

Eine Link-Datenbank zur Integration von Virtual Engineering-Daten*

Ingolf Geist und Stephan Vornholt
 Otto-von-Guericke-Universität Magdeburg
 Universitätsplatz 2, 39106 Magdeburg
 {ingolf.geist|stephan.vornholt}@ovgu.de

Zusammenfassung

Virtual Engineering beschreibt den kollaborativen Prozess der Produktentwicklung in den Phasen der Konstruktion und Simulation/Analyse mit Hilfe von computergestützten Methoden in unterschiedlichen Entwicklungsabteilungen. Dabei entsteht in verschiedenen Entwicklungsbereichen eine Vielzahl von heterogenen, komplex-strukturierten Dokumenten, Produktdaten und Wissen, welche voneinander abhängen. Da Produktmodelle zur Integration von Konstruktion und Verhaltensanalyse die Sichten der Entwickler in den verschiedenen Bereichen einschränken, soll in dieser Arbeit eine Link-Datenbank vorgestellt werden, die durch externe Links und Annotationen Abhängigkeiten verwaltet, ohne die lokalen Datenmodelle zu verändern. Die Integration erfolgt, ähnlich zu Social-Tagging-Systemen, durch die interaktive Annotation der Daten in ihrer jeweiligen Entwicklungsumgebung und der Erstellung von Abhängigkeiten zwischen den Bereichen.

1 Einleitung

Im globalen Wettbewerb ist es notwendig, die Entwicklungszyklen von Produkten zu verkürzen, um schneller auf den Markt reagieren und neue Produkte kostengünstig anbieten zu können. Die konsequente und durchgehende Nutzung von computergestützten Methoden und virtuellen Prototyping-Techniken [9] in den Phasen Konstruktion und Simulation/Analyse des Produktlebenszyklus [6] ist eine Möglichkeit diese Anforderung zu erfüllen. Die Entwicklungsprozesse, *Virtual Engineering* (VE) genannt, nutzen CAD-Systeme zur Konstruktion sowie verschiedenste Simulationswerkzeuge (CAE-Systeme) zur frühen Analyse des Verhaltens eines Produktes. Eine effiziente Unterstützung erfordert eine Integration von CAD- und CAE-Daten.

Abb. 1 skizziert drei Datenquellen und verschiedene Aktionen in einem VE-Prozess zur Entwicklung eines Industrieroboters. Ausgehend von dem Ergebnis des konzeptuellen Entwurfs (nicht gezeigt) wird in einem CAD-System der Roboter konstruiert. Um zu überprüfen, ob das initiale Design die gegebenen Anforderungen erfüllt, wird ein Simulationsmodell mit Hilfe der Informationen aus dem CAD-Modell erstellt und parametrisiert. Dieser Schritt erfolgt manuell oder automatisch durch angepasste Werkzeuge (z.B. [3]). Mit Hilfe zusätzlicher Produktdaten über Elektromotoren und Materialien wird das Simulationsmodell vervollständigt. Anschließend werden verschiedene Alternativen getestet. Die Ergebnisse der Tests müssen an die Konstrukteure zurückgegeben werden, um eventuelle Änderungen ausführen zu können. Diese Schritte werden zyklisch durchgeführt bis das Produktdesign den Anforderungen entspricht. Aus der Analyse verschiedener VE-Prozesse ergeben sich folgende Anforderungen für die Datenverwaltung:

Integration: In den verschiedenen Entwicklungsbereichen, *Domänen* genannt, liegen Daten über dieselben Artefakte, d.h. Real-World-Objekte, aus unterschiedlichen Blickwinkeln vor. Die

*Unterstützt durch die Europäische Kommission: EFRE COMO C1-3201201 und C3-3201201.

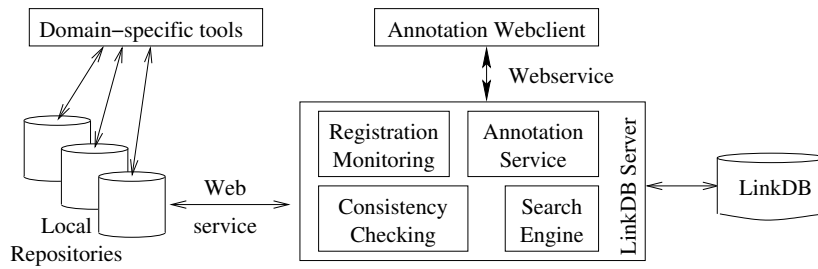


Abbildung 2: Architektur

die Suche nach Ressourcen und Experten und implementiert eine Abhängigkeitskontrolle zwischen Domänen. Diese Dienste sind über Webservice-Schnittstellen verfügbar und somit in anwendungsspezifische Werkzeuge integrierbar. Ein webbasierter Annotations-Client bietet einen domänenunabhängigen Zugriff auf die LinkDB-Dienste.

