Janert, Norman
Max-Planck-Ring 8c L63
98693 Ilmenau
Telefon: 03677 896103
E-Mail: NJanert@GMX.de
Studiengang: Wirtschaftsinformatik
Matrikel-Nr.: 27351

Dokumentation im Bereich der Softwareentwicklung

Studienarbeit
am Institut für Theoretische und Technische Informatik
der Technischen Universität Ilmenau

Fachbereich Prozessinformatik
Betreuender Hochschullehrer: Prof. Dr.-Ing. habil. Ilka Philippow
Weiterer Betreuer: Dipl.-Inf. Detlef Streitferdt

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Einleitung

Im Gegensatz zu dem viel beachteten und sehr populären Thema der Sotwarequalität wird dem Bereich der Dokumentationsqualität, trotz der allgemein anerkannten negativen Auswirkungen mangelhafter Dokumente, nur eine geringe Beachtung beigemessen. Dies ist beispielsweise an der geringen Menge von wissenschaftlichen Publikationen über dieses Thema und den wenigen existierenden, aber veralteten, Ausarbeitungen zu erkennen. Aus diesem Grund ist es nicht verwunderlich, dass sich nur wenige brauchbare standardisierte Instrumentarien der Qualitätsmessung und Standardisierung der Softwaredokumentation finden lassen und das darüber hinaus keine international anerkannte Prozessbeschreibung der Dokumentationsvorgänge existiert.

In der gesamten Computer- und Softwareindustrie ist das Problem mangelhafter Dokumentation seit langem bekannt und immer wieder viel diskutiert worden. Insbesondere für die Endanwender der erstellten Produkte ist eine qualitativ hochwertige Dokumentation von großer Bedeutung, da diese eine Produktivsetzung der Software ohne Schwierigkeiten oder Verzögerungen ermöglicht. Eine gute Dokumentation verringert nicht nur die Aus- und Weiterbildungskosten des eingesetzten Personals, sondern baut darüber hinaus Hemmschwellen gegenüber der Anwendung ab. Die positiven Auswirkungen ausgefeilter Dokumentationsstrategien beschränken sich jedoch nicht nur auf die Zielgruppe des Produktes, sondern wirken sich direkt auf die Qualität und Durchführung der Softwareerstellung und -wartung aus.

1.2 Zielstellung und Vorgehensweise

Da die Erstellung einer qualitativ hochwertigen Dokumentation, vor allem in kleinen und mittleren Softwareprojekten, einen nicht zu unterschätzenden Aufwand darstellt, soll in der folgenden Arbeit untersucht werden was unter dem Begriff Softwaredokumentation zu verstehen ist. In diesem Rahmen ist es, neben einer genauen Begriffsbestimmung, unumgänglich das Umfeld des Softwaredokumentationsprozesses genauer zu untersuchen, um eine zielgruppenbezogene Dokumentationserstellung zu gewährleisten und diese in die jeweilige organisatorische Unternehmensstruktur einzuordnen. In einem weiteren Schritt müssen die Arten der zu erstellenden Dokumente ermittelt werden um sie den einzelnen Phasen der Softwareerstellung zuordnen zu können.

Ein weiteres Ziel dieser Untersuchungen ist die Ausarbeitung qualitativer Anforderungen an die Softwaredokumentation. Um diese objektiv messen zu können, ist es notwendig aktuelle Standards zu betrachten und untereinander in Relation zu setzen.

Abschließend soll die Unterstützung der Umsetzung des erarbeiteten Rahmenwerkes der Softwaredokumentation durch aktuelle standardisierte Werkzeuge und Programme analysiert werden, um zu ermitteln wie und ob die gestellten Anforderungen an die Dokumentation automatisiert realisiert werden können.

