

mehr zum thema:
www.osthus.de
www.xprogramming.com

MODELLGETRIEBENE SOFTWARELÖSUNGEN FÜR „REAL-TIME UNTERNEHMEN“

Die Zukunft gehört den agilen Unternehmen, die fähig sind sich schnell und zielgerichtet auf neue Anforderungen auszurichten. Diese Erkenntnis erfordert auch ein neues Denken und Handeln bei der Entwicklung von IT-Systemen. Ausgehend von sieben Prinzipien beschreibt der Artikel ein Vorgehensmodell für die Entwicklung von IT-Systemen, die einerseits robust sein sollen und deren Prozesse andererseits flexibel in „Echtzeit“ änderbar sind.

Agile Unternehmen sind in der Lage ohne wesentliche Zeitverzögerung für das Unternehmen relevante Information aufzunehmen und in entsprechende, den Geschäftserfolg unterstützende Aktivitäten und Strukturen umzusetzen. Die Gartner Group hat in diesem Zusammenhang den Begriff *Real-Time Enterprise (RTE)* geprägt.

Nahezu gleichgültig in welchem Unternehmensbereich: IT-Systeme bilden das Rückgrad, auf dem Unternehmen ihre Echtzeit-Strukturen ausbilden. Ob IT-Systeme in der Lage sind die Vision eines RTE zu unterstützen, entscheidet sich schon bei deren Erstellung.

Bezogen auf die Softwareentwicklung werden in den letzten Jahren zunehmend neue agile Vorgehensmodelle wie „Extreme Programming“ oder „Adaptive Software Development“ diskutiert und eingesetzt. Diese agilen Modelle sind die Antwort der Software-Ersteller auf die Erkenntnis, dass agilen Unternehmen die Zukunft gehört.

Die Softwareentwicklung war Jahrzehnte von dem so genannten Wasserfallmodell geprägt. Dieses basiert auf Leitgedanken oder Prinzipien, von denen einige hier kurz erwähnt seien:

- Ein Erstellungsprozess ist nur dann tragfähig, wenn er in wohl definierten Prozessschritten abläuft.
- Jeder Prozessschritt muss beendet sein, bevor der nächste beginnen kann.
- Jeder Prozessschritt enthält die vollständige Komplexität des geforderten Systems. Beim Übergang von einem Schritt zum nächsten kommen lediglich Details hinzu.
- Was gut dokumentiert ist, wird auch entsprechend gut realisiert.

Stellt man diese Prinzipien teilweise oder ganz in Frage, so lässt sich entsprechend das

Wasserfallmodell nicht mehr anwenden und es müssen Prinzipien gefunden werden, die den RTE-Anforderungen Rechnung tragen.

Sieben Entwicklungsprinzipien

Bevor man ein neues Modell der Softwareentwicklung auswählt, anpasst oder erstellt, muss man sich über die Anforderungen von RTE im Klaren sein und daraus Prinzipien für die Entwicklung ableiten. Was sind die Anforderungen eines RTE an die Softwareentwicklung? Um diese Frage zu beantworten, benennen wir zunächst einige Eigenschaften eines RTE:

- Da das RTE adaptiv auf seine Umwelt reagiert, ändern sich die Geschäftsprozesse in zunehmend kürzeren Zeiten. Häufig haben sich die Geschäftsprozesse schon geändert, bevor das entsprechende System fertig gestellt wurde.
- Um Wettbewerbsvorteile zu erzielen, müssen Prozesse „anders“ sein und die IT-Systeme müssen sich an den Prozessen ausrichten. Dies trifft insbesondere für die Bereiche zu, in denen sich das Unternehmen von seinen Wettbewerbern unterscheiden will, also z. B. in Forschung und Entwicklung.
- Prozesse, die einen hohen Automatisierungsgrad erlauben, sollen automatisiert ablaufen um Zeitvorteile zu generieren; gleichzeitig muss es möglich sein, diese Prozesse in einem (gewissen) Rahmen auf neue Gegebenheiten kurzfristig anzupassen.

Aus diesen zentralen Echtzeit-Eigenschaften erwachsen folgende Anforderungen:

- Businessprozesse müssen managbar sein. Dies umfasst die Prozess-Modellierung,

die autoren



Dr. Alfred Oswald
(E-Mail: alfred.oswald@osthus.de) ist Geschäftsführer und Projektmanager bei der Osthus GmbH und verantwortlich für die fachliche Konzeption und das Projektmanagement des Entwicklungskonzepts JEDI.



