Ein Referenz-Prozess der Software-Migration

(erweiterte Kurzfassung)

Ellen Ackermann
Andreas Winter
Universität Koblenz-Landau
Institut für Softwaretechnik,
D-56016 Koblenz
(ackerm|winter)@uni-koblenz.de

Rainer Gimnich
IBM Software Group
Enterprise Integration Solutions (EIS)
Wilhelm-Fay-Str. 30-34,
D-65936 Frankfurt
gimnich@de.ibm.com

Zusammenfassung

Prozessmodelle der Software-Entwicklung und der Software-Migration wurden bisher im Allg. unabhängig voneinander entwickelt. Ausgehend von einer umfangreichen Literaturstudie zu Prozessmodellen der Software-Migration, wird im Folgenden ein generisches Prozessmodell entwickelt, das Aktivitäten, Akteure und Artefakte der Softwaremigration in ein inkrementelles und iteratives Prozessmodell der Software-Entwicklung integriert.

1. Einleitung

Softwaresysteme durchlaufen einen mehr oder weniger vorgegebenen *Lebenslauf*. Auf die initiale System-Entwicklung folgt zunächst eine Phase der kontinuierlichen Weiterentwicklung, in der das System an geänderte Anforderungen angepasst wird. Werden solche Änderungen aufgrund der nicht mehr ausreichenden Software-Architektur zu komplex, werden nur noch Fehlerkorrekturen vorgenommen, bis auch diese wirtschaftlich nicht mehr sinnvoll sind, und das System durch ein neues abgelöst wird [12].

Die *Software-Migration* zielt darauf ab, die Phase der kontinuierlichen Weiterentwicklung möglichst lange zu erhalten, bzw. die Änderbarkeit wiederzuerlangen. Migration bezeichnet die Überführung von Softwaresystemen in eine andere Zielumgebung oder in eine neue Form, ohne hierbei deren Funktionalität zu ändern. Die neue Zielumgebung ermöglicht dann die Weiterentwicklung des Softwaresystems und verlängert dessen Nutzungsdauer.

Prozessmodelle der Software-Migration erlauben die geplante, dokumentierte und kontrollierte Durchführung dieser Migrationsvorhaben. In den letzten Jahren wurden diverse Vorgehensmodelle zur Softwaremigration vorgestellt (vgl. [13]). Diese Vorgehensmodelle wurden aber unabhängig von Prozessmodellen zur Software-Entwicklung erstellt. Die Kombination mit iterativen und inkrementellen Prozessmodellen, wie beispielsweise dem Rational Unified Process [10], die ihrerseits kaum Unterstützung für Migrationsvorhaben umfassen [9], erfolgte bislang nicht.

Dieser Beitrag skizziert die wesentlichen Aufgabenbereiche der Software-Migration (Kapitel 2) und kombiniert diese mit iterativen Software-Entwicklungsmodellen (Kapitel 3) [1]. Ziel ist die Entwicklung eines generischen Referenz-Prozesses zur Durchführung von Migrationsvorhaben, deren Gültigkeit in realen Migrationsprojekten zu validieren ist.

2. Kernbereiche der Migration

Wichtige Grundlagen zur Herleitung eines Bezugsrahmens für die Entwicklung eines generischen Referenzprozesses zur Software-Migration stellen bereits publizierte *Prozessmodelle aus Forschung und Praxis* dar, die jeweils unterschiedliche Schwerpunkte der Software-Evolution thematisieren.

Allgemeine Beschreibungen von Migrations- bzw. Evolutionsprozessen decken einen relativ breiten Einsatzbereich ab (Chicken Little-Methode [7], Allgemeiner Migrationsprozess [16], Reengineering-Factory [6], Renaissance-Methode [19]). Weitere Prozessmodelle konzentrieren sich ausschließlich auf die *Datenmigration* (MIKADO [2], Butterfly-Methode [5]) oder auf die *Migration in objektorientierte Software* (Objektorientierte Migration [14], FAMOOS [3]). Der „Risk-Managed Modernization"-Ansatz (RMM [17], [18]) fokussiert auf die *Planungsaktivitäten* von Modernisierungsprojekten. *Vorgehensweisen* zur Modernisierung von Altsystemen, die konkrete Projekterfahrungen widerspiegeln, wurden von Softwarehäusern vorgestellt [8].

Aus diesen Prozessmodellen konnten *wesentliche Aktivitäten von Migrationsvorhaben* identifiziert werden, die sich hinsichtlich ihrer logischen Zusammengehörigkeit als *Kernbereiche* zusammenfassen lassen. Diese Kernbereiche lassen sich grundsätzlich in unterstützende und in migrationsspezifische Kernbereiche unterteilen (vgl. Abb. 1). *Unterstützende Kernbereiche* zielen auf die bereichsübergreifende Unterstützung in Migrationsprojekten (Projektmanagement, Konfigurations- und Änderungsmanagement, Entwicklungsumgebung). *Migrationsspezifische Kernbereiche* beziehen sich auf operative Migrationstätigkeiten, die zur Überführung in das Zielsystem beitragen.

