# FMC als Mittel der Architekturfindung im Reengineering

Andreas Knöpfel, Peter Tabeling
Hasso–Plattner–Institute for IT–Systems Engineering
{knoepfel, tabeling}@hpi.uni-potsdam.de

25. April 2005

## Zusammenfassung

Die Identifikation der zentralen Systemstrukturen ist unverzichtbar für das Reengineering komplexer, über lange Zeit gewachsener Software–Systeme. Abgleich und Integration von Aussagen über das System aus unterschiedlichen Informationsquellen sind hierzu unverzichtbar. Durch Anwendung des Modellierungsansatzes FMC lassen sich in diesem Kontext üblicherweise auftretende Verständnis- und Kommunikationsprobleme effektiv reduzieren.

## 1 Einleitung

Komplexe informationelle System entstehen üblicherweise durch wiederholte Restrukturierung, Erweiterung und Integration bestehender Systeme. Die Überführung eines bestehenden Ursprungssystems in das gewünschte Zielsystem, ist Aufgabe des Reengineerings. Grundsätzlich lassen sich dabei folgende Ansätze unterscheiden:

1. Rekonfiguration durch Austausch oder Entfernen bestehender Einzelkomponenten, sowie durch die Integration neuer Komponenten
2. Änderung bzw. Erweiterung der Parametrisierung/Programmierung bestehender Komponenten
3. Migration des Systems oder von Teilen des Systems durch Extraktion von Konzepten und deren Neuimplementierung

Daher ist es zunächst erforderlich, eine präzise Vorstellung von der Architektur des Ursprungssystem zu entwickeln (Reverse–Engineering). Von diesem Modell ausgehend lassen sich dann – idealerweise in Zusammenarbeit von Kunden, Anwendern, Analysten und Systemarchitekten – Anforderungen und Lösungsansätze für das Zielsystem formulieren. Die effektive und effiziente Kommunikation zwischen den Beteiligten kann dabei den Erfolg des Vorhabens maßgeblich beeinflussen.

## 2 Problem der Architekturfindung

In der Analyse bestehender Systeme kann man prinzipiell zwei Ansätze der Architekturfindung unterscheiden [5]: Beim *Top–Down–Ansatz* findet ausgehend vom Wissen um die Anwendung des Systems – gewonnen durch Interviews, Sichtung von Dokumentation und Analyse des Systembetriebes – eine schrittweise verfeinernde Abbildung auf zuständige Systemkomponenten statt. Dem gegenüber steht der *Bottom–Up–Ansatz*. Hierbei wird ausgehend vom Quellcode versucht, durch geeignete Abstraktionen eine Abbildung auf die Konzepte des Anwendungsgebietes zu erreichen, um so die Architektur des Systems zu rekonstruieren.

In der Praxis zeigt sich, dass im allgemeinen eine Kombination beider Ansätze erforderlich ist [3]. So ermöglichen Werkzeuge eine automatisierte Analyse der Software–Strukturen um im Bottom–Up–Ansatz einen ersten Überblick über die Implementierung zu erhalten. Allerdings sind wesentliche Informationen, die für ein Verständnis des Systems erforderlich sind, im allgemeinen nicht Bestandteil der Software. Dazu gehört das Wissen über die Nutzung des Systems durch die Umgebung, sowie wie die Gründe, die im Falle von Design–Alternativen zu einer bestimmten Entscheidung geführt haben. Auf der anderen Seite bleiben bei strikter Anwendung des Top–Down–Ansatzes ungenutzte, aber möglicherweise zukünftig relevante Systemkomponenten und deren Eigenschaften unentdeckt.

Ziel ist es, die Ergebnisse beider Ansätze miteinander in Einklang zu bringen und zu integrieren. Das Problem besteht in der Heterogenität der einfließenden Informationsquellen und ihrer Darstellung: Von Haus aus verwenden Anwender, Analysten, Designer und Entwickler meist eine unterschiedliche Terminologie und Notation, in der eine unterschiedliche Sicht auf das System zum Ausdruck kommt.

Unverzichtbar ist daher ein übergreifend gültiges begriffliches Fundament, auf dem eine gemeinsame Systemvorstellung aufgebaut werden kann. Dazu gehören nachvollziehbare, leistungsfähige Konzepte zur Abstraktion sowie eine leicht verständliche, präzise und anwendungsunabhängige Notation.

Der Einsatz objekt–orientierter Methoden hat sich dabei in der Praxis auf der Ebene konzeptioneller Architekturmodelle nur als bedingt geeignet erwiesen [4, 1]. So erscheinen Konzepte wie die Identifikation funktionaler Zuständigkeiten für eine erste Verständnisbildung wichtiger als die Klassifizie-

rung von Objekten und Generalisierung. Die in der Praxis häufig unzureichende Trennung in der Darstellung von von Beschreibungs- und Systemstrukturen, von Objekt- und Typbeziehungen erschwert das Verständnis zusätzlich [2].

