

Requirements Engineering und die UML

Mit Modellen zu stabilen Anforderungen

Anforderungen sind der Schlüssel zum Erfolg von IT-Projekten. Das hat sich längst herumgesprochen. Dennoch scheitern immer wieder Projekte an unzureichenden Anforderungen. Und das, obwohl mit der UML eine fundierte Technik eingesetzt wird. Woran liegt's? Oft an zwei verbreiteten Missverständnissen. Sie lauten: Die Anforderungen werden zu Beginn eines Projekts entwickelt und dann abgearbeitet. Und: Die Use Cases der UML sind die Anforderungen.

Wenn Zusatzarbeiten in IT-Projekten nötig werden, hat das fast immer dieselbe Ursache: schlechte, unvollständige oder instabile Anforderungen. Man kann es in Zahlen fassen: 70 bis 85% der Kosten für unvorhergesehene Zusatzarbeiten werden durch unzureichende Anforderungen hervorgerufen – Kosten, die 30% und mehr der gesamten Entwicklungskosten erreichen, können schnell den Rahmen des Budgets sprengen. Warum haben so viele IT-Projekte Probleme mit den Anforderungen? Die Antwort klingt beinahe zu einfach: Das Requirements Engineering, also die Entwicklung von Anforderungen, ist eine der schwierigsten Aufgaben bei der Software-Entwicklung. Und leider auch

eine mit weit reichenden Konsequenzen. Gehen wir dem Problem auf den Grund: Was ist eine Anforderung, wie sollten Anforderungen aussehen und wie gestaltet man einen Prozess der zu guten Anforderungen und anforderungsgerechten Systemen führt?

Eine Anforderung ist eine Eigenschaft, die ein IT-System besitzen, oder ein Verhalten, das es zeigen soll. Systemeigenschaften oder Systemverhalten zu Beginn eines Projekts so vollständig zu beschreiben, dass die Anforderungen „gebaut“ werden können, ist kaum möglich. Dazu fehlen Verständnis, Wissen und Entscheidungen, die erst bei der Modellierung des Systems erarbeitet werden. Daraus folgt: Requirements En-

gineering ist ein Prozess, der im Wechselspiel mit der Systemmodellierung ablaufen sollte. Der Prozess des Requirements Engineering lässt sich also nicht auf eine Zeitspanne zu Projektbeginn begrenzen. Er findet auf allen Detaillierungsniveaus des Systementwurfprozesses statt.

Von der Anforderung – zum Modell – zur Anforderung

Requirements Engineering beginnt mit dem Festhalten der allgemeinen Statements of Need der Auftraggeber und Stakeholder eines Projekts. Die Statements of Need leiten sich aus Unternehmenszielen ab und werden textuell dokumentiert. Sie müssen im nächsten Schritt zu Anwenderanforderungen an das neue IT-System konkretisiert werden. Dabei kommen die Use Cases der UML ins Spiel: Ein Use Case beschreibt eine Folge von Aktionen, die ein System ausführen kann und die ein wahrnehmbares Ergebnis liefern oder einen Nutzen schaffen.

Use Cases stellen eine spezielle Sicht auf das System dar – die Sicht der Anwender. Sie werden deshalb konsequent in der Sprache der Anwender formuliert. Die Gesamtheit der Use Cases – mit ihren beschreibenden Texten, Use-Case-Diagrammen und ergänzenden Sequenz- oder Aktivitätsdiagrammen – kann man als ein Systemmodell verstehen, ein Modell der Benutzung, eine äußere Sicht, die die Interaktion von System und Anwendern spezifiziert. Ein Use Case ist nach dieser Betrachtungsweise keine Anforderung.