Die **Anforderungs-Analyse** bezieht sich auf die Erhebung und Verwaltung aller Migrationsanforderungen. Funktionale Anforderungen – soweit sie für die Migrationsdurchführung relevant sind - werden allerdings aus dem Legacy-

System oder dem Wissen darüber extrahiert, da Migrationen die zugrundeliegende fachliche Funktionalität nicht verändern (Grundprinzip der Migration). Neben der Extraktion von Anforderungen und dem problemspezifischen Verstehen von Programmen, Schnittstellen und Daten durch Reverse-Engineering-Techniken zielt die **Legacy-Analyse/ Aufbereitung** gleichzeitig auf eine Bewertung des Wiederverwendbarkeitspotentials und die Aufbereitung im Sinne einer wiederverwendbaren und zerlegbaren Struktur.

Die Legacy-Bewertung bildet die Basis für die **Strategie-Auswahl**, die für jedes Migrationspaket festlegt, ob es durch Konversion, Wrapping oder Neuentwicklung migriert werden kann. Mit der Auswahl der Transformationsstrategie einher geht die Festlegung einer angemessenen Übergabestrategie, die sich entweder auf eine inkrementelle oder eine „Big-Bang"-Übergabe bezieht.

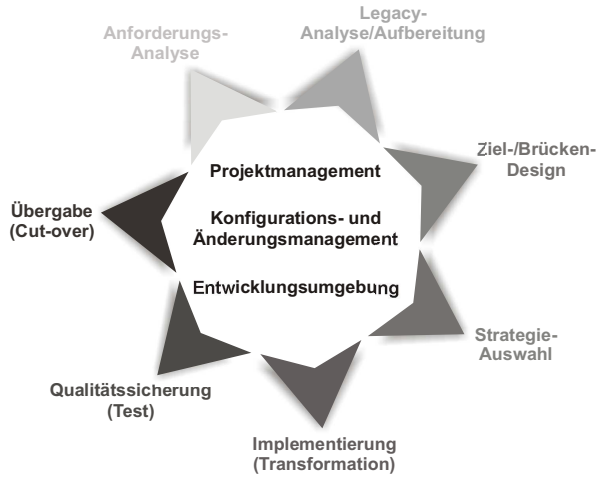

Abb. 1 Kernbereiche der Software-Migration

Die Entwicklung des **Ziel-Designs** fokussiert auf den Entwurf des Zielsystems und der Zielumgebung. Da sich während einer inkrementellen Umstellung Legacy- und Zielsystem parallel im Einsatz befinden, ist zur Sicherstellung der Interoperabilität zusätzlich der Entwurf einer Übergangsarchitektur (**Brücken-Design**) erforderlich. Ziel-Design und Strategie-Auswahl bedingen sich gegenseitig.

Die **Implementierung** bezieht sich auf die Bereitstellung von Transformationswerkzeugen und deren Anwendung zur Überführung der Legacy-Pakete (**Transformation**). Hierzu gehören auch erforderliche zusätzliche Implementierungen für Wrapper, Gateways, Neuentwicklungen, etc. Zur Überprüfung der funktionalen Äquivalenz zwischen Legacy- und Zielsystem ist die Durchführung entsprechender Regressionstests als Teil der **Qualitätssicherung** unabdingbar.

Die **Übergabe (Cut-over)** bezeichnet die konkrete Umstellung vom Legacy- auf das Zielsystem und behandelt insbesondere die Verteilung und Installation der migrierten Pakete sowie das Training der Anwender gemäß der zuvor festgelegten Übergabestrategie.

Diese Kernbereiche bilden das Grundgerüst zur Entwicklung eines generischen Migrationsprozesses. Den „Best Practices" der Softwareentwicklung folgend, wird eine iterative und inkrementelle Vorgehensweise angestrebt. Die Aktivitäten der Kernbereiche werden während des Migrationsvorhabens mit unterschiedlicher Intensität wiederholt durchlaufen und führen in jeder Iteration zu einer Ergänzung bzw. Verfeinerung der resultierenden Artefakte (inkrementell).

3. Einbettung der Kernbereiche in den Unified Process

Viele Softwareentwicklungsprozesse beschreiben Vorgehensweisen, die sich in erfolgreichen Projekten bewährt haben. Migrationsprozesse können diese Erfahrungen ausnutzen, indem sie diese Vorgehensweisen übernehmen und entsprechend adaptieren. Da die Analysen der publizierten Prozessmodelle deutliche Parallelen zwischen Migrations- und Entwicklungsprozessen aufzeigen, legen diese Gemeinsamkeiten die Integration beider Prozesstypen nahe. In einen generischen Migrationsprozess fließen so neben den migrationsspezifischen Prozesselementen typische Aktivitäten, Artefakte und Akteure der klassischen Softwareentwicklung (in entsprechend modifizierter Form) ein. Verglichen mit einer Neuentwicklung sind reine Entwicklungsaktivitäten jedoch mit deutlich geringerer Intensität zu bearbeiten [15].

