Zur Bedeutung von Spezifikationen in verschiedenen Teilaufgaben der Entwicklung kundenspezifischer Software

Claus Lewerentz Heinrich Rust

I-21/1997

Inhaltsverzeichnis

1 Einführung

1.1 Zweck der Arbeit

Formale Spezifikationen werden für verschiedene Aufgaben im Softwareentwicklungsprozeß als hilfreich angesehen. Sie dienen der eindeutigen Festlegung von Anforderungen an das Gesamtsystem, dem Nachweis der Übereinstimmung von Spezifikationen auf verschiedenen Abstraktionsebenen, dem Nachweis der Übereinstimmung von Spezifikation und Implementierung oder der Festlegung von Verantwortlichkeiten von Teilmodulen.

Bei einem solchen Spektrum an Möglichkeiten ist klar, daß dieselbe Notation kaum alle möglichen Teilaufgaben gleichermaßen gut unterstützen kann. Es stellt sich die Frage, wie dies systematisch untersucht werden sollte. Eine Vorgehensweise für eine solche Untersuchung soll in dieser Arbeit vorgestellt werden.

Fraser et al.[FKV94] geben einen Überblick über verschiedene formale Spezifikationsnotationen und die Art und Weise ihres Einsatzes. Sie betrachten die Frage unter dem Gesichtspunkt, wie von einer informellen Spezifikation zu einer formalen fortgeschritten wird.

Hall [Hal90] und später Bowen und Hinchey [BH95b] versuchten, mit einer Anzahl von Mißverständnissen aufzuräumen, denen sie in Bezug auf die Anwendung formaler Spezifikationsnotationen begegnet sind. In der hier erfolgenden Untersuchung wird ein solches Mißverständnis systematisch widerlegt: daß nämlich im gewöhnlichen Softwareprozeß nur Prüfungstätigkeiten durch formale Spezifikationen unterstützt werden können. Hier wird folgende These belegt:

> Spezifikationen mit höchsten Exaktheitsansprüchen sind besonders in den Reflexions- und Kommunikationstätigkeiten der Softwareentwicklung nutzbringend anwendbar.

Reflexionstätigkeiten sind solche, in denen sich eine einzelne Person über einen Sachverhalt Klarheit verschaffen möchte. Spezifikationsnotationen sind hier hilfreich, weil die Notation es dieser Person erleichtern kann, einen komplexen Sachverhalt übersichtlich zu strukturieren.

Kommunikationstätigkeiten sind solche, in denen sich mehrere Personen über einen Sachverhalt verständigen. Spezifikationsnotationen sind hier hilfreich, weil sie Sachverhalte mit guter Kontrollierbarkeit der nötigen Kontextinformation darzustellen gestatten.

Bowen und Hinchey [BH95a] weisen darauf hin, durch welche Elemente im Softwareentwicklungsprozeß die Anwendung formaler Methoden zu ergänzen sind. Sie machen deutlich, daß die Anwendung formaler Methoden nicht Ersatz, sondern Ergänzung bewährter herkömmlicher Methoden der Softwareentwicklung sein muß.

Um eine Grundlage für die Bewertung von Spezifikationsnotationen zu schaffen, bietet es sich an, einen Katalog von Bewertungskriterien zu erarbeiten. Solche Kataloge von Bewertungskriterien wurden bereits von Leveson et al.[LHHR94], von Ardis et al.[ACJ+ 96] und von Knight et al.[KDGN97] erarbeitet. Insbesondere Ardis et al. und Knight et al. weisen darauf hin, daß sie bei der Suche nach solchen Bewertungskriterien von Phasenmodellen der Softwareentwicklung ausgegangen sind. Wie sie allerdings aus den von ihnen benutzten Phasenmodellen die Kriterien ermittelten, erläutern sie nicht.

Auch in der vorliegenden Arbeit werden Bewertungskriterien dadurch erarbeitet, daß verschiedene Teilaufgaben im Prozeß der Softwareentwicklung identifiziert werden, in denen die Benutzung einer Spezifikationsnotation eine Rolle spielen kann. Verschiedene Teilaufgaben können verschiedene Anforderungen an Notationen bedingen.

Folgen für Kommunikationsanforderungen fallen allerdings systematisch unter den Tisch, wenn von der Beteiligung verschiedener Personen in den Projektphasen abstrahiert wird. Es ist aber unsere These, daß die Unterstützung effizienter Kommunikation eines der wichtigsten Kriterien für den Erfolg einer formalen Notation ist. Dies macht eine Untersuchung der Rollen nötig, die in den jeweiligen Teilaufgaben wichtig sind: So gibt es Auftraggeber, Anwender, Auftragnehmer, Anforderungsanalytiker, Entwerfer, Implementierer, Tester mit je spezifischen Aufgaben. Der Vorteil gegenüber der bloßen Beschreibung der Arbeitsphasen liegt darin, daß hier explizit wird, daß auch in derselben Arbeitsphase verschiedene Personen an der Aufgabenerledigung beteiligt sind, deren Voraussetzungen und Anforderungen an die Spezifikationsnotation möglicherweise verschieden sind.

Bei diesem Vorgehen für die Entwicklung von Bewertungskriterien hat das zugrundegelegte Bild vom Softwareentwicklungsprozeß einen großen Einfluß auf die Gewichtung verschiedener Eigenschaften von Spezifikationsnotationen. In dieser Arbeit wird ein Bild zugrundegelegt, das die Implementierung eines kundenspezifischen DV-Systems modelliert. Infolgedessen spielt die Kommunikation mit dem Kunden eine zentrale Rolle bei der Entwicklung der Anforderungen an das zu entwickelnde System.

1.2 Einordnung

In [Rus94, Seite 77–85] wurde eine Untersuchung der Leistungen formaler Spezifikationsnotationen vorgenommen und es wurden verschiedene Kritiken daran referiert. Die vorliegende Arbeit paßt sich in diesen Kontext ein. Daher sollen die dort erarbeiteten Ergebnisse kurz dargestellt und ergänzt werden.

Der Ort dieser Arbeit ist die Diskussion, welche Rolle Formalität und Mathematisierung im Prozeß der Softwareentwicklung hat. Vom Einsatz formaler Methoden werden dabei folgende Ergebnisse erhofft:

- Die einengende Notation erschwert die Beschreibung der Anforderungen an ein System, indem sie zu größerer Eindeutigkeit zwingt. Damit zwingt die Bearbeiter auch, sich genauer mit dem System auseinanderzusetzen: [MMM85], [Hal90], [dR91].
- Gewisse Inkonsistenzen werden beim Formulierungsversuch explizit: [MMM85], [MDL87], [Hal90].
- Nur von einfachen Entwürfen können wichtige Eigenschaften mit vertretbarem Aufwand formal bewiesen werden. Werden daher formale Beweise wichtiger Systemeigenschaften verlangt, so gewöhnen sich die Entwerfer mit der Zeit daran, einfacher zu entwerfen [MDL87].
- Die Aufmerksamkeit der Entwerfer werden auf die durch die Formalismen explizit ausgedrückten Systemaspekte fokussiert: [Par77], [Flo85].
- Die präzise Semantik der benutzten Notationen erzeugt Eindeutigkeit bei Ergebnissen: [Par77], [Win90], [dR91].
- Eine formale Notation, die mehrere Abstraktionsebenen überdeckt, ermöglicht die Konsistenzprüfung der Ebenen miteinander: [GHH+92].

Einige Befürchtungen beim Einsatz formaler Methoden sind die folgenden:

- Die Fundierung der Formalismen in der (informellen) Intuition wird bei Betonung der Formalismen leicht übersehen. So entsteht die Gefahr, daß Formalismen blind angewendet werden: [Nau85], [Nau89].
- Die in vielen Softwareentwicklungsprojekten zentrale (informelle) Frage der Angemessenheit des entstehenden Systems für die Arbeitswelt der künftigen Nutzer wird bei Betonung von Formalismen unterbewertet: [Flo85], [MMM85].
- Die Betonung eines formalisierten Gesamtweges von der Anforderungsaufnahme zum endgültigen Produkt lenkt die Aufmerksamkeit von besser geeigneten „kleinen“ Formalismen und von der Möglichkeit und der Notwendigkeit von Präzision bei nichtformalisierten Anforderungen und Entwicklungsschritten ab: [Nau82].
- Die Anwendung formaler Methoden bedeutet einen sehr hohen Aufwand, der durch die Ergebnisse nicht gerechtfertigt wird.

Auf die Frage, welche der angeführten Argumente in welcher Situation die bestimmenderen seien, steht eine systematisch gefundene Antwort aus.

In der vorliegenden Arbeit wird nun der Versuch gemacht, eine Grundlage zu präsentieren, auf der die Bedeutung der verschiedenen Standpunkte systematisch gegeneinander abgewogen werden können. Dies geschieht durch eine Analyse eines Softwareentwicklungsprozesses mit dem Ziel, mögliche Nutzen von Spezifikationsnotationen und die dafür hilfreichen Eigenschaften solcher Notationen zu identifizieren.

2 Katalog der Einsatzsituationen und Tätigkeiten

Im vorangehenden Abschnitt wurde beschrieben, daß die Einsatzsituationen sich durch die zu erledigenden Teilaufgaben, durch die dabei auszufüllenden Rollen, und endlich durch die spezifischen Kommunikationsprobleme kennzeichnen lassen, die bei den Rollenträgern relevant werden können.

