September 2001

PDM-Basisdienste für die integrierte virtuelle Produktentstehung

Dimitri Giwerzew
B.I.M.-Consulting mbH, dimitrij.giwerzew@bim-consulting.de

Holger Wierschin
B.I.M.-Consulting mbH, holger.wierschin@bim-consulting.de

Georg Paul
Otto-von-Guericke-Universität Magdeburg, paul@iti.cs.uni-magdeburg.de

In: Buhl, Hans Ulrich, u.a. (Hg.) 2001. *Information Age Economy*; 5. Internationale Tagung Wirtschaftsinformatik 2001. Heidelberg: Physica-Verlag

ISBN: 3-7908-1427-X

PDM-Basisdienste für die integrierte virtuelle Produktentstehung

Dimitri Giwerzew, Holger Wierschin
B.I.M.-Consulting mbH

Georg Paul
Otto-von-Guericke-Universität Magdeburg

Zusammenfassung: In diesem Beitrag wird die integrierte virtuelle Produktentstehung im BMBF-Leitprojekt iViP[1] (Innovative Technologien und Systeme für die integrierte Virtuelle Produktentstehung) vorgestellt. Es wird dabei näher auf die PDM-Basisdienste als zentraler Bestandteil des Clusters "Infrastruktur Datenmanagement" eingegangen. Diese haben die Aufgabe, Dienste zur bedarfsgerechten Verwaltung von Produktdaten über den gesamten Prozeß der virtuellen Produktentstehung zur Verfügung zu stellen. Um dieser Aufgabe gerecht zu werden, bedarf es einer neuartigen Softwarearchitektur, die ausführlich vorgestellt wird. Nach der Vorstellung der Architektur folgt ein Anwendungsszenario für das Customizing der PDM-Basisdienste zur Definition und Erweiterung von Diensten durch den Administrator.

Schlüsselworte: integrierte virtuelle Produktentstehung, Basisdienste, Product Data Management, Repository, PDM-Enablers

1 Einleitung

Die Unterstützung von Produktentwicklungsprozessen durch IT-Entwicklungswerkzeuge im großen Stil wird ohne PDM-Systeme nicht möglich sein [GrGe97]. Große und mittelständische Unternehmen haben dies bereits erkannt und fördern seit einiger Zeit deren Einführung.

Die in den PDM-Systemen zu verwaltenden Daten sind Teil des integrierten Produktmodells [GrAn93]. Das integrierte Produktmodell ist ein Informationsmodell, das sämtliche Produktdaten umfaßt, die im Verlauf des Produktlebenszyklus ent-

[1] Das Leitprojekt iViP wird aus Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) unter Förderkennzeichen 02PL10xxx gefördert und von der Projektträgerschaft Produktion und Fertigungstechnologien betreut.

stehen. Unter Produktdaten wird die formale Darstellung von Fakten, Konzepten und Instruktionen über ein Produkt oder eine Menge von Produkten verstanden, die zur manuellen oder automatisierten Kommunikation, Interpretation und Verarbeitung geeignet ist [Satt98].

Heutige PDM-Systeme verwalten mit Erfolg alle Produktdaten in der technischen Produktionsvorbereitung. Der Produktlebenszyklus eines Erzeugnisses wird durch die Verwaltung aller im Vertrieb, in der Projektierung, der Konstruktion, der Arbeitsplanung und dem Auftragswesen erforderlichen Unterlagen wie Angebot, Zeichnung, Stückliste, Arbeitsplan und NC-Programm genau nachgebildet.

Trotz ihrer Mächtigkeit sind die meisten PDM-Systeme proprietäre Lösungen einzelner Anbieter. Die Architektur läßt solche Aspekte wie Modularität, Flexibilität, Skalierbarkeit, Einsatz standardisierter Schnittstellen und Plattformneutralität häufig vermissen. Besonders aber ist das Fehlen eines expliziten Repository zu nennen, das die Struktur und Semantik der System-Objekte definiert, verwaltet und zur Verfügung stellt [Saak97].

Eines der Ziele für die integrierte virtuelle Produktentstehung ist deshalb die Unabhängigkeit von den proprietären Lösungen der PDM-Systeme durch die Schaffung eines konfigurierbaren Repository [HaLe93] und der konsequenten Anwendung anerkannter Standards. Hierzu zählen beispielsweise CORBA [OMG00a] und die PDM-Enablers [OMG00b].

Die Einbindung anderer Applikationen, wie z.B. eines CAD-Systems, erfolgt meistens individuell für jedes PDM- und CAD-System. An dieser Stelle kann eine offene Integrationsplattform eine Hilfe schaffen.

In diesem Zusammenhang wird in den folgenden Abschnitten zunächst die integrierte virtuelle Produktentstehung im iViP-Projekt vorgestellt und dann speziell auf die PDM-Basisdienste eingegangen. Ersteres umfaßt eine kurze Vorstellung des Projektes sowie dessen Software-Architektur.

2 Integrierte virtuelle Produktentstehung im iViP-Projekt

Der Stand der Technik auf dem Gebiet der virtuellen Produktentstehung ist ausführlich in [SpKr97] beschrieben. Unter einem virtuellen Produkt wird dabei der zentrale Informationsträger einer vollständig rechnerbasierten Produktentwicklung verstanden. Mit einer virtuellen Produktentstehung wird eine Umgebung assoziiert, in der die vollständige Repräsentation eines Produktes in digitaler Form erstellt, getestet und visualisiert wird.

