Die Rolle von Requirements Engineering und IT-Architektur bei der Entwicklung betrieblicher Informationssysteme

Helmut Duschinger

BeOne München GmbH
Emmy-Noether-Str. 4
80992 München
helmut.duschinger@beone-group.com

Abstract: Betriebliche Informationssysteme werden in Projekten fachlich und technisch konzipiert und dann umgesetzt. Requirements Engineering ist dabei eine Basis-Kompetenz, die nicht nur in frühen Phasen, sondern an allen Auftraggeber- / Auftragnehmer-Schnittstellen der Projektorganisation zum Einsatz kommt. IT-Architekturen bilden Strukturen, die Anforderungen erfüllen und sind Erklärungsmodelle, um komplexe Systeme verstehen zu können. Am Beispiel der fachlichen Architektur werden Wechselwirkungen zum Requirements Engineering aufgezeigt. Analysen bis zur inhaltlichen Essenz und sprachliche Exaktheit können vom Requirements Engineering auf die Architektur-Entwicklung übertragen werden.

1 Schritte bei der Entwicklung betrieblicher Informationssysteme

Betriebliche Informationssysteme für die Kernprozesse von Unternehmen sind komplex und langlebig, ihre Erstellung ist anspruchsvoll. Im Folgenden werden Requirements Engineering und IT-Architektur vor dem Hintergrund umfangreicher Erfahrung mit der Entwicklung betrieblicher Informationssysteme als Individualsoftware betrachtet.

Betriebliche Informationssysteme werden in Projekten mit vielen – teils parallel, teils sequentiell ablaufenden – Einzelschritten erstellt. Aus einer Vielzahl von Vorgehensmodellen, wie z. B. V-Modell oder Unified Process kann jedes Projekt einen passenden Projektablauf ableiten. Zur Einordnung der Rollen von Requirements Engineering und IT-Architektur wird von nachfolgendem grundlegenden Ablauf bei der Software-Erstellung ausgegangen (Abbildung 1). Im Weiteren spielt es keine Rolle, ob die Schritte in einem Projekt sequentiell oder iterativ, einmal oder mehrfach durchlaufen werden.

Der Projektauftrag definiert, welches betriebliche Informationssystem gebaut werden soll, welche Rahmenbedingungen eingehalten werden müssen und welche Mittel zur Verfügung stehen. Mit ihm werden die Anforderungen an das Projekt definiert, welches das System erstellen wird. In der Regel liegen die Anforderungen erst in sehr grober Form vor und werden später durch Methoden des Requirements Engineering verfeinert.

Im Rahmen der fachlichen Konzeption wird mit den Fachleuten des Unternehmens die vom System zu unterstützende fachliche Aufgabenstellung geklärt. Dazu werden die fachlichen Anforderungen zunächst möglichst lösungsneutral betrachtet. Damit ein IT-System programmiert werden kann, wird dieses – als Lösung der Aufgabenstellung – in einem Fachkonzept fachlich gestaltet, und z. B. in Form von Use Cases, Dialogbeschreibungen und Druckausgaben präzise beschrieben. Bei dieser fachlichen Gestaltung ergeben sich nicht selten Rückwirkungen auf die Aufgabenstellung, die auf Grund neuer Möglichkeiten und Sichtweisen modifiziert werden kann.

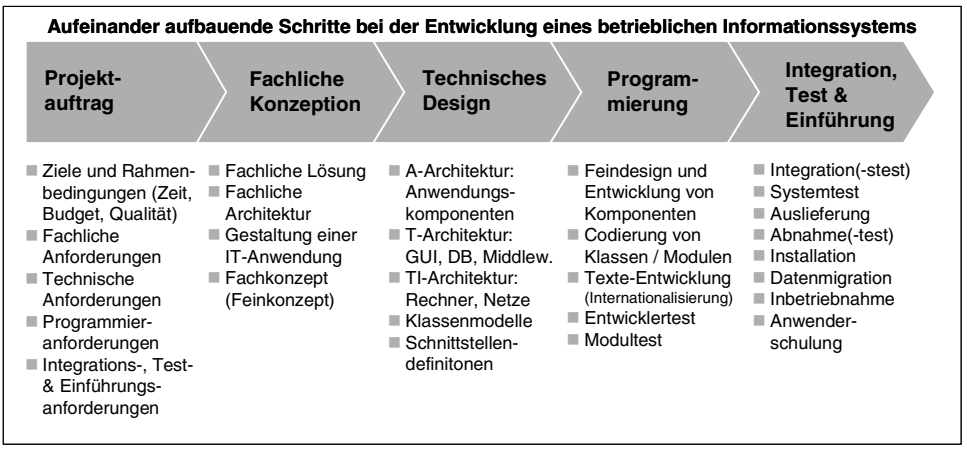

Abbildung 1: Schritte bei der Entwicklung eines betrieblichen Informationssystems