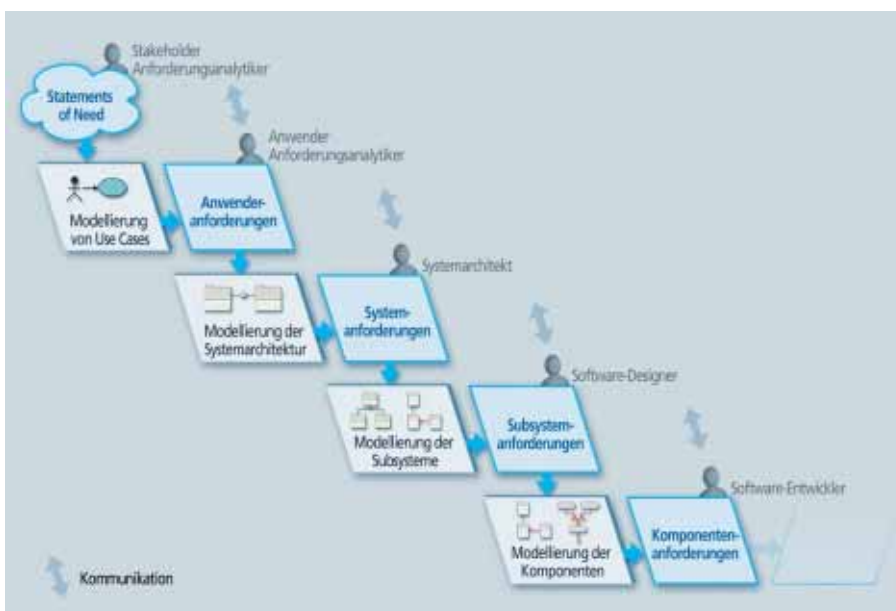


Bild 1: Requirements Engineering und Systemmodellierung in enger Wechselwirkung.

Vielmehr „stecken“ in einem Use Case meist mehrere Eigenschaften oder Verhaltensschritte des Systems. Sie sollten herausgearbeitet und einzeln als Anwenderanforderungen festgehalten werden.

Ausgangspunkt für den nächsten Schritt des Requirements Engineering ist die Frage: Welche Elemente muss das neue System auf Architekturniveau besitzen, um die Anwenderanforderungen zu erfüllen? Die aus den Use Cases abgeleiteten Anwenderanforderungen werden zum Ausgangspunkt der Entwicklung der Systemarchitektur. Sie wird mit den Mitteln der UML – am besten geeignet sind Komponenten- und Package-Diagramme – modelliert. Für die entworfene Systemarchitektur werden dann detailliertere Systemanforderungen formuliert.

Wie kann man die Systemanforderungen realisieren? Diese Frage führt zu einer weiteren Zerlegung der entworfenen Architektur und damit zu einem Systemmodell aus „feineren“ Subsystemen. Die Subsysteme des Systemmodells werden mithilfe von Package-, Klassen-, Sequenz- oder Zustandsdiagrammen der UML spezifiziert. Aus diesem Modell werden dann Subsystemanforderungen abgeleitet. Wie realisiert man die Anforderungen an ein Subsystem?

Die Frage wird durch die Modellierung von Komponenten beantwortet, aus denen detaillierte Komponentenanforderungen entwickelt werden. Sie werden wiederum in Modelle umgesetzt. Dieser wechselseitige Prozess der Verfeinerung von Anforderungen und Modellen wird solange fortgesetzt, bis die Anforderungen „baubar“ sind, das heißt, das Niveau von Arbeitsaufträgen an einzelne Software-Entwickler erreicht haben.

Übrigens: Erfolgen Realisierung und Integration dann in umgekehrter Reihenfolge – angefangen bei den feinsten Komponenten bis zum Gesamtsystem –, dann fügt sich der skizzierte Prozess für das Requirements Engineering in das Konzept eines V-Modells für die Software-Entwicklung ein. Dieses allgemein gültige Konzept liegt vielen Vorgehensmodellen zugrunde. Unter anderem auch dem neuen Entwicklungsstandard für IT-Systeme des Bundes, dem V-Modell XT.

Am Requirements Engineering sind neben Anforderungsanalytikern erst Stakeholder, Auftraggeber und Anwender, dann Systemarchitekten, Software-Designer und schließlich Software-Entwickler beteiligt. Abstrahiert man die Rolle der UML dabei, so wird klar: Sie bietet allen Beteiligten auf jeder Ebenen der Software-Entwicklung einheitliche (Denk-) Modelle oder – anders ausgedrückt – eine gemeinsame grafische Sprache.

