Integration von Entwicklung und Einführung betrieblicher Standardanwendungssysteme durch Softwareproduktmodelle

Jörg Ritter, Institut OFFIS, Escherweg 2, 26121 Oldenburg

Einleitung

Wird von bekannten Problemen des Softwareengineerings gesprochen, so tauchen Begriffe wie *Software (Development) Process, Software Product Model, Reuse, Software Configuration Management* auf. Alle damit verbundenen Fragestellungen sind auf den eigentlichen Entwicklungsprozeß und selten auf die Anwendung von Standardanwendungssystemen und den damit verbundenen Systemeinführungsprozeß ausgerichtet. Beim Einsatz von betrieblichen Standardanwendungssystemen bleibt der Anwender von diesen herstellerbezogenen Fragestellungen fast unberührt. Es interessiert ihn wenig, welchen Softwareprozeß der Hersteller "lebt", wie gut er seine Entwürfe und Implementierungen wiederverwendet oder wie er eine lauffähige Version zusammenstellt. Der Anwender will sich auch nicht mit abstrakten Softwareentwicklungstechniken (z.B. Frameworks, Design Patterns) auseinandersetzen, die ihn immer noch zum Programmieren zwingen. Ihn interessieren hingegen stärker die Einsatzmöglichkeiten bestehender Systeme und die Frage, wie er möglichst schnell und kostengünstig zu einer "passenden" Kombination und Konfiguration dieser Systeme kommt. Aus diesem Grund wird in diesem Papier die folgende Position vertreten:

Die betriebliche Anwendungssystemgestaltung unterzieht sich einem starken Wandel weg von der reinen Entwicklung hin zur Konfiguration von Standardanwendungssystemen. Softwareentwicklungsprozesse, die speziell die Entwicklung von anwenderseitig zu konfigurierenden betrieblichen Standardanwendungssystemen berücksichtigen, werden bislang nicht ausreichend erforscht. Deshalb müssen Softwareproduktmodelle entwickelt werden, durch die der Entwicklungsprozeß beim Hersteller und der Einführungsprozeß beim Anwender stärker integriert werden. Softwarehersteller müssen dazu Modelle ihrer Softwarekomponenten liefern, aus denen konstruktiv Modelle kundenindividueller Systeme aufgebaut werden können. Diese Modelle der Hersteller müssen auf standardisierten Sprachen/Metamodellen aufbauen. Die Integration muß zudem über Zwischenschritte (Vorkonfigurationen) vorgenommen werden können, die den Übergang von abstrakten Softwareprodukten zu den verschiedenen inhaltlich geprägten Instanzen (Konfigurationen) erleichtern.

Entwicklung Betrieblicher Standardanwendungssysteme

Betriebliche Standardanwendungssysteme (BSAS) entstehen nicht unmittelbar aus den Anforderungen eines einzelnen Anwenders, wie es z.B. bei der Individualentwicklung der Fall ist. Vielmehr entstehen sie aus einem iterativen, zunehmend anwenderanonymen Softwareentwicklungsprozeß. Durch zunehmende Abstraktion vom konkreten Anwender und damit vom Anwendungsfall werden diese Systeme allgemeingültiger und können letztlich zu branchenübergreifenden Standards werden. Die Abstraktion wird nicht direkt durch Techniken z.B. der objektorientierten Softwareentwicklung, die u.a. in wiederverwendbaren Bibliotheken und Frameworks zum Ausdruck kommt, erreicht, sondern durch Konfigurations- und Parametrierungstechniken ("Customizing"). Somit haben diese Entwicklungstechniken ihren Nutzen primär in der Produktion, wirken sich aber so gut wie gar nicht auf den Endanwender aus.