2 Theoretische Betrachtungen

2.1 Begriffsklärung

Der Dokumentationsbegriff ist historisch eng mit dem Bibliotheks- und Dokumentationswesen verbunden. Aus diesem Grund ist in den bisherigen Publikationen oftmals eine ausschließliche Betrachtung dieses traditionellen Ansatzes zu finden, der sich hauptsächlich mit den statischen Prozessen des Sammeln, Ordnen und Speichern von Daten befasst. Erst in moderneren Betrachtungen [1] wird eine Verbindung zwischen der Dokumentation und den zu realisierenden Informationsaufgaben hergestellt. In diesem Kontext wird die Dokumentation zunehmend als das zweckorientierte Sammeln, Erfassen, Ordnen und Aufschließen von Dokumente betrachtet.

Im Rahmen der heutigen informationstechnischen Möglichkeiten sollte der Begriff „Dokument“ in einer Losen Bindung verwendet werden, da sowohl moderne Dokumentationstechniken und -medien zur Verfügung stehen, als auch ein konkreter Aufgabenbezug, inklusive darauf abgestimmter Zielgruppen, der Softwaredokumentation besteht.

„Der Begriff „Dokumentation“ bezeichnet umgangssprachlich sowohl den Prozess, d.h. die Tätigkeit des Dokumentierens, als auch das Produkt, d.h. das Ergebnis dieser Tätigkeit.“[Leh94, 15] Aufgrund vielfältiger Aufgaben der Dokumentation im Softwareumfeld, wie der Kommunikation aller an der Entwicklung beteiligten Personen, der Unterstützung des Projektmanagements, der Hilfestellung hinsichtlich der Produkteinführung, usw., ist eine einheitliche Begriffsdefinition nur schwer möglich. In diesem Rahmen ist in der Fachliteratur eine Konzentration auf Begriffsabgrenzungen, der Beschreibung von Zielen, auf Anwendungsbereiche sowie auf formale Anforderungen und allgemeinen Merkmalen hinsichtlich der Softwaredokumentation zu erkennen.

Aus der übergeordneten organisatorischen betrieblichen Sicht wird die Dokumentation als das Management unternehmensinterner und -externer Dokumente definiert. Unternemensinterne Dokumente stellen hierbei sowohl die klassische Ablage (z.B. Ordner) dar, als auch Mikrofilme und Datenträger für die Speicherung digitaler Informationen. Unter externen Dokumenten ist hingegen die Einbeziehung von Informationsdiensten, Bibliotheksdiensten und Datenbankanbietern zu verstehen. Das Dokumente stellt in diesem Kontext eine Informationsmenge dar, welche innerhalb der Anwendung oder des Vorgangs eine logische Einheit bildet. Diese verfügt hierbei neben einer physikalischen Struktur über eine logische Struktur, unter der die semantische Anordnung der Elemente und deren Layout verstanden wird. Die Dokumentation wird in dieser

[1] vgl. [Leh94]

Betrachtung sehr allgemein gehalten, da sie aus der unternehmensinternen Sicht eher eine Querschnittsfunktion darstellt, welche sich neben der Softwaredokumentation mit den Fragen der Archivierung, dem Einsatz von Technik, der Erfüllung gesetzlicher Auflagen und mit der Realisierung strategischer Visionen beschäftigt.

Unter der Einbeziehung der dargestellten prozess- und produktorientierten Begriffsdefinition kann der Dokumentationsbegriff in sechs verschiedene Aspekte bzw. Bedeutungsebenen eingeteilt werden. Diese bestehen aus:

- Wissenschaft (Lehre über das Dokumentationswesen)
- Dokumentationswesen (Gesamtheit aller Dokumentationssysteme)
- Dokumentationssystem (inkl. aller Schnittstellen zur Umwelt)
- Dokumentationsmethode (der Vorgehensplan für die Dokumentenerstellung)
- Dokumentationsprozess (die Tätigkeiten der Dokumentation)
- Dokumentationsprodukt (das Ergebnis der Dokumentation)

Unter Berücksichtigung dieser Aspekte definiert Scheibl [2] den Dokumentationsprozess als das Erstellen, Sammeln, Speichern, Ordnen, Auswählen, Verbreiten, Nutzbarmachen, Ändern und Vernichten von Informationen welche in Dokumenten enthalten sind. Das Dokument wird in diesem Kontext als die kleinste in der Dokumentation auftretende abgeschlossene Einheit betrachtet. Die eigentliche Dokumentation ergibt sich wiederum aus der geordneten Bündelung dieser atomaren Elemente zu einer wichtigen und verlässlichen Einzelbeschreibung der verschiedenen Bereiche der Softwareerstellung und -benutzung.