Das Ziel des iViP-Projektes besteht in der Schaffung einer IT-Systemwelt für die integrierte virtuelle Produktentstehung in einer heterogenen Umgebung. Es ist von Anfang an geplant, die Zusammenarbeit zwischen Softwarefirmen, Endproduktherstellern, Zulieferern und wissenschaftlichen Institutionen zu fördern [KrTa99].

Im Mittelpunkt der virtuellen Produktentstehung im iViP steht der 'Digitale Master', der als verbindlicher Informationsträger sämtliche für die Produktentstehung und für alle Folgephasen relevanten Daten eines Produktes in digitaler Form enthalten soll.

Das Projekt gliedert sich in sechs Cluster. Die ersten beiden Cluster befassen sich mit der Infrastruktur zum Prozeß- und Datenmanagement, die nächsten drei entwickeln Anwendungssysteme zur Unterstützung der virtuellen Produktentstehung und schließlich gestaltet das letzte Cluster Referenzmodelle und Maßnahmen zur Marktvorbereitung für eine Verbreitung der iViP-Ergebnisse. Es gibt ebenfalls projektübergreifende Arbeitsgruppen: Architektur und Integration, Software-Recht, Software-Qualität, Security, PDM-Enablers sowie Public Relations und Marketing.

Zu einer branchenübergreifenden Realisierung der Integration sind vier Anwendungsszenarien aus den Bereichen der Zulieferindustrie, des Werkzeugmachinenbaus, des Automobilbaus und der Entwicklungsdienstleister in Entwicklung.

Die iViP-Softwarearchitektur besteht aus diesen drei wesentlichen Teilen: iViP-Integrationsplattform, iViP-Applikationen und iViP-Client (vgl. Abbildung 1).

Die iViP-Integrationsplattform ist der Kern der Softwarearchitektur und stellt die Basisfunktionalitäten für die Integration unterschiedlicher Anwendungssysteme bereit. So gesehen ist sie ein Framework [RaSt92] auf Systemebene für die integrierte virtuelle Produktentstehung. Da die Schnittstellen der Integrationsplattform offen liegen, bestehen für kleine und mittelständische Softwareanbieter keine in der Systemarchitektur begründeten Marktschranken. Die Integrationsplattform verfügt über einen Komponenten-Manager, einen Session-Manager, einen User-Manager und einen Trader. Der Komponenten-Manager enthält alle Informationen über Funktionalitäten, Dienste und Komponenten, die während einer Session dem Benutzer zur Verfügung gestellt werden können. Der Session-Manager verwaltet die Benutzerdaten während einer Sitzung (z.B. An-/Abmelden der Benutzer, Loggig, Überprüfungen der Berechtigungen usw.). Der User-Manager dient zur einheitlichen Benutzerverwaltung. Über den Trader wird die Bereitstellung der Software on Demand ermöglicht, was z.B. durch das Angebot der iViP-Dienste aus dem Inter-/Intra-/Extranet durchgeführt werden kann.