Es ist begründet worden, warum ein Phasenmodell als heuristisches Hilsmittel bei der Identifizierung der Teilaufgaben benutzt werden soll.

Im folgenden wird nun eine Liste von Teilaufgaben aufgestellt. Als Hauptbereiche werden dabei Vorarbeiten, Anforderungsanalyse, Entwurf/Implementierung, Test und produktiver Einsatz des Systems unterschieden. Diese Hauptaufgaben werden in weitere Teilaufgaben aufgespalten, eventuell mehrfach.

Wenn auch die Hauptbereiche und Teilaufgaben in Anlehnung an Phasenmodelle der Softwareentwicklung aufgegliedert wurden, so ist doch die hier angebotene Darstellung nicht als zeitlicher Ablauf zu verstehen. Vielmehr soll das Schema die verschiedenen im Verlaufe eines Softwareentwicklungsprozesses anfallenden Aufgaben übersichtlich zu ordnen helfen. Die skizzierte Reihenfolge ist also nicht normativ zu verstehen.

Zu jeder der aufgesuchten Teilaufgaben werden einige Rollen identifiziert.

1. **Vorarbeiten:** Unter diesen Stichwort sind Aufgaben zusammengefaßt, die wenigstens einmal vor der Phase der detaillierten Anforderungsdefinition anfallen. Insofern kann man hier nun doch von einer Phase sprechen. Hierzu gehören insbesondere die Aufgaben, die beim Auftraggeber anfallen, aber auch Aspekte, die mit der Projektorganisation zusammenhängen.

   Bereits in dieser Phase kann eine Spezifikationsnotation eine Rolle spielen.

   - **1.1. Ausschreibung eines DV-Entwicklungsauftrags:** Je detaillierter eine Ausschreibung ist und je klarer die Vorgaben über den zu vergebenden Entwicklungsauftrag, desto leichter fällt die Einschätzung der Solidität der daraufhin eingehenden Angebote.

     Tätigkeiten und Anforderungen: Gewisse Fachabteilungen melden Bedarf für ein Datenverarbeitungssystem an; eine Einkaufsabteilung erstellt die Ausschreibung. Die Notation muß leicht erlernbar und der Beschreibung des Bedarfs angemessen sein.

   - **1.2. Angebotserstellung, Machbarkeitsstudie, Aufwandsabschätzung, Vertragsgestaltung:** Der Auftragnehmer eines Softwareentwicklungsprojektes muß für die Formulierung des abzuliefernden Angebots die technische Machbarkeit einschätzen und eine Aufwandsabschätzung vornehmen.

     Das Angebot wird zudem gewöhnlich die Grundlage für die Gestaltung des Vertrages sein, den Auftragnehmer und Auftraggeber miteinander abschließen. Es kann insofern einen rechtlich bindende Ausdruck bekommen.

     Ökonomische Gründe machen eine gute Aufwandsschätzung wichtig, da sowohl zu hohe als auch zu niedrige Aufwandsschätzungen Nachteile nach sich ziehen.

     Grundlage für solide Schätzungen sind Erfahrungen mit vergleichbaren Systemen sowie ein möglichst detailliertes Bild über die in einem Projekt zu

erledigenden Aufgaben. Genau sind diese Aufgaben immer erst nach Abschluß des Projektes bekannt.

Enthält das Angebot eine gute, also möglichst detaillierte Spezifikation der im Projekt zu behandelnden Aufgaben, so haben Auftragnehmer und Auftraggeber eine sichere Planungsgrundlage.

Tätigkeiten und Anforderungen:

- 1.2.1. Der Auftragnehmer führt eine Machbarkeitsstudie durch, schätzt den zu erwartenden Ressourcenbedarf und erstellt ein entsprechendes Angebot. Die Beschreibung der zu erfüllenden Anforderungen muß Aufwandsabschätzungen ermöglichen, besondere Probleme deutlich machen.
- 1.2.2. Auftragnehmer und Auftraggeber einigen sich auf einen Vertrag, der die Auftragsvergabe beschreibt. Die Notation muß für Auftragnehmer wie Auftraggeber nachvollziehbar sein, die zu erbringenden Leistungen sollen möglichst eindeutig formulierbar sein.

- **1.3. Teamzusammenstellung:** Die Teamzusammenstellung ist eine wichtige Phase bei der Vorbereitung eines Projektes: Hier werden einerseits zum Teil zukünftige Rollen zugewiesen, wie etwa die Projektleiterschaft, andererseits müssen die Personen so ausgewählt werden, daß eine erfolgreiche Projektbearbeitung erwartet werden kann.

  Neben einer hinreichenden personellen Ausstattung des Projektes spielen hier vor allem die Vorkenntnisse der Personen eine besondere Rolle, die das Projekt bearbeiten sollen. Damit deutlich wird, welche Vorkenntnisse sinnvoll sind, muß eine für diesen Zweck hinreichende Spezifikation der Aufgabenstellung vorliegen.

  Die Spezifikation kann also bei der Teamzusammenstellung nützlich sein, wenn man ihr entnehmen kann, welche Kenntnisse bei der Bearbeitung des Projektes eine besondere Rolle spielen können.

  Tätigkeiten und Anforderungen:

  - 1.3.1. Für das Projekt wird ein geeigneter Projektleiter gesucht. Die im Verlauf des Projektes anfallenden Planungs- und Organisationsaufgaben, wie sie aufgrund des Vertrags und aufgrund vorheriger Erfahrungen mit dem Auftraggeber erwartet werden, müssen abgeschätzt werden.
  - 1.3.2. Es werden Experten für die verschiedenen spezifischen Kenntnisse gesucht. Mit den möglichen Mitgliedern des Projektteams wird geklärt, ob sie die ihnen jeweils zugedachten Aufgaben verantwortlich wahrnehmen können. Es muß erkennbar sein, welche Fachexpertise das Projekt voraussetzt. Dies wird vereinfacht, wenn Spezialkenntnisse in den Anforderungen an das zu erstellende System explizit thematisiert werden.

- **1.4. Planung der iterativen und inkrementellen Entwicklung des Softwaresystems:** Es wird gewöhnlich empfohlen, umfangreichere Softwaresysteme in mehreren Stufen zu entwickeln, also die gesamte zu implementierende Funktionalität in aufeinander aufbauende Gruppen einzuteilen.

  Wenn man ein solches Vorgehen wählt, so hat dies den Vorteil, daß die verschiedenen Teilaufgaben, die im Laufe der Systementwicklung zu erle-

digen sind, nicht nur einmal eintreten, sondern daß der gesamte Entwicklungszyklus mehrfach durchlaufen wird. Damit ist es möglich, in früheren Durchläufen Erfahrungen zu machen, aus denen man für die späteren Durchläufe lernen kann.

Ein solches inkrementelles Vorgehen kann durch eine Spezifikationsnotation unterstützt werden, die die explizite Beschreibung von Funktionsstufen unterstützt.

Tätigkeiten und Anforderungen:

- 1.4.1. Das Projektteam erarbeitet einen Plan für die inkrementelle Anfertigung des Gesamtsystems. Dies wird vereinfacht, wenn sich die Anforderungen an das Gesamtsystem in mehrere Inkremente aufgeteilt dokumentieren lassen.
- 1.4.2. Die Planung erfolgt zusammen mit dem Auftraggeber und den künftigen Anwendern. Der Auftraggeber muß einschätzen, ob der vom Projektteam erarbeitete Projektplan aus seiner Sicht sinnvoll ist, ob sich beispielsweise kritische Entwurfsentscheidungen möglichst früh in Prototypen niederschlagen, die eine Prüfung durch die zukünftigen Anwender ermöglichen. Dies wird vereinfacht, wenn kritische Entwurfsphasen im Entwicklungsplan explizit thematisiert werden.

- **1.5. Planung des Konfigurationsmanagements:** Im Rahmen des Konfigurationsmanagements wird geregelt, wie verschiedene Versionen des Programms verwaltet werden sollen. So wird es von vielen Programmprodukten im Laufe der Zeit verschiedene Releases geben, in denen Fehlerkorrekturen und Funktionserweiterungen vorgenommen werden, und oft gibt es verschiedene Varianten des Programms für verschiedene Laufzeitumgebungen.

  Die Gestaltung dieser Versionen läßt sich nicht gut planen, soweit sie auf der Notwendigkeit von Fehlerkorrekturen und Änderungen der Anforderungen an das System beruhen. Für solche Fälle muß nur ein Schema vorgesehen sein, das die systematische Verwaltung der verschiedenen Versionen ermöglicht.

  Manche Aspekte der Versionenverwaltung lassen sich allerdings bereits vor der Durchführung eines Projektes planen. Wenn es verschiedene Einsatzbedingungen (etwa verschiedene Betriebssysteme) gibt, auf denen ds System arbeiten muß, oder wenn die Gesamtfunktionalität in mehreren Iterationen implementiert werden soll.

  Tätigkeiten und Anforderungen: Das Projektteam muß planen, welche Varianten des zu Systems zu entwickeln sind, und wie verschiedene Versionen und Varianten verwaltet werden sollen. Dies wird vereinfacht, wenn die Anforderungsdokumente Varianten und Inkremente explizit ausweisen. Dies wird vereinfacht, wenn die Anforderungsdokumente Varianten und Inkremente explizit ausweisen.

