianca Falcidieno: Part-based Annotation of Virtual 3D Shapes. In *Proceedings of Cyberworlds '07, spec. sess. on NASAGEM workshop*, (pp. pp. 427–436). IEEE Computer Society Press, 2007.

[ARSF07b] Marco Attene, Francesco Robbiano, Michela Spagnuolo and Bianca Falcidieno: Semantic Annotation of 3D Surface Meshes based on Feature Characterization. *Lecture Notes in Computer Science*, volume Vol. 4816 (SAMT'07 Procs.):pp. 126–139, 2007.

[AS04] R. Amar and J. Stasko: A Knowledge Task-Based Framework for Design and Evaluation of Information Visualizations. In *IEEE Symposium onInformation Visualization (INFOVIS 2004)*, (pp. 143–150). IEEE, Austin, TX, 2004. ISBN 0-7803-8779-3. ISSN 1522-404X. Best Paper.

[AS07] S. Avidan and A. Shamir: Seam carving for content-aware image resizing. In *International Conference on Computer Graphics and Interactive Techniques*. ACM Press New York, NY, USA, 2007.

[BB87] Tommaso Bolognesi and Ed Brinksma: Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, volume 14(1):25–59, 1987. Appeared also in: P. H. J. van Eijk, C. A. Vissers and M. Diaz (Editors), The Formal Description Technique LOTOS, Elsevier Science Publishers B. V., (North-Holland), 1989.

[BB98] G. Badros and A. Borning: The Cassowary linear arithmetic constraint solving algorithm: Interface and implementation. Technical Report UW-CSE-98-06-04, University of Washington, Seattle, Washington, June 1998.

[BC06] Jeanette Bautista and Giuseppe Carenini: An Integrated TaskBased Framework for the Design and Evaluation of Visualizations to Support Preferential Choice. In *Proceedings AVI'06*, (pp. 217–224). ACM, Venezia, Italy, May 23–26 2006.

[BDF+06] Norman Biehl, Antje Düsterhöft, Peter Forbrig, Georg Fuchs, Daniel Reichart and Heidrun Schumann: Advanced Multi-Modal User Interfaces for Mobile Devices - Integration of Visualization, Speech Interaction and Task Modeling. In *Proceedings 17th IRMA International Conference*. Washington D.C., USA, May 21st-24th 2006. (Short Paper).

[Ben00] Todd Bentley: Task Modelling Review of Methods, Tools, and Other. Technical Report CSIRO/MIS 2000/155, Swinburne University of Technology / CSIRO Mathematical and Information Sciences, Melbourne, Australia, August 2000.

[BEPW00] Klaus Backhaus, Bernd Erichson, Wulff Plinke and Rolf Weiber: *Multivariante Analysemethoden: eine anwendungsorientierte Einführung*. 9. "uberarbeitete und erweiterte Auflage edition. Springer Verlag, Berlin, 2000.

[Ber82] Jacques Bertin: *Graphische Darstellungen und die graphische Weiterverarbeitung der Information*. de Gruyter Berlin, 1982. Übersetzt und bearbeitet von Wolfgang Scharfe.

[BG05] Stefan Bruckner and Eduard Gröller: VolumeShop: An Interactive System for Direct Volume Illustration. In *Proc. IEEE Visualization Conference*, (p. 85). IEEE Computer Society, Los Alamitos, CA, USA, 2005. ISBN 0-7803-9462-3.

[BHLT06] T. Bray, D. Hollander, A. Layman and R. Tobin: Namespaces in XML 1.0 (Second Edition), 2006.
`http://www.w3.org/TR/xml-names/`

[BJR00] Grady Booch, Ivar Jacobson and Jim Rumbaugh: OMG Unified Modeling Language (UML) Specification, March 2000.

[BKO00] Andreas Butz, Antonio Krüger and Patrick Olivier (editors): *Proc. Smart Graphics – Papers from the AAAI Spring Symposium (Technical Report SS-00-04)*, 2000. ISBN 978-1-57735-110-8.

[BMPW00] O. Buyukkokten, H. G. Molina, A. Paepcke and T. Winograd: Power Browser: Efficient Web Browsing for PDAs. In *Proc. of the Conf. on Human Factors in Computing Systems, CHI'00*, (pp. 430–437), 2000.

[Bre94] C.A. Brewer: Color Use Guidelines for Mapping and Visualization. In *Visualization in Modern Cartography*, (pp. 123–147). Elseview Science, Tarrytown, NY, 1994.

[Bre99] C.A. Brewer: Color Use Guidelines for Data Representation. In *Proc. of the Section on Statistical Graphics*, (pp. 55–60). American Statistical Association, Alexandria, VA, 1999.

[BRS01] S. Beckhaus, F. Ritter and T. Strothotte: Guided exploration with dynamic potential fields: the CubicalPath System. *Computer Graphics Forum*, volume 20(4):201–210, December 2001.

[BRT95] L. Bergman, B. E. Rogowitz and L. A. Treinish: A Rule-based Tool for Assisting Colormap Selection. In *Proc. of IEEE Visualization*, (pp. 118–125), 1995.

[BSF+94] Eric A. Bier, Maureen C. Stone, Ken Fishkin, William Buxton and Thomas Baudel: A Taxonomy of See-Through Tools. In *Proceedings of CHI'94 conference*, (pp. 358–364). Addison-Wesley, April 1994.

[BSP+93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton and Tony D. DeRose: Toolglass and Magic Lenses: The See-Through Interface. *Computer Graphics*, volume 27(Annual Conference Series):73–80, 1993.

[BSP97] E. Bier, M. Stone and K. Pier: Enhanced Illustration Using Magic Lens Filters. *Computer Graphics and Applications, IEEE*, volume 17(6):62–70, November/December 1997. ISSN 0272-1716.

[BTM+01] Greg J. Badros, Jojada J. Tirtowidjojo, Kim Marriott, Bernd Meyer, Will Portnoy and Alan Borning: A constraint extension to scalable vector graphics. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, (pp. 489–498). ACM, New York, NY, USA, 2001. ISBN 1-58113-348-0.

[BZM+04] I. Bruder, A. Zeitz, H. Meyer, B. Hänsel and A. Heuer: FlyingDoc: An Architecture for Distributed, User-friendly, and Personalized Information Systems. In *20th Int. Conf. on Data Engineering*, (p. 849), 2004.

[Cas91] S. M. Casner: A Task-Analytic Approach to the Automated Design of Graph Presentations. In *ACM Transactions on Graphics*, volume 10, (pp. 111–151), April 1991.

[CCF95] M. Sheelagh T. Carpendale, David J. Cowperthwaite and F. David Fracchia: 3-Dimensional Pliable Surfaces: For the Effective Presentation of Visual Information. In *ACM Symposium on User Interface Software and Technology*, (pp. 217–226), 1995.

[CCT01] Gaëlle Calvary, Joëlle Coutaz and David Thevenin: A Unifying Reference Framework for the Development of Plastic User Interfaces. In *Engineering for Human-Computer Interaction* (edited by Murray Little and Laurence Nigay), *Lecture Notes in Computer Science*, volume 2254, (pp. 173–192). Springer Berlin / Heidelberg, 2001. ISBN 978-3-540-43044-5.

[CCT+03] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon and Jean Vanderdonckt: A Unifying Reference Framework for multi-target user interfaces. *Interacting with Computers*, volume 15(3):289–308, June 2003. ISSN 0953-5438. Computer-Aided Design of User Interface.

[CD05] D. Cook and S.K. Das: *Smart Environments*. Wiley-Interscience, 2005.

[Che10] Chaomei Chen: Information visualization. *Wiley Interdisciplinary Reviews: Computational Statistics*, volume 2(4):387–403, 2010.

[Chi00] Ed H. Chi: A Taxonomy of Visualization Techniques using the Data State Reference Model. In *INFOVIS*, (pp. 69–76), 2000.

[Cla99] James Clark: XSL Transformations (XSLT), Version 1.0, W3C Recommendation. http://www.w3.org/TR/xslt, 1999.

[CM97] Stuart K. Card and Josh Mackinlay: The structure of the information visualization design space. *Information Visualization, IEEE Symposium on*, volume 0:92, 1997.