Dr. Torsten Osthus
(E-Mail: torsten.osthus@osthus.de) ist Geschäftsführer und Projektleiter bei der Osthus GmbH und verantwortlich für Marketing und Vertrieb.

die Prozess-Konfiguration, das Prozess-Monitoring und die Prozess-Optimierung.

- Da die IT-Systeme Prozesse abbilden, an (neue) Prozesse adaptierbar sein sollen und einen hohen Automatisierungsgrad haben sollen, entstehen IT-Landschaften und Systeme mit erheblicher Komplexität, die beherrscht werden muss.
- Die IT-Architektur muss nachhaltig sein, d. h. redundanzfreie Strukturen, service-orientierte Architektur, wieder verwendbare Komponenten (ein Problem, eine Lösung).

Aus den genannten Anforderungen können die in **Tabelle 1** dargestellten sieben Entwicklungsprinzipien abgeleitet werden.

Das Vorgehensmodell

Um die in **Tabelle 1** dargestellten Prinzipien in die Praxis umzusetzen, benötigen wir ein Vorgehensmodell, das die mit den Prinzipien verbundenen Charakteristika abbilden kann.



Nr. Prinzipien	Erläuterungen
1. Der Prozess der Softwareentwicklung ist als komplexes, adaptives System zu behandeln.	Der Prozess der Softwareentwicklung ist nicht mehr über „wasserfall“-artige Strukturen abbildbar. Die Anforderungen an die Systeme ändern sich gegebenenfalls mehrmals während des Erstellungsprozesses. Außerdem ist die Komplexität der erstellten Systeme oft so hoch, dass es nicht möglich ist, die Spezifikation der Anforderungen vollständig vorzugeben.
2. Geschäftsprozessgestaltung (Analyse und Design) gehören untrennbar zur robusten Gestaltung von Architektur und Begriffswelt (Geschäftsobjekte): Struktur und Prozess gehören zusammen.	Sehr oft ist die Geschäftsprozessmodellierung eine Domäne spezieller Abteilungen. Damit kann der adaptive Prozess zwischen Struktur und Prozess nicht einsetzen: Die Geschäftsprozesse beeinflussen die Gestaltung der Architektur und der Geschäftsobjekte und umgekehrt.
3. Die Trennung von Geschäftsprozessgestaltung, Systemanalyse und Systemdesign berücksichtigt evtl. vertragliche Rahmenbedingungen, ist ansonsten aber eher willkürlich.	Geschäftsprozessgestaltung, Systemanalyse und Systemdesign sind ein mehrfach iterativer Prozess. Im Rahmen dieses Prozesses werden Geschäftsprozesse verallgemeinert und fließen im Rahmen eines Abstraktionsprozesses in Design-Strukturen.
4. Der Systemgestalter muss federführend den gesamten Zyklus der Softwareentwicklung abdecken.	Die Güte der Software hängt wesentlich davon, ob ein oder mehrere Personen, die Systemgestalter, den gesamten Prozess der Softwareerstellung konstruktiv begleiten. Hier bietet sich im Idealfall ein Tandem aus Kunde und Software-Ersteller an.
5. Die Identifikation und das Design von Mustern auf allen Ebenen (Kommunikation, Prozess, Begriffswelt, Design, Architektur) entsprechend den Zielen (Strategien und Prinzipien) ist zentral für die Erstellung eines zukunftsfähigen Systems: <ul style="list-style-type: none"> • Verhaltensmuster • Organisationsmuster • Prozessmuster • Analysemuster: Domänenmuster, Applikationsmuster, Architekturmuster usw. • Designmuster: Entwurfsmuster, Architekturmuster, Schnittstellenmuster usw. 	Das Denken in Modellen und Mustern ist zum einen entscheidend, um die nötige Abstraktion für die Systemerstellung zu erhalten, und zum anderen überkommene Muster zu identifizieren und Redundanzen zu vermeiden.
6. Offene, visuelle Kommunikation ist das Mittel der Wahl, um sowohl den Entwicklungsprozess als auch das System adaptiv zu erstellen.	Ein Bild sagt mehr als 1.000 Worte: Visualisierung dient sowohl der Problemlösung, der Lösungsfindung als auch der Ergebnisdokumentation. Elementar ist hierbei, dass die Bilder in der Analyse und in der Designphase und umgekehrt verwendet werden, um die Konsistenz der Modelle und Begriffswelt sicherzustellen.
7. Die Softwareentwicklung erfolgt modellgetrieben: Es wird soviel automatisiert, wie sinnvoll ist, aber nicht mehr.	Die Automatisierung des Softwareentwicklungsprozesses ist von entscheidender Bedeutung, um die Entwickler von Routine-tätigkeiten zu entlasten und der technologie-freien Realisierung näher zu kommen. Gleichzeitig eröffnet sie die Möglichkeit der Vision einer Echtzeit-Softwareentwicklung, indem möglichst viele der Anforderungen über Modelle abgebildet werden, die die Generierung von Code erlauben.