2. **Anforderungsanalyse:**

   - **2.1. Analyse eines bestehenden Altsystems:** Die Anforderungsanalyse wird wesentlich erleichtert, wenn ein Altsystem existiert, dessen Funktionalität im Detail aufgenommen und dokumentiert werden kann. Auf der

Grundlage einer detaillierten Spezifikation des Altsystems kann analysiert werden, welche der Systemeigenschaften verbesserungswürdig sind und welche beibehalten werden müssen.

Eine Spezifikation kann bei der Analyse eines existierenden Altsystems eine Darstellung des Ist-Zustands sichern.

Tätigkeiten und Anforderungen:

- 2.1.1. Das Projektteam muß die von einem bestehenden Altsystem ausgefüllten Funktionen möglichst vollständig und eindeutig dokumentieren. Dies wird vereinfacht, wenn die Notation, die zur Dokumentation der Funktionalität des Altsystems benutzt wird, alle wesentlichen Eigenschaften des Altsystems darzustellen gestattet, insbesondere auch nichtfunktionale Anforderungen wie Effizienz, Ausfallwahrscheinlichkeit, Bedienungsfreundlichkeit. Zu erhaltende Eigenschaften sind ebenso zu kennzeichnen wie zu verbessernde Eigenschaften.
- 2.1.2. Die augenblicklichen Anwender müssen den Mitgliedern des Projektteams bei der Erarbeitung einer Dokumentation des Altsystems behilflich sein und müssen diese Dokumentation auf Korrektheit und Vollständigkeit prüfen. Die zur Dokumentation benutzte Notation muß daher den Anwendern verständlich sein, Fehler und Lücken in der Dokumentation müssen auf der Grundlage ihrer täglichen Arbeit mit dem Altsystem erkennbar werden.

- **2.2. Erarbeitung einer Anforderungsspezifikation:** Anwenderwünsche, Untersuchung der Einsatzbedingungen, positive wie negative Eigenschaften eines Altsystems, Beobachtungen der täglichen Arbeit der Anwender, Befragungen der Fachexperten, Effizienzanforderungen und die Anfertigung von Prototypen werden gewöhnlich zur Sammlung von Anforderungen an das entstehende Neusystem benutzt. Es ist wichtig, die so gesammelten Anforderungen in Formen zu dokumentieren, die die Kommunikation über die so festgehaltenen Anforderungen mit den Fachleuten und den künftigen Anwendern erleichtert.

  Die Anforderungsspezifikation hat, als Grundlage der weiteren Projektarbeit, eine Doppelfunktion: Der Auftraggeber steht dafür ein, daß er eine Implementierung, die die explizit aufgelisteten Eigenschaften aufweist, akzeptieren wird, und der Auftragnehmer weiß, welche Eigenschaften das herzustellende Produkt aufweisen muß, damit es letztendlich akzeptiert wird.

  Gewöhnlich ist es jedoch dem Auftraggeber nicht möglich, alle wesentlichen Anforderungen vor den ersten Einsatzversuchen des Systems zu formulieren, und ebensowenig kann der Auftragnehmer in jedem Fall bereits früh abschätzen, ob alle gewünschten Systemeigenschaften mit vertretbarem Aufwand implementiert werden können. Die Anforderungsspezifikation wird daher im Laufe der Entwicklung, der Anfertigung von Prototypen usw. geändert werden.

  Es muß jedoch eine Verfahrensweise gefunden werden, die die Veränderungen der Anforderungen an das Programm während der Entwicklung beschränkt.

  Diese verschiedenen Aspekte des Umgangs mit einer Anforderungsdefinition können durch eine geeignete Notation unterstützt werden. Eine Notation mit gut festgelegter Semantik kann eine weitgehende Eindeutigkeit der

Formulierung garantieren. Eine gut verständliche Notation kann garantieren, daß Auftragnehmer und Auftraggeber ähnliche Vorstellungen über die vom System zu erfüllenden Anforderungen haben. Sie kann sichern, daß Änderungen hinsichtlich der Anforderungen von einem Beteiligten nicht lediglich durch veränderte Interpretation der Spezifikationsdokumente den anderen Beteiligten aufgezwungen werden können.

Der Zwang, eine Spezifikation in einer formalen Notation festzuhalten, macht Definitionslücken deutlicher, weil unklare Konzepte hier weniger leicht mit unklaren Begriffen ausgedrückt werden können.

Tätigkeiten und Anforderungen:

- 2.2.1. Das Projektteam muß eine detaillierte Dokumentation der Anforderungen an das neu zu entwickelnde System vornehmen. Auf der Grundlage dieser Dokumentation ist die weitere Entwicklung vorzunehmen. Dies wird erleichtert, wenn die Notation Konsistenzprüfungen erlaubt, Definitionslücken erkennbar macht und Bewertung durch Laien ermöglicht.
- 2.2.2. Von den künftigen Anwender wird eine Prüfung der Dokumentation der Anforderungen an das neue System erwartet. Dies ist nur möglich, wenn die Konzeption des zu entwickelnden Systems ihnen in verständlicher Form präsentiert wird.
- 2.2.3. Von den Auftraggebern wird erwartet, das fertige Anforderungsdokument als Detaillierung der bei der Vertragsvergabe festgelegten Aufgaben des Auftragnehmers schriftlich zu bestätigen. Dies wird er nur tun können, wenn er das Anforderungsdokument versteht. Die für die Dokumentation der Anforderungen benutzte Notation muß daher für Auftraggeber nachvollziehbar sein.

- **2.3. Definition der Schnittstellen und Verteilung der Verantwortlichkeiten bei Vergabe des Auftrags an mehrere Auftragnehmer:** Wenn der Auftrag auf mehrere rechtlich unabhängige Auftragnehmer aufgeteilt wird, so ist eine klare Festlegung der Verantwortlichkeiten und eine ebenso klare Definition der Schnittstellen vorzusehen. Auch diese Spezifikation kann durch eine dazu geeignete Notation unterstützt werden.

  Tätigkeiten und Anforderungen:

  - 2.3.1. Der Auftraggeber muß sichern, daß die Anforderungen an das Gesamtsystem durch die Zusammenarbeit der von verschiedenen Auftragnehmern zu entwickelnden Teilsysteme erfüllt werden. Hierfür ist eine exakte Beschreibung der Verantwortlichkeiten der verschiedenen Auftragnehmer nötig, da sonst bei Aufteilung der Arbeit unter mehrere Auftragnehmer die Gefahr besteht, daß Teilaufgaben „verlorengehen“.
  - 2.3.2. Die verschiedenen Auftragnehmer werden auf eine klare Definition des von ihnen zu entwickelnden Teilsystems Wert legen, um der Gefahr zu begegnen, daß ihnen im Laufe der Zeit der Entwicklung mehr Aufgaben zugeteilt werden, als sie anfangs annehmen konnten. Dies wird durch eine Notation zur Festlegung der Verantwortlichkeiten ermöglicht, die die klare Abgrenzung von Teilaufgaben voneinander unterstützt.

- **2.4. Sicherung des gemeinsamen Verständnisses der Anforderungsspezifikation zwischen zukünftigen Nutzern und Entwicklern:** Eine schwer

zu mißdeutende Notation hilft, den Konsens unter den Projektbeteiligten über das zu implementierende System zu sichern.

Tätigkeiten und Anforderungen:

- 2.4.1. Die künftigen Anwender haben ein Interesse daran, zu verstehen, welche Eigenschaften das System aufweist, mit dem sie einmal arbeiten sollen, um auf die Entwicklung des Systems Einfluß zu nehmen. Die Anforderungsspezifikation muß daher in einer Form entwickelt werden, die von den künftigen Anwendern verstanden werden kann.
- 2.4.2. Auch das Projektteam hat ein Interesse daran, möglichst frühzeitig Rückmeldungen von den künftigen Anwendern über Entwurfsfehler und vergessene Anforderungen an das Gesamtsystem zu erhalten. So ist es auch im Interesse des Projektteams, eine Notation zu benutzen, die bei der Kommunikation mit den künftigen Anwendern die Diskussion auf wesentliche Fehlermöglichkeiten fokussiert. Dies kann erleichtert werden, wenn die Systembestandteile, bei denen Fehler oder Definitionslücken besonders kritisch sind, in der benutzten Notation explizit wiedergegeben werden müssen.

- **2.5. Prototyperstellung, Prüfung der prototypisch implementierten Funktionalität:** Prototypen dienen den Entwicklern unter anderem der Prüfung ihres Verständnisses der Anforderungen. Sie dienen zudem den künftigen Systemanwendern dazu, sich besser vorstellen zu können, was das spezifizierte System leisten wird. Endlich gibt es Anforderungen an ein entstehendes System, die sich nur sehr schwierig explizit schriftlich oder graphisch erfassen lassen. Ein Prototyp kann dem künftigen Anwender die frühzeitige Bewertung auch solcher Entwurfsentscheidungen ermöglichen, die sich nur schwer durch explizite verbale oder formale Spezifikationen darstellen lassen.

  Auftraggeber haben ein Interesse daran, frühzeitig Produkte in die Hand zu bekommen, an denen sie messen können, ob ihre Erwartungen über die Interpretation der explizit niedergelegten Spezifikation vom Auftragnehmer erfüllt werden.

  Auftragnehmer haben andererseits ein Interessen daran, vom Auftraggeber möglichst früh Bestätigungen ihrer Interpretation der Aufgabenstellung zu erhalten.