[CMN83] Stuart K. Card, T.P. Moran and A. Newell: *The psychology of human-computer interaction*. Erlbaum, 1983.

[CMS99] Stuart K. Card, Jock D. Mackinlay and Ben Shneiderman: *Readings in information visualization: using vision to think*. The Morgan Kaufmann series in interactive technologies, Morgan Kaufmann, San Francisco, Calif., 1999.

[Con03] L. L. Constantine: Canonical Abstract Prototypes for Abstract Visual and Interaction Design. In *Proceedings DSV-IS* (edited by J. A. Jorge et al.), (pp. 1–15). LNCS 2844, Springer Verlag, Berlin, 2003.

[CR98] Ed Huai-hsin Chi and John T. Riedl: An Operator Interaction Framework for Visualization Systems. In *Proceedings IEEE Symposium on Information Visualization 1998*, (pp. 63–70), 1998.

[Dam97] Andries van Dam: Post-WIMP User Interfaces. *Communications of the ACM*, volume 40(2):63–67, February 1997. ISSN 0001-0782.

[Den99] B. D. Dent: *Cartography, Thematic Map Design*. fifth edition. McGraw-Hill, 1999.

[DeV90] Pamela B. DeVinne (editor): *The Tormont Webster's Illustrated Encyclopedic Dictionary*. Tormont Publications Inc, 1990. ISBN 2-921171-32-5.

[DH02] H. Doleisch and H. Hauser: Smooth Brushing for Focus and Context Visualization of Simulation data in 3D. *Journal of WSCG*, volume Vol. 10:147–154, 2002.

[DMO01] S. DeRose, E. Maler and D. Orchard: XML Linking Language (XLink) Version 1.0, 2001.
`http://www.w3.org/TR/xlink/`

[DWE02] J. Diepstraten, D. Weiskopf and T. Ertl: Transparency in Interactive Technical Illustrations. *Computer Graphics Forum*, volume 21(3):317–325, 2002.

[DWE03] J. Diepstraten, D. Weiskopf and T. Ertl: Interactive Cutaway Illustrations. *Computer Graphics Forum*, volume 22(3):523–532, 2003.

[ED06] Geoff Ellis and Alan Dix: Enabling Automatic Clutter Reduction in Parallel Coordinate Plots. In *IEEE Transactions on Visualization and Computer Graphics*, volume 12, (pp. 717–724). IEEE, Baltimore, Maryland, USA, Sept.-Oct. 2006.

[ED07] G. Ellis and A. Dix: A Taxonomy of Clutter Reduction for Information Visualisation. *IEEE Transactions on Visualization and Computer Graphics*, volume 13(6):1216–1223, 2007. ISSN 1077-2626.

[EK05] J.L. Encarnação and T. Kirste: Ambient Intelligence: Towards Smart Appliance Ensembles. In *From Integrated Publication and Informations Systems to Virtual Information and Knowledge Environments*, (pp. 261–270). Springer, 2005.

[ELMS91] P. Eades, W. Lai, K. Misue and K. Sugiyama: Preserving the mental map of a diagram. In *Proceedings of CompuGraphics*, volume 91, (pp. 24–33), August 1991.

[EVP01] Jacob Eisenstein, Jean Vanderdonckt and Angel Puerta: Applying model-based techniques to the development of UIs for mobile computers. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, (pp. 69–76). ACM Press, New York, NY, USA, 2001. ISBN 1-58113-325-1.

[FBD+05] Peter Forbrig, Gregor Buchholz, Anke Dittmar, Andreas Wolff and Daniel Reichart: Model-Based Software Development and Usability Testing. In *Workshop proceedings INTERACT*. Rome, September 2005.

[FD06] Z. Ben Fredj and D. A. Duce: GraSSML: Accessible Smart Schematic Diagrams for All. In *W4A'2006 International Cross-Disciplinary Conference on Web Accessibility, ACM International Conference Proceeding Series*, volume 134, (pp. 57–60). ACM Press, Edinburgh, Scotland, UK, May 2006. ISBN 1-59593-281-X.

[FFIT00] I. Fujishiro, R. Furuhata, Y. Ichikawa and Y. Takeshima: GADGET/IV: A Taxonomic Approach to Semi-Automatic Design of Information Visualization Applications Using Modular Visualization Environment. In *Proceedings of IEEE Information Visualization*, (p. 77ff.), 2000.

[FFJ03] Jon Ferraiolo, Jun. Fujisawa and Dean Jackson: Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR/SVG11/, January 2003. `http://www.w3.org/TR/SVG11/`

[FHS08] Georg Fuchs, Mathias Holst and Heidrun Schumann: 3D Mesh Exploration for Smart Visual Interfaces. In *Proc. 10th Intl. Conference on Visual Information Systems* (edited by Monica Sebillo, Guiliana Vitiello and Gerald Schaefer), volume LNCS 5188, (pp. 80–91). Springer Verlag, Salerno, Italy, 11-12 September 2008.

[FLH+06] G. Fuchs, M. Luboschik, K. Hartmann, K. Ali, Th. Strothotte and H. Schumann: Adaptive Labeling for Interactive Mobile Information Systems. *10th International Conference Information Visualisation (IV'06)*, volume 0:453–459, 2006. ISSN 1550-6037.

[FMP97] L. D. Floriani, P. Magillo and E. Puppo: Building and traversing a surface at variable resolution. In *Proceedings of the 8th conference on Visualization '97*, (p. 103–ff.). IEEE Computer Society Press, 1997.

[FMS93] Steven Feiner, Blair Macintyre and Dorée Seligmann: Knowledge-based augmented reality. *Communications of the ACM*, volume 36(7):53–62, 1993. ISSN 0001-0782.

[FRS11] Georg Fuchs, René Rosenbaum and Heidrun Schumann: Progressive Imagery with Scalable Vector Graphics. In *IS&T/SPIE Electronic Imaging 2011 – Multimedia on Mobile Devices*. San Francisco, USA, January 23–27 2011.

[FRSF06] Georg Fuchs, Daniel Reichart, Heidrun Schumann and Peter Forbrig: Maintenance Support – Case Study for a Multimodal Mobile User Interface. In *Multimedia on Mobile Devices II*, *Proceedings of SPIE*, volume 6074. The International Society for Optical Engineering, January 2006. ISBN 0-8194-6114-8.

[FRW08] Peter Forbrig, Daniel Reichart and Andreas Wolff: User Interfaces from Task Models. In *MDSE*. Berlin, 2008.

[FS06] Georg Fuchs and Heidrun Schumann: Visualization in Mulitmodal User Interfaces of Mobile Applications. In *Proceedings 17th IRMA International Conference*. Washington D.C., USA, May 21st-24th 2006.

[FS09] Georg Fuchs and Heidrun Schumann: Smart Visual Interfaces: Adaptive Anzeige graphischer Modelldaten. *CAD-CAM Report*, volume 1/2-09:50–53, Januar 2009.

[FSS07] Georg Fuchs, Hans-Jörg Schulz and Heidrun Schumann: Presenting Technical Drawing on Mobile Handhelds. In *Proceedings of 18th IRMA International Conference*. IRMA, Vancouver, Canada, May 2007.

[FTS07] Georg Fuchs, Conrad Thiede and Heidrun Schumann:: Pluggable Lenses for Interactive Visualizations. In *Poster Compendium of InfoVis'07*, October 2007.

[FTSS09] Georg Fuchs, Conrad Thiede, Mike Sips and Heidrun Schumann: Device-based Adaptation of Visualizations in Smart Environments. In *Workshop Collaborative Visualization on Interactive Surfaces (CoVIS), IEEE VisWeek 2009*. Atlantic City, USA, 11th-16th October 2009.

[Fur86] G. W. Furnas: Generalized fisheye views. *Human Factors in Computing Systems*, volume CHI '86:16–23, April 1986.

[FWDR06] Peter Forbrig, Andreas Wolff, Anke Dittmar and Daniel Reichart: Tool Support for an Evolutionary Design Process using XML and User-Interface Patterns. In *Proc. CUSEC 2006*. Montreal, January 2006.