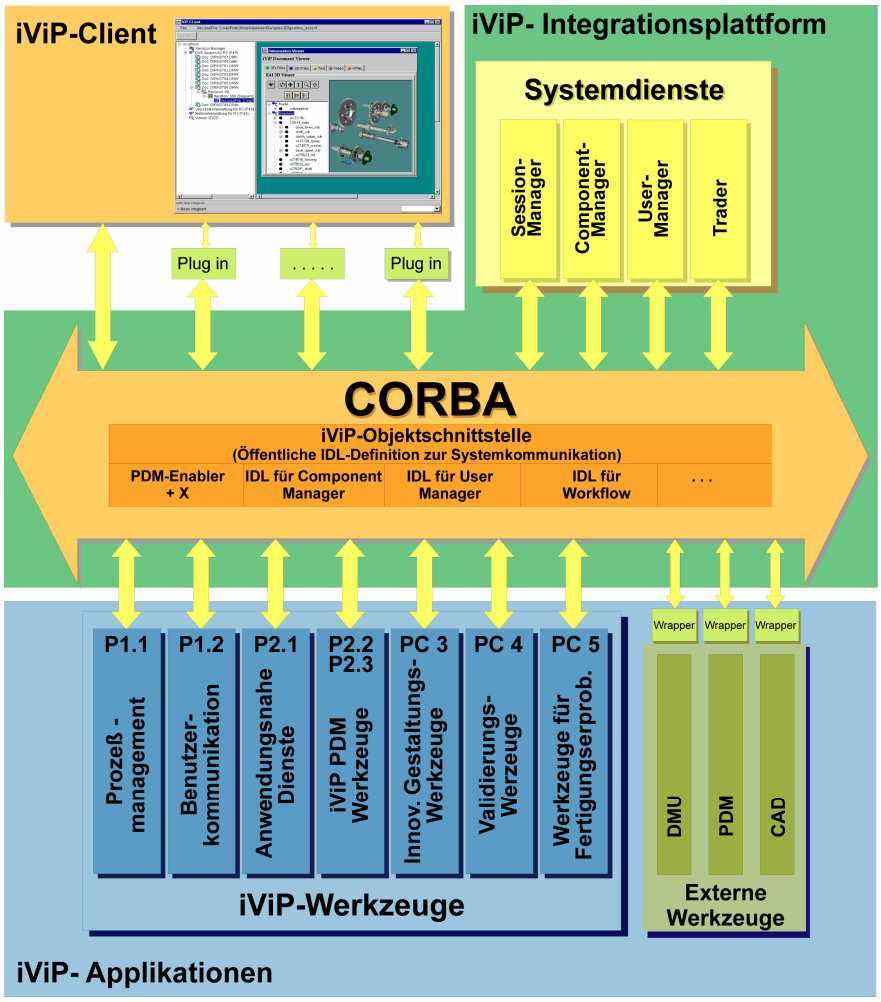

Abbildung 1: iViP-Softwarearchitektur

Die iViP-Applikationen umfassen sowohl iViP-Werkzeuge als auch externe Werkzeuge (z.B. CAD-Systeme), die mit Hilfe von Wrappern in die iViP-Plattform integriert werden können. Zu den iViP-Werkzeugen zählen u.a. neuartige Werkzeuge zur Kommunikation und Kooperation [LuMa00], zum Produktdatenmanagement (vgl. Abschnitt 3), zur Spezifikation in frühen Phasen, zum Komplexitätsmanagement oder zur Simulation.

Der iViP-Client stellt ein Framework auf der Oberflächenebene zur Darstellung der Benutzerschnittstellen der Komponenten dar. Mit Hilfe der Client-Funktionalitäten ist es möglich, einfache Parameterabfragen durchzuführen und die elementaren Ergebnisse direkt darzustellen. Weiterhin ist es möglich, die von den Liefe-

ranten der Komponenten implementierten Java Beans als InnerFrame zu präsentieren. Das Starten externer Anwendungen als OuterFrame ist ebenfalls möglich.

3 PDM-Basisdienste

Die PDM-Basisdienste gehören zu dem Cluster “Infrastruktur Datenmanagement” und übernehmen innerhalb des Projektes iViP die Aufgabe, Dienste zur bedarfsgerechten Verwaltung von Produktdaten über den gesamten Prozeß der virtuellen Produktentstehung zur Verfügung zu stellen. In der iViP-Architektur (vgl. Abbildung 1) sind sie den iViP-Applikationen, konkret den iViP-Werkzeugen zuzuordnen. Es werden Grundfunktionen eines PDM-Systems entwickelt, die dynamische, verteilt ablaufende Produktentstehungsprozesse unterstützen sollen.

Die PDM-Basisdienste gliedern sich in einen Kern und ein Definitions-Toolset, das zur Konfiguration des Repository dient (vlg. Abbildung 2).

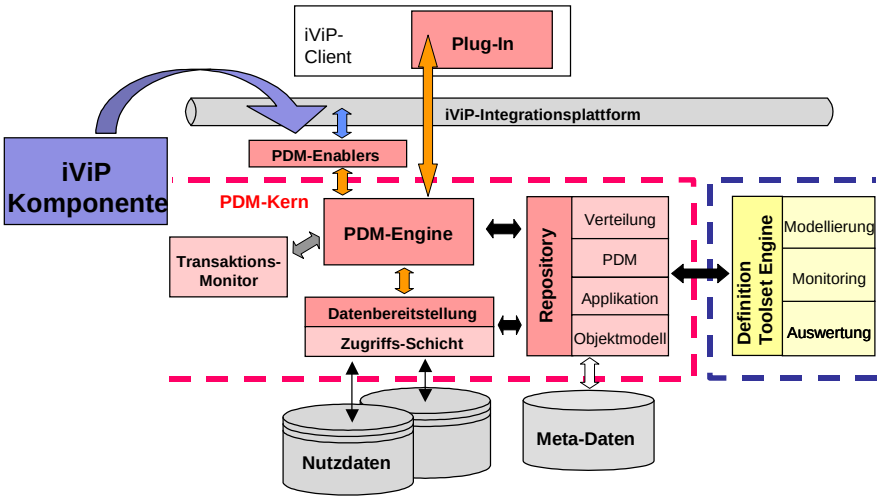

Abbildung 2: Architektur der PDM-Basisdienste

Der Kern besteht aus dem Repository, der Datenbereitstellung & Zugriffs-Schicht, der PDM-Engine und dem Transaktions-Monitor. Der Zugriff auf die PDM-Basisdienste durch die anderen iViP-Komponenten erfolgt mittels PDM-Enablers. Es wird ebenfalls ein Plug-In in den iViP-Client angeboten, wobei der Zugriff auf die PDM-Basisdienste dann über eine eigene generische CORBA-Schnittstelle erfolgt.

3.1 Repository

Das Bereitstellen einer flexiblen, an verschiedene Anforderungen anpaßbaren Architektur für PDM-Basisdienste, die von der Struktur der zu verwaltenden Objekte und Dienste sowie deren Abbildung auf Datenquellen abstrahiert, setzt das Vorhandensein einer dynamisch konfigurierbaren Metabeschreibung für diese Dienste, Objekte und deren Abhängigkeiten untereinander voraus. Anhand dieser Metabeschreibung können die einzelnen Schichten der Architektur zur Laufzeit ihre eigenen Fähigkeiten bestimmen und sich auf geänderte Randbedingungen anpassen. Die Bereitstellung einer solchen Metabeschreibung wird durch das Repository realisiert. Die Lösung dieses Aspektes ist ein Schwerpunkt der Forschungsbemühungen der Autoren.