  Beide Wünsche lassen sich prinzipiell durch ein auf der Anfertigung von frühen Prototypen basierendes Vorgehen erfüllen. Wenn aber der Auftragnehmer sich die Erfüllung gewisser Funktionalität durch einen Prototypen bescheinigen lassen möchte, stellt sich das Problem, daß Prototypen nur gewisse Teile der zu implementierenden Gesamtfunktionalität abdecken. Damit ist in einer Bescheinigung die vom Auftraggeber akezptierte Implementierung eines Teils des Gesamtfunktionsumfangs zu spezifizieren.

  Tätigkeiten und Anforderungen:

  - 2.5.1. Vielen zukünftigen Anwendern fällt es besonders leicht, die Angemessenheit von Entwurfsentscheidungen zu beurteilen, wenn sie diese Entwurfsentscheidungen in Form von prototypischen Programmen präsentiert bekommen. So wird es ihnen erleichtert, sich klar zu machen, welche Folgen eine Entwurfsentscheidung für ih-

re künftige Arbeit haben werden. Da allerdings Prototypen definitionsgemäß nicht alle Eigenschaften des endgültigen Produkts aufweisen, müssen die zukünftigen Anwender wissen, welche Teilfunktionalität sie in einem Prototyp erwarten können und welche nicht, welche Nichtübereinstimmungen der Eigenschaften des Prototypen mit den Anforderungen an ein einsatzfähiges System also kritisch sind, und welche nicht. Dies wird erleichtert, wenn explizit dokumentiert wird, welche Teilfunktionalität durch einen bestimmten Prototyp abgeprüft werden soll.

  - 2.5.2. Das Projektteam hat großes Interesse daran, möglichst viel über die wirklichen Anforderungen mit Hilfe eines Prototypen zu lernen. Andererseits wünscht es, die von den zukünftigen Anwendern die durch einen Prototypen überprüften Systemeigenschaften, die im Prototyp nur in impliziter Form vorliegen, in expliziter Form zu dokumentieren. Dies wird durch eine Notation vereinfacht, die die kritischen Eigenschaften des Prototypen exakt und für die zukünftigen Anwender nachvollziehbar zu dokumentieren gestattet.

- **2.6. Qualitätssicherung hinsichtlich der Spezifikationsdokumente:** Die Dokumente, die die Spezifikation des zu implementierenden Gesamtsystems enthalten, dienen bei manchen Vorgehensweise als Grundlagen für viele weitere Schritte. Fehler in diesen Dokumenten können äußerst kostspielig werden, weil sie eventuell viele Korrekturen notwendig machen. Daher ist eine sorgfältige Qualitätssicherung dieser Dokumente von großer Bedeutung.

  Vollständigkeits- und Konsistenzprüfungen sind leichter durchzuführen, wenn die Dokumentation in einer Form vorliegt, die Definitionslücken und Inkonsistenzen deutlicher zutage treten läßt. Diese Eigenschaften können von einer geeigneten Notation unterstützt werden, insbesondere dann, wenn sich die Konzepte Konsistenz und Vollständigkeit formalisieren lassen.

  Tätigkeiten und Anforderungen:

  - 2.6.1. Das Projektteam wird, um eine sichere Planungsgrundlage zu haben, auf möglichst weitgehende Korrektheit und Vollständigkeit der Spezifikationsdokumente hinsichtlich der zentralen Anforderungen großen Wert legen. Eine Prüfung der konzeptionellen Korrektheit und Vollständigkeit durch das Projektteam kann durch eine Notation erleichtert werden, in der die wesentlichen Anforderungen vor den untergeordneten hervorgehoben werden können. Eventuell können formale Verfahren zur Konsistenzprüfung eingesetzt werden.
  - 2.6.2. Auch die zukünftigen Anwender haben ein Interesse daran, daß die Planungsgrundlage des Projektteams den wirklichen Anforderungen angemessen ist. Sie können aber Fehler und Lücken nur dann feststellen, wenn sie diese Planungsgrundlage verstehen und ihre Bedeutung für ihre tägliche Arbeit einsehen können. Eine Spezifikationsnotation sollte daher den zukünftigen Anwendern diese beiden Aktivitäten erleichtern.

3. **Entwurf und Implementierung:**

- **3.1. Grobentwurf:** Im Grobentwurf wird die vom Gesamtsystem zu leistende Funktionalität auf eine Anzahl von Teilsystemem aufgeteilt. Dabei wird allerdings keine sehr detaillierte Festlegung der Funktionalität angestrebt, im Vordergrund steht vielmehr die Herausarbeitung der grundlegenden Konzepte, die bei einer Festlegung der Details allzu leicht undeutlich wird.

  Um die Übersichtlichkeit zu erhalten, erfolgt diese Aufteilung in Teilsysteme in hierarchischer Weise, und zwar so, daß jedes System in eine überschaubare Zahl von Teilsystemen aufgeteilt wird. Diese Teilsysteme sollten Sinneinheiten mit klaren Verantwortungen bilden, so daß man sie ihrerseits bei Bedarf weiter analysieren kann, ohne stets ihren Einsatzkontext voll präsent haben zu müssen.

  So wird in einer hierarchisch von den abstrakteren zu den konkreteren Systembeschreibungsebenen fortschreitenden Analyse ein Strukturbaum entworfen, der beschreibt, aus welchen eventuell wiederum in sich zu gliedernden Teilmodulen das Gesamtsystem aufgebaut werden muß. Diesen Teilmodulen sind jeweils spezifische, klar abgrenzbare Aufgaben zuzuweisen, deren gemeinsame Erfüllung die Einhaltung der Spezifikation des Gesamtsystems garantiert.

  Dem Grobentwurf lassen sich einige Unteraufgaben zuweisen, in denen Spezifikationsnotationen eine besondere Rolle spielen.

  - **3.1.1. Identifikation einer Hierarchie von zu implementierenden Teilmodulen:** Es muß eine Hierarchie von zu implementierenden Teilmodulen aufgebaut werden. Dabei sind den Teilmodulen und dem aus diesen Teilmodulen zusammengesetzten Modul gewisse Verantwortlichkeiten zuzuweisen. Die Zuweisung von Verantwortlichkeiten an die Module erfolgt beim Grobentwurf noch nicht allzu detailliert, damit auf der gegebenen Abstraktionsebene das System besser überschaubar bleibt.
    Zudem wünscht man sich die Möglichkeit, die Teilmodule zur weiteren Bearbeitung verschiedenen Personen oder Arbeitsgruppen zuzuweisen.
    Auch diese Tätigkeiten sind durch eine Spezifikationsnotation unterstützbar. Wenn diese die Aufteilung des Gesamtsystems in Module explizit macht, so kann man aus dem modularen Aufbau der Spezifikation Hinweise auf den modularen Aufbau des Entwurfes und der Implementierung entnehmen. Auch diese Zuweisung von qualitativen Verantwortlichkeiten läßt sich durch eine passende Notation unterstützen, wenn gewisse der in der Spezifikation festgehaltenen Anforderungen an ein Modul als zentral, andere als ergänzend ausgezeichnet werden.
    Tätigkeiten und Anforderungen:
    * 3.1.1.1. Das Projektteam teilt das zu entwickelnde Gesamtsystem in Teilsysteme auf und weist den Teilsystemen Verantwortlichkeiten zu. Dabei wird eine allzu weitgehende Detailliertheit vermieden. Eine Notation, die diese Aufteilung in Teilsyssteme unterstützt, muß eine allzu weitgehende Exaktheit vermeidbar machen, aber doch die Zuteilung von Verantwortlichkeiten zu Teilsystemen unterstützen.

* 3.1.1.2. Das Ergebnis des Grobentwurfs sollte von den zukünftigen Anwendern auf Problemangemessenheit hin überprüft werden. Es müßte überprüft werden, ob alle wesentlichen Verantwortlichkeiten des Gesamtsystems durch die Verantwortlichkeiten der Teilsysteme abgedeckt werden. Dies kann durch eine Notation vereinfacht werden, die die Verantwortlichkeiten des Gesamtsystems mit den Verantwortlichkeiten der Teilsysteme zu vergleichen gestattet.

– **3.1.2. Spezifikation des Systems auf verschiedenen Abstraktionsebenen:** Beim Grobentwurf entstehen verschiedene Darstellungen desselben Gesamtsystems, allerdings auf verschiedenen Abstraktionsebenen. Wenn eine Spezifikationsnotation die verschiedenen Abstraktionsebenen unterstützt, dann ist es möglich, den Grobentwurf mit Hilfe dieser Spezifikationsnotation zu unterstützen.
Tätigkeiten und Anforderungen:
  * 3.1.2.1. Das Projektteam entwickelt Spezifikationen des Gesamtsystems auf zunehmend tiefer liegenden Abstraktionsebenen, also mit zunehmend mehr Details. Damit muß gesichert werden, daß eine Spezifikation des Gesamtsystems auf einer detaillierteren Systemebene die kritischen Eigenschaften aufweist, die für eine Spezifikation des Systems auf einer übergeordneten Systemebene mit den zukünftigen Anwendern zusammen geprüft worden sind. Dies wird erleichtert, wenn die verschiedenen Systemspezifikationen leicht miteinander verglichen werden können, um sie so auf Übereinstimmung hin zu überprüfen.
  * 3.1.2.2. Je detallierter die Systemspezifikationen sind, desto mehr Schwierigkeiten werden die zukünftigen Anwender dabei haben, diese Spezifikationen nachzuvollziehen und zu überprüfen, soweit die verschiedenen Entwurfsentcheidungen nicht als Bestandteile von Prototypen direkt prüfbar gemacht werden können. Prüfung der Spezifikationen auf detaillierteren Ebenen durch die zukünftigen Anwender kann also durch Ausführbarkeit der Spezifikationen erleichtert werden.