[GF04] Carl Gutwin and Chris Fedak: Interacting with big interfaces on small screens: a comparison of fisheye, zoom, and panning techniques. In *Proceedings of Graphics Interface (GI '04)*, (pp. 145–152). Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. ISBN 1-56881-227-2.

[GFF+07] Martin Giersich, Peter Forbrig, Georg Fuchs, Thomas Kirste, Daniel Reichart and Heidrun Schumann: Towards an Integrated Approach for Task Modeling and Human Behavior Recognition. In *Proc. HCI International 2007: 12th International Conference on Human-Computer Interaction*, *LNCS*, volume 1, (pp. 1109–1118). Springer, Beijing, China, July 22-27 2007.

[GFS05] Henning Griethe, Georg Fuchs and Heidrun Schumann: A Classification Scheme for Lens Techniques (Short Paper). In *13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'2005)*. Plzen, Czech Republic, January 31th - February 4th 2005.

[GGSC98] Amy Gooch, Bruce Gooch, Peter Shirley and Elaine Cohen: A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, (pp. 447–452). ACM, 1998.

[GH97] Michael Garland and Paul S. Heckbert: Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, (pp. 209–216). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997. ISBN 0-89791-896-7.

[GHS06] T. Götzelmann, K. Hartmann and Th. Strothotte: Contextual Grouping of Labels. In *17th Conference on Simulation and Visualization* (edited by T. Schulze, G. Horton, B. Preim and S. Schlechtweg), (pp. 245–258). SCS Publishing House, 2006.

[Goe07] Daniel Goebel: *Semantische Anreicherung polygonaler Netze zur Unterstützung der interaktiven Exploration*. Master Thesis (Diplomarbeit), University of Rostock, April 29 2007.

[GSSF04] A. Gaffar, D. Sinnig, A. Seffah and P. Forbrig: Modeling patterns for task models. In *TAMODIA '04: Proceedings of the 3rd annual conference on Task models and diagrams*, (pp. 99–104). ACM, New York, NY, USA, 2004. ISBN 1-59593-000-0.

[GWH01a] Michael Garland, Andrew Willmott and Paul S. Heckbert: Hierarchical face clustering on polygonal surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, (pp. 49–58). ACM, New York, NY, USA, 2001. ISBN 1-58113-292-1.

[GWH01b] Michael Garland, Andrew Willmott and Paul S. Heckbert: Hierarchical face clustering on polygonal surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, (pp. 49–58). ACM, New York, NY, USA, 2001. ISBN 1-58113-292-1.

[HA05] Arthur H. ter Hofstede and Wil M. van der Aalst: YAWL: Yet Another Workflow Language. *Information Systems*, volume 30(4):245–275, 2005.

[Hau06] Helwig Hauser: *Scientific Visualization: The Visual Extraction of Knowledge from Data*, chapter Generalizing Focus+Context Visualization, (pp. 305–327). Springer-Verlag Berlin Heidelberg, 2006. ISBN 9783540307907.

[HB03] Mark A. Harrower and Cynthia A. Brewer: ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps. *The Cartographic Journal*, volume 40(1):27–37, 2003.

[HBP+07] J. Huang, B. Bue, A. Pattath, D. Ebert and K. Thomas: Interactive Illustrative Rendering for Mobile Devices. *IEEE Computer Graphics and Applications*, volume 27(3):48–56, 2007.

[HDD+93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle: Mesh optimization. In *Proc. SIGGRAPH '93: 20th annual conference on Computer Graphics and Interactive Techniques*, (pp. 19–26). ACM Press, New York, NY, USA, 1993.

[Hei09] Thomas Heider: *A Unified Distributed System Architecture for Goal-based Interaction with Smart Environments*. PhD Thesis, University of Rostock, 2009.

[HF01] Kasper Hornbaek and Erik Frokjaer: Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, (pp. 293–300). ACM Press, 2001. ISBN 1-58113-327-8.

[HG94] G. Hake and D. Grünreich: *Kartographie*. 7th edition. deGruyter, Berlin, 1994.

[HK00] Jiawei Han and Micheline Kamber: *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[HK05] Thomas Heider and Thomas Kirste: Smart Environments and Self-Organizing Appliance Ensembles. In *Mobile Computing and Ambient Intelligence: The Challenge of Multimedia* (edited by Nigel Davies, Thomas Kirste and Heidrun Schumann). Number 05181 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany, 2005.

[HK07a] Thomas Heider and Thomas Kirste: Automatic vs. Manual Multi-Display Configuration: A Study of User Performance in a Semi-Cooperative Task Setting. In *Proc. of BCS HCI Group Conference*. Lancaster, UK, 2007.

[HK07b] Thomas Heider and Thomas Kirste: Minimizing Cognitive Load by Automatic Display Management. In *Proc. Ubicomp Workshop'07*, 2007.

[HM90] R. B. Haber and D. A. McNabb: Visualization idioms: A conceptual model for scientific Visualization Systems. In *Visualization in Scientific Computing* (edited by B. Shriver, G. M. Nielson and L. Rosenblum), (pp. 74–93). IEEE Computer Society Press, Los Alamitos, 1990.

[HM01] Ivan Herman and M. Scott Marshall: GraphXML - An XML-Based Graph Description Format. In *Proceedings of the 8th International Symposium on Graph Drawing*, (pp. 52–62). GD '00, Springer-Verlag, London, UK, 2001. ISBN 3-540-41554-8.

[HO95] K.-T. Huang and D. Overhauser: A novel graph algorithm for circuit recognition. *IEEE International Symposium on Circuits and Systems*, volume 3:1695–1698, 30 Apr-3 May 1995.

[Hol07] Mathias Holst: *Effiziente auflösungsvariable Oberflächenbeschreibung mit hybriden Modellen*. PhD Thesis, University of Rostock, 2007.

[Hop99] Axel Hoppe: *Validierung und Nachbearbeitung von gerenderten Bildern*. PhD. Thesis, Otto-von-Guericke Unversity of Magdeburg, 1999. Shaker Verlag, Aachen.

[HR98] JoAnn T. Hackos and Janice C. Redish: *User and Task Analysis for Interface Design*. Wiley Computer Publishing, 1998. ISBN 0-471-17831-4.

[HS06] Mathias. Holst and Heidrun Schumann: Efficient Rendering of High-Detailed Objects Using A Reduced Multi-Resolution Hierarchy. In *Proceedings of GRAPP 2006*, (pp. 3–10), February 2006.

[HS07] M. Holst and H. Schumann: Surfel-Based Billboard Hierarchies for Fast Rendering of 3D-Objects. In *Proc. Eurographics Symposium on Point-based Graphics*, (pp. 72–79). Praha, Sept. 2007.

[HSH90] H. Rex Hartson, Antonio C. Siochi and Deborah H. Hix: The UAN: a user-oriented representation for direct manipulation interface designs. *ACM Trans. Inf. Syst.*, volume 8(3):181–203, 1990. ISSN 1046-8188.

[IDE] IDELIX Software Inc.: Pliable Display Technology White Paper. `www.idelix.com`

[IKN98] Laurent Itti, Christof Koch and Ernst Niebur: A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20(11):1254–1259, November 1998.

[Jes05] Stefan Jeschke: *Accelerating the Rendering Process Using Impostors*. PhD Thesis, University of Rostock, 2005.

[JK96] Bonnie E. John and David E. Kieras: The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Computer-Human Interaction*, volume 3(4):320–351, 1996. ISSN 1073-0516.

[JPE00] ISO/IEC 15444-1: JPEG 2000 Core Coding System, December 2000.

[JPE01a] ISO/IEC 15444-2: JPEG 2000 Extensions, November 2001.

[JPE01b] ISO/IEC 15444-3: Motion JPEG 2000, November 2001.

[KB09] Antonio Krüger and Andreas Butz: Smart Graphics. *it - Information Technology*, volume 51(3):129–130, 05 2009. ISSN 1611-2776.

[KCY93] Arie Kaufman, Daniel Cohen and Roni Yagel: Volume Graphics. *IEEE Computer*, volume 26(7):51–64, July 1993.