Die Parametrierung und Konfiguration findet im Rahmen der durch die Entwicklung definierten Grenzen statt. Ein Problem dabei ist, daß durch die zunehmende Abstraktion (und der daraus resultierenden aufwendigeren kundenindividuellen Spezialisierung) die Möglichkeiten der Systeme vom Anwender nicht mehr schnell genug nachvollzogen werden können. Es werden Produkte geschaffen, die zwar immer universeller werden, über deren Einsatzgebiete aber selbst der Entwickler nur noch spekulieren kann. Dieses Problem wird um so gravierender, je feiner die Granularität (und damit größer der Anwendungsbereich) der kombinierbaren Komponenten (=> Component Ware) wird.
Der Dokumentation der Systeme fällt deswegen eine zentrale Rolle zu. In herkömmlichen Dokumenten (z.B. Handbüchern) können nicht alle Einsatzmöglichkeiten beschrieben werden. Die Dokumentation (auch in Form heutiger Softwarereferenzmodelle) hat somit oft exemplarischen Charakter. Beschriebene Interaktionen mit dem System können stark variieren, da sie von der Konfiguration des Systems abhängen. Das Dilemma ist, daß abstraktere Beschreibungen dagegen vom Anwender nicht so leicht verstanden werden.

Einführung Betrieblicher Standardanwendungssysteme

Im Gegensatz zur Individualsoftwareentwicklung formuliert der Anwender seine Anforderungen nicht mehr gegenüber einem Softwarehersteller sondern gegenüber einem Markt, der ihm konfigurierbare Systeme (und in Zukunft vielleicht auch beliebig kombinierbare Komponenten) anbietet. Nach der Analyse der Anforderungen eines Anwenders (mit eventueller Beraterunterstützung) werden die geeignetsten Produktkombinationen ausgewählt. Hierbei nimmt der Anwender Abstriche bzgl. seiner Anforderungen hin, die er aber häufig erst später bemerkt.
Die ausgewählten Produkte - handelt es sich nicht um relativ einfache Systeme - müssen den Wünschen des Anwenders entsprechend konfiguriert werden. Diese Konfiguration ist kompliziert und erfordert detailierte Kenntnisse des Systems. So wird z.B. in [PIE93] berechnet, daß in dem R/2-Modul RM der SAP bei der Pflege von 5.000 Teilestammsätzen und 45 materialspezifisch zu pflegenden Parametern über 200.000 Steuergrößen einzustellen sind. Die detailierte Analyse der Anforderungen, der Entwurf und die Durchführung der Konfiguration eines BSAS nehmen darum nicht selten mehrere Personenjahre in Anspruch.
Betrachtet man heutige Einführungsprojekte, so fehlt es oft am frühzeitigen System-Know-How der Anwender und Berater. Hinzu kommt, daß Neuentwicklungen schlecht oder gar nicht dokumentiert sind. Konsequenzen verschiedener Konfigurationseinstellungen können oft nur unter erheblichen Kosten in Form von Prototypen durchgespielt werden. Scheinen die BSASe in vielen Fällen zwar trotzdem die kostengünstigere (und oft auch die bessere) Alternative zu sein, so lassen sich doch bei Auswahl und Konfiguration noch Kosten einsparen. Dazu muß schnell erkannt werden, ob die geforderten Geschäftsprozesse durch das BSAS unterstützt werden und welche Alternativen an Modulen und Konfigurationen es gibt.

Anforderungen an die Integration von Entwicklung und Einführung

Vor dem Hintergrund der durch die Standardisierung von Anwendungssystemen bedingten Anonymisierung der Beziehung zwischen Entwickler und Anwender müssen Mittel gefunden werden, diese Lücke zu überbrücken. Dazu müssen Kommunikationswege zwischen beiden Seiten (Hersteller, Anwender) etabliert werden, die dem Anwender die Standardanwendungssysteme und ihre Einsatzmöglichkeiten leichter verständlich machen. Von einigen Herstellern werden heute schon sogenannte Softwarereferenzmodelle ([BEC97]) entwickelt, die (wie z.B. das R/3-Referenzmodell der SAP, [KEL97]) die aufbauorganisatorischen Möglichkeiten, Datenstrukturen,

die Grundfunktionalität und typische Geschäftsprozesse des Systems mittels Informationsmodellen (z.B. auf Basis der ARIS-Architektur, [SCHE91]) beschreiben. Diese Modelle stellen einen wichtigen Ausgangspunkt beim Abgleich von Anforderungen des Anwenders und Leistungen der entsprechenden Systeme dar, dienen als Diskussions-, Design- und Schulungsgrundlage für Einführungsteams und bieten einen Zugang zum "Customizing".

