4 Kollaborative Analyse und Design

4.1 Einführung

Brooks [4.9] schreibt in seinem bekannten Essay zur Softwareentwicklung: „Software products are among the most complex of man-made systems, and software by its very nature has intrinsic, essential properties (e. g., complexity, invisibility, and changeability) that are not easily addressed.“ Die Gestaltung von interaktiven Softwaresystemen erfordert eine Betrachtung aus verschiedenen Perspektiven, damit eine hohe Produktqualität erreicht werden kann. Dazu sind in der Regel interdisziplinäre Zusammenarbeit und die aktive Einbeziehung von Benutzern und anderen Beteiligten in den Entwicklungsprozess notwendig, um ein gemeinsames Verständnis des Gestaltungsproblems und möglicher Lösungsansätze entwickeln zu können.

Dieses Kapitel macht Sie mit speziellen Methoden und Techniken vertraut, die kollaborative Analyse- und Designprozesse unterstützen. Es werden Beschreibungsmittel vorgestellt, die allgemein verständlich sind und die Darstellung und Diskussion des Warums, Was und Wie des zu entwickelnden Systems ermöglichen. In Kombination mit UML unterstützen sie eine qualitativ hochwertige Softwareentwicklung.

4.1.1 Besonderheiten und Qualität von Software

Software ist die Menge von Programmen, Verfahren, zugehörige Dokumentation und Daten, die mit dem Betrieb eines Computersystems zu tun haben [4.26]. Es wird dabei zwischen System- und Anwendungssoftware unterschieden. Unter Systemsoftware versteht man die Programme, die die Verbindung zu einer spezifischen Hardware herstellen. Dazu gehören etwa Betriebssysteme, Datenbankverwaltungssysteme oder Webserver. Anwendungssoftware ermöglicht die Abarbeitung von Anwendungsprogrammen, die die Erledigung von Aufgaben in bestimmten Anwendungsgebieten unterstützen sollen. In diesem Kapitel geht es um Anwendungssoftware, insbesondere um interaktive Systeme.

Software ist ein immaterielles Produkt, das nur in seinen Wirkungen beim Ablauf auf einem Computer bzw. indirekt über seine Dokumentation beobachtbar ist. Daher sind Fehler in der Softwarekonstruktion oft nicht leicht zu erkennen. Balzert [4.3] und Ludewig & Lichter [4.37] charakterisieren Software weiterhin wie folgt:

- Software wird nicht gefertigt, sondern nur entwickelt. Es gibt keine Fertigungskosten, wenn man als Fertigung die Reproduktion des ersten Musters versteht. Kopie und Original sind völlig gleich.
- Software unterliegt keinem Verschleiß, altert aber trotzdem.
- Softwarefehler entstehen nicht durch Abnutzung. Für Software gibt es keine Ersatzteile.
- Software ist im Allgemeinen leichter und schneller änderbar als ein anderes technisches Produkt. Dabei können kleinste Änderungen durchaus zu massiven Änderungen im Verhalten der Software führen.
- Software ist schwer zu „vermessen". Der Nachweis des wunschgemäßen Funktionierens ist schwieriger als bei anderen technischen Produkten.
- Software ist ein immaterielles Produkt

In [4.58] wird Software als eine Form der Explikation von Wissen mithilfe von Computersprachen verstanden. Es werden folgende Besonderheiten bezüglich ihrer Entwicklung, Distribution und Konsumption beschrieben.

- *Potenzielle Unabschließbarkeit der Entwicklung:* Die Entwicklung von Software kann immer weitergeführt werden. Software kann z. B. an neue Hardware angepasst oder ihre Funktionalität erweitert werden.
- *Potenzielle Unabschließbarkeit des Produktionsprozesses:* Die wachsende Komplexität von Software und sich ändernde Bedingungen (z. B. durch neue Produktanforderungen) tragen dazu bei, dass Software fehleranfällig ist. Es können immer wieder Fehler gefunden werden. Verbesserungen oder alternative Problemlösungen sind möglich bzw. sogar nötig.
- *Parallele Entwicklung:* Software ist ein symbolisches Produkt, das einfach versendet und verteilt werden kann. Weiterhin kann Software modularisiert werden. Dadurch wird eine parallele Entwicklung der Software durch Entwickler, die auch an verschiedenen Orten arbeiten können, möglich.
- *Verschränkung von Produktion und Anwendung:* Die Anforderungen an ein Produkt sind selten vollständig im Voraus bestimmbar. Sie können sich z. B. erst während der Nutzung eines Prototyps ergeben. Software muss in der Regel iterativ entwickelt werden, und es können immer wieder Anpassungen notwendig werden. Der Aufwand an Dienstleistungen, der mit dem Einsatz von Software verbunden ist, ist daher relativ hoch.