[Kea98] T. Alan Keahey: The Generalized Detail-in-Context Problem. In *Proceedings of the IEEE Symposium on Information Visualization*, (pp. 44–51). IEEE Visualization, IEEE, Research Triangle Park, NC, October 19-20 1998.

[Kea99] T.A. Keahey: Area-normalized thematic views. In *Proceedings of International Cartography Assembly*, 1999.

[Kei05] Daniel A. Keim: Scaling Visual Analytics to Very Large Data Sets. In *Workshop on Visual Analytics*. University of Darmstadt, Germany, June 4th 2005.

[KGJV83] S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi: Optimization by Simulated Annealing. *Science*, volume 220(4598):671–680, May 1983.

[KHS04] M. Klima, P. Halabala and P. Slavik: Semantic information visualization. In *Proc. 19th Intl. CODATA Conference*, 2004.

[Kie04] David Kieras: *The Computer Science and Engineering Handbook*, chapter Task Analysis and the Design of Functionality, (pp. 1–26). 2nd edition. CRC Inc., Boca Raton, 2004.

[KJDG+06] H.S. Kim, C. Joslin, T. Di Giacomo, S. Garchery and N. Magnenat-Thalmann: Device-based decision-making for adaptation of three-dimensional content. *The Visual Computer*, volume 22(5):332–345, 2006.

[KK93] Peter R. Keller and Mary M. Keller: *Practical Data Visualization*. 1^{st} edition. IEEE Computer Society Press, Los Alamitos, 1993.

[Kli09] Martin Klima: *Collaborative Work in Mobile Environment*. PhD Thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Graphics and Interaction, February 2009.

[KLK03] Hae-Kwang Kim, Nam-Yeol Lee and Jin-Woong Kim: 3D Graphics Adaptation System on the Basis of MPEG-21 DIA. In *SG 2003: Procceedings Third International Symposium on Smart Graphics* (edited by Andreas Butz, Antonio Krüger and Patrick Olivier), *Lecture Notes in Computer Science*, volume LNCS 2733, (pp. 206–211). Springer-Verlag Berlin Heidelberg, Heidelberg, Germany, July 2-4 2003.

[KLS00] M. Kreuseler, N. López and H. Schumann: A Scalable Framework for Information Visualization. In *Proceedings IEEE Symposium on Information Visualization (InfoVis 2000)*, (pp. 27–36). Salt Lake City, USA, October 2000.

[KLT05] Sagi Katz, George Leifman and Ayellet Tal: Mesh segmentation using feature point and core extraction. *The Visual Computer*, volume 21(8–10):649–658, September 2005.

[KMH01] Robert Kosara, Silvia Miksch and Helwig Hauser: Semantic Depth of Field. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, (pp. 97–105). IEEE Computer Society, Washington, DC, USA, 2001. ISBN 0-7695-1342-5.

[KMH02a] Robert Kosara, Silvia Miksch and Helwig Hauser: Focus+Context Taken Literally. *IEEE Computer Graphics and Applications*, volume 22:22–29, 2002. ISSN 0272-1716.

[KMH+02b] Robert Kosara, Silvia Miksch, Helwig Hauser, Johann Schrammel, Verena Giller and Manfred Tscheligi: Useful Properties of Semantic Depth of Field for Better F+C Visualization. In *Proceedings of the symposium on Data Visualisation 2002*, (pp. 205–210). VISSYM '02, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, May 27-29 2002. ISBN 1-58113-536-X.

[KMS04] M. Klima, Z. Mikovec and P. Slavik: Adaptation of Graphical Data in Collaborative Environment. In *Workshop User Interfaces for All*, 2004.

[KRS03] B. Karstens, R. Rosenbaum and H. Schumann: Visual Interfaces for mobile handhelds. In *Proceedings HCI International (HCII'03)*. Crete, Greece, June 2003.

[Krü98] Antonio Krüger: Automatic graphical abstraction in intent-based 3D-illustrations. In *AVI '98: Proceedings of the working conference on Advanced visual interfaces*, (pp. 47–56). ACM, New York, NY, USA, 1998.

[LA94] Y. K. Leung and M. D. Apperley: A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Human-Computer Interaction*, volume 1(2):126–160, June 1994.

[LAS04] W. Li, M. Agrawala and D. Salesin: Interactive image-based exploded view diagrams. In *Proc. Graphics Interface (GI'04)*, (pp. 203–212), 2004.

[LCCV03] K. Luyten, T. Clerckx, K. Coninx and J. Vanderdonckt: Derivation of a dialog model from a task model by activity chain extraction. In *Proceedings of DSV-IS* (edited by J. Jorge, N.J. Nunes and J.F. e Cunha). LNCS 2844, Springer, 2003.

[LFSa] LFS – Landesforschungsschwerpunkt IuK: Multimediales Content-Management in Mobilen Umgebungen mit Multimodalen Nutzungsschnittstellen (LFS-M6C) / Proaktive verteilte Informationssysteme (LFS-ProVIS). http://www.m6c.de/ (last accessed 2010-03-31).

[LFSb] LFS-MA – Landesforschungsschwerpunkt Mobile Assistenzsysteme. http://www.lfs-ma.de/ (last accessed 2010-03-31).

[LG05] Feng Liu and Michael Gleicher: Automatic image retargeting with fisheye-view warping. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, (pp. 153–162). ACM Press, New York, NY, USA, 2005. ISBN 1-59593-271-2.

[LH94] M.M. Loughlin and J.F. Hughes: An Annotation System for 3D Fluid Flow Visualization. In *Proceeding IEEE Conference on Visualization'94*, (pp. 273–279), October 17–21 1994.

[LJMM+05] Víctor López-Jaquero, Francisco Montero, José P. Molina, P. González and A. Fernández-Caballero: A Seamless Development Process of Adaptive User Interfaces Explicitly Based on Usability Properties. In *Engineering Human Computer Interaction and Interactive Systems* (edited by Rémi Bastide, Philippe Palanque and Jörg Roth), *Lecture Notes in Computer Science*, volume 3425, (pp. 289–291). Springer Berlin / Heidelberg, 2005.

[LM09] Uwe von Lukas and Benjamin Mesing: Virtual Engineering im Schiffbau. *Economic Engineering: Intelligente Methoden, Prozesse und Technologien*, volume 6:62–65, 2009.

[LPKD01] E. Lin, C. Podilchuk, T. Kalker and E. Delp: Streaming video and rate scalable compression: What are the challenges for watermarking? In *Proceedings of SPIE - Security and Watermarking of Multimedia Contents III*, volume 4314, (pp. 116–127), 2001.

[LRC+02] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson and R. Huebner: *Level of Detail for 3D Graphics*. Morgan Kaufmann, 2002.

[LRS10] M. Luboschik, A. Radloff and H. Schumann: A new weaving technique for handling overlapping regions. In *Proceedings AVI 2010 – Advanced Visual Interfaces*, 2010.

[LSC08] Martin Luboschik, Heidrun Schumann and Hilko Cords: Particle-Based Labeling: Fast Point-Feature Labeling without Obscuring Other Visual Features. *IEEE Transactions on Visualization and Computer Graphics (TVCG) / Proceedings of IEEE Information Visualization (InfoVis'08)*, volume 14(6):1237–1244, November-December 2008.

[Lup07] Markus Lupp: OpenGIS Styled Layer Descriptor Profile of the Web Map Service, August 14th 2007.

[Luy04] Kris Luyten: *Dynamic User Interfaces Generation for Mobile and Embedded Systems with Model-Based user Interface Development*. Ph.D. thesis, Universiteit Maastricht, Maastricht, 2004.

[LV03] Quentin Limbourg and Jean Vanderdonckt: *The Handbook of Task Analysis for Human-Computer Interaction*, chapter Comparing Task Models for User Interface Design, (pp. 135–154). Lavoisier, 2003.

[LVM+04] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, Murielle Florins and Daniela Trevisan: UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces. In *Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages"*, (pp. 55–62). Press, 2004.

[LXMZ03] Hao Liu, Xing Xie, Wei-Ying Ma and Hong-Jiang Zhang: Automatic browsing of large pictures on mobile devices. In *11th ACM International Conference on Multimedia*. Berkeley, 2003.