Die Aufgabe des Repository besteht also in der Abbildung der Struktur und Semantik der anzubietenden Objekte und Dienste. Dies wird über die Metabeschreibung der in den verschiedenen Schichten der Architektur benötigten Objekte und Methoden erreicht. Das Bereitstellen der Metabeschreibung erfolgt dynamisch durch eine anwendungsfallorientierte Konfiguration über das Definitions-Toolset.

Das Repository selbst ist in seiner Struktur statisch und wird als Ganzes in einer Datenquelle abgelegt. Der Zugriff auf das Repository erfolgt transparent über CORBA. Es wird in vier Schichten gegliedert, die auch Subrepositories genannt werden. Jedes Subrepository ist jeweils extra als CORBA-Dienst implementiert. Im einzelnen sind es: das Objektmodellrepository, das Applikationsrepository, das PDM-Repository und das Verteilungsrepository. Diese vier Schichten werden im folgenden genauer beschrieben.

Das **Objektmodellrepository** stellt die unterste Schicht des gesamten Repository dar. Daran schließen sich die anderen Subrepositories an. Die Aufgabe besteht in der Bereitstellung einer Metabeschreibung für die Objektmodelle des PDM-Kerns. Zudem beschreibt das Objektmodellrepository das Mapping der Objektmodelle auf die tatsächlichen Daten- und Funktionsstrukturen der primären und sekundären Datenquellen. Somit besteht das Objektmodellrepository aus dem Metaobjektmodell (vgl. Abbildung 3) und aus dem Mappingmodell.

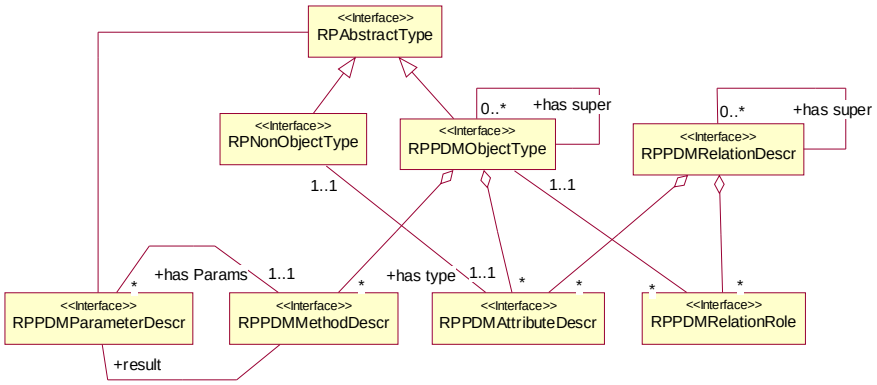

Abbildung 3: Metaobjektmodell

Zur Zeit wird ein Mapping für relationale Datenbanken (welche als Industriestandard angesehen werden können) unterstützt. Da es sich dabei u.a. um die Problematik des objekt-relationalen Mappings handelt, werden verschiedene Strategien zur Abbildung von Vererbungshierarchien auf Tabellen unterstützt.

Um den Zugriff auf die Nutzdaten über die Zugriffs-Schicht zu ermöglichen, wird in dem Mappingmodell zusätzlich zu den eigentlichen Mapping-Informationen wie Tabellenname zu einer Klasse, Spaltenname zu einem Attribut noch die Information über die tatsächliche Nutzdatenquelle in Form eines weltweit eindeutigen Schlüssels gespeichert.

Im **Applikationsrepository** ist die Zuordnung von PDM-Funktionen (Businessklassen) zu den Objekten aus dem Objektmodell vorzunehmen. Dies ist Grundlage für die Definition von PDM-spezifischen Applikationen und ermöglicht die Definition von spezifischen Sichten auf die Objektmodelle, die vom Objektmodellrepository bereitgestellt werden. Eine wesentliche Aufgabe des Applikationsrepository ist die Versionierung und Strukturierung der Objektmodelle. Das heißt, es stellt für das PDM-Repository eine gültigkeits- und versionsbezogene Sicht auf die Metaobjektmodelle bereit.

Im **PDM-Repository** werden Informationen über die Funktionsbausteine der PDM-Engine abgelegt. Im speziellen werden hier die Businessklassen beschrieben, welche die angebotenen Funktionalitäten der PDM-Engine repräsentieren. Hier werden Daten definiert, die ursprünglich Bestandteil der Nutzdaten von PDM-Systemen bilden, die aber übergreifend von Interesse sind. Mit der Verlagerung dieser Daten in das Repository stehen diese unabhängig von den Nutzdatenquellen zur Verfügung und können auch von anderen Teilprojekten benutzt werden. Diese Daten werden auch als Basisdaten bezeichnet. Eine weitere Aufgabe des PDM-Repository ist die Definition der Prüfabläufe, die den Klassen aus dem Objektmodellrepository zugeordnet werden können.