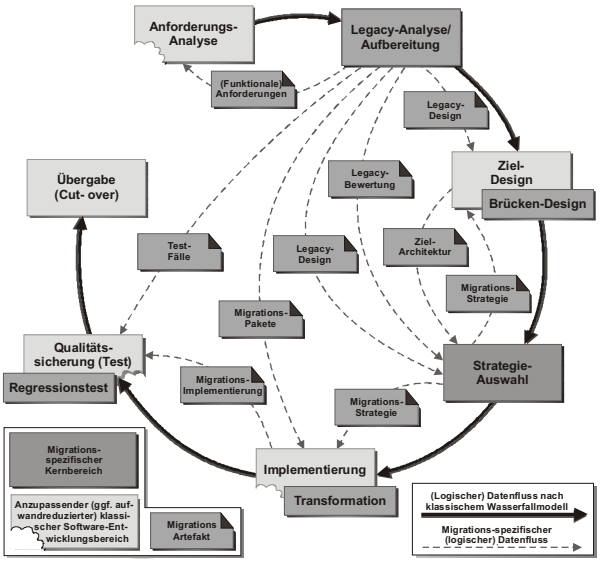

Abb. 2 Integration der Aktivitäten

Abbildung 2 skizziert die Integration des Migrationsprozesses in einen iterativen und inkrementellen Software-Entwicklungsprozess. Die Integration von Vorgehensweisen der Softwaremigration und der Softwareentwicklung erfordert die Beibehaltung gemeinsamer, die Anpassung ähnlicher, die Eliminierung irrelevanter und die Ergänzung fehlender Aktivitäten, Akteure und Artefakte. Beispielsweise werden nicht-funktionale Migrationsanforderungen partiell durch Techniken des traditionellen Anforderungsmanagements erhoben (Kernbereich Anforderungs-Analyse), wäh-

rend funktionale Anforderungen durch Techniken des Reverse Engineering (Kernbereich Legacy-Analyse) aus dem Altsystem extrahiert werden. Reverse-Engineering-Techniken werden in der Regel in traditionellen Entwicklungsprozessen nicht berücksichtigt.

Als eine geeignete Integrationsbasis bietet sich der *Rational Unified Process (RUP)* an. Er weist nicht nur eine zum entwickelten Migrationsprozess ähnliche Struktur auf, sondern unterstützt auch eine iterative und inkrementelle Vorgehensweise. Zudem bietet er durch die umfangreiche Werkzeugunterstützung (RUP Modeler, Organizer, Builder [4]) eine wertvolle Integrationsumgebung an, die eine Prozessanpassung an die jeweiligen Anforderungen eines Migrationsprozesses ermöglicht.

Als Anknüpfungspunkte zur Integration dienen semantisch äquivalente Prozesselemente. Einige Kernbereiche der Migration (Anforderungs-Analyse, Ziel-/Brücken-Design, Implementierung (Transformation), Qualitätssicherung (Test), Übergabe (Cut-over), Konfigurations-/Änderungsmanagement, Projektmanagement, Entwicklungsumgebung) können mit den Bereichen des RUP, den sogenannten Disziplinen (Anforderungen, Analyse & Design, Implementierung, Test, Übergabe, Konfigurations-/Änderungsmanagement, Projektmanagement, Umgebung), verschmolzen werden. Hierzu sind nicht erforderliche Aktivitäten, Artefakte, und Akteure des RUP auszublenden, abweichende Elemente zu modifizieren und zusätzliche Elemente hinzuzufügen. Weiterhin werden Kernbereiche, die nicht durch den RUP abgedeckt werden, neu definiert und mit den RUP-Disziplinen verknüpft (Legacy Analysis, Strategy Selection).

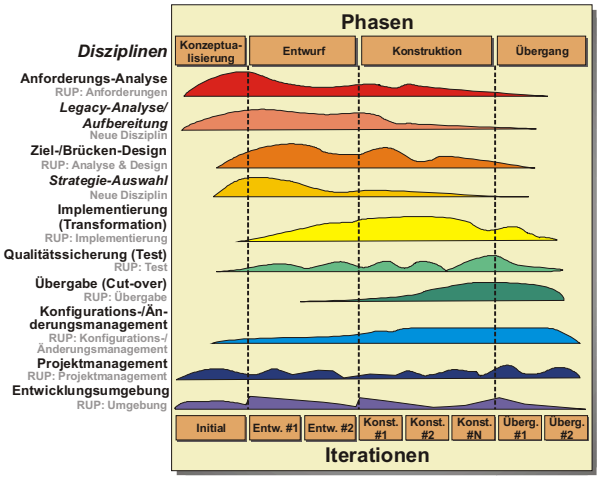

Abb. 3 Unified Migration Process

In Abbildung 3 wird die Integration des Migrationsprozesses in den RUP skizziert. Der resultierende Unified Migration Prozess (UMP) stellt einen generischen Migrationsprozess dar, der an die jeweiligen Anforderungen eines Migrationsvorhabens anpassbar ist.