Diese sehr allgemeine und theoretische Definition des Dokumentationsbegriffes bedarf im Folgenden einer weiteren Präzisierung und Einschränkung. Dies geschieht durch die Fokussierung auf die technische Dokumentation, beispielsweise durch Hoffman und Schlummer[3]. Den Kernpunkt dieser Betrachtungen bildet die technische Dokumentation, welche als die strukturierte Sammlung aller notwendigen und zweckorientierten Informationen über ein auf technischem Wege hergestelltes Produkt und seiner Verwendung verstanden wird. Dies umfasst beispielsweise System-, Anwender und Kundendokumentation sowie Produktbeschreibungen im Rahmen der Planung, Entwicklung und des Marketing / Vertriebs. Ausgeschlossen von der technischen Dokumentation sind hingegen Dokumente wie Prospekte, Kataloge oder Preislisten, da diese mit der Produkterstellung nicht direkt in Verbindung stehen, sondern der Produktverwertung zugerechnet werden können.

[2] vgl. [Sch85]
[3] vgl. [Sch90]

Eine solche präzisierte Definition kann nun auf den Prozess der Softwareerstellung abgebildet werden. In diesem Kontext können unter der technischen Dokumentation alle Unterlagen, welche bei der Erstellung eines Programmes entstehen, verstanden werden. Um diese Unterlagen detailliert zu beschreiben, ist es im Folgenden notwendig das Umfeld der Dokumentation zu analysieren. Dies ermöglicht es die mit ihr in Verbindung stehenden Organisationseinheiten und Zielgruppen abzugrenzen und somit die Arten der auftretenden Dokumente zu ermitteln. Diese können in einem weiteren Schritt in die Abschnitte der Softwareentwicklung eingeordnet werden, um so den Prozess der Softwaredokumentation darzulegen.

2.2 Dokumentationsumfeld

Unter dem Bereich, welcher sich mit der Erstellung der Dokumentation beschäftigt, wird in der Regel ein Teil des unternehmensinternen organisatorischen Umfeldes verstanden.

Im Rahmen der Fokussierung auf den Aspekt der Softwaredokumentationserstellung wird dieser, auf die Personen und Organisationseinheiten welche mit der Erstellung von Benutzer- und Systemdokumentationen direkt oder indirekt in Verbindung stehen, eingegrenzt[4]. Hierbei stellen diese Prototypen für Rollen und Funktionen im Kontext der Softwareentwicklung dar. Um die Dokumentation in die Organisationsstruktur der Unternehmung einordnen zu können, müssen verschiedene Faktoren berücksichtigt werden. Unternehmensgröße, Projektumfang, Nutzung und Verbreitung der Software, allgemeine unternehmensinterne Gegebenheiten sowie die konkrete Projektorganisation beeinflussen den Dokumentationsprozess direkt und variieren von Unternehmen zu Unternehmen und somit von Softwareprojekt zu Softwareprojekt[5].

2.2.1 Unternehmensgröße und Projektumfang

Die beiden ersten Faktoren, Unternehmensgröße und Projektumfang, haben einen hohen Einfluss auf die Aufbauorganisation der unternehmensinternen Dokumentationsstrukturen. Um diesen Bereich genauer abzugrenzen ist es unumgänglich sich mit folgenden Fragen näher auseinander zu setzen. Wer sind die Verfasser der Dokumentation? Wer sollten die Verfasser der Dokumentation sein? Was sind die Ursachen für Widersprüche zwischen den beiden vorhergehenden Fragen? Für eine Beantwortung dieser Fragen ist es notwendig zwischen System- und Benutzerdokumentation zu unterscheiden.