Tabella 1: Sieben Entwicklungsprinzipien

Verbindet man auf Basis der sieben Prinzipien die Anforderungen eines agilen Entwicklungsprozesses mit denen flexibler Geschäftsprozesse und rationeller Technologie, so folgt daraus das in **Abbildung 1** dargestellte Vorgehensmodell, das wir im Folgenden als „Real-Time Entwicklungsmodell“ bezeichnen. Wie die Abbildung erkennen lässt, untergliedern wir den Entwicklungsprozess grob in drei Phasen:

- Systemanalyse
- Systemdesign & Codierung
- Lieferung & Installation

Hierbei ist zu beachten, dass dieses „Phasenmodell“ nicht dem klassischen Wasserfallmodell unterliegt, in dem die Phasen nacheinander abgearbeitet werden und keine Rückkopplung zu den vorhergehenden Phasen stattfindet.

Im Gegensatz zum Wasserfallmodell erfolgen bei diesem „Phasenmodell“ immer wieder Rückkopplungen über die einzelnen Phasen hinweg, was zu einer adaptiven und iterativen Entwicklung führt. Das bedeutet, dass sich Entwickler mehr mit der Fachlichkeit und Berater mehr mit der Technologie beschäftigen müssen. Das Modell verbindet damit die Vorteile einer systematischen und strukturierter Vorgehensweise mit den Anforderungen agiler Softwareentwicklung.

Systemanalyse und Businessprozess-Management (BPM)

Die Phase Systemanalyse des Real-Time-Entwicklungsmodells ist im Wesentlichen geprägt durch die Anforderung, Businessprozesse „managbar“ zu machen, und gliedert sich in die vier Hauptaktivitäten:

- Businessprozess-Analyse
- Architektur
- Oberfläche
- Glossar

Der Begriff Analyse trifft hierbei nur einen Teil der Aktivitäten in der „Phase Analyse“: Die vorhandenen Prozesse (Ist-Prozesse) werden analysiert, vorhandene Prozesse werden modifiziert, neue Prozesse werden kreiert, Prozesse werden vereinheitlicht, verallgemeinert und zusammengelegt. Damit werden aus den vielen Ausnahmen der Anwender robuste Kernprozesse gebildet, die trotzdem die spezifischen Anforderungen der Anwender berücksichtigen.



Ebene	Beispiel
Geschäftsprozess	Der Substanzlogistik Geschäftsprozess ist „Aufnahme, Lagerung, Konfektionierung und Verteilung von Substanzen“.
Hauptprozess	Zu den Hauptprozessen gehören z. B. <ul style="list-style-type: none"> • „Wirkstoffe und Formulierungen bestellen“ • „Wirkstoffe und Formulierungen nachfordern“ Ein Hauptprozess besteht aus mehreren Teilprozessen.
Teilprozess	Z. B. „384er MTP' nachfordern“: Ein Teilprozess kann mehrere Anwendungsfälle enthalten.
Workflow-Prozess (WFP)	Ein Workflow-Prozess kann aus mehreren Hauptprozessen > Teilprozessen > Anwendungsfällen bestehen.
Anwendungsfall	Z. B. „Arbeitsrack leeren“: Dieser Anwendungsfall ist z.B. im Teilprozess „384er MTP' enthalten“
Servicefall	Der Begriff des Servicefalls wird im Rahmen des Systemdesigns verwendet. Ein Anwendungsfall wird in der so genannten Business-Logik durch gegebenenfalls mehrere Servicefälle abgebildet (siehe service-orientierte Architektur)

Tabelle 2: BPM – Prozesshierarchie

Damit ein flexibles BPM möglich wird, ist die Einführung einer Prozesshierarchie sinnvoll. Hierdurch wird eine Strukturierung der Prozesse mit unterschiedlicher Granularität möglich. **Tabelle 2** gibt einen Überblick über das verwendete Modell der Prozesshierarchie (mit Beispielen aus der Substanzlogistik eines Chemieunternehmens).