Im technischen Design wird das Fachkonzept unter Berücksichtigung weiterer Rahmenbedingungen des Projektauftrags in die Strukturen einer Software umgesetzt. Dabei werden Anwendungskomponenten modelliert, technische Datenhaltungen und der Zugriff darauf konzipiert, die Einbettung in die Ablaufumgebung eines Application Servers und die Versorgung weiterer technischer Schnittstellen festgelegt.

Bei der Programmierung werden die fachlich und technisch konzipierten Abläufe und Datenstrukturen mit der gewählten Programmiersprache auf Basis der vorhandenen Systemsoftware vollständig in allen Aspekten und mit formaler Präzision ausformuliert und durch den Entwickler komponentenweise getestet.

Bei der Integration, dem Test und der Einführung werden alle Komponenten des Systems zum Gesamtsystem integriert und im Zusammenspiel getestet. Mit System- und Abnahmetest wird überprüft, ob alle Anforderungen an das System erfüllt werden, bevor es schließlich in Betrieb genommen wird.

Die aufgezeigte Schrittfolge ist insbesondere auch für serviceorientierte Architekturen geeignet, bei denen Services erst fachlich modelliert und dann auf IT-Komponenten abgebildet werden.

2 Requirements Engineering an den Schnittstellen zwischen Auftraggeber und Auftragnehmer

Nach [Ru04] ist eine Anforderung eine Aussage über eine Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen. Gute Anforderungen sind vollständig, korrekt, konsistent, prüfbar, eindeutig, verstehbar, gültig und aktuell. Anforderungsdokumente unterscheiden sich von anderen Konzeptdokumenten u. a. dadurch, dass die Anzahl der im Dokument formulierten Anforderungen bekannt ist und zu jeder genannten Anforderung klar ist, wie ihre Erfüllung geprüft wird.

Deshalb sind Anforderungen in einem Projekt vor allem an den Schnittstellen von Auftraggeber/Auftragnehmer-Beziehungen interessant. Der Auftraggeber teilt dem Auftragnehmer seine Anforderungen mit und prüft nach Lieferung der Ergebnisse, ob alle seine Anforderungen erfüllt sind. Für diesen Prüfschritt – die Abnahme der Ergebnisse – ist also Voraussetzung, dass Anzahl und Prüfkriterien der Anforderungen beiden Seiten bekannt sind. Typisch ist, dass Anforderungen zunächst nur grob vom Auftraggeber vor Auftragserteilung formuliert werden, und dass der Auftragnehmer eine Verfeinerung und Vervollständigung der Anforderungsdokumentation vornehmen und mit dem Auftraggeber abstimmen wird. Requirements Engineering ist also eine Tätigkeit an der Schnittstelle Auftraggeber/Auftragnehmer, die von beiden Seiten ausgeführt wird.

Eine immer vorhandene und wichtige Auftraggeber/Auftragnehmer-Schnittstelle besteht zwischen Projektsponsor und Projekt und wird durch den Projektauftrag gebildet. Deshalb wird häufig eine frühe Projektphase die Requirementsphase genannt (z. B. im Unified Process). Je nach Projekorganisation und Sourcing-Strategien kann es aber mehrere Auftraggeber/Auftragnehmer-Beziehungen geben, wobei kein grundsätzlicher Unterschied zwischen internen und externen Auftragnehmern besteht. Wird als Beispiel für eine späte Projektphase der Softwaretest an einen Dienstleister vergeben, so sind an dieser Schnittstelle Anforderungen an den Auftragnehmer zu definieren und nach Erbringung der Leistung zu prüfen. Dabei wird man in diesem Fall einen Teil der bereits erhobenen Anforderungen an die Software für den Test durchreichen und weitere, für diese Auftraggeber/Auftragnehmer-Beziehung spezifische (Prozeß-)Anforderungen ergänzen.