– **3.1.3. Prüfung der Übereinstimmung der Systemspezifikationen auf verschiedenen Abstraktionsebenen:** Wenn die Spezifikationsnotation die Darstellung des untersuchten Teilsystems auf verschiedenen Abstraktionsebenen ermöglicht, dann ist eventuell eine Prüfung der Konsistenz der Beschreibungen auf den verschiedenen Abstraktionsebenen denkbar. Auf diese Weise ließe sich prüfen, ob die angebliche Verfeinerung eines Systementwurfes wirklich eine Verfeinerung darstellt, ob also die in der abstrakteren Darstellung geforderten Eigenschaften von dem konkreteren Modell aufgewiesen werden.
Tätigkeiten und Anforderungen:
  * 3.1.3.1. In Fällen mit besonderen Gefahren im Falle der Nichtübereinstimmung einer detaillierteren Spezifikation mit der abstrakteren könnte das Projektteam eine formale Verifikation der Konsistenz der verschieden weit detaillierten Entwürfe anstreben. Dies wird durch eine Notation mit exakt definierter Semantik erleichtert, die zudem die Modellierung des Systems auf den ver-

schiedenen Abstraktionsstufen unterstützt.

- **3.2. Feinentwurf:** Im Feinentwurf wird die Struktur der beim Grobentwurf identifizierten Teilmodule genauer festgelegt. Hier werden exakte Definitionen der Systemkomponenten vorgenommen. Das Ergebnis des Feinentwurfs sind Entwurfsdokumente eines hohen Detaillierungsgrades, die nur noch wenige Entwurfsentscheidungen für die eigentliche Implementierung offen lassen.

  Auch im Feinentwurf fallen verschiedene Teilaufgaben an, die sich durch die Benutzung geeigneter Spezifikationsnotationen unterstützen lassen:

  - **3.2.1.1. Detaillierte Spezifikation der zu implementierenden Teilmodule:** Die Eigenschaften der im Grobentwurf identifizierten Teilmodule sind detailliert zu spezifizieren.
    Tätigkeiten und Anforderungen:
    * 3.2.1.1. Das Projektteam nimmt eine detaillierte Spezifikation der beim Grobentwurf identifizierten Module vor. Hier spielt insbesondere die exakte Festlegung der Semantik der Spezifikationsnotation eine besondere Rolle, da hier nicht mehr nur die grobe Festlegung der Konzepte und Verantwortlichkeiten der Komponenten genügt.
  - **3.2.2. Festlegung der Schnittstellen der zu implementierenden Teilmodule:** Ergebnis des Feinentwurfs sind nicht nur Beschreibungen der von den einzelnen Modulen umzusetzenden Funktionalität, sondern auch Festlegungen der Schnittstellen zu anderen Modulen.
    Tätigkeiten und Anforderungen:
    * 3.2.2.1. Das Projektteam definiert die Schnittstellen der zu implementierenden Teilmodule. Die hierfür benutzte Notation wird sich weitgehend an die zu benutzende Programmiersprache anlehnen, am günstigsten wird hier direkt eine Benutzung von Sprachelementen dieser Programmiersprache sein.
  - **3.2.3. Erarbeitung einer Implementierung für die Teilmodule durch schrittweise Verfeinerung:** Die detaillierte Spezifikation der Teilmodule beschreibt die Aufgaben, nicht aber die Art und Weise, wie diese Aufgaben gelöst werden. Für jedes Teilmodul werden daher schrittweise die Anforderungen in Entwurfsentscheidungen umgesetzt. Dabei sind die Verfeinerungen auf Korrektheit zu überprüfen.
    Tätigkeiten und Anforderungen:
    * 3.2.3.1. Für jedes der exakt definierten Module wird das Projektteam aus der detaillierten Spezifikation mit Hilfe schrittweiser Verfeinerung eine Implementierung erarbeiten. Dies wird durch eine Notation unterstützt, die den konzeptionellen Abstand zwischen der Spezifikation und der Programmiersprache in mehreren Schritten zu überwinden gestattet, da sich nur so die Erhaltung wichtiger Eigenschaften der Spezifikation auf dem Weg hin zur Implementierung mit erträglichem Aufwand nachweisen läßt. Im Extremfall könnte das Projektteam hier wiederum einen formalen Nachweis der Übereinstimmung der verschiedenen Verfeinerungsstufen anstreben, der eine exakt festgelegte Semantik sowie eine Werkzeugunterstützung für die benutzte Spezifkationsnotation voraussetzt.

- **3.3. Benutzerhandbücher:** Auch die Anfertigung von Handbüchern für die künftigen Benutzer des entstehenden Systems ist eine wichtige Aufgabe im Verlauf des Programmentwicklungsprozesses. Auch diese Aufgabe kann durch eine Spezifikationsnotation unterstützt werden:

  Identifikation von typischen Systeminteraktionen und Beschreibung ihrer Durchführung: Benutzerhandbücher müssen die verschiedenen Interaktionsmöglichkeiten beschreiben, die ein Benutzer beim Umgang mit einem System hat. Wenn die Anforderungsanalyse ein Dokument erzeugt hat, dem sich die typischen Interaktionen des Benutzers mit dem zugrundeliegenden System bereits auf einfache Weise entnehmen lassen, so ist diese Aufgabe bei der Anfertigung eines Benutzerhandbuches durch die Anforderungsanalyse bereits im wesentlichen vorbereitet.

  Tätigkeiten und Anforderungen:

  - 3.3.1. Identifikation von typischen Systeminteraktionen
    - 3.3.1.1. Das Projektteam beschreibt die verschiedenen Interaktionsmöglichkeiten des Benutzers mit dem System. Eine Spezifikation der Anforderungen an das System kann bei der Überprüfung der Vollständigkeit des Benutzerhandbuchs helfen. Bei der Anfertigung des Benutzerhandbuchs können spezielle Konventionen und Notationen hilfreich sein, die die verschiedenen Interaktionsmöglichkeiten des Benutzers mit dem System ordnen können.
    - 3.3.1.2. Die künftigen Anwender prüfen die Verständlichkeit des Benutzerhandbuchs. Nicht jeder Benutzer wird das Benutzerhandbuch ständig einsehen müssen. Zur Dokumentation der Interaktionen für das Benutzerhandbuch eignen sich sich also nur solche Notationen, die auch dem Benutzer, der nur sporadisch das Benutzerhandbuch einsieht, verständlich sind.

- **3.4. Implementierung:** Mit dem Begriff „Implementierung“ wird die Umsetzung der im Feinentwurf detailliert spezifizierten Module in Programmcode beschrieben. Auch diese Aufgabe kann durch die Benutzung geeigneter Spezifikationsnotationen unterstützt werden:

  - **3.4.1. Umsetzung der im Feinentwurf spezifizierten Module:** Wenn der Feinentwurf bereits in einer Notation dokumentiert wurde, der einer Programmiersprache ähnelt, so kann die Umsetzung der Spezifikationen in Programmcode dadurch wesentlich vereinfacht sein, eventuell sind die Spezifikationen sogar bereits selbst ausführbar.
    Andererseits kann die Implementierung der im Feinentwurf erstellten Modulspezifikationen durch eine Notation, deren Sprachmittel bestimmte Implementierungen suggeriert, erschwert werden, wenn diese Implementierungen in der zu benutzenden Programmiersprache schwierig oder nur ineffizient umzusetzen sind.
    Tätigkeiten und Anforderungen:
    - 3.4.1.1. Das Projektteam setzt die während des Feinentwurfs entwickelte detaillierte Spezifikation jedes Moduls in Programmcode um. Dies kann durch eine Spezifikationssprache erleichtert werden, die den konzeptuellen Schritt von der detaillierten Spezifikation zur Programmiersprache klein hält. Suggeriert die Spezifikationsnotation aber Sprachbestandteile, die in der gewählten Imple-

mentationssprache nicht vorliegen, so kann die Implementierung durch diese Wahl der Notation sogar erschwert werden.

– **3.4.2. Quellcodedokumentation:** Die Quellcodedokumentation dient der Lesbarkeit der Implementierung. Diese ist wichtig, wenn Fehler gesucht werden oder wenn das entstehende Programm später geändert werden soll.
Da eine Implementierung gewöhnlich viele Details enthält, die die abstrakteren Ideen, auf denen ein Entwurf beruht, nicht leicht deutlich werden lassen, ist es für die Verständlichkeit des Programmtextes wichtig, daß auch die abstrakteren Konzepte im Programmtext wiedergefunden werden. Dies kann ermöglicht werden, wenn sich die Spezifikationsnotation zur lesbaren Wiedergabe im Quelltext eignet.
Tätigkeiten und Anforderungen:
  * 3.4.2.1. Das Projektteam muß darauf achten, daß der Quellcode lesbar bleibt. In diesem spiegeln sich jedoch nicht nur die Konzeptionen, auf denen der Entwurf beruht, sondern vielerlei technische Beschränkungen und implementationsbedingte Details. Zur Erhöhung der Lesbarkeit des Quellcodes ist es daher hilfreich, wenn dieser die abstrakteren Darstellungen noch enthält, in denen die klaren Ideen noch nicht durch die technischen Details überdeckt wurden. Eine Spezifikationsnotation, die sich zur Wiedergabe im Quelltext des Programms eignet, kann daher die Dokumentation des Entwurfsprozesses vereinfachen und damit eine verbesserte Lesbarkeit des Programmtextes ermöglichen.