Effiziente Tool-Unterstützung

Wichtig für erfolgreiches Requirements Engineering ist, dass die Anforderungen über alle Stufen der Zerlegung gleichartig beschrieben und zentral verwaltet werden. Sie müssen zu jedem Zeitpunkt im Projekt eindeutig identifizierbar sein und einen definierten Bearbeitungsstatus besitzen. Anforderungen, die mit diesen Eigenschaften ausgestattet sind, bilden nicht nur die Basis für die Entwicklungstätigkeiten. Sie dienen auch als Grundlage für eine zeitnahe Projektplanung.

Anforderungen sollten außerdem navigierbar sein, so dass nachvollzieh-

bar bleibt, welche Anforderungen auseinander hervorgegangen sind. Diese Eigenschaft erlaubt es dem Projektleiter, die klassischen Instrumente der Impact-Analyse als Entscheidungshilfe heranzuziehen, wenn er die Auswirkungen von Anforderungsänderungen abschätzen muss. Damit beantwortet er Fragen wie: Mit wem bekomme ich Probleme, wenn ich diese Anforderung jetzt zurückstelle? Und: Ist eine von einer Änderung betroffene Anforderung vielleicht schon in Bearbeitung? Noch mehr Transparenz der Entwicklung wird erreicht, wenn das eingesetzte Werkzeug für das Anforderungsmanagement und das verwendete UML-Tool miteinander kooperieren, das heißt wenn nicht nur zwischen Anforderungen, sondern auch von einer Anforderung zu den in Beziehung stehenden UML-Modellelementen navigiert werden kann.

So mit Tools unterstützt, stellt der beschriebene Prozess für das Requirements Engineering einen Riesenschritt hin zu guten Anforderungen und anforderungsgerechten Systemen dar.

Ursula Meseberg

Ursula.Meseberg@microTOOL.de

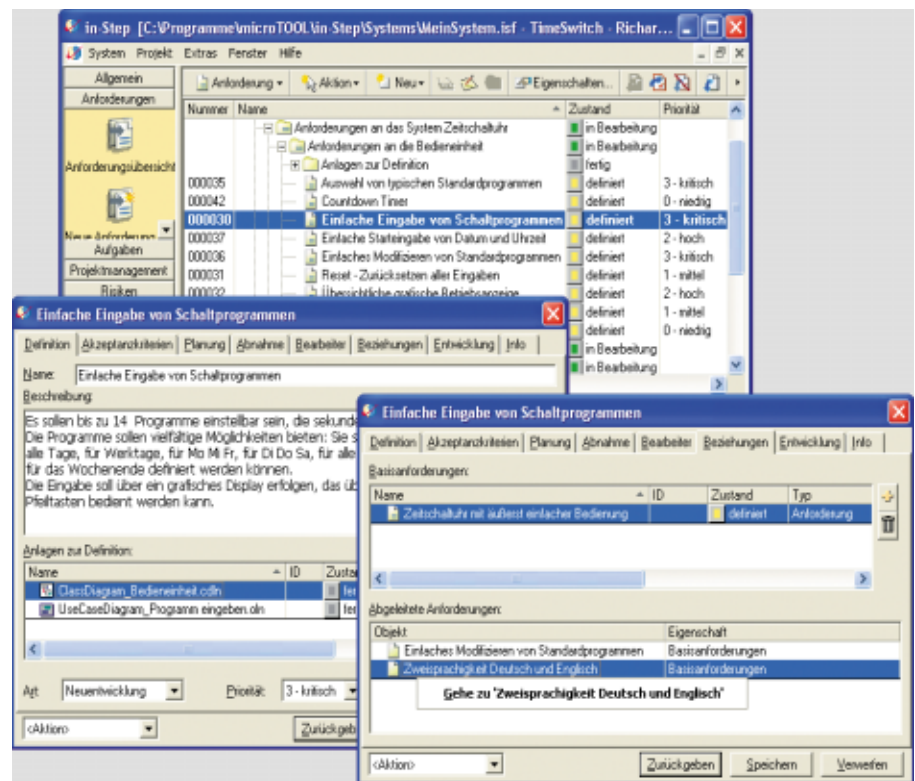


Bild 2: Ein Beispiel für die Dokumentation von Anforderungen: formularbasiert, mit maschinell gepflegten Beziehungen zur Navigation und Referenzen auf Diagramme in einem UML-Tool.