[LZ04] Rong Liu and Hao Zhang: Segmentation of 3D Meshes through Spectral Clustering. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, (pp. 298–305). IEEE Computer Society, Washington, DC, USA, 2004. ISBN 0-7695-2234-3.

[Mac86] J. Mackinlay: Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, volume 5(2):110–141, April 1986.

[MAI] MAIKE – Mobile Assistenzsysteme für Intelligente Kooperierende Räume und Ensembles, http://maike.lfs-ma.de/. http://maike.lfs-ma.de/ (last accessed 2010-03-31).

[MAX] MAXIMA – Mobile Assistenzsysteme für eXpertengestütztes Instandhaltungs-MAnagement, http://maxima.lfs-ma.de/. http://maxima.lfs-ma.de/ (last accessed 2010-03-31).

[May07] Julia Mayer: The New Main Research Areas of Fraunhofer IGD. *Computer Graphics topics*, volume 19(2):18–19, 2007.

[McE94] Alan M. McEachren: *Some Truth With Maps: A Primer on Symbolization and Design*. Resource Publications in Geography, Assn of Amer Geographers, April 1994.

[MH04] Deborah L. McGuinness and Frank van Harmelen: OWL Web Ontology Language, W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-features/, 2004. Http://www.w3.org/TR/owl-features/.

[Mia00] John Miano: *Compressed Image File Formats*. Addison-Wesley, 2000. ISBN 0-201-60443-4.

[MKP02] José P. Molina M. Martínez, Rob Koenen and Fernando Pereira: MPEG-7: The Generic Multimedia Content Description Standard, Part 1. *IEEE MultiMedia*, volume 9(2):Peiya Liu, April - June 2002. ISSN 1070-986X.

[MKS02] Z. Mikovec, M. Klima and P. Slavik: Manipulation of complex 2D/3D scenes on mobile devices. In *Proceedings of the Second IASTED International Conference Visualization, Imaging, and Image Processing* (edited by J.J. Villanueva), (pp. 161–166). IASTED/ACTA Press, Anaheim, 2002.

[MM99] I. Mani and M.T. Maybury: *Advances in Automatic Text Summarization*. The MIT Press, 1999.

[MMM04] C. McCormack, K. Marriott and B. Meyer: Adaptive layout using one-way constraints in SVG. In *Proceedings of third Annual Conference on Scalabe Vector Graphics (SVG Open)*, 2004.

[MMS04] Kim Marriott, Bernd Meyer and Peter Stuckey: Towards Flexible Graphical Communication Using Adaptive Diagrams. In *Advances in Computer Science - ASIAN 2004, Lecture Notes in Computer Science (LNCS)*, volume 3321, (pp. 380–394). Springer Berlin / Heidelberg, December 2004. ISBN 978-3-540-24087-7.

[MMT02] Kim Marriott, Bernd Meyer and Laurent Tardif: Fast and efficient client-side adaptivity for SVG. In *Proceedings of the 11th international conference on World Wide Web (WWW'02)*, (pp. 496–507). ACM, New York, NY, USA, 2002. ISBN 1-58113-449-5.

[MPS02] Giulio Mori, Fabio Paterno and Carmen Santoro: CTTE: Support for developing and analyzing task models for interactive system design. *IEEE Transanctions on Software Engineering*, volume 28(8):797–813, August 2002.

[MPS+04] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno and J. Rossignac: Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *SM '04: Proceedings of the 9th ACM symposium on Solid modeling and applications*, (pp. 339–344). Eurographics Association, Aire-la-Ville, Switzerland, 2004. ISBN 3-905673-55-X.

[MS91] J. Marks and S. Shieber: The Computational Complexity of Cartographic Label Placement. Technical Report TR-05-91, Center for Research in Computing Technology, Harvard University, 1991.

[MSK03] Zdenek Mikovec, Pavel Slavik and Martin Klima: Human-Computer Communication in Special Environments. In *Proceedings of the IADIS International Conference WWW/Internet*, (pp. 763–766), 2003.

[MSK+06] Carmen Sanz Merino, Mike Sips, Daniel A. Keim, Christian Panse and Robert Spence: Task-at-hand interface for change detection in stock market data. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, (pp. 420–427). ACM, New York, NY, USA, 2006. ISBN 1-59593-353-0.

[Mun09] Tamara Munzner: A Nested Process Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics*, volume 15:921–928, 2009. ISSN 1077-2626.

[MuS09] MuSAMA Project Homepage. http://www.musama.de, 2009.

[MVL09] Benjamin Mesing, Matthias Vahl and Uwe von Lukas: Ein ganzheitlicher Ansatz für den Einsatz von VR im schiffbaulichen Sektor. In *Augmented und Virtual Reality in der Produktentstehung : Grundlagen, Methoden und Werkzeuge*, (pp. 275–288). HNI-Verlagsschriftenreihe 252, Gausemeier, Jürgen, Heinz Nixdorf Institut Paderborn, 2009.

[MW99] Alan P. Mangan and Ross T. Whitaker: Partitioning 3D Surface Meshes Using Watershed Segmentation. *IEEE Transactions on Visualization and Computer Graphics*, volume 5(4):308–321, 1999. ISSN 1077-2626.

[MZ03] Yu-Fei Ma and Hong-Jiang Zhang: Contrast-based image attention analysis by using fuzzy growing. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, (pp. 374–381). ACM, New York, NY, USA, 2003. ISBN 1-58113-722-2.

[NC98] D. Nister and C. Christopoulos: Lossless region of interest with a naturally progressive still image coding algorithm. In *Proceedings of ICIP*, volume 3, (pp. 856–859), 1998.

[Noc07] Thomas Nocke: *Visuelles Data Mining und Visualisierungsdesign für die Klimaforschung (Visual Data Mining and Visualization Design for Climate Research).* Dissertation (PhD Thesis), University of Rostock, Faculty of Computer Science and Electrical Engineering, 2007. (in German).

[Nor06] Chris North: Toward Measuring Visualization Insight. *IEEE Computer Graphics and Applications*, volume 26(3):6–9, 2006.

[NPF+04] W. Narzt, G. Pomberger, A. Ferscha, D. Kolb, R. Müller, J. Wieghardt, R. Bidner, H. Hörtner and C. Lindinger: PARIS - Personal Augmented Reality Information System. In *2nd ACM International Conference on Mobile Systems, Applications and Services*, June 2004.

[NS04] Thomas Nocke and Heidrun Schumann: Goals of Analysis for Visualization and Visual Data Mining Tasks. In *Proceedings of CODATA Workshop "Information Visualization, Presentation, and Design"*. Prague, Czech Republic, March 29^{th}-31^{st} 2004.

[Ope05] Open Geospatial Consortium: *OpenGIS Filter Encoding Implementation Specification, Version 1.1.0 (final).* Open Geospatial Consortium Inc., 2005.

[OW00] Thomas J. Overbye and Jamie D. Weber: New Methods for the Visualization of Electric Power System Information. In *IEEE Symposium on Information Visualization (InfoVis'00)*, 2000.

[Pat00] Fabio Paternò: *Model-based design and evaluation of interactive applications.* Springer Verlag, 2000.

[Pat03] Pattaya: MPEG-21 Digital Item Adaptation AM, 2003.

[PD84] Thomas Porter and Tom Duff: Compositing digital images. *SIGGRAPH Computer Graphics*, volume 18:253–259, January 1984. ISSN 0097-8930.

[PG88] R.M. Pickett and G.G. Grinstein: Iconographic Displays For Visualizing Multidimensional Data. In *Proc. 1988 IEEE International Conference on Systems, Man, and Cybernetics*, (pp. 514 – 519), 8-12 Aug 1988.

[Pla04] Catherine Plaisant: The challenge of information visualization evaluation. In *AVI'04: Proceedings of the working conference on Advanced visual interfaces*, (pp. 109–116). ACM, New York, NY, USA, 2004. ISBN 1-58113-867-9.

[PMM97] F. Paternò, C. Mancini and S. Meniconi: ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *Proceedings of Interact'97*, (pp. 362–369). Chapman & Hall, Sydney, 1997.

[Por07] Clemens Portele: OpenGIS Geography Markup Language (GML) Encoding Standard, Oktober 5th 2007.