Damit kann Requirements Engineering nicht nur als bestimmte Phase im Projektverlauf der Software-Entwicklung gesehen werden, sondern als Basiskompetenz, die an allen in der Projektorganisation vorgesehenen Schnittstellen zwischen einem Auftraggeber und einem Auftragnehmer zum Tragen kommt. Requirements Engineering legt die Grundlage, dass die richtigen Ergebnisse geliefert werden. Je nach Art der gewünschten Ergebnisse werden Lösungskompetenzen benötigt, diese richtig zu erstellen [FHW07]. Geht es um die Erstellung von Software, so gehört die Fähigkeit, IT-Architekturen zu entwickeln, zu diesen Lösungskompetenzen.

3 Architekturen betrieblicher Informationssysteme

Definitionen für IT-Architekturen gibt es viele. Die Definition „Unter der IT-Architektur wird die Gesamtheit aus verwendeter Hard- und Software, den Geschäftsprozessen und der Organisationsstruktur eines Unternehmens verstanden.“ ist breit gefasst. Im Folgenden werden fokussiertere, für betriebliche Informationssysteme bewährte und auf den Zweck der Software-Erstellung ausgerichtete Architekturbegriffe verwendet:

Nach [BCK03] ist eine Architektur eine Struktur von Elementen mit ihren wesentlichen Eigenschaften und den Beziehungen zwischen den Elementen. Diese Definition kann bei der Entwicklung betrieblicher Informationssysteme auf unterschiedliche Gegenstandsbereiche angewendet werden.

Angewandt auf die fachliche Aufgabenstellung kann – unabhängig von der konkreten Lösung durch ein IT-System – eine Geschäftsarchitektur entwickelt werden. Sie strukturiert die Aufgabenstellung lösungsneutral und dient als Erklärungsmodell für die Anforderungsstruktur.

Angewandt auf ein IT-System, das die fachliche Aufgabenstellung löst, kann die fachliche Architektur (wie funktioniert die Lösung fachlich?) und die Systemarchitektur (wie funktioniert die Lösung softwaretechnisch?) unterschieden werden (Abbildung 2). Für die Systemarchitektur können wiederum drei Sichten unterschieden werden [Sie04]: die A-Architektur (Struktur der Anwendungskomponenten), die T-Architektur (Struktur der techniknahen Komponenten) und die TI-Architektur (Struktur der technischen Infrastruktur).

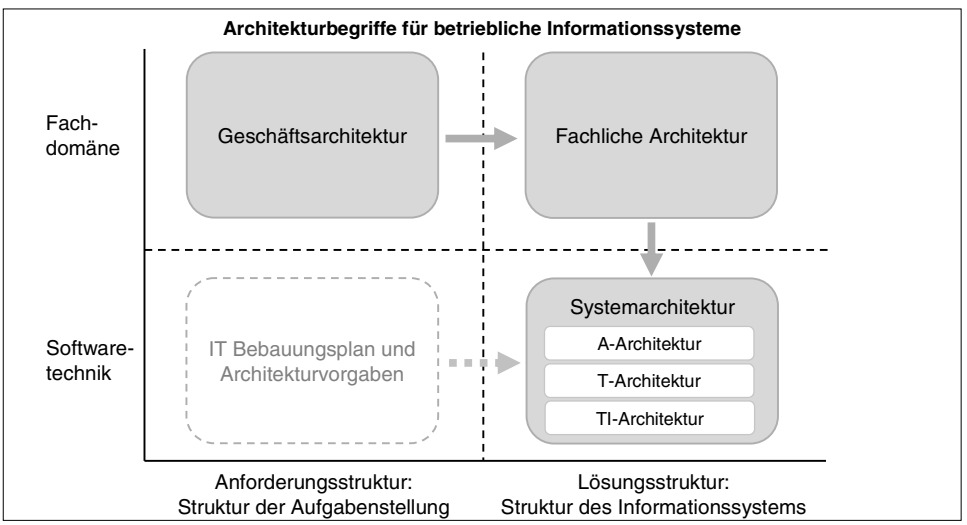

Abbildung 2: Systematik der Architekturbegriffe

In Projekten helfen Architekturen, Themen zu strukturieren; sie unterstützen als Erklärungsmodelle die Kommunikation im Projekt und machen Komplexität beherrschbar:

- Die Geschäftsarchitektur strukturiert die Aufgabenstellung. Sie ist geeignet zur Strukturierung der fachlichen Anforderungen des Projektauftrags und hilft, die Aufgabenstellung zu erklären.