## 3 Der FMC–Ansatz

FMC steht als Abkürzung für *Fundamental Modeling Concepts* und ist primär ein Ansatz, der das Denken und Kommunizieren über komplexe informationelle Systeme auf der Ebene von Architekturmodellen vereinfacht[6]. Im Kern steht ein kompaktes Metamodell, dessen Grundgedanke die konsequente Trennung von drei Kategorien unterschiedlicher Systemstrukturen darstellt: Aufbau, Ablauf und Wertestruktur.

**Aufbaustrukturen** Jedes Systemmodell ist durch eine Aufbaustruktur charakterisiert, nach der man sich das System als Gebilde miteinander interagierender Systemkomponenten vorstellen kann. Von zentraler Bedeutung ist die Unterscheidung in aktive Komponenten, die Akteure genannt werden und passive Komponenten, die Speicher oder Kanäle darstellen. Akteure können miteinander kommunizieren, indem sie über Kanäle Informationen (Werte) austauschen. Speicher dienen der Ablage von Informationen oder als Aktionsfelder für Operationen auf Daten.

**Verhaltensstrukturen** Prinzipiell können Akteure zueinander nebenläufig agieren. Ihr tatsächliches Verhalten hängt vom jeweiligen Akteurstyp und der festgelegten Aufbaustruktur ab. Mit diesem Wissen lässt sich die Kausalstruktur bestimmen.

**Wertebereichsstruktur** Das Repertoire möglicher Werte, das auf den unterschiedlichen Kanälen und Speichern im System beobachtbar ist, wird durch die Wertebereichsstruktur definiert.

Jedes FMC–Modell legt eine bestimmte Aufbaustruktur, eine Verhaltensstruktur und eine Wertebereichsstruktur fest. Eine präzise, kompakte, grafische Notation, die an etablierten Standards orientiert ist, unterstützt die Kommunikation. *Blockdiagramme* dienen zur Darstellung von Aufbaustrukturen, *Petri-Netze* zur Darstellung von Kausalstrukturen und *Entity-Relationship-Diagramme* zur Darstellung von Wertebereichsstrukturen.

Der Grundgedanke, der die Architekturfindung mit FMC leitet, ist die Suche nach Zuständigkeiten und die klare Trennung von Betrachtungsebenen. Wer tut was, wann und womit? Welche Komponenten sind auf Ebene des Anwendungssystems sichtbar, welche auf Ebene der Plattform?

Im Falle komplexer Systeme wird man üblicherweise mehrere Modelle erstellen, die unterschiedliche Aspekte des Systems in unterschiedlicher Auflösung zeigen. Systemkomponenten können durch konkrete Objekte, wie Benutzer oder PCs, repräsentiert sein. In anderen Fällen ist die Vorstellung von abstrakten Dienstleistern, Kanälen oder Speichern, deren Abbildung auf konkrete Implementierungsstrukturen nicht trivial ist, angemessener. So kann es zweckmäßig sein, ein ganzes ERP–System als Speicher für betriebswirtschaftliche Daten anzusehen. Abstraktionen dieser Art ermöglichen die nahtlose Modellierung heterogener Systeme in ihrer Umgebung, was das Verständnis komplexer Systeme entscheidend erleichtert.

## 4 Erfahrungen und Ausblick

In FMC sind praktische Erfahrungen aus aus mehr als zwei Jahrzehnten eingeflossen. In zahlreichen Projekten großer Firmen wie SAP, Siemens, ALCATEL oder BMW hat FMC dazu beitragen das Verständnis für das jeweilige System zu schärfen [4]. Dazu gehört die erfolgreiche Portierung des R/3–Systems der SAP auf die AS/400 von IBM, die maßgeblich auf den Einsatz von FMC–Modellen gestützt war.

Als Hilfsmittel zur Architekturfindung und Kommunikation entwickelt, ist FMC aus unserer Sicht ein ideales Werkzeug entsprechende Aufgaben innerhalb des Reengineering informationeller Systeme zu unterstützen.

## Literatur


[1] Robert L. Glass. The Naturalness of Object Orientation: Beating a Dead Horse? *IEEE Software*, 19(3):104, 103, May/June 2002.

[2] Bernhard Gröne, Andreas Knöpfel, and Peter Tabeling. Component vs. Component: Why We Need More Than One Definition. In *Proceedings of ECBS2005, Model-Based Development of Computer Based Systems Workshop*, April 2005.

[3] Rick Kazman, Liam O´Brien, and Chris Verhoef. Architecture Reconstruction Guidelines, Third Edition CMU/SEI-2002-TR-034, ESC-TR-2002-034. Technical report, Carnegie Mellon, Software Engineering Institute, November 2003.

[4] Frank Keller. Über die Rolle von Architekturbeschreibungen im Software-Entwicklungsprozess. Phd. theses, Hasso-Plattner Institute for IT-Systems Engineering, 2003.

[5] Scott Tilley. A Reverse-Engineering Environment Framework, CMU/SEI-98-TR-005 ESC-TR-98-005. Technical report, Carnegie Mellon University, Software Engineering Institute, April 1998.

[6] Siegfried Wendt and Frank Keller. FMC: An Approach Towards Architecture-Centric System Development. In *Proceedings of 10th IEEE Symposium and Workshops on Engineering of Computer Based Systems, Huntsville Alabama USA (2003)*, pages 173–182, April 2003.