Allerdings weisen die heute bekannten Modelle (und Werkzeuge) ernste Mängel auf. In der Regel fehlt den verwendeten Modellierungssprachen (z.B. der EPK im R/3-Referenzmodell) die nötige formale Grundlage (siehe [RUM95]). Wichtige Eigenschaften, wie z.B. Kompositionalität von Produktmodellen werden nicht gewährleistet. Ferner gibt es keine herstellerübergreifenden Standards zur Beschreibung von Standardsoftware-produkten, vergleichbar z.B. mit dem Industriestandard STEP (ISO 10303, International Standard for the computer-interpretable representation and exchange of product data) oder seinen Erweiterungen PAREX (ISO 14959, Working draft for "Parametric Representation and Exchange") für Produktmodelle.

Ferner sollten diese Modelle einen anwendungsbezogenen Zugang bieten, da der Anwender sich nicht primär mit software-architektonischen Fragestellungen beschäftigen möchte, sondern sich eine Sicht auf ein Modell wünscht, die zuerst seine fachlichen Fragestellungen beantwortet. Entsprechende Werkzeuge müssen ihn bei Selektion, Modifikation und Kombination der Produktmodelle unterstützen. Dem Anwender muß es ermöglicht werden, möglichst früh seine Terminologie durch entsprechende Werkzeuge einzubringen.

Die Modelle sollten außerdem möglichst direkt aus dem Softwareentwicklungsprozeß der zugehörigen Softwareprodukte entstehen und integraler Bestandteil dieser Produkte sein. Nur so kann einigermaßen gewährleistet werden, daß Produkt und Modell zueinander passen.

Ideen für eine Integration

In diesem Abschnitt sollen Ideen für die Integration von Entwicklung und Einführung von BSASen aufgeführt werden. Dabei wird die zur Zeit vorherrschende dreiteilige Organisationsstruktur bis hin zur Einführung dieser Systeme - Softwarehersteller, Berater und Anwender - beibehalten. Die Aufgabeninhalte der beteiligten Parteien ändern sich allerdings. So werden sich die Berater zunehmend mit der Zusammenstellung und Vorkonfiguration von Komponenten auseinandersetzen. Sie sollen deswegen hier als Konfigurierer bezeichnet werden.

Der Ausgangspunkt für die Integration sind Softwareproduktmodelle, die kombinierbare und konfigurierbare Komponenten beschreiben. Diese könnten aus Dokumenten der Softwareentwicklung (z.B. UML-Dokumente, [UML97]) abgeleitet werden. Voraussetzung dafür ist ein herstellerübergreifendes Systemverständnis. Dazu müssen formale Systemmodelle (siehe [RUM96]) entwickelt werden, auf denen die Beschreibungssprachen für Produktmodelle aufbauen. Die Beschreibungssprachen werden um Konzepte zur Parametrisierung und zur Definition von Constraints, mit denen die Variantenvielfalt der zugehörigen Produkte ausgedrückt werden kann, erweitert.

Um dem Anwender die Anpassung der abstrakten Systeme auf seine Anforderungen zu erleichtern, kann diese komplexe Aufgabenstellung in einer Vorstufe durch (Vor-)Konfigurierer übernommen werden, die branchen- und/oder betriebstypische (Teil)-Systemkonfigurationen erstellen. Sie konstruieren aus den abstrakten Produktmodellen inhaltlich bezogene Systeme, die aber immer noch weiter individualisierbar sind.