- Die fachliche Architektur strukturiert die Lösung der Aufgabenstellung aus fachlicher Sicht, indem Abläufe und wesentliche Informationsobjekte grundsätzlich festgelegt und Verarbeitungen strukturiert werden. Sie hilft, die Erfüllung fachlicher Anforderungen auf hohem Abstraktionsniveau zu erklären.

- Die A-Architektur detailliert die fachliche Architektur, indem sie auf konkret in der Software umgesetzte Anwendungskomponenten verfeinert wird und deren Schnittstellen fachlich festgelegt werden. Die A-Architektur hilft, die Umsetzung fachlicher Anforderungen anhand konkreter Software-Artefakte (z. B. Komponenten und Klassen) detailliert zu erklären.

- Die T-Architektur stellt technische Komponenten und deren Schnittstellen dar, die für Komponenten der A-Architektur höherwertige technische Dienste bereitstellen, z. B. indem sie die Ansteuerung technischer Schnittstellen vereinfachen. Die T-Architektur berücksichtigt fachliche Anforderungen und hilft, die Erfüllung technischer Anforderungen zu erklären.

- Die TI-Architektur stellt die technische Ablaufumgebung – bestehend aus Hardware-, Netz- und Systemsoftware-Komponenten – für das System dar.

Architekturen sind Modelle, die jeweils für einen bestimmten Zweck einen relevanten Ausschnitt eines Systems darstellen. Architektur-Entwicklung ist – wie Requirements Engineering – eine Basis-Kompetenz. Sie bedeutet die Fähigkeit, Strukturen entwickeln zu können, die allen Anforderungen gerecht werden und diese zweckdienlich dokumentieren und kommunizieren zu können. Auch im Requirements Engineering kann Architektur-Entwicklung genutzt werden, z. B. um Anforderungen anhand einer Geschäftsarchitektur zu strukturieren.

4 Beispiel: Fachliche Architektur und Requirements

Jedes IT-System funktioniert nach dem Schema Eingabe – Verarbeitung – Ausgabe. Damit die Verarbeitung programmiert werden kann, muss vollständig verstanden sein, wie die gewünschte Ausgabe abhängig von fachlichen Einflussfaktoren entsteht. Gegenseitige Wechselwirkungen zwischen den Einflussfaktoren verhindern, dass der Beitrag einzelner Einflussfaktoren zur Ausgabe isoliert betrachtet werden kann. Die fachliche Komplexität des IT-Systems ist also umso höher, je mehr Einflussfaktoren und Prinzipien zu berücksichtigen sind und je stärker diese in gegenseitiger Wechselwirkung stehen.

Anforderungen an Eingaben, Verarbeitungen und Ausgaben werden durch Techniken des Requirements Engineering ermittelt und dokumentiert. Bei der fachlichen Gestaltung eines IT-Systems gibt es verschiedene Möglichkeiten der Umsetzung, die u. a. durch den Zuschnitt von Use Cases und fachlichen Komponenten festgelegt werden. Durch den gewählten Zuschnitt der fachlichen Lösung ergeben sich in der Regel weitere Fragestellungen, die durch Detaillierung und Präzisierung der Anforderungen zu klären sind.

Auf Basis der fortschreitenden Klärung der Anforderungen und einer gewählten Gestaltung und Strukturierung des IT-Systems wird in der Regel eine Feinspezifikation erstellt – das Fachkonzept – das meist u. a. Use Cases, ein fachliches Datenmodell sowie Dialog- und Druckspezifikationen enthält. Nicht selten entsteht durch den Umfang solcher Feinkonzepte (mehrere hundert bis tausende Seiten) ein Problem mit der Übersicht und der Konsistenzsicherung: Wie stellt man sicher, dass beispielsweise 60 Use Cases, 120 Entitäten im Datenmodell, 40 Dialoge und weitere Detailbeschreibungen konsistent die Anforderungen erfüllen?

Zur Beherrschung der Komplexität wird die Entwicklung und Dokumentation einer fachlichen Architektur mit folgenden Inhalten als Bestandteil des Fachkonzepts empfohlen:

- Visualisierung der fachlichen Funktionsweise, also der Struktur der grundsätzlichen Lösungsidee, die alle Einflussfaktoren und Prinzipien berücksichtigt (bewährt haben sich anschauliche, plakative Grafiken fachlicher Konzepte),
- Übersicht über die zur Umsetzung der Lösungsidee bestimmten Prozesse,
- Übersicht über die zur Unterstützung dieser Prozesse bestimmten fachlichen Komponenten des IT-Systems,
- Übersicht über die für die Einbettung der Anwendung in das Unternehmen betroffenen Schnittstellen aus fachlicher Sicht.