Prozesse werden insbesondere in der Phase der Identifikation und Festlegung der Hauptprozesse mittels Prozess-Ablauf-Diagrammen visualisiert. Wenn die Hauptprozesse im Wesentlichen festliegen, empfiehlt sich für die Darstellung der Teilprozesse, der Anwendungsfälle und Workflow-Prozesse der Übergang zu einer tabellarischen Darstellung.

Begriffe werden in einem Glossar gesammelt, klarer gefasst und anhand ihrer Prozesse an einer konsistenten Verwendung über alle Prozesse hinweg überprüft.

Anhand der Prozesse und der Begriffswelt werden Komponenten, d.h. „Kästchen“ gezeichnet, mit deren Hilfe man Prozesse „fassbar“ macht. Die Menge aller Komponenten, ihre Beziehungen untereinander sowie die Schnittstellen über die Systemgrenzen hinweg bezeichnen wir als (Analyse-) Systemarchitektur. Wichtig ist es, jedem der Kästchen einen Namen zu geben und diesen wesentliche Funktionen, die aus den Prozessen ermittelt wurden, zuzuordnen. Alle internen und externen Schnittstellen werden dargestellt. Dies vermittelt ein unmittelbares Gefühl für die Komplexität des Systems.

Wenn sich innerhalb des Kernteams das Verständnis bezüglich der Prozesse, der Begriffswelt und der Systemarchitektur ange nähert hat, ist es Zeit, die gewonnenen Erkenntnisse an Hand von zentralen Masken zu überprüfen und einige wesentliche Prozesse an Hand von Maskensequenzen „durchzuspielen“. Auch hier wird sich herausstellen, dass die Gestaltung der Masken die Prozesse, die Begriffswelt und die Systemarchitektur beeinflusst und umgekehrt.

In den meisten Fällen beeinflussen sich Prozesse, Begriffe, Architektur und Masken gegenseitig – der Klärungsprozess wird mehrmals durchlaufen und ergibt am „Ende“ ein in sich konsistentes System.

Während der gesamten Analysephase werden die Ziele, die mit dem System verfolgt werden, an den übrigen Analyseergebnissen gespiegelt. Gegebenenfalls werden auch die Ziele überarbeitet und die Strategie zur Erlangung der Ziele wird angepasst.

Bei dem Wort „Ende“ ist Vorsicht geboten, denn wie die obige Abbildung erkennen lässt, kann man nur von einem vorläufigen Abschluss sprechen.

Die Phase Systemanalyse wird im Kontext einer auftragsorientierten Abarbeitung als beendet angesehen, wenn als Arbeitsergebnis eine Systemanalyse-Spezifikation (Pflichtenheft) vorliegt, die vom Projektleiter des Kunden abgenommen wird.

Übergang zum Systemdesign

Um einen Bruch zwischen Analyse und Design zu verhindern, haben sich folgende

Maßnahmen mit rückkoppelnder Wirkung in die Analysephase bewährt:

- Entwickler werden an der Analysephase beteiligt. Damit wird sichergestellt, dass Domänenwissen in die Design- und Kodierungsphase übernommen wird und Designkompetenz schon in der Analysephase eingebracht werden kann.
- Für komplexe Systeme, die zudem noch in einer neuen Technologie erstellt werden, ist es sinnvoll das System- und das Technologiewissen in Form von die Kodierung begleitenden Workshops zu vermitteln. Hierbei übernimmt der Systemgestalter, der die Systemanalyse wesentlich mitgestaltet hat, die Rolle des Coaches für die Vermittlung des Systemwissens.
- Zusätzlich ist es sinnvoll, den Designprozess und seine Ergebnisse an dem Systemverständnis der Anwender zu spiegeln. Workshops zum Projektfortschritt in Form von Reviews von Design-dokumenten und/oder Reviews des bisher realisierten Systems wirken dem Risiko der „Systementfremdung“ entgegen.

Im Rahmen des Systemdesigns werden Dokumente zur Benutzungsoberfläche, zur Business-Logik, zu den Schnittstellen und zur Komponentenarchitektur erstellt und iterativ gepflegt.

Die Agilität während der Phase Systemdesign und Kodierung wird weiterhin durch die Leitgedanken des „Extreme Programming“ unterstützt wie z.B. „Pair Programming and Testing“, inkrementelle und kleine Releases, kontinuierliche Integration und Tests.