Die Aufgabe des **Verteilungsrepository** besteht darin, die Verteilung von Daten zwischen verschiedenen Standorten und den Abgleich von asynchron durchgeführten Änderungen zwischen diesen Standorten verwalten und speziell konfigurieren zu können. Solche Verteilung wird i.a. als Replikation [Dada96; HeHe96] bezeichnet. Für die Durchführung der Replikation stellen die eingesetzten Datenhaltungssysteme in der Regel die benötigten Mechanismen bereit. Die Voraussetzung für den Aufbau einer Verteilungsumgebung und deren Anpassung an Änderungen der darin betrachteten Objektmodelle setzt die Verfügbarkeit einer Metabeschreibung dieser Umgebungen voraus. Eine derartige Metabeschreibung stellt das Verteilungsrepository bereit und bildet damit die Basis für die Administration der Verteilungsumgebung.

3.2 Definitions-Toolset

Um den Aufwand für die Anpassungen möglichst gering zu halten, wird mit dem Definitions-Toolset (DTS) eine Sammlung von Tools zur komfortablen grafischen Modellierung und Auswertung des Repository der PDM-Basisdienste bereitgestellt. Neben den grafischen Editoren werden auch textuelle Editoren angeboten.

Das Definitions-Toolset besteht aus den folgenden Komponenten:

- Die DTS-Engine fungiert als Schnittstelle zu den einzelnen Subrepositories der PDM-Basisdienste und stellt u.a. die Modellkonsistenz sicher.
- Die Modellierungskomponente verfügt über verschiedene grafische und textuelle Editoren, mit denen die Inhalte der Subrepositories modifiziert werden können. Bei den grafischen Notationen der einzelnen Editoren werden soweit wie möglich anerkannte Standards (z.B. UML [OMG00c]) verwendet.
- Die Auswertungskomponente unterstützt die Analyse der im Repository enthaltenen Informationen.
- Die Monitoring-Komponente erlaubt eine Visualisierung bzw. Auswertung der Nutzdaten des PDM-Systems. Dies geschieht unter Zuhilfenahme der Informationen aus den einzelnen Subrepositories und der Defintions-Toolset-Engine.

Die zur Beschreibung des Repository notwendigen Elemente werden in Partialmodelle aufgeteilt, die weitgehend unabhängig voneinander mit spezifizierten Beziehungen ausgeprägt werden. Zu diesen Partialmodellen zählen der Objektmodell-Editor und der Prüfabläufe-Editor. Der Editor zur Modellierung des Verteilungsrepository ist momentan in Arbeit.

Der Objektmodell-Editor dient zur Verwaltung des Objektmodell- und des Applikationsrepository. Zur grafischen Darstellung wird die UML-Notation verwendet. Das DTS bietet die Möglichkeit, die UML-Klassenmodelle in das bzw. aus dem Objektmodell- und Applikationsrepository zu exportieren bzw. zu importieren. Es

ist ebenfalls möglich, die UML-Klassenmodelle in eine XMI-Datei [OMG00d] zu exportieren bzw. eine XMI-Datei zu importieren.

Der Prüfabläufe-Editor wird zur Verwaltung von Prüfabläufen des PDM-Repository verwendet. Diese Prüfabläufe können den Elementen des Objektmodellrepository zugeordnet und somit deren mögliche Lebenszyklen definiert werden. Die grafische Darstellung wurde an die Notation der *Statechart Diagrams* aus der UML angelehnt. Auch hier gibt es die Möglichkeit des Exports aus dem DTS in das PDM-Repository bzw. des Imports aus dem PDM-Repository in das DTS.

3.3 Datenbereitstellung & Zugriffs-Schicht

Zusammen mit der Zugriffs-Schicht implementiert die Datenbereitstellung das sogenannte „Objectwarehouse“, welches für die Bereitstellung der Produktdaten zuständig ist. Die Datenbereitstellung ist unabhängig von bestimmten Datenquellen und stellt Objekte heterogener Quellen den darüberliegenden Funktionen der PDM-Engine bereit. Zentrales Element der Datenbereitstellung ist das generische Objekt. Die Daten eines generischen Objekts werden in einer dynamischen Attributliste gespeichert. Dessen Identität wird in einem eigenständigen Objekt repräsentiert, das sowohl einen eindeutigen Kennzeichner, als auch den Klassennamen speichert.

Um gleichzeitig auf verschiedene, heterogene Datenquellen zugreifen zu können, besteht die Zugriffs-Schicht aus verschiedenen Adaptoren, welche den (lesenden und schreibenden) Zugriff auf jeweils eine spezifische Datenquelle implementieren. Bisher wurde der JDBC-Adapter realisiert, der Nutzdaten einer relationalen Datenbank in generische Objekte überführt. Die Mapping-Informationen sind im Objektmodellrepository gespeichert. Bis jetzt wurden folgende relationale Datenbanken getestet: ACCESS, SQL-Server, Oracle und HypersonicSQL.