4. **Test:**

- **4.1. Modul- und Integrationstest:** Beim Modul- und Integrationstest geht es darum, die Funktionalität der implementierten Einzelmodule sowie der Module, die aus Teilmodulen bestehen, anhand von Beispielen zu überprüfen. Auch diese Prüfung läßt sich durch die Wahl bestimmter Spezifikationsnotationen unterstützen. Es lassen sich zwei wichtige Teilaufgaben unterscheiden: der Entwurf von Testfällen, und die eigentliche Durchführung der Tests.

  – **4.1.1. Entwurf von Testfällen für die implementierten Module:** Für den Entwurf von Testfällen für die Prüfung eines implementierten Moduls ist nötig, nach Anwendungsfällen für das implementierte Modul zu suchen, die verschiedene Bereiche der Funktionalität des Moduls abdecken. Dafür kann man, wenn man beispielsweise an einen Black-Box-Test denkt, verschiedene Kriterien unterscheiden: Test mit einigen typischen Eingaben des Moduls; Test mit einigen untypischen, aber zulässigen Eingaben an das Modul; Test mit unzulässigen Eingaben an das Modul zur Prüfung der Robustheit.
  Diese Auswahl von verschiedenen Eingaben kann durch die Spezifikation unterstützt werden, wenn diese typische von untypischen und unzulässigen Eingaben explizit unterscheidet.
  Tätigkeiten und Anforderungen:
    * 4.1.1.1. Die Testplaner entwerfen Testfälle für jedes implementierte Modul. Dabei werden Testfälle für typische Eingaben an das Modul, für erlaubte untypische Eingaben an das Modul und für

nichterlaubte Eingaben an das Modul benötigt (mit diesen wird die Ausnahmebehandlung geprüft). Die Generierung von Testfällen wird erleichtert, wenn die Spezifikation des Moduls die Unterscheidung von typischen und von untypischen Eingaben vornimmt und wenn sie auch die Ausnahmebehandlungen für das Modul definiert.

- **4.1.2. Durchführung der Tests:** Bei der Durchführung der Tests ist der Vergleich der tatsächlichen Ergebnisse mit den erwarteten wichtig. Dafür muß der Spezifikation entnommen werden können, welche Ausgaben bei welchen Eingaben an das System erwartet werden.
  Tätigkeiten und Anforderungen:
  - 4.1.2.1. Die Testdurchführer benutzen die von den Testplanern entworfenen Testfälle zur Prüfung der implementierten Module. Dafür müssen Spezifikationen der Testfälle vorliegen. Zusätzlich müssen die erwarteten Ausgaben der Module spezifiziert sein, eventuell mit Grenzen für zulässigen Zeit- und Speicherbedarf. Die Testdurchführung kann also durch eine Notation unterstützt werden, die die wesentlichen zu prüfenden Eigenschaften der Testfälle zu spezifizieren gestattet.

- **4.2. Systemtest, Abnahmetest:** Der System- und der Abnahmetest sind Tests der Gesamtfunktionalität des Systems. Als Systemtest wird dabei ein Test bezeichnet, der vom Auftragnehmer selbst durchgeführt wird, als Abnahmetest ein solcher Test durch den Auftraggeber.

  Auch für System- und Abnahmetest lassen sich die zwei Teilaufgaben unterscheiden, die auch schon bei Modul- und Integrationstests wichtig waren, nämlich die Entscheidung, mit welchen Testfällen der Test durchgeführt werden soll, und die Durchführung der Tests mit der Prüfung, welche Ausgaben erwartet werden.

  Tätigkeiten und Anforderungen:

  - 4.2.1. Testplaner gibt es sowohl auf Seiten des Auftragnehmers für den Systemtest, als auch auf Seiten des Auftraggebers für den Abnahmetest. Die Anforderungsspezifikation kann eine Grundlage für die Erarbeitung der Testfälle bilden.
  - 4.2.2. Die Durchführer der Tests vergleichen die Systemreaktionen in den geplanten Testfällen mit den richtigen Reaktionen des Systems. Dies wird erleichtert, wenn die korrekten Systemreaktionen aus der Anforderungsspezifikation leicht zu entnehmen sind.

5. **Produktiver Einsatz:** An die Entwicklung des Systems schließt sich der Übergang zu produktiven Einsatz des entwickelten Systems an. Auch hier lassen sich verschiedene Teilaufgaben unterscheiden: Die Einführungsphase, und die Einsatzphase selbst.

   - **5.1. Einführung des Systems:** Die Einführung eines neuen Systems macht oft einen Pilotbetrieb oder einen Parallelbetrieb sinnvoll. Diese Phase läßt sich als ausgedehnter Test unter Betriebsbedingungen verstehen.

     Es ist zu erwarten, daß in diesem Pilotbetrieb eine Vielzahl von zusätzlichen Anforderungen an das System deutlich wird, insbesondere dann, wenn es keine Prototypen des entstehenden Systems gegeben hat,

oder wenn die Prototypen nicht intensiv geprüft worden sind. Zudem werden in den Tests unentdeckt gebliebene Fehler auftreten.

Aus diesem Grund ist es wichtig, Fehler und Änderungsanforderungen systematisch zu erfassen und zu priorisieren. Diese Priorisierung kann erleichtert werden, wenn der Anforderungsspezifikation entnommen werden kann, welche Anforderungen an das System besonders wichtig sind.

Die nachträglichen Änderungen der Anforderungen müssen dokumentiert werden. Dabei ist darauf zu achten, daß die Anforderungen konsistent bleiben.

Tätigkeiten und Anforderungen:

- 5.1.1. Das Projektteam muß Änderungswünsche, die bei der Einführung des Systems oder im Rahmen eines Pilotbetriebs bekannt werden, in die Anforderungsspezifikation einarbeiten. Dabei ist darauf zu achten, daß die Konsistenz gewahrt bleibt. Dies wird erleichtert, wenn die zur Dokumentation der Änderungswünsche benutzte Notation der Notation entspricht, in der die Anforderungsspezifikation vorliegt.

- **5.2. Pflege:** Wenn das System im Pilotbetrieb ausreichend getestet wurde, kann es für den täglichen Einsatz benutzt werden. In dieser Phase nun liegt ein relativ gut getestetes System vor, bei dem jedoch Benutzungsprobleme, Fehler und Anforderungsänderungen auftreten können. Damit sind die folgenden Teilaufgaben zu lösen:
  - **5.2.1. Benutzerberatung:** Eine für die Benutzer gut verständliche Systemdokumentation kann Unklarheiten beseitigen helfen und persönlichen Beratungsbedarf bei den Anwendern vermindern. Hier kann eine Notation helfen, die für die Anwender leicht verständlich ist. Zusätzlich ist gegebenenfalls eine Beratungsstelle einzurichten.

    Tätigkeiten und Anforderungen:

    * 5.2.1.1. Das Projektteam muß die Grundlage einer guten Benutzerberatung schaffen, indem es den Anwendern und einer eventuellen Beratungsstelle gute Handbücher zur Verfügung stellt. Die Handbücher müssen zwei Bedingungen erfüllen: Sie müssen die Verhaltensweise und die Benutzung des Systems auf korrekte Weise wiedergeben, und sie müssen für die zukünftigen Anwender und die Mitglieder der Beratungsstelle verständlich sein. Die Lösung dieser Aufgabe wird dem Projektteam erleichtert, wenn sich die Notationen, die für die Systemspezifikation und für das Benutzerhandbuch benutzt werden, nicht allzu stark voneinander unterscheiden.
    * 5.2.1.2. Die zukünftigen Anwender und die Mitglieder einer Beratungsstelle müssen die Benutzerdokumentation verstehen. Wenn die Benutzerdokumentation nur sporadisch benutzt wird, so ist wesentlich, daß nur Notationen eingesetzt werden, die dem Zielpersonenkreis vertraut sind. Gleichzeitig muß die Benutzerdokumentation vollständig sein, also alle in der Praxis nötigen Interaktionsmöglichkeiten mit dem System beschreiben.
  - **5.2.2.1. Nachträglicher Änderungsbedarf:** Fehler, die nach der Einführung des Systems auftreten, müssen gegebenenfalls identifiziert, erfaßt und priorisiert werden. Die Systemspezifikation kann

dabei helfen, zu entscheiden, ob ein bestimmtes Systemverhalten als Fehler anzusehen ist oder nicht, und sie kann dabei helfen, klare Priorisierungskriterien zu finden.

Tätigkeiten und Anforderungen:

* 5.2.2.1. Die Anwender werden auch nach der Einführung des Systems Änderungsbedarf anmelden, indem sie etwa Fehler und Anforderungsänderungen melden. Das Pflegeteam muß jeweils entscheiden, ob ein bestimmtes Systemverhalten als Fehler zu werten ist oder nicht. Wenn ein Fehler erkannt wurde oder eine Programmänderung aufgrund einer Anforderungsänderung zugestanden wurde, muß das Pflegeteam diesen Fehler oder diesen Änderungsbedarf dokumentieren und priorisieren. Die Priorisierung wird erleichtert, wenn in der Anforderungsspezifikation zentrale von peripheren Anforderungen an das System unterschieden werden, so daß auf dieser Basis entschieden werden kann, mit welcher Priorität ein Änderungsbedarf zu behandeln ist.