Der Prozeß der Wiederverwendung der Produktmodelle (Selektion, Kombination und Modifikation) soll hier über einen (geschäfts)-prozeßorientierten Ansatz realisiert werden, der sich den Zusammenhang "Geschäftsprozeßdefinition = Menge von Traces (Abläufe) der Komponenten" zunutze macht: Ausgehend von

einer Komponente kann der Konfigurierer sich Aktionsfolgen dieser Komponente heraussuchen. Wird eine Interaktion (= Kommunikation mit einer anderen Komponente) selektiert, kann der Benutzer aus der Liste der kompatiblen Komponenten eine heraussuchen (=Kombination) und weitere Aktionsfolgen aufbauen. Aus diesen (Inter)-Aktionsfolgen können dann sukzessive Geschäftsprozesse aufgebaut werden. Parallel zum Aufbau werden die Parameter und Constraints der selektierten Komponenten aus dem Produktmodell mit denen der schon selektierten Komponenten kombiniert und überpüft. Parameter können vom Konfigurierer gesetzt werden (Modifikation) und wirken sich entsprechend der Constraints auf die Gesamtkonfiguration und so z.B. auf die möglichen Traces aus. Durch das Offenlassen von Interaktionen und Nichtbesetzen von Parametern werden Teil- oder Vorkonfigurationen erreicht, die z.B. branchen- oder betriebstypische Lösungen darstellen können.

Der Endanwender kann diese Vorkonfigurationen nutzen, um sie mit seinen Anforderungen abzugleichen bzw. um durch Modellvarianten die für ihn geeignetste Lösung zu finden. Eine ausgewählte Modellkonfiguration kann als Grundlage für die spätere Systemkonfiguration dienen.

Die geforderte Anwendungsbezogenheit kann durch Verwendung von Sprachen erreicht werden, die für den Anwender eine hohe Akzeptanz besitzen. Neben etablierten (aber leider nicht standardisierten) Sprachen zur Beschreibung von Strukturen von Systemen, wie z.B. ER-Modellen oder Klassendiagrammen, setzen sich im Umfeld der BSASe vor allem die Ereignisgesteuerten Prozeßketten (EPK, [SCHE91]) zur Beschreibung des Verhaltens durch. Diese "geschäftsprozeßorientierten" Sprachen haben im Kontext der BSASe einen hohen Stellenwert und sollen deswegen im folgenden betrachtet werden.

Wie schon angedeutet, können Geschäftsprozeßsprachen (mit entsprechendem formalen Fundament) zur fachlichen Beschreibung von Mengen von Traces der zugrundeliegenden Softwareprodukte verstanden werden. Konkrete Geschäftsprozeßbeschreibungen lassen sich also konstruktiv über Traces aus den Beschreibungen der Produktmodelle ableiten. Der These, daß Anwender sich mit formalen Sprachen schwer tun und semi-formale Sprachen leichter nachvollziehbar sind [AMB96], wird hier nicht gefolgt. Vielmehr müssen Wege gefunden werden, Sachverhalte, die mittels formaler (oder besser "präziser") Sprachen ausgedrückt werden, dem Anwender nahezubringen. Dafür bieten sich Diagrammtechniken aber auch die natürliche Sprache an, wenn man sie durch einen sprachkritischen Ansatz mittels Orthogrammatiken (siehe [LOR87]) von Inkonsistenzen und Mehrdeutigkeiten bereinigt. Im Rahmen des hier vorgestellten Konzeptes wird eine Kombination aus diagramm- und orthogrammatischen Techniken vorgeschlagen: Elementaraussagen werden auf Basis einer Orthosprache formuliert, temporallogische Zusammenhänge werden durch Diagrammtechniken ausgedrückt. Dabei kann diese Kombination sowohl auf Ebene der Produktmodelle als auch auf Ebene der daraus resultierenden Konfigurationen genutzt werden.

Für die hier geschilderte Nutzung der Produktmodelle bedarf es eines einheitlichen und genormten Konfigurationsrahmens, dessen sich der Konfigurierer bedient. Dieser Rahmen sorgt dafür, daß Einstellungen, die während der Konfiguration einer Komponente vorgenommen wurden, sich auch auf andere Komponenten auswirken. Hierzu können Propagierungsmechanismen herangezogen werden, wie sie aus dem *Software Configurations Management* (siehe [ZEL97]) bekannt sind.