3 Link-Datenbank

Die Struktur der LinkDB folgt grundsätzlich der Struktur einer Folksonomy, d.h. es existieren Ressourcen, Nutzer, Annotationen und die Beziehungen zwischen diesen Mengen [8].

Werte in Annotationen können verschiedenen Datentypen angehören. Die *Datentypen* sind in der Menge $\mathcal{T} = \text{String} \cup \text{Integer} \cup \text{Date} \cup \dots$ zusammengefasst. Darüber hinaus gibt es zwei spezielle Mengen: $\text{URI} \subset \text{String}$ umfasst alle Uniform Resource Identifier (URI) und $\text{Label} \subset \text{String}$ beinhaltet alle gültigen Bezeichner.

Eine VE-Ressource repräsentiert ein Objekt in einem lokalen Daten-Repository, das durch eine URI eindeutig identifiziert ist. Die Koordinaten in einer 2D- oder 3D-Geometrie können in der URI kodiert werden, d.h. Ressourcen können auch mittels der Geometrie spezifiziert werden. Desweiteren wird der Ressource in der LinkDB ein Bezeichner zugeordnet. Somit wird die Menge der *VE-Ressourcen* als $\mathcal{R} = \text{URI} \times \text{Label}$ definiert. In der Beispiel-Datenbank in Abb. 3 werden Ressourcen aus drei Quellen (CAD-Modell, Mehrkörpersimulation (MBS), E-Motor-DB) als Boxen dargestellt. Auf die URIs wurde in der Darstellung verzichtet, aber jede Ressource stellt einen Link zu einem lokalen Objekt dar.

Ein Nutzer, z.B. $u2 = (\text{user2}, \text{Hans Müller})$ in Abb. 3, wird durch einen eindeutigen Nutzernamen sowie wie den realen Namen dargestellt. Reale Namen ermöglichen die direkte Kommunikation und werden deshalb in Enterprise-Folksonomies eingesetzt [4]. Im vorgestellten System ist die Menge der *VE-Nutzer* als $\mathcal{U} = \text{Label} \times \text{String}$ spezifiziert. Weitere Nutzerdaten wie Kontaktdaten können hinzugefügt werden.

VE-Ressourcen stehen in verschiedenen Beziehungen zueinander, die entweder domänenspezifisch sind (z.B. `partOf`-Beziehung in einem CAD-System), in allen Domänen vorkommen (z.B. `versionOf`-Beziehung) oder die Abhängigkeiten zwischen Domänen widerspiegeln. Weiterhin existieren Beziehungen zwischen Ressourcen und Nutzern. Die Menge der *VE-Beziehungen* ist als gerichtete Kantenmenge $\mathcal{E} = \mathcal{V} \times \mathcal{V} \times \text{Label}$ definiert, wobei $\mathcal{V} = \mathcal{U} \cup \mathcal{R}$ die Menge der Knoten ist. Jede Kante hat einen Bezeichner, wodurch mehrere Kanten pro Knotenpaar möglich sind. In Abb. 3 sind diese Beziehungen als durchgehende Linien dargestellt, auf die Bezeichner wurde in der Darstellung verzichtet.

Ressourcen und Beziehungen haben weder einen Typ noch eine Beschreibung. Diese Informationen sollen durch Annotationen geliefert werden. Eine Annotation ist eine bezeichnete Menge von Label-Wert-Paaren. Annotationen haben eine komplexe Struktur, sind aber wie in einem Social-Tagging-System frei durch den Nutzer definierbar. Die Menge aller Label-Wert-Paare ist als $\text{LV} = \text{Label} \times \mathcal{T}$ definiert und $\mathcal{P}(\text{LV})$ bezeichnet die entsprechende Potenzmenge. Somit stellt die Menge $\mathcal{A} = \text{Label} \times \mathcal{P}(\text{LV})$ die Menge der *Annotationen* dar. In Abb. 3 ist z.B. die Annotation (*manuell*, $\{(\text{description}, \text{manuelle Übernahme})\}$) dargestellt, welche eine Kante als