3.4 PDM-Engine

Die PDM-Engine stellt das Herz der PDM-Basisdienste dar. Hier sind die Methoden hinterlegt, welche die Basisfunktionalität eines PDM-Systems darstellen. Zu diesen Basisfunktionalitäten gehören beispielsweise: Statusverwaltung, Sperren von Objekten gegen parallele Änderung, Versionsverwaltung, Stücklistenverwaltung, Dokumentenverwaltung, Projekteverwaltung, Historie, Suchdienst, Customizing, Überprüfungen der Zugriffsberechtigungen und Gültigkeitsverwaltung.

Dem Anwender werden diese Basisfunktionen mit Hilfe eines Plug-Ins in den iViP-Client bereitgestellt, anderen iViP-Komponenten ist der Zugriff über die PDM-Enablers Schnittstelle möglich. Eine Basisfunktion wird als Businessobjekt implementiert. Businessobjekte werden im Transaktions-Monitor (vgl. Abschnitt 3.7) verwaltet und greifen ihrerseits auf generische Objekte des Objectwarehouse

zu. Für die Ausführung einer Basisfunktion werden Konfigurationsdaten über das PDM-Repository bzw. über das Objektmodell- und das Applikationsrepository abgefragt.

3.5 Plug-In für den iViP Client

Das Plug-In für den iViP-Client ermöglicht den interaktiven Zugriff auf die PDM-Engine. Die Implementierung erfolgt als Java-Anwendung unter Nutzung der Swing-Bibliotheken zur Gestaltung der Oberflächen. Das Plug-In kann – sobald sich die PDM-Engine an der Integrations-Plattform angemeldet hat – vom iViP-Client aus gestartet werden. Dabei erscheint es als InnerFrame innerhalb des iViP-Clients (vgl. Abbildung 4)

Im linken Bereich des Plug-Ins befinden sich Strukturansichten für die verwalteten PDM-Objekte. Die aktuelle Strukturansicht zeigt die Dokumentenstruktur mit Beziehungen zu Dateien. Im rechten Bereich ist die Formularansicht eines konkret ausgewählten Objektes – hier einer Datei – zu sehen, in der beispielsweise Attributwerte geändert werden können und die Datei ein- und ausgecheckt werden kann.

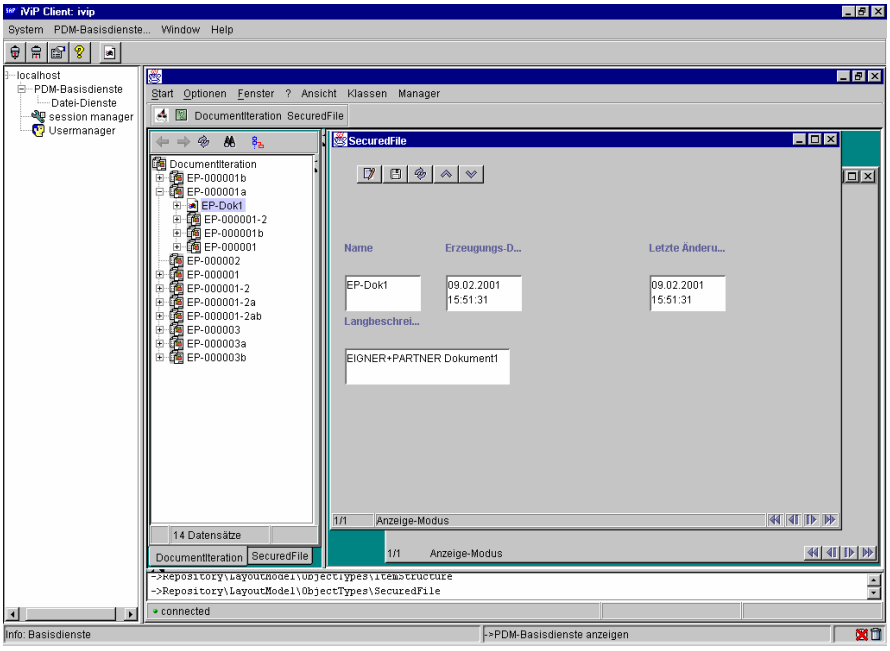

Abbildung 4: Plug-In der PDM-Basisdienste für den iViP-Client

3.6 PDM-Enablers Schnittstelle

Während das Plug-In für den iViP-Client den interaktiven Zugriff auf die PDM-Engine ermöglicht, ist die Implementierung der PDM-Enablers Schnittstelle dafür zuständig, anderen Applikationen (speziell anderen iViP-Komponenten) den Zugriff auf die PDM-Engine zu ermöglichen. Im Unterschied zum STEP-Standard [ISO94; Owen97], der für den Offline-Zugriff gedacht ist, dient die Spezifikation der PDM-Enablers zum Online-Zugriff auf die Produktdaten.

Durch die Spezifikation der Object Management Group (OMG) ist der Aufbau und die Funktionsweise der PDM-Enablers grundlegend definiert. Eine einheitliche Auslegung dieser Spezifikation zwecks Implementierung ist jedoch notwendig, da in der Spezifikation kein explizites Datenmodell definiert ist. Eine solche Auslegung erfolgt im Rahmen der Arbeitsgruppe “PDM-Enablers“ innerhalb des iViP-Projekts, wobei hier die Entwicklung des Standards für CAD-Enablers [OMG00e] ebenfalls verfolgt wird. Hierdurch wird sichergestellt, daß andere Dienste tatsächlich transparent auf produktbeschreibende Daten zugreifen können, unabhängig davon, welcher (iViP-konforme) Dienst diese Schnittstelle bereitstellt.