Automatisierung des Softwareentwicklungsprozesses

Die bisher beschriebenen Modelle sind notwendige, aber nicht hinreichende Bedingungen für zukunftsrobuste IT-Systeme. Eine weitere Dimension ist die Rationalisierung der technologischen Realisierung, damit Änderungen der Anforderungen im oder nach dem Projekt auch technologisch schnell und mit geringem Aufwand integriert werden können. Um den Ablauf vom Systemdesign bis zur Lieferung zu rationalisieren, werden drei Elemente benötigt:

- Automatische Codegenerierung
- Automatisches *Deployment*
- Agiles *Change Request Management*



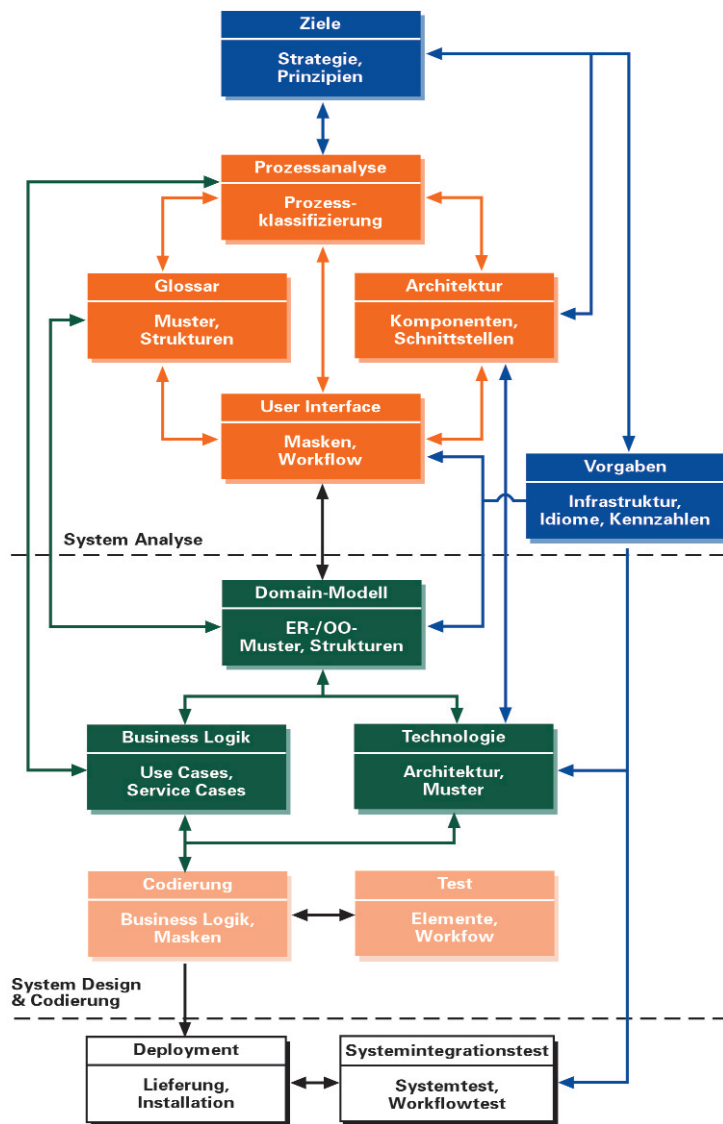


Abb. 1: Real-Time-Entwicklungsmodell

Zur automatischen Codegenerierung wird die MDA-Entwicklungsplattform *JEDI* (*Java Enterprise Development Intelligence*) eingesetzt. Mit dem Ansatz werden 80% des Programmcodes sowie das komplette *Deployment* automatisch erzeugt. Mit der Plattform ist es möglich Änderungen am Projekt in „Echtzeit“ und mit hoher Qualität zu integrieren.

Fazit

Das beschriebene Modell verbindet innovative Konzepte agiler Softwareentwicklung, des Businessprozess-Managements und der *Model Driven Architecture* (MDA) in einer Strategie. Damit können zukunftsrobuste Prozesse und Architekturen mit der für agile Unternehmen nötigen Flexibilität in IT-Systemen abgebildet werden. ■

Literatur

- [Col04] W. Colsman, Ein generatives Entwicklungskonzept, in: *JavaSPEKTRUM* 4/04
- [Osw04] A. Oswald, W. Colsman, Ein Weg zum Einsatz der Model Driven Architecture in größeren Projekten, in: *OBJEKTSPEKTRUM Online-Spezialausgabe „Model Driven Software Development“*, Oktober 2004