[PR02] B. Preim and F. Ritter: Techniken zur Hervorhebung von Objekten in medizinischen 3d-Visualisierungen. In *Simulation und Visualisierung*, (pp. 187–200), 2002.

[PS02] F. Paternò and C. Santoro: One Model, Many Interfaces. In *Proceedings of the 4^{th} International Conference on Computer-Aided Design of User Interfaces*, (pp. 143–154). Kluwer Academics Publishers, 2002.

[Pue96] Angel R. Puerta: The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development. In *Proc. of CADUI'96: Computer-Aided Design of User Interfaces*, 1996.

[Pue97] A.R. Puerta: Model-Based Interface Development Environment. *IEEE Software*, volume 14(4):40–47, July-Aug. 1997.

[PZB02] Thomas Phan, George Zorpas and Rajive Bagrodia: An Extensible and Scalable Content Adaptation Pipeline Architecture to Support Heterogeneous Clients. In *22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02)*, (p. 507). IEEE Computer Society, Vienna, Austria, July 02-05 2002. ISBN 0-7695-1585-1. ISSN 1063-6927.

[Ras97] Wolf-Dieter Rase: Fischauge-Projektionen als kartographische Lupen. *Salzburger Geographische Materialien, Angewandte Geographische Informationsverarbeitung*, volume IX(26):115–122, 1997.

[Rau99] Uwe Rauschenbach: The Rectangular Fish Eye View as an efficient method for the transmission and display of large images. In *Proceedings IEEE International Conference on Image Processing (ICIP99)*. Kobe, Japan, October 25-28 1999.

[RDFW08] Daniel Reichart, Anke Dittmar, Peter Forbrig and Maik Wurdel: Tool Support for Representing Task Models, Dialog Models and User-Interface Specifications. In *Proc. DSV-IS 2008*. Kingston, Canada, 2008.

[Rei03] T. Reichenbacher: Adaptive methods for mobile cartography. In *Proceedings of the 21st International Cartographic Conference (ICC)*, (pp. 1311–1323). International Cartographic Association (ICA), Durban, South Africa, 10–16 August 2003.

[RF05] Daniel Reichart and Peter Forbrig: Multiple-User Interfaces based on Task, User and Object Models. In *Dagstuhl Seminar Proceedings 05181 "Mobile Computing and Ambient Intelligence: The Challenge of Multimedia"*. Dagstuhl, May 2005.

[RFS08] René Rosenbaum, Georg Fuchs and Heidrun Schumann: Region-wise meta-data in JPEG2000-encoded imagery. In *VIE 2008*. Xi'an/China, July 29th - August 01st 2008.

[Rhe99] Penny Rheingans: Task-based Color Scale Design. In *Proc. of Applied Image and Pattern Recognition*, (pp. 35–43). SPIE, 1999.

[RKMG94] S. F. Roth, J. Kolojejchick, J. Mattis and J. Goldstein: Interactive graphic design using automatic presentation knowledge. In *CHI'94: Proceedings Human Factors in Computing Systems*, (pp. 112–117). ACM, April 1994.

[RLS+96] S.F. Roth, P. Lucas, J.A. Senn, C.C. Gomberg, M.B. Burks, P.J. Stroffolino, A.J. Kolojechick and C. Dunmire: Visage: a user interface environment for exploring information. *Information Visualization, IEEE Symposium on*, volume 0:3, 1996.

[RLS11] Axel Radloff, Martin Luboschik and Heidrun Schumann: Smart Views in Smart Environments. In *Proceedings Smart graphics 2011*, 2011. (submitted).

[RM90] Steven F. Roth and Joe Mattis: Data Characterization for Intelligent Graphics Presentation. In *Proceedings of the Computer-Human Interaction Conference (CHI'90)*, 1990.

[Ros06] René Rosenbaum: *Mobile Image Communication using JPEG2000*. PhD Thesis, University of Rostock, Rostock/Germany, December 11 2006.

[RRS01] Uwe Rauschenbach, René Rosenbaum and Heidrun Schumann: A flexible Polygon Representation of Multiple Overlapping Regions of Interest for Wavelet-based Image Coding. In *Proceedings IEEE International Conference on Image Processing (ICIP)*. Thessaloniki, Griechenland, Okt 2001.

[RS98] Uwe Rauschenbach and Heidrun Schumann: Flexible embedded image communication using levels of detail and regions of interest. In *Proceedings of the workshop on Interactive Applications of Mobile Computing*. Rostock, Germany, November 24-25 1998.

[RS99] Uwe Rauschenbach and Heidrun Schumann: Demand-driven Image Transmission with Levels of Detail and Regions of Interest. *Computers and Graphics*, volume 23(6):857–866, 1999.

[RS09] René Rosenbaum and Heidrun Schumann: Progressive refinement – more than a means to overcome limited bandwidth. In *Proceedings of Electronic Imaging - Visualization and Data Analysis 2009*. San Jose/USA, January 18 - 22 2009.

[RT03] R. Rosenbaum and D. Taubman: Remote display of large raster images using JPEG2000 and the rectangular FishEye-View. In *IEEE-WSCG2003*. Plzen/Czech Republic, February 03-07 2003.

[RTS06] R. Rosenbaum, Ch. Tominski and H. Schumann: *Encyclopedia of E-Commerce, E-Government, and Mobile Commerce*, chapter Graphical Contents on Mobile Devices. Idea Group Publishing, 2006.

[SC01] K. Schneider and J. Cordy: Abstract user interfaces: A Model and a Notation to Support Plasticity in Interactive Systems. In *Proceedings of DSV-IS*, (pp. 40–58), 2001.

[Sch96] E. Schlungbaum: Model-based User Interface Software Tools – Current State of Declarative Models. Technical Report GIT-GVU-96-30, Georgia Institute of Technology, Atlanta, GA 30332-0280, November 1996.

[Sch06] R. Schmitz: *Ein Modell zur Beschreibung von Visualisierungszielen beim Einsatz von Farbkodierung zur Darstellung multivariater Daten.* Master thesis, University of Rostock, 2006. (in German).

[Sch10] Hans-Jörg Schulz: *Explorative Graph Visualization.* PhD thesis, University of Rostock, June 2010.

[SE96] E. Schlungbaum and T. Elwert: Automatic user interface generation from declarative models. In *Proc. CADUI'96*, (pp. 3–18), 1996.

[SF91] Dorée Duncan Seligmann and Steven Feiner: Automated Generation of Intent-based 3D Illustrations. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, (pp. 123–132). ACM Press, New York, NY, USA, 1991.

[Sha08] Ariel Shamir: A survey on Mesh Segmentation Techniques. *Computer Graphics forum*, volume 27(6):1539–1556, 2008.

[She09] Don Sherman: Technology of the Year: GM's Two-Mode Hybrid System. *Automobile Magazine*, volume 2:1–2, February 2009.

[Shn96] Ben Shneiderman: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization. In *Proceedings of the IEEE Symposium on Visual Languages*, (pp. 336–343), Sep 1996.

[Sie06] Siemens AG: Advanced Augmented Reality Technologies for Industrial Service Applications (ARTESAS). http://www.artesas.de, 2006.

[SKKM+05] L. Skorin-Kapov, H. Komericki, M. Matijasevic, I. Pandzic and M. Mosmondor: MUVA: a Flexible Visualization Architecture for Multiple Client Platforms. *Journal of Mobile Multimedia*, volume 1(1):3–17, 2005.

[SM00] Heidrun Schumann and Wolfgang Müller: *Visualisierung – grundlagen und allgemeine Methoden.* Springer Verlag, 2000.

[SMA+09] Peter Shirley, Steve Marschner, Michael Ashikhmin, Michael Gleicher, Naty Hoffman, Garrett Johnson, Tamara Munzner, Erik Reinhard, Kelvin Sung, William B. Thompson, Peter Willemsen and Brian Wyvill: *Fundamentals of Computer Graphics*, chapter 27: Visualization, (pp. 675–707). Third edition edition. Taylor & Francis Ltd, 2009.