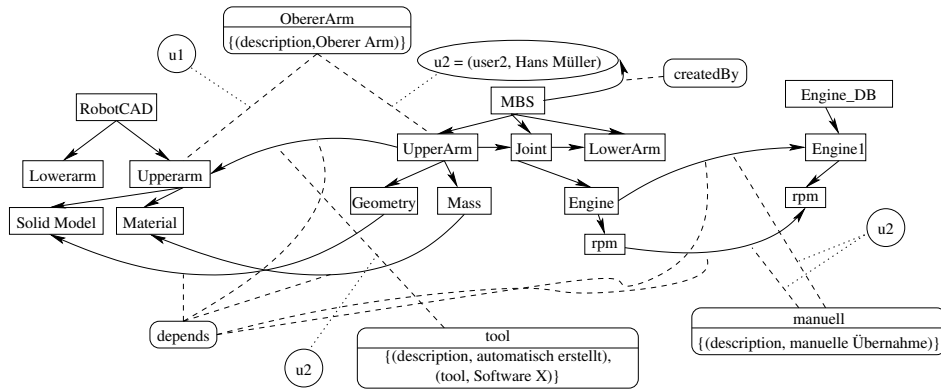


Abbildung 3: Beispiel-Datenbank

manuelle Übernahme von Parameterwerten beschreibt.

Der Zusammenhang zwischen Ressourcen \mathcal{R} und Kanten \mathcal{E} , Nutzern \mathcal{U} sowie Annotationen \mathcal{A} wird durch Beziehung *Annotierung* ausgedrückt. Die Beziehung ist als Relation $\mathcal{AN} = (\mathcal{R} \cup \mathcal{E}) \times \mathcal{U} \times \mathcal{A} \times \text{Date}$ definiert. In Abb. 3 werden Annotierungen durch gestrichelte und gepunktete Linien dargestellt. Das Element $(UpperArm, u2, Oberer Arm, date) \in \mathcal{AN}$ bedeutet (siehe Abb. 3), dass der Nutzer *user2* der Ressource *UpperArm* im Mehrkörpermodell (MBS) die Annotation *Oberer Arm* zum Zeitpunkt *date* zugeordnet hat. Wie in der Abbildung zu sehen, kann eine Annotation verschiedenen Ressourcen zugeordnet werden. Der Zeitpunkt der Annotierung ist nötig, um den Status der Produktentwicklung mit der Annotation in Beziehung setzen zu können, welches gerade bei iterativen Produktentwicklungsprozessen wichtig ist [7]. Nachdem alle Komponenten beschrieben sind, ist die Link-Datenbank definiert als

$$LinkDB = (\mathcal{R}, \mathcal{U}, \mathcal{E}, \mathcal{A}, \mathcal{AN}).$$

In Abb. 3 sind Beziehungen von Ressourcen zwischen verschiedenen Domänen mit Hilfe der Annotation *depends* bezeichnet. Diese Annotation wird für alle interdomänen-Beziehungen verwendet und ist eine vordefinierte *Systemannotation*. Weitere Systemannotationen sind u.a. *versionOf* zur Darstellung einer Versionsabhängigkeit oder *createdBy* zur Darstellung, dass eine Ressource durch einen Nutzer erzeugt wurde.

4 Integrationsprozess und weitere Applikationen

Aufbauend auf der Architekturbeschreibung und der Datenbankstruktur wird in diesem Abschnitt der interaktive Prozess der Integration beschrieben, welcher sukzessiv die Datenbank aufbaut. Der Prozess besteht folgenden Schritten:

1. Registrierung von Datenquellen: Nutzer registrieren sich und die lokalen Repositories beim LinkDB-Server. Damit stehen die Daten für weitere Dienste zur Verfügung. Die lokalen Repositories stellen sicher, dass die Daten in einem XML-Format beschrieben werden, bspw. ModelicaXML für die Simulationssprache Modelica, und über eine URI zugreifbar sind.

2. Erstellung von Ressourcen: Mit Hilfe des Annotationsclients, der XML-Daten darstellen kann, exportiert der Nutzer seine lokalen Daten und deren Beziehungen als VE-Ressourcen und VE-Beziehungen. Dadurch werden diese Daten für die Annotierung und die Nutzung durch andere User zugänglich.

3. Erstellung von Annotationen: Durch die Zuweisung von Annotationen an eigene und fremde Ressourcen beschreiben Nutzer die VE-Daten in einem Bottom-Up-Prozess. Somit wird sukzessive eine Beschreibung der Informationen aufgebaut.