– **5.2.3. Behebung von Fehlern und Bearbeitung von Änderungswünschen:** Auch bei der Behebung der nach Einführung des Systems aufgetretenen Fehler kann die Systemspezifikation hilfreich sein, indem sie etwa bei der Lokalisierung des Fehlers im Sourcecode benutzt werden kann, weil etwa die Dokumentation dabei hilft, die relevanten Programmteile zu identifizieren.

  Tätigkeiten und Anforderungen:

  * 5.2.3.1. Das Pflegeteam muß darauf achten, daß infolge von Fehlerkorrekturen und Anforderungsänderungen die Systemdokumentation konsistent gehalten wird. Insbesondere darf etwa nicht nur der Programmcode geändert werden, ohne daß die zugehörigen Entwürfe angepaßt werden. Dies wird erleichtert, wenn die zu einem Stück Quellcode gehörigen Entwürfe leicht zugänglich sind. Am besten wäre es wohl, wenn Quellcode, Entwurfsdokumente, Codedokumentation und Benutzerdokumentation integriert wären.

– **5.2.4. Durchführung von Regressionstests vor Benutzung der neuen Version:** Bekanntlich besteht bei Programmcodeänderungen aufgrund von Fehlerkorrekturen eine große Gefahr, daß weitere Fehler eingeführt werden. Zum einen muß daher eine solche Änderung am Programmcode sehr sorgfältig und unter Heranziehung möglichst vieler Zusatzinformationen erfolgen, damit in einem Review geprüft werden kann, ob die Änderung zu Folgefehler führen kann. Zum anderen ist nach einer solchen Fehlerkorrektur ein Regressionstest nötig, der die Erfüllung der Gesamtfunktionalität im Anschluß an eine solche Änderung des Quellcodes prüft.

  Tätigkeiten und Anforderungen:

  * 5.2.4.1. Das Pflegeteam muß nach der Korrektur von Fehlern oder nach der Entwicklung von Entwurfs- und Codeänderungen aufgrund veränderter Anforderungen gründliche Tests der Funktionalität durchführen, bevor eine neue Version des Programms in Benutzung genommen werden kann. Eine sorgfältige Dokumentation der auszuführenden Testfälle zusammen mit den erwarteten

Ergebnissen kann diese Tests vereinfachen.

Eine solche Liste von Teilaufgaben kann nicht vollständig sein. Es wurde aber ein relativ breites Spektrum von Aufgaben identifiziert, so daß auf dieser Grundlage eine Analyse verschiedener Spezifikationsnotationen auf ihre Anwendungsbereiche hin möglich ist.

3 Ergebnisse

Bei der Analyse der im vorigen Aubschnitt gesammelten Tätigkeiten und Anforderungen lassen sich verschiedene Klassen von Anforderungen unterscheiden. Manche der Anforderungen betreffen soziale Funktionen von Spezifikationsnotationen, bei anderen steht die Bedeutung für soziale Interaktionen nicht im Vordergrund. Dies schließt aber nicht aus, daß auch diese Anforderungen für soziale Interaktionen eine Rolle spielen können.

Bei den weniger an sozialen Interaktionen orientierten Anforderungen lassen sich solche finden, bei denen eine Prüfungsfunktion überwiegt, während bei anderen ein Konstruktionsaspekt im Vordergrund steht.

Insgesamt ergibt sich damit das folgend angegebene Schema. Für jede der Anforderungsklassen werden Beispiele angegeben.

- Anforderungen mit primär sozialen Funktionen
  - Vermeidung von Mißverständnissen durch Eindeutigkeit, Unterstützung der Kommunikation durch Klarheit, leichte Verständlichkeit und leichte Einsehbarkeit der Bedeutung für die tägliche Arbeit: 1.2.2., 2.1.1., 2.1.2., 2.2.2., 2.2.3., 2.4.1., 2.6.2., 3.1.1.2., 3.1.2.2., 3.3.1.2., 4.2.2.
  - eindeutige, eventuell juristisch verwendbare Zuordnung von Verantwortlichkeiten zu Personen(-gruppen); eindeutige Festlegung von Schnittstellen von Teilsystemen: 2.3.1., 2.3.2.
- Anforderungen ohne primär soziale Funktionen
  - Anforderungen mit primärem Bezug auf Konstruktion
    - Wiederbenutzung von Spezifikationselementen für die Anfertigung des Benutzerhandbuchs, für die Planung von Testfällen: 3.3.1.1., 4.1.1.1., 4.2.1., 5.2.1.1.
    - Erhaltung der Konsistenz aufeinander bezogener Systembestandteile durch gemeinsame Pflege: 3.4.2.1., 5.2.2.2.
  - Anforderungen mit primärem Bezug auf Prüfung
    - Gliederung einer Gesamtspezifikation in Sinneinheiten: 1.4.1., 1.5.
    - explizite Repräsentierbarkeit der zentralen Konzepte (Beispiele: Bedarf, Aufwand, Planungsaufgaben, kritische Entwurfsphasen, Priorität bei Fehlern etc.), Fokussierung der Aufmerksamkeit durch Unterscheidung von Zentralem und Peripherem: 1.1.1., 1.2.1., 1.3.1., 1.3.2., 1.4.2., 2.1.1., 2.4.2., 2.5.1., 2.5.2., 3.1.1.1., 3.3.1.1., 4.1.1.1., 4.1.2.1., 4.2.1., 4.2.2., 5.2.2.1., 5.2.4.1.

- leichte Erlernbarkeit, evtl. auch für DV-Laien: 1.1.1., 2.1.2., 2.2.1, 3.1.1.2., 3.3.1.2., 5.2.1.2.
- Abdeckung mehrerer Abstraktionsstufen: 3.1.2.1., 3.1.3.1., 3.2.2.1., 3.2.3.1., 3.4.1.1., 5.1.1.
- Benutzbarkeit in mathematischen Beweisen durch formale Exaktheit, in Konsistenzprüfungen, Prüfung Erfüllung kritischer Anforderungen an das System, Entdeckung von Definitionslücken: 2.2.1., 2.6.1., 3.1.3.1., 3.2.1.1., 3.2.3.1.

Aus dieser Anordnung der gefundenen Ergebnisse lassen sich für das zugrundegelegte Modell der Softwareentwicklung die folgenden Schlüsse ziehen:

- Die sozialen, dabei insbesondere die kommunikativen Funktionen von Spezifikationsnotationen sind von zentraler Bedeutung.
- Spezifikationsnotationen spielen insbesondere für Prüfungsaufgaben die zentrale Rolle. Für Konstruktionsaufgaben sind sie weniger wichtig.
- Formale Exaktheit ist nur an wenigen Stellen von besonderer Wichtigkeit, und auch dies nur bei bestimmten Anforderungen.
- Viele nicht auf formaler Exaktheit basierende Eigenschaften von Spezifikationsnotationen können die individuelle Prüfung von Eigenschaften von Entwürfen erleichtern.

Der erste und der letzte dieser Punkte belegen gemeinsam die anfangs formulierte These, daß die Unterstützung von Kommunikations- und Kognitionsakten eine zentrale Anforderung an Spezifikationsnotationen ist. Einschränkend ist zu sagen, daß diese These in dieser Arbeit nur für Projekte bestätigt wurde, deren Struktur in etwa die bei der Identifikation der Einsatzsituationen aufgezählten Teilaufgaben aufweist.

4 Folgerungen und Beispiel

Wir ziehen einige Folgerungen für die Auswahl von Formalismen. Einerseits erfolgt eine solche Auswahl natürlich anwendungsgebietsspezifisch, vergleiche etwa Fraser et al.[FKV94]. Beispiel: Z ist eher geeignet für Datenbanken und Transformationssysteme, schlechter für reaktive Systeme; für Statecharts gilt umgekehrtes.

Bei Beachtung unserer These, die Kommunikation und Reflexion betont, können wir andere Folgerungen ziehen:

- Eine Spezifikationsmethode darf nicht unter Absehung von den spezifischen Personen ausgewählt werden, die mit ihr arbeiten sollen. Wenn den Personen eine geeignete Spezifikationsmethode vertraut ist, so muß das natürlich genutzt werden. Wenn den Personen keine geeigneten Spezifikationsmethoden vertraut sind, dann muß nach einer solchen gesucht werden, die für sie leicht zu lernen ist, weil die Konzepte, auf denen die Notation aufbaut, den diesen Personen vertrauten Konzepten entsprechen. In jedem Fall muß Vertrautheit mit einer Notation gegen Eignung der Notation für einen vorliegenden Anwendungsfall abgewogen werden.

- Für neuartige Notationen darf nicht nur ihre technische Eignung für die Beschreibung von Modellen des Anwendungsbereichs berücksichtigt werden, sondern auch die konzeptionelle Nähe der Grundbegriffe zur Begriffswelt der beteiligten Personen.

- Für die Funktion einer Notation als Kommunikationsmittel und Reflexionshilfe können Eigenschaften, die in semantischer Hinsicht irrelevant sind, wie etwa die Darstellung einer Spezifikation in textueller oder in graphischer Form, eine erhebliche Rolle spielen.