Die so definierte fachliche Architektur ist ein Erklärungsmodell, an dem das System in seiner gesamten Funktionsweise auf der Ebene von Überblicksdarstellungen vollständig und präzise erklärt werden kann. Vollständig bezieht sich dabei auf den Funktionsumfang des Systems, nicht auf die Detailtiefe der Erklärung.

Zudem gibt die fachliche Architektur die Strukturen vor, in die alle weiteren Inhalte der Feinspezifikation top-down eingeordnet werden können. Beispielsweise können Use Cases als Detaillierung der systemunterstützten Aktivitäten der Geschäftsprozesse verstanden werden. Wenn die Gliederung der Use Case Dokumentation der Gliederung der Prozessübersicht folgt, findet man schnell von der Übersicht zum Detail.

Auf Basis dieser Definition von fachlicher Architektur und der damit verbundenen Forderung, dass die umfangreichen Inhalte des Fachkonzepts in den Strukturen dieser Architektur beschrieben werden, wird die Prüfung des Fachkonzepts gegen die Anforderungen deutlich erleichtert: mittels Szenarien können alle Abläufe an Überblicksdarstellungen nachvollzogen und anschaulich gegen die Anforderungen geprüft werden.

5 Anwendung von RE-Techniken auf die Architektur-Entwicklung

Requirements Engineering ist eine analytische Tätigkeit, bei der die Anforderungen ermittelt, dokumentiert und abgestimmt werden. Die daran anschließende Software-Entwicklung ist eine konstruktive Tätigkeit, bei der das IT-System fachlich gestaltet und konzipiert, technisch entworfen und schließlich umgesetzt wird. Erst das Ergebnis der Umsetzung kann belastbar gegen die Anforderungen getestet werden. Weil die Umsetzung mit Programmiersprachen formale Genauigkeit und Vollständigkeit erfordert, ist sie detailreich und komplex. Damit ein IT-System verstanden und erklärt werden kann, sind zweckgerichtete Architekturen notwendig, die die Struktur der Umsetzung vorausdenken, dokumentieren und als Erklärungsmodell geeignet sind. Vielfältige Schnittstellen und Wechselwirkungen zwischen Requirements Engineering und IT-Architektur wurden dargestellt.

Darüberhinaus können weitere, im Requirements Engineering etablierte Zielsetzungen und Techniken gewinnbringend auf die Architektur-Entwicklung übertragen werden. Zum Beispiel:

- Sprachliche Genauigkeit: in [Ru04] werden Techniken erläutert, wie natürlichsprachliche Aussagen zur geforderten Genauigkeit und Aussagekraft von Anforderungen umgeformt werden können. Die sprachliche Methode hilft auch konstruktiv bei der fachlichen Gestaltung des zu entwickelnden Systems, wie [Or97] bereits gezeigt hat.

- Analyse bis zur Essenz: Anforderungsanalyse begnügt sich nicht mit der Aufzeichnung der von den Stakeholdern zuerst genannten Anforderungen. Vielmehr werden zugrundeliegende Ziele, Zwecke und Absichten erforscht und daraus essentielle Anforderungen entwickelt. Dieser Ansatz kann auf die Architektur-Entwicklung übertragen werden, als Konstruktion aus der Essenz: eine tragfähige und langfristig stabile Architektur entsteht, wenn sie die fachlichen Einflussfaktoren und Prinzipien in ihrer tatsächlichen Komplexität erfasst.

Literaturverzeichnis

[BCK03] Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley, Boston, 2003

[FHW07] Fahney, R., Herrmann, A., Weissbach, R.: Wie viel Requirements Engineering steckt im Software Engineering, Beitrag zum Workshop anlässlich der SE2007 Conference on Software Engineering, Hamburg, 2007

[Or97] Ortner, E.: Methodenneutraler Fachentwurf. Zu den Grundlagen einer anwendungsorientierten Informatik. B. G. Teubner Verlag, Stuttgart, Leipzig, 1997

[Ru04] Rupp, C.: Requirements-Engineering und -Management. Professionelle, iterative Anforderungsanalyse für die Praxis. Carl Hanser Verlag, München, Wien, 2004.

[Si04] Siedersleben, J.: Moderne Softwarearchitektur. Umsichtig planen, robust bauen mit Quasar. dpunkt Verlag, Heidelberg, 2004