4. Erstellung von Abhängigkeiten: Interdomänen-Beziehungen werden im Annotationsclient erstellt. Die Beziehungen erhalten jeweils die Annotation *depends*. In Zukunft sollen spezielle Transformationswerkzeuge(z.B.[3]) integriert werden, um die Aufgabe zu erleichtern.

5. Beschreibungen von Abhängigkeiten: Interdomänen-Beziehungen werden ebenfalls mittels Annotierungen durch den Nutzer beschrieben.

Durch die Datenbank können die im Abschnitt 1 genannten Aufgaben wie folgt erfüllt werden: **Integration bei Beibehaltung der Entwicklersichten:** Bei der Erstellung einer neuen Version eines Datenobjekts werden die Beziehungen der zugehörigen Ressource überprüft. Werden hierbei Abhängigkeiten durch Verfolgung von Kanten zu anderen Domänen erkannt, können die jeweiligen Nutzer durch das System über Änderungen benachrichtigt werden.

Dokumentation und Beschreibung: Annotationen beinhalten automatisch die Beschreibung der Daten. Durch die Annotierungen werden sie zu den Datenobjekten direkt zugeordnet. Somit wird es möglich, Beschreibungen zu speichern, im VE-Prozess allgemein zugänglich zu machen und erweiterte Suchmöglichkeiten zu implementieren.

Soziale Interaktion: Da Nutzer explizit modelliert sind, ist es möglich wie in Social-Tagging-Systemen [4] "Experten" für bestimmte Aufgaben zu finden, indem man deren annotierte Ressourcen untersucht.

5 Zusammenfassung

Ziel dieser Arbeit war die Entwicklung einer Struktur für eine Link-Datenbank, die Darstellung der Architektur und die zugehörigen Integrationsprozesse als Grundlage der interaktiven Integration und Beschreibung von heterogenen, komplex-strukturierten Daten in VE-Prozessen. Im Moment wird das System umgesetzt. Im Anschluss soll eine erste Akzeptanz bei Projektpartnern im Bereich des Maschinenbaus getestet werden. In einem weiteren Schritt sollen die Annotationen durch die Nutzung von Ontologien organisiert werden, wie bspw. in [2] für wissenschaftliche Daten vorgeschlagen. Durch die Informationen in den Ontologien sind weitere Konsistenzprüfungen möglich. Weiterhin wird daran gearbeitet, domänenspezifische Anwendungen mit dem LinkDB-Server zu koppeln, sowie automatische CAD-zu-CAE-Transformationsprozesse zur Erzeugung von Abhängigkeiten zu nutzen. Beides soll dazu beitragen, die Akzeptanz im Anwendungsbereich zu erhöhen.

Literatur

- [1] Sabeur Bettaieb and Frederic Noel. A generic architecture to synchronise design models issued from heterogeneous business tools: towards more interoperability between design expertises. *Engineering with Computers*, 2007. online first.
- [2] Michael Gertz and Kai-Uwe Sattler. Integrating Scientific Data through External, Concept-Based Annotations. In *VLDB 2002 Workshop EEXTT*, volume 2590 of *Lecture Notes in Computer Science*, pages 220–240. Springer, 2002.
- [3] Tamas Juhasz and Ulrich Schmucker. Automatic Model Conversion to Modelica for Dymola-based Mechatronic Simulation. In *Modelica 2008, March 3rd-4th, 2008*, pages 719 – 726. The Modelica Association, 2008.
- [4] David R. Millen, Jonathan Feinberg, and Bernard Kerr. Dogear: Social bookmarking in the enterprise. In *CHI '06: Proc. of the SIGCHI conference*, pages 111–120. ACM, 2006.
- [5] Gregory M. Mocko and Steven J. Fenves. A Survey of Design – Analysis Integration Issues. Technical Report NISTIR 6996, National Institute Of Standards and Technology, May 2003.
- [6] Gerhard Pahl, Wolfgang Beitz, Jörg Feldhusen, and K. H. Grote. *Konstruktionslehre : Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung*. Springer, Berlin, 2007.
- [7] Noel Titus, Eswaran Subrahmanian, and Karthik Ramani. Folksonomy and Designing: An Exploration. In *Proc. ASME IDETC/CIE 2007*. ASME, 2007.
- [8] T. Vander Wal. Folksonomy Coinage and Definition. <http://vanderwal.net/folksonomy.html>, February 2007.
- [9] F. Zorriassatine, C. Wykes, R. Parkin, and N. Gindy. A survey of virtual prototyping techniques for mechanical product development. *Journal of Engineering Manufacture*, 217:513 – 530, 2003.