Wichtig ist weiterhin, daß der Anwender seine Terminologie in die Modelle einbringen kann. Im Produktmodell definierte Begriffe können dazu im Kontext der jeweiligen Anwendung spezialisiert und genauso, wie die übrigen Parameter in andere Komponenten propagiert werden.

Die hier geschilderte "geschäftsprozeßorientierte" Anwendungssicht kann nur eine von vielen Sichten auf die Produktmodelle sein. Bedingt durch den exemplarischen Charakter der Geschäftsprozeßbeschreibungen müssen

andere Sichten, wie z.B. eine Komponentensicht, die Komponenten, ihre Parameter und ihre Verbindungen zu anderen Komponenten beschreibt, vorhanden sein.

Die geforderte Integration von Softwareprodukt (Komponente) und Produktmodell kann durch *Introspektionsmechanismen* (vgl. [SUN97a]) erreicht werden, über die der Konfigurierer (und später auch der Anwender) die Möglichkeiten des Systems mittels entsprechender Werkzeuge analysiert. SUNs Introspektionsmechanismen beschränken sich heute ausschließlich auf strukturelle Eigenschaften (Schnittstellen, Parameter, Constraints) der Komponenten (JavaBeans). Sie müßten um verhaltensorientierte Beschreibungsmittel (Zustandsmaschinen) ergänzt werden.

Offene Fragen

Eigentlich "neu" an den präsentierten Ideen sind weniger die verwendeten Techniken, als vielmehr das damit verbundene Plädoyer, sie stärker in den Kontext von betrieblichen Standardanwendungssystemen einzubeziehen. Viele, vor allem technische Fragen, wurden hier vernachlässigt. Die skizzierten Softwareproduktmodelle sind zuerst auf rein fachlicher Ebene anzusiedeln. Technische Problemstellungen, wie z.B. die Schnittstellen- und Verhaltenskompatibilität, die Versionsverwaltung, die bei dem Austausch von Komponenten berücksichtigt werden müssen, und der Zusammenhang zwischen fachlicher und technischer Ebene wurden in den Ausführungen vernachlässigt.

Literatur

[AMB96] M. Amberg, R. Striemer, and M. Weske. *Modellierung von Workflows - Ein Rahmenmodell.* GI Arbeitskreis Workflow, 1996.

[BEC97] J. Becker, M. Rosemann, and R. Schütte. Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung. Arbeitsbericht Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster, Münster, März 1997.

[UML97] G. Booch, J. Rumbaugh, and I. Jacobson. Unified Modeling Language - Notation Guide. 1997.

[KEL97] G. Keller, and T. Teufel. *SAP R/3 prozeßorientiert anwendenEdition SAP.* Addison-Wesley, 1997.

[LOR87] P. Lorenzen. *Lehrbuch der konstruktiven Wissenschaftstheorie.* B.I.-Wissenschaftsverlag, Mannheim, Wien, Zürich, 1987.

[PIE93] M. Pietsch. PAREUS-RM - ein Tool zur Unterstützung der Konfiguration von PPS-Parametern im SAP-System R/2. *Wirtschaftsinformatik*, 35(5): pages 434-445, 1993.

[RUM95] F. Rump. Ereignisgesteuerte Prozeßketten zur formal fundierten Geschäftsprozeßmodellierung. *Informationssystem-Architekturen*, 2(2): pages 94-96, 1995.

[RUM96] B. Rumpe. Formale Methodik des Entwurfs verteilter objektorientierter Systeme. Technische Universität München, München, 1997.

[SCHE91] A.-W. Scheer. *Architektur integrierter Informationssysteme.* Springer-Verlag, Berlin, 1991.

[SUN97a] SUN. *JavaBeans.* Sun Microsystems, 1997.

[ZEL97] A. Zeller. Configuration Management with Version Sets. PhD thesis, Technische Universität Braunschweig, Braunschweig, 1997.