Das größte Interesse innerhalb von iViP bezieht sich auf das Dokumenten- und Produktstrukturmanagement. Daher sind zum überwiegenden Teil die notwendigen Module bereits implementiert, um anderen Projektpartnern die Implementierung von entsprechenden Clients zu ermöglichen.

3.7 Transaktions-Monitor

Der Transaktions-Monitor hat im wesentlichen die Aufgabe, die Ausführung der Funktionen in der PDM-Engine zu steuern und zu protokollieren. Die zentrale Komponente ist dabei der Basisdienstemanager (BADIMA). Zusammen mit den anderen Komponenten des Transaktions-Monitors wird eine einheitliche Verwaltung der Businessobjekte realisiert, wodurch die Wartbarkeit beim Hinzufügen oder Entfernen neuer Funktionen gewährleistet wird.

Ohne an dieser Stelle auf die Details einzugehen, wird hier kurz die Arbeitsweise des Transaktions-Monitors beschrieben. Es wird eine Initialisierungs- und eine Ausführungsphase unterschieden. Während der Initialisierungsphase werden alle im Repository definierten Businessobjekte erzeugt und beim BADIMA registriert. Zur Ausführung einer PDM-Funktion wird das geeignete Businessobjekt gesucht und sodann wird an dieses Objekt die Ausführung der Funktion delegiert. Dies wird in einem Vorlauf validiert. Dazu wird an alle registrierten Businessobjekte ein entsprechender Vertrag geschickt, den diese ablehnen oder annehmen. Das Ergebnis einer Vertragsbildung fließt in die Ablaufsteuerung einer Methode mit ein und wird entsprechend protokolliert.

4 Anwendungsszenario für die PDM-Basisdienste

Gegenstand dieses Abschnittes ist ein Szenario für das Customizing der PDM-Basisdienste zur Definition und Erweiterung derselben durch den Administrator.

Die Ausgangssituation sei wie folgt definiert. Es soll weitgehend ohne Programmieraufwand eine Artikelverwaltung bereitgestellt werden, die über das Plug-In der PDM-Basisdienste im iViP-Client angesprochen werden kann. Die Vorgehensweise hierzu wird im folgenden vorgestellt (vgl. Abbildung 5).

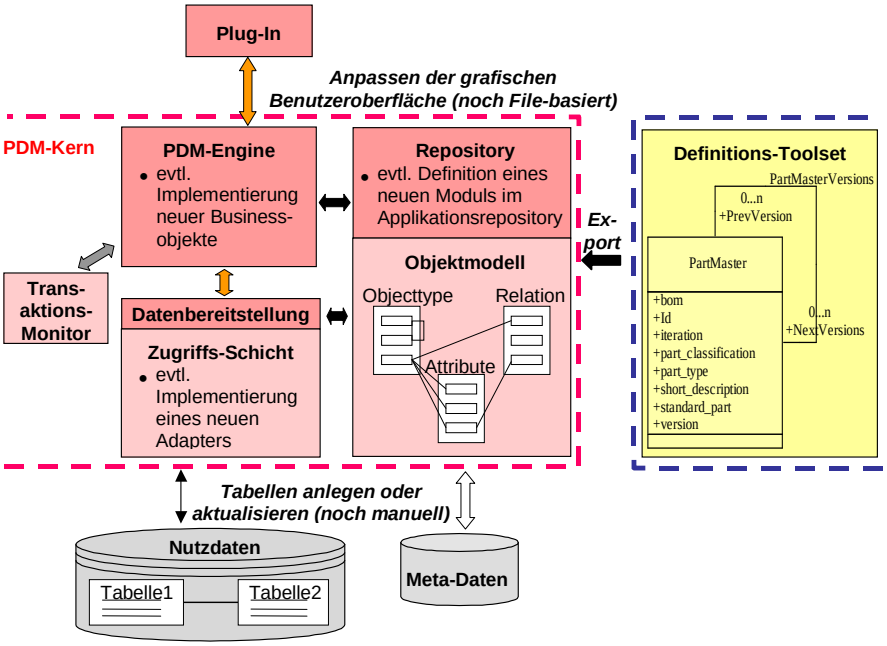

Abbildung 5: Beispiel für das Customizing der PDM-Basisdienste

- Die dem Artikel entsprechende Klasse “PartMaster” wird mit Hilfe des Definitions-Toolsets definiert. Hier besteht die Möglichkeit, diese Klasse einem Modul zuzuordnen, was sich im Applikationsrepository in der Strukturierung der Metamodelle widerspiegelt.
- Um die Änderungen im Definitions-Toolset im Repository zu speichern, wird ein Export durchgeführt.
- Der nächste Schritt besteht in der Anpassung der Datenbank-Schemata für Nutzdaten. Derzeit wird dies noch manuell durchgeführt.
- Die Erweiterung der Zugriffs-Schicht um einen neuen Adapter ist nur in seltenen Fällen notwendig (z.B. Zugriff auf Objekt-Datenbanken)

- Nach der Änderung auf der Datenebene ist es notwendig, die grafische Benutzeroberfläche für den Artikel (Artikel-Formular) zu definieren. Derzeit erfolgt das noch auf Basis einer Konfigurations-Datei.
- Nun muß das so entstandene Formular mit den Funktionalitäten versehen werden. Dazu werden entweder die bereits existierenden Businessobjekte verwendet oder neue implementiert.