Wann hat Software eine hohe Qualität? Was unterscheidet gute von schlechter Software?

Qualität im alltäglichen Sprachgebrauch ist ein Synonym für die Güte eines Produktes oder eines Prozesses. Etwas hat beispielsweise eine „gute Qualität", wenn es langlebig ist. Der Begriff der Qualität lässt sich, wenn überhaupt, nur schwer definieren. Garvin [4.21] schlägt fünf Perspektiven vor, aus denen Qualität betrachtet werden kann.

- *Transzendente Perspektive:* Qualität ist ein anzustrebendes Ideal. Wir können Qualität erkennen, aber nicht analysieren und definieren.
- *Benutzerbezogene Perspektive:* Qualität ist Tauglichkeit für einen Zweck. Es wird angenommen, dass Benutzer individuelle Wünsche und Bedürfnisse besitzen und dass die Produkte, die diese am besten befriedigen, eine hohe Qualität besitzen.
- *Fertigungsbezogene Perspektive:* Qualität ist die Übereinstimmung des Produktes mit der Anforderungsspezifikation. Jede Abweichung von der Spezifikation bedeutet Einbußen in der Qualität.

- *Produktbezogene Perspektive*: Qualität ist an die inhärenten Eigenschaften (Attribute) eines Produktes gebunden.
- *Wertorientierte Perspektive:* Qualität ist definiert in Bezug auf Kosten und Preis. Mit anderen Worten, die Qualität hängt von der Anzahl der Kunden ab, die bereit sind, für das Produkt zu zahlen.

Die angegebenen Perspektiven auf die Qualität eines Produktes weisen unterschiedliche Annahmen und Schwerpunkte auf. Beispielsweise wird in der benutzer- und in der fertigungsbezogenen Perspektive ein Produkt von außen betrachtet, in der produktbezogenen Perspektive dagegen von innen. Im transzendenten Ansatz nimmt man an, dass Qualität keiner Seite (Produkt oder Benutzer) zugeschrieben und auch nicht gemessen werden kann. In der wertbezogenen Perspektive gibt es eine Vermischung des Qualitätsbegriffes mit dem Wertbegriff. Es geht um „Qualität, die man sich leisten kann“. Die Betrachtung der verschiedenen, sich teilweise widersprechenden Ansätze zur Definition von Qualität mag auf den ersten Blick akademisch erscheinen. Für eine erfolgreiche Softwareentwicklung ist es jedoch wichtig, dass im Verlaufe eines Projektes verschiedene Sichten auf die Produktqualität eingenommen werden und dass der Vorgang des Perspektivwechsels bewusst gestaltet wird. Im Folgenden gehen wir näher auf die produkt-, benutzer- und fertigungsbezogene Perspektive ein.

Die Qualität von Softwareprodukten wird, aus der **Produktperspektive** gesehen, an der Erfüllung bestimmter Produktmerkmale gemessen. Dabei gibt es verschiedene Vorschläge für Attribute, die einen wesentlichen Einfluss auf die Qualität von Softwareprodukten besitzen. Beispielsweise beschreibt das standardisierte, 1991 eingeführte Qualitätsmodell ISO 9126 (seit 2005: ISO/IEC 25000) relevante Qualitätsattribute in Form einer Hierarchie und empfiehlt, die Erfüllung der Attribute der letzten Hierarchieebene zu messen. Der Standard gibt dabei nicht im Detail vor, wie diese Messungen auszusehen haben. Zu den Hauptattributen dieses Modells gehören die folgenden:

- *Funktionalität:* Menge von Attributen, die das Vorhandensein der Funktionen und deren spezifizierten Eigenschaften betreffen, die die Anforderungen an die Software erfüllen sollen.
- *Zuverlässigkeit:* Menge von Attributen, die die Fähigkeit des Softwareproduktes betreffen, seine Leistung für den definierten Zeitraum und unter den festgelegten Bedingungen aufrechtzuerhalten.
- *Benutzbarkeit:* Menge von Attributen, die sich auf den Nutzungsaufwand einer Software für eine angegebene Benutzergruppe und der subjektiven Bewertung der Benutzung durch diese Benutzer beziehen.
- *Effizienz:* Menge von Attributen, die das Verhältnis zwischen Leistung und den unter festgelegten Bedingungen eingesetzten Ressourcen betreffen.
- *Wartbarkeit:* Menge von Attributen, die den Aufwand für festgelegte Änderungen betreffen (z. B. Korrekturen, Verbesserungen, Anpassungen der Software an veränderte Umgebungsbedingungen bzw. an Änderungen in den Anforderungen)
- *Übertragbarkeit:* Menge von Attributen, die die Fähigkeit der Software betreffen, in eine andere Umgebung übertragen zu werden (z. B. andere Hardware- oder Softwareumgebung, anderes organisatorisches Umfeld).