[SMS07] Samuel Silva, Joaquim Madeira and Beatriz Sousa Santos: There is More to Color Scales than Meets the Eye: A Review on the Use of Color in Visualization. In *Proc. of International Conference Information Visualisation (IV)*, 2007.

[Son07] Henry Sonnet: *Embedding Metadata in Computer Graphics for Interaction.* PhD thesis, Otto-von-Guericke University Magdeburg, 2007.

[SS02] Th. Strothotte and S. Schlechtweg: *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation.* Morgan Kaufman Publisher, Los Altos, 2002.

[Sta00] C. Stary: TADEUS: Seamless Development of Task-Based and User-Oriented Interfaces. *IEEE Transactions on Systems, Man and Cybernetics*, volume 30(5):509–525, 2000. ISSN 1083-4427.

[Sto03] M. C. Stone: *A Field Guide to Digital Color.* A.K. Peters, 2003.

[Sto07] M. C. Stone: Color in Information Display. In *Tutorial, IEEE Visualization Conference.* Sacramento, USA, Oct. 2007.

[STR+05] Vidya Setlur, Saeko Takagi, Ramesh Raskar, Michael Gleicher and Bruce Gooch: Automatic image retargeting. In *MUM '05: Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, (pp. 59–68). ACM Press, New York, NY, USA, 2005. ISBN 0-473-10658-2.

[Suo09] Xiaoyuan Suo: *A Task-Centered Visualization Design Environment and a Method for Measuring the Complexity of Visualization Designs.* PhD thesis, Georgia State University, 2009.

[SW04] F. Y. Shih and Y. Wu: Fast Euclidean Distance Transformation in Two Scans Using a 3x3 Neighborhood. *Computer Vision and Image Understanding*, volume 93(2):195–205, 2004.

[SW07] Dieter Schmalstieg and Daniel Wagner: Experiences with Handheld Augmented Reality. In *The 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007)*, (pp. pp. 3–15). IEEE and ACM, November 2007.

[SWF+07] Daniel Sinnig, Maik Wurdel, Peter Forbrig, Patrice Chalin and Ferhat Khendek: Practical Extensions for Task Models. In *Task Models and Diagrams for User Interface Design*, *Lecture Notes in Computer Science*, volume 4849/2007, (pp. 42–55). Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-77221-7. ISSN 0302-9743 (Print) 1611-3349 (Online).

[SWTS05] P. Schulze-Wollgast, Ch. Tominski and H. Schumann: Enhancing Visual Exploration by appropriate Color Coding. In *Proc. of International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG).* Plzen, Czech Republic, Febr. 2005.

[TAHS06] Christian Tominski, James Abello, Frank van Ham and Heidrun Schumann: Fisheye Tree Views and Lenses for Graph Visualization. In *Proceedings IV'06*, (pp. 17–24). IEEE Computer Society, Los Alamitos, CA, USA, 2006. ISSN 1550-6037.

[TC05] James J. Thomas and Kristin A. Cook (editors): *Illuminating the Path: The Research and Development Agenda for Visual Analytics.* IEEE Computer Society Press, August 2005. ISBN 0-7695-2323-4.

[TC06] James J. Thomas and Kristin A. Cook: A Visual Analytics Agenda. In *IEEE Computer Graphics and Applications*, volume 26, (pp. 10–13). IEEE Computer Society, Los Alamitos, CA, USA, January 2006. ISSN 0272-1716.

[Tel07] Alexandru C. Telea: *Data Visualization: Principles and Practice.* A K Peters, Ltd., 2007. ISBN 978-1-56881-306-6.

[TFS08a] Conrad Thiede, Georg Fuchs and Heidrun Schumann: Smart Lenses. In *Proc. 9th International Symposium on Smart Graphics (SG 2008)* (edited by Andreas Butz, Brian Fischer, Antonio Krüger, Patrick Oliver and Marc Christie), *Lecture Notes in Computer Science*, volume LNCS 5166, (pp. 178–189). Springer Verlag, Rennes, France, August 27th - 29th 2008. ISBN 978-3-540-85410-4. ISSN 0302-9743 (Print) 1611-3349 (Online).

[TFS08b] Christian Tominski, Georg Fuchs and Heidrun Schumann: Task-Driven Color Coding. In *Proc. 12th International Conference Information Visualisation (IV'08).* IEEE Computer Society, London, 8 - 11 July 2008.

[Thi10] Conrad Thiede: *Visuelle Informationsdarstellung in Smart Environments.* PhD Thesis, University of Rostock, Rostock, Germany, 2010.

[TIP05] Christian Tietjen, Tobias Isenberg and Bernhard Preim: Combining Silhouettes, Surface, and Volume Rendering for Surgery Education and Planning. In *EUROVIS 2005: Eurographics / IEEE VGTC Symposium on Visualization*, (pp. 303–310), 2005.

[TM01] D. Taubman and M. Marcellin: *JPEG2000: Image compression fundamentals, standards and practice*. Kluwer Academic Publishers, Boston, November 2001. ISBN 978-0792375197.

[TMM00] J.J. Tirtowidjojo, K. Marriott and B. Meyer: Extending SVG with Constraints. In *Proceedings of the 6th Australian World Wide Web Conference*, 2000.

[Tom06] Christian Tominski: *Event-Based Visualization for User-Centered Visual Analysis*. PhD thesis, University of Rostock, Germany, 2006.

[Træ02] Hallvard Trætteberg: *Model-based User Interface Design*. PhD thesis, Institutt for datateknikk og informasjonsvitenskap - Norges teknisk-naturvitenskapelige universitet, 2002.

[Tre99] Lloyd A. Treinish: Task-Specific Visualization Design. *IEEE Computer Graphics and Applications*, volume 19(5):72–77, September/October 1999. ISSN 0272-1716.

[TTS09] Conrad Thiede, Christian Tominski and Heidrun Schumann: Service-Oriented Information Visualization for Smart Environments. In *Proc. International Conference Information Visualisation (IV'09)*. Barcelona, Spain, 2009.

[VÓ3] Pere Pau Vázquez: *On the Selection of Good Views and its Application to Computer Graphics*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, April 2 2003.

[VCWP96] John Viega, Matthew J. Conway, George Williams and Randy Pausch: 3D magic lenses. In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, (pp. 51–58). ACM, New York, NY, USA, 1996. ISBN 0-89791-798-7.

[VFSG06] Ivan Viola, Miquel Feixas, Mateu Sbert and Eduard Gröller: Importance-Driven Focus of Attention. *IEEE Transactions on Visualization and Computer Graphics*, volume 12(5):933–940, October 2006.

[Vio05] Ivan Viola: *Importance-Driven Expressive Visualization*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, June 2005.

[VJ01] Paul Viola and Michael Jones: Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[WFRS07] Maik Wurdel, Peter Forbrig, T. Radhakrishnan and Daniel Sinnig: Patterns for Task- and Dialog-Modeling. In *Human-Computer Interaction. Interaction Design and Usability, Lecture Notes in Computer Science*, volume 4550/2007, (pp. 1226–1235). Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-73104-7. ISSN 0302-9743 (Print) 1611-3349 (Online).

[Whi02] A. W. White: *The Elements of Graphic Design: Space, Unity, Page Architecture, and Type*. Allworth Press, 2002.

[WKM+09] I. Woo, S.Y. Kim, R. Maciejewski, D.S. Ebert, T.D. Ropp and K. Thomas: SDViz: A Context-Preserving Interactive Visualization System for Technical Diagrams. In *Computer Graphics Forum*, volume 28, (pp. 943–950). The Eurographics Association, Blackwell Publishing, 2009.

[WL90] S. Wehrend and C. Lewis: A problem-oriented classification of visualization techniques. In *Proceedings of IEEE Visualization*, (pp. 139–143), 1990.

[Woy06] Sebastian Woyda: *Darstellung technischer Strukturen und Diagramme auf PDA*. Diplomarbeit (Master thesis), University of Rostock, 2006.

[WPF04] Marco A. Winckler, Philippe Palanque and Carla M.D.S. Freitas: Tasks and scenario-based evaluation of information visualization techniques. In *TAMODIA '04: Proceedings of the 3rd annual conference on Task models and diagrams*, (pp. 165–172). ACM Press, New York, NY, USA, 2004. ISBN 1-59593-000-0.