Abschließend wenden wir zum Zwecke der konkreten Illustration unserer Konzepte diese Folgerungen auf das Gebiet der Spezifikation von Anforderungen an reaktive Systeme an.

Für die Spezifikation von Eigenschaften solcher Systeme werden sehr verschiedene Notationen vorgeschlagen (vergleiche etwa [LL95]):

- Temporale Logiken erlauben die Formulierung von Anforderungen an reaktive Systeme durch modallogische Formeln [MP92].

- Prozeß-Algebren nutzen hochabstrakte Programmiersprachen, die von jedem Verhalten außer der Kommunikation von Prozessen abstrahieren [Hoa85, Mil89].

- Der Statecharts-Ansatz modelliert reaktive Systeme als System kommunizierender und hierarchisch organisierter endlicher Automaten modelliert [Har87]. Er nutzt eine graphische Notation. Argos [Hal93] ist eine Umsetzung der wichtigsten Statecharts-Ideen im Rahmen einer exakt definierten synchronen Semantik.

- SDL erlaubt die Modellierung eines reaktiven Systems als aus mehreren Komponenten aufgebaut, die je einzeln durch einen endlichen Automaten beschrieben sind und asynchron miteinander kommunizieren.

- Esterel [Hal93] ist eine synchrone Programmiersprache, deren Syntax an herkömmliche imperative Programmiersprachen angelehnt ist.

- Lustre [HCP91, Hal93] ist eine synchrone Programmiersprache, die Datenflußkonzepte unterstützt.

- VHDL und Verilog sind Notationen, die zur Beschreibung von digitalen Schaltkreisen eingesetzt werden. Sie gehören zu den synchronen Sprachen. Sie erlauben sie sowohl die bloß verhaltensmäßige als auch die strukturelle Beschreibung von Funktionsgruppen.

- Auch herkömmliche Programmiersprachen erlauben gewöhnlich die die Abarbeitung mehrerer Prozesse und deren Kommunikation untereinander. Hierbei werden wesentliche Funktionen oft vom Betriebssystem übernommen. Ein Nachteil ist die übberreiche Semantik der Notationen, die automatische Verifikationen erschwert. Daher wird dieser Ansatz gewöhnlich nicht unter die „formalen Methoden“ gezählt. Bei eindeutig definierter Semantik der Programmiersprache und der Kooperationsmechanismen sind die Ansprüche der Formalität aber auch hier erfüllt.

Wir nehmen für unser Beispiel an, daß die Entwickler gemeinsam mit einem Auftraggeber ein reaktives System entwerfen wollen, etwa eine Steuerung für die Fertigungszelle aus [LL95]. Wir prüfen, welche der Notationen zur Unterstützung von Reflexion und Kommunikation für die beteiligten Personen geeignet ist. Wir gehen davon aus, daß keine der Notationen den beteiligten Personen bereits vertraut ist. Daher ist zu prüfen, bei welchen wir mit einer besonders problemarmen Aneignung rechnen dürfen. Dies kann nicht ohne Befragung der beteiligten Personen geschehen. Nehmen wir an, sie seien teils C-Programmierer, teils DV-Laien.

Wir erwarten, daß sich C-Programmierer schnell in die Verwendung von herkömmlichen Programmiersprachen für die Programmierung reaktiver Systeme eindenken können. Auch Esterel dürfte, als imperative Programmiersprache, zu den rascher anzueignenden Notationen gehören. Schwierig dürfte die Benutzung der allzu abstrakten Temporalen Logiken und der Prozeß-Algebren sein. Die Einarbeitung in VHLD oder Verilog wird leichter sein, aber nicht so leicht wie die Benutzung von Esterel. Bei Lustre wird das Datenflußkonzept zunächst gewöhnungsbedürftig sein.

Für den DV-Laien werden all diese Notationen schwierig zu benutzen sein. Auch die programmiernahen Notation, die dem C-Entwickler vertraut sind, sind dem DV-Laien fremd, da er über Konzepte wie Variable, Zuweisung oder Prozedur nicht verfügt.

SDL und Statecharts rsp. Argos stellen einen Sonderfall dar. Die graphischen Notationen, die bei SDL für Flußdiagramme und in Statecharts für die Beschreibung der hierarchisch und nebenläufig organisierten endlichen Automaten benutzt werden, stellen den Kontrollfluß, der bei reaktiven Programmen gewöhnlich die Hauptquelle der Komplexität darstellt, in einer für viele Menschen intuitiv gut faßlichen Weisen dar. Damit erwarten wir, daß sich bei der Kommunikation mit Laien diese Notationen als besonders hilfreich erweisen.

5 Zusammenfassung

Wir haben begründet, warum wir für die Benutzung formaler Notationen im Softwareentwicklungsprozeß die Berücksichtigung von Kommunikations- und Reflexionsfunktionen für unerläßlich halten. Wir haben wir ein Prozeßmodell für die Entwicklung kundenspezifischer Software herangezogen und durch Identifikation von Rollen, die in den verschiedenen Aufgaben wichtig werden, die Kommunikationsfunktion von Spezifikationen thematisierbar gemacht. Am Beispiel der Auswahl einer Spezifikationsnotation für reaktive Systeme haben wir unsere Auswahlkriterien illustriert.

Literatur

[ACJ+96] Mark A. Ardis, John A. Chaves, Lalita Jategaonkar Jagadeesan, Peter Mataga, Carlos Puchol, Mark G. Staskauskas und James Von Olnhausen. A Framework for Evaluating Specification Methods for Reactive Systems. *IEEE Transactions on Software Engineering*, 22(6):378–389, Juni 1996.

[BH95a] Jonathan P. Bowen und Michael G. Hinchey. Ten Commandments of Formal Methods. *IEEE Computer*, 28(4):56–63, April 1995.

[BH95b] J.P. Bowen und M.G. Hinchey. Seven More Myths of Formal Methods. *IEEE Software*, 12(4):34–41, Juli 1995.

[dR91] W.-P. de Roever. Foundations of Computer Science: Leaving the Ivory Tower. *EATCS Bulletin*, Seiten 455–492, Juni 1991.

[FKV94] Martin D. Fraser, Kuldeep Kumar und Vijay K. Vaishnavi. Strategies for Incorporating Formal Specifications in Software Development. *Communications of the ACM*, 37(10):74–86, Oktober 1994.

[Flo85] Christiane Floyd. On the Relevance of Formal Methods to Software Development. *Proc. TAPSOFT/Formal Methods and Software Development:Berlin*, 2:1–11, März 1985.

[GHH+92] Chris George, Peter Haff, Klaus Havelund, Anne E. Haxthausen, Robert Milne, Claus Bendix Nielsen, Søren Prehn und Kim Ritter Wagner. *The RAISE Specification Language*. Prentice Hall, New York, 1992.

[Hal90] A. Hall. Seven Myths of Formal Methods. *IEEE Software*, Seiten 11–19, September 1990.

[Hal93] Nicholas Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, 1993.

[Har87] David Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.

[HCP91] N. Halbwachs, P. Caspi und D. Pilaud. The synchronous dataflow programming language Lustre. *Proceedings of the IEEE*, 79(9):1305–1320, September 1991.

[Hoa85] C.A.R. Hoare. *Communcating Sequential Processes*. Prentice Hall, Hemel Hempstead, 1985.

[KDGN97] John C. Knight, Colleen L. DeJong, Matthew S. Gibble und Luís G. Nakano. Why Are Formal Methods Not Used More Widely? In *4th NASA Langley Formal Methods Workshop*, 1997.

[LHHR94] Nancy G. Leveson, Mats P. E. Heimdahl, Holly Hildreth und John D. Reese. Requirements Specification for Process-Control Systems. *IEEE Transactions on Software Engineering*, 20(9):684–707, September 1994.

[LL95] Claus Lewerentz und Thomas Lindner. *Formal Development of Reactive Systems*. Springer-Verlag, Berlin,Heidelberg, 1995.

[MDL87] H. D. Mills, M. Dyer und R. C. Linger. Cleanroom Software Engineering. *IEEE Software*, Seiten 19–24, September 1987.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall, Hemel Hempstead, 1989.

[MMM85] L. Mathiassen und A. Munk-Madsen. Formalization in Systems Development. *Proc. TAPSOFT*, 2:101–116, März 1985. Formal Methods and Software Development:Berlin.

[MP92] Zohar Manna und Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1992.

[Nau82] Peter Naur. Formalization in Program Development. *BIT Bd.22*, Seiten 437–453, 1982.

[Nau85] Peter Naur. Intuition in Software Development. *Proc. TAPSOFT: Formal Methods and Software Development*, 2:60–79, März 1985.

[Nau89] Peter Naur. The Place of Strictly Defined Notation in Human Insight. In P. Dybjer, L. Hallnäs, B. Nordström, K. Petersson und J. M. Smith, Hrsg., *Proc. of the Workshop on Programming Logic*, Seiten 429–443, Göteborg, Mai 1989.

[Par77] D. L. Parnas. The Use of Precise Specifications in the Development of Software. *IFIP Congress Proceedings*, Seiten 861–867, 1977.

[Rus94] Heinrich Rust. *Zuverlässigkeit und Verantwortung*. Vieweg, Braunschweig,Wiesbaden, 1994.

[Win90] J. M. Wing. A Specifier's Introduction to Formal Methods. *Computer*, Seiten 8–24, September 1990.