An dieser Stelle sei bemerkt, daß solche Anpassungen wie Erweiterungen bestehender Klassen um weitere Attribute generell ohne Programmierung möglich sind.

5 Ausblick

Die PDM-Basisdienste haben im Moment einen Stand erreicht, der dadurch gekennzeichnet ist, daß die ersten lauffähigen Prototypen an die Projektpartnern ausgeliefert werden. Diese Prototypen werden von den Software-Entwicklern, den akademischen Partnern und den Industriepartnern validiert. Die ersten Ergebnisse dieser Bewertung sind positiv und zeigen, daß solche Anforderungen wie Modularität, Flexibilität, Skalierbarkeit, Einsatz standardisierter Schnittstellen, Plattformneutralität und explizites Repository grundlegend erfüllt sind.

Zudem ist die Mitarbeit am iViP Projekt mit einem großen Erfahrungsgewinn in der Anwendung neuer Informations-Technologien und Architekturen verbunden. Diese Erfahrungen und Teilergebnisse des Projektes fließen in die weitere Produktentwicklung der einzelnen Partner ein.

Die zukünftigen Aktivitäten bei den PDM-Basisdiensten umfassen die Weiterentwicklung bestehender und die Bereitstellung neuer Dienste. Zu den Highlights zählen die Konzeption und eine prototypische Realisierung des Monitorings (Datenextraktion und –auswertung) sowie des Grafik-Repository. Unter dem letzteren wird eine Meta-Beschreibung der Elemente grafischer Benutzeroberflächen sowie deren Bezug zu den Daten unter Verwendung von XML [W3C00] verstanden.

Die Ergebnisse und Bestrebungen aus dem iViP-Projekt finden auch internationales Interesse. So wurde z.B. auf Initiative von iViP zwecks Interoperabilität der Implementierung des PDM-Enablers Standards ein Implementors' Forum gegründet, wo unter anderem auch die Mitglieder der OMG teilnehmen.

Literatur

[Dada96] P. Dadam: „Verteilte Datenbanken und Client/Server-Systeme“, Springer-Verlag, Berlin Heidelberg, 1996

[GrAn93] H. Grabowski, R. Anderl, A. Polly: „Integriertes Produktmodell“, Beuth Verlag GmbH, 1993

[GrGe97] H. Grabowski, K. Geiger: „Neue Wege zur Produktentwicklung“, Raabe-Verlag, 1997

[HaLe93] H.-J. Habermann, F. Leymann: „Repository: eine Einführung“, Oldenbourg Verlag GmbH, München, 1993

[HeHe96] A. A. Helal, A. A. Heddaya, B. B. Bhargava: „Replication Techniques in Distributed Systems“, Kluwer Academic Publishers, 1996

[ISO94] International Organization for Standardization (ISO): “Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles”, ISO Document 10303-1

[KrTa99] F.-L. Krause, T. Tang, U. Ahle: „Systementwicklungen für die integrierte virtuelle Produktentstehung“, VDI-Bericht 1497, 1999

[LuMa00] U. v. Lukas, A. Mähler, E. Scheinhof: „Kommunikation und Kooperation in der integrierten virtuellen Produktentstehung“, GI-Fachtagung „CAD 2000 – Kommunikation, Kooperation, Koordination“, Berlin, 2000

[OMG00a] Object Management Group (OMG): „Common Object Request Broker Architecture Specification“, OMG Document formal/2000-11-03

[OMG00b] Object Management Group (OMG): „Product Data Management Enablers v1.3 Publication draft“, OMG Document mfg/00-07-02

[OMG00c] Object Management Group (OMG): „Unified Modeling Language Specification“, OMG Document formal/2000-03-01

[OMG00d] Object Management Group (OMG): „XML Metadata Interchange (XMI) Specification V1.1“, OMG Document formal/2000-11-02

[OMG00e] Object Management Group (OMG): „CAD Services V1.0 Request For Proposal“, OMG Document mfg/2000-06-07

[Owen97] J. Owen: „STEP: An Introduction“, Information Geometers Ltd., Winchester, 1997

[RaSt92] F. Rammig, B. Steinmüller: „Frameworks und Entwurfsumgebungen“, In: Informatik-Spektrum, 1992, Bd. 15, S. 33-43

[Saak97] G. Saake: „Föderierte Datenbanksysteme - Plattform für zukünftige EDM/PDM-Anwendungen“, EDM/PDM-Workshop, Magdeburg, 1997

[Satt98] K.-U. Satter: „Tool-Komposition in integrierten Entwurfsumgebungen“, Dissertation, Magdeburg, 1998

[SpKr97] G. Spur, F.-L. Krause: „Das virtuelle Produkt: Management der CAD-Technik“, Carl Hanser Verlag München Wien, 1997

[W3C00] World Wide Web Consortium (W3C): „Extensible Markup Language (XML) 1.0 (Second Edition)“, W3C Document TR/2000/REC-xml-20001006