[WSF08] Maik Wurdel, Daniel Sinnig and Peter Forbrig: Task Model Refinement with Meta Operators. In *Interactive Systems. Design, Specification, and Verification*, *Lecture Notes in Computer Science*, volume 5136/2008, (pp. 300–305). Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-70568-0. ISSN 0302-9743 (Print) 1611-3349 (Online).

[Wur09] Maik Wurdel: Towards an Holistic Understanding of Tasks, Objects and Location in Collaborative Environments. In *Human Centered Design*, *Lecture Notes in Computer Science*, volume 5619/2009, (pp. 357–366). Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-02805-2. ISSN 0302-9743 (Print) 1611-3349 (Online).

[WVE98] M. van Welie, G van der Veer and A. Eliëns: An Ontology for Task World Models. In *DSV-IS'98*. Springer, Abingdon, 1998.

[X3D04] X3D: ISO/IEC 19775:2004 – Extensible 3D, 2004.

[ZCF02] Michelle X. Zhou, Min Chen and Ying Feng: Building a Visual Database for Example-based Graphics Generation. In *IEEE Symposium on Information Visualization (InfoVis 2002)*, (pp. 23–30), 2002.

[ZF98] Michelle X. Zhou and Steven K. Feiner: Visual task characterization for automated visual discourse synthesis. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, (pp. 392–399). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998. ISBN 0-201-30987-4.

[ZHHM07] Yongwang Zhao, Chunyang Hu, Yonggang Huang and Dianfu Ma: Collaborative Visualization of Large Scale Datasets Using Web Services. In *Proc. of Intl. Conf. on Internet and Web Applications and Services*, 2007.

List of Abbreviations

2D	two-dimensional
3D	three-dimensional
AR	Augmented Reality
CAD	Computer-Aided Design
CoU	Context of Use
CSS	Cascading Stylesheets
CTT	Concur Task Tree
CSVG	Constraint Scalable Vector Graphics
CKM	Content and Knowledge Management
DoI	Degree of Interest
DOM	Document Object Model
DSL	domain-specific language
DSRM	Data State Reference Model
GIF	Graphics Interchange Format
GML	Geographic Markup Language
GUI	Graphical User Interface
HCI	Human–Computer Interaction
HVAC	Heating, Ventilation, Air Conditioning
I/O	input/output
JPEG	Joint Photographic Expert Group
LAN	Local Area Network
WLAN	Wireless Local Area Network
LFS	Landesforschungsschwerpunkt
M6C	Multimedia Content Management in Mobile environments with Multi-Modal user interfaces
LoD	Level of Detail
MaTE	Mask and Task Editor
MBUID	Model-based User Interface Development
MDD	Model-driven Design
MUI	Multiple User Interface
MuSAMA	Multimodal Smart Appliance ensembles for Mobile Applications
NPR	Non-photorealistic Rendering
OWL	Web Ontology Language
PDA	Personal Digital Assistant
PDF	Portable Document Format
PIM	platform-independent model
PNG	Portable Network Graphics
PSM	platform-specific model

RCP	Rich Client Platform
RDF	Resource Description Framework
RoI	Region of Interest
SOA	service-oriented architecture
SVG	Scalable Vector Graphics
UA	user agent
UI	User Interface
VI	Visual Interface
UML	Unified Modeling Language
VR	Virtual Reality
VRML	Virtual Reality Markup Language
W3C	World Wide Web Consortium
WIMP	Windows, Icons, Menus, Pointing Device
WWW	World Wide Web
XLink	XML Linking Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language for Transformations

Thesis Statements

1. Visual representation has always been an important communication medium to convey complex facts. In order to be effective, a visual representation must be adapted to its respective context of use. *Smart Graphics* aim to move beyond the current requirement that designers anticipate every data, task and technological scenario, and instead facilitate the dynamic generation and presentation of content. A key goal of Smart Graphics is the development of *Smart Visual Interfaces*.

2. A Smart Visual Interface is a context-sensitive user interface that strives to present to the user those information and with the level of detail required for her current situation as effectively as possible. It must therefore be determined WHAT content (which information) must be presented WHY (which task) to WHOM (user), WHEN (in the context of composite workflows) and WHERE (which output device). Based on these questions a visual representation is generated, and as the context changes, the content representation *adapts* accordingly. These '5Ws' thus comprise the *Context of Use* (CoU) of visual representations within smart visual interfaces.

3. Adaptation of graphical content is linked to its *scalability*. Scalability corresponds to the ability of content representations to accommodate for changes to the CoU. Graphical content can be distinguished into four principal *visual data types* according to the degree of inherent scalability: raster graphics, 2D vector graphics, 3D graphics, and information visualizations.

4. The objective of *task-based adaptation* is to adapt the *graphical content* (WHAT) associated with the task at hand (WHY + WHEN) in such a way that the content's visual representation (HOW) provides a *good initial view* with respect to information relevant to the current working step. In doing so, the adaptation process is subject to output device constraints (WHERE). Initial views can always be interactively manipulated allowing the user to fine-tune the visual representation according to her expertise, preferences, and situational requirements.

5. The two primary challenges in providing task-specific visual representations are to provide a suitable *task description*, and to provide means for an *adequate adaptation control* on the basis of this description.

6. As a starting point for the task description, a general *task model* is used. This model is the result of prior domain task analysis and represents a *hierarchical decomposition of the task at hand* into composite task with subtasks. It further captures temporal and causal relations between subtasks.

7. The developed PAGE/FEATURE *concept* allows to enrich this general task model with a description of relevant task context information with respect to scalable visual representations. PAGES comprise a *general, data type-independent description of graphical content*. FEATURES represent a *conceptual organization of graphical content into distinct content elements* for which the *relevance with respect to the task at hand* is specified. PAGES and FEATURES thus abstract from peculiarities of the underlying visual data type. Based on these information, adaptation of task-specific visual representations is effected in the smart visual interface.

8. The concept of an *adaptation pipeline* provides the bridge between the conceptual task context specification in the enriched task model and concrete "Smart-X" display techniques comprising the functional building blocks of smart visual interfaces. The adaptation pipeline comprises five stages – view selection, geometry, visual attributes, view space and annotations – that allow to effect different aspects of visual representations by corresponding *adaption operations*. To this end, task-specific FEATURE relevance is mapped onto the parameter domains of the respective display technique. The adaptation pipeline therefore serves as a *general framework for the integration of display techniques* for the different visual data types.

9. The feasibility of the PAGE/FEATURE and adaptation pipeline concepts has been shown for all four of the contemplated visual data types. This included a discussion of viable adaptation operations on every stage of the adaptation pipeline as well as examples on how to integrate existing display techniques to utilize them as "Smart-X" components in smart visual interfaces. Thereby certain aspects of graphical content preparation and task context specification are manual tasks by necessity. Suitable authoring tools for each visual data type have been introduced that support content authors during smart visual interface design.

10. Six novel "Smart-X" display techniques have been developed that address aspects of task-specific visual representation generation that where found to be open research questions: *belt-based raster image distortion*, image-based *space-efficient remote labeling*, *smart exploded view diagrams* from vector graphics, an approach to *smart circuit schematic representation*, *Smart Lenses* for FEATURE-based local adaptation, and a systematic approach to *task-driven color coding*.

11. The research conducted in the scope of this thesis has been flanked by the Landesforschungsschwerpunkt (LFS) research project and the MuSAMA research training school. The developed concepts and techniques have been integrated into a joint *e-manual application demonstrator* for the LFS. Here, a *smart visual interface component* has been implemented that shows the PAGE/FEATURE approach based on enriched task models is suitable for *integration into an overall model-driven software/interface design process*. Moreover several publications co-authored with participants of the research training school substantiate the developed concepts' *applicability to Smart Meeting Room scenarios*.

Erklärung

Ich, Georg Fuchs, erkläre, dass ich die vorgelegte Dissertationsschrift mit dem Thema: *Task-based Adaptation of Graphical Content in Smart Visual Interfaces* selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, ohne die (unzulässige) Hilfe Dritter verfasst und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Bad Oldesloe, 11. April 2011