4. Zusammenfassung und Ausblick

Die Migration von Softwaresystemen ist eine Disziplin der Softwaretechnik, deren Beherrschung und industrielle Nutzung nach einem adäquaten Prozessmodell verlangt. Dieses Papier skizziert einen generischen Migrationsprozess, der relevante Aktivitäten, Akteure und Artefakte der Migrationsprozesse in das iterative und inkrementelle Vorgehensmodell des Rational Unified Processes einbettet. Das vorgestellte Referenzprozessmodell soll in weiteren Arbeiten entlang der auf RePro 2005 präsentierten Fallbeispiele validiert werden.

Literatur

[1] E. Ackermann: Ein Referenz-Prozessmodell zur Migration von Softwaresystemen, Diplomarbeit, Universität Koblenz, erscheint 2005.

[2] D. Aebi: Re-Enginerring und Migration betrieblicher Nutzdaten. Dissertationsschrift, Universität Zürich, Schweiz, 1996.

[3] H. Bär, M. Bauer, O. Ciupke, S. Demeyer, S. Ducasse, M. Lanza, R. Marinescu, R. Nebbe, O. Nierstrasz, M. Przybilski, T. Richner, M. Rieger, C. Riva, A. Sassen, B. Schulz, P. Steyaert, S. Tichelaar, J. Weisbrod: The FAMOOS Object-Oriented Reengineering Handbook. 1999. www.iam.unibe.ch/~famoos /handbook.

[4] A. Bencomo: Extending the RUP, Process Modeling. http://www-128.ibm.com/developerworks/rational/library/05/323_extrup1

[5] J. Bisbal, J. Grimson, D. Lawless, B. Wu.: Legacy Information Systems: Issues and Directions. IEEE Software, Vol. 16, No. 5, 103-111, 1999.

[6] J. Borchers, K. Hildebrand: Vorgehensmodell für das Software Reengineering. In: Kneuper, R.; Müller-Luschnat, G.; Oberweis, A. (Hrsg.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung. Teubner, Stuttgart/Leipzig, 1998.

[7] M.L. Brodie, M. Stonebraker: Migrating Legacy Systems. Morgan Kaufmann, San Francisco, California, 1995.

[8] collogia AG: Reengineering - Modernisierung von Altsystemen. Köln. Version 5.4. http://www.collogia.de/95.0.html.

[9] R. Gimnich, A. Winter: Workflows der Software-Migration, Softwaretechnik-Trends, (25)2, Mai 2005, 22-24.

[10] P. Kruchten: The Rational Unified Process, An Introduction. Addison-Wesley, Upper Saddle River, 3rd edition, 2004.

[11] U. Kuhlmann, A. Winter: Softwarewartung und Prozessmodelle in Theorie und Praxis, Softwaretechnik-Trends, (24)2, Mai, 2004, 37-38.

[12] V. Rajlich, K. Bennett: A Staged Model for the Software Lifecycle, IEEE Computer}, July 2000, 66-71.

[13] U. Kuhlmann, H. Sneed, A. Winter: Workshop Reengineering Prozesse, Fallstudien, Methoden, Vorgehen, Werkzeuge. Fachberichte Informatik, 11/2004, Universität Koblenz-Landau, 2004.

[14] H. Sneed: Objektorientierte Softwaremigration, Addison-Wesley, Bonn, 1999.

[15] H. Sneed H.: Aufwandsschätzungen von Software-Reengineering-Projekten. Wirtschaftsinformatik 45(6):599–610

[16] H. Sneed, M. Hasitschka, M. Teichmann: Software-Produktmanagement, Wartung und Weiterentwicklung bestehender Anwendungssysteme. Dpunkt Verlag, Heidelberg, 2004.

[17] R. Seacord, D. Plakosh, G. Lewis: Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices. The SEI Series in Software Engineering. Addison-Wesley, Boston, 2003.

[18] J. Bergey, L. O'Brian, D. Smith: DoD Software Migration Planning. Technical Report CMU/SEI-2001-TN-012, Carnegie Mellon University, Pittsburgh, 2001.

[19] I. Warren: The Renaissance of Legacy Systems: Method Support for Software-System Evolution. Springer, London, 1999