Die Systemdokumentation sollte von der thematischen Sachlage betrachtet von Programmierern oder DV-Spezialisten erstellt werden, da diese am besten mit den internen Strukturen und Anforderungen der Software vertraut sind[6]. Welche Aspekte der Systemdokumentation von welchem Spezialisten abgedeckt werden, wird stark durch den Faktor der Projektorganisation festgelegt. Diese könnte idealtypisch wie folgt gestaltet sein[7]. Der Projektleiter trägt die Gesamtverantwortung des Projektes und ist dem Management des Unternehmens unterstellt, sowie rechenschafts- und ergebnispflichtig. Er plant, steuert und kontrolliert das Projekt. Die Stelle des Projektmanagers legt die konkreten Rahmenbedingungen für die Projektdurchführung fest. Diese

[4] vgl. [Rup87] / 18
[5] vgl. [Leh94] / 73
[6] vgl. [Leh94] / 79
[7] vgl. [Bal00] / 144ff

werden durch den Softwaremanager, welcher mit den Aufgaben der strategischen Realisierung, des Konfliktmanagements und der Erstellung von Problemlösungsstrategien betraut ist, dem Rechtsverantwortlichen, und dem Qualitätsbeauftragten umgesetzt. Auf der untersten Ebene der Projektorganisation sind Systemanalytiker, welche das System modellieren und Schnittstellen identifizieren, Softwarearchitekten, die den abstrakten Entwurf sowie Design und Architektur der Software erstellen, Programmierer, deren Aufgabe darin besteht die Architektur umzusetzen und konkrete Algorithmen zu implementieren, Anwendungsspezialisten, welche das Anwendungsgebiet aufgabengerecht strukturieren um so eine ganzheitliche Anwendungsschicht zu schaffen und Softwareergonomen, die als Arbeitswissenschaftler konzeptionell und experimentell tätig sind, zu finden. Jede einzelne dieser Organisationseinheiten deckt, bedingt durch ihre jeweiligen Spezialisierungen, einen Teil der Systemdokumentation ab und nimmt so direkt Einfluss auf diese.

Im Rahmen der Benutzerdokumentation sind in vielen Fällen, insbesondere bei der Erstellung von Individualsoftware, ebenfalls diejenigen Programmierer die Verfasser, welche bei der Entwicklung der Software aktiv beteiligt waren[8]. Dies ist nicht darin begründet, dass sie sich besonders für diese Arbeit interessieren oder eignen, sondern das diese Aufgaben oft einfach an sie delegiert werden. Eine solche Aufgabenverteilung kann sich jedoch negativ auf die Qualität der Benutzerdokumentation auswirken, da Programmierer meist froh sind, dass das Projekt beendet wurde und nur mehr mit reduziertem Einsatz bei der Sache sind. Entwickler fühlen sich nicht für die Dokumentation verantwortlich oder sehen sie als eine zweitrangige Aufgabe an, insbesondere wenn aufgrund eines hohen Termindrucks nur unzureichende Zeitressourcen für diese Aufgabe veranschlagt werden[9]. In großen Unternehmen existieren oft Mitarbeiter, welche auf die Erstellung von Benutzerhandbücher spezialisiert sind. Hierbei handelt es sich um speziell ausgebildete Fachleute, wie technische Autoren, Technikredaktuere sowie Instruktoren, die entweder direkt in die Projektorganisation eingebunden sind oder separaten Dokumentationsabteilungen angehören. Technische Autoren sind auf die Erstellung von Dokumentationen spezialisiert und können, im Gegensatz zu Programmierern, die Dokumentation parallel zur Softwareerstellung verfassen[10]. Instruktoren werden hingegen zumeist in einer unternehmenseigenen Schulungs- oder Anwenderunterstützungsabteilungen organisiert. Dies bringt den großen Vorteil mit sich, dass während der Entwicklung von Schulungsunterlagen umfangreiches Wissen über das Softwareprodukt erworben wird und es somit ein logischer Schritt ist die Benutzerdokumentation aus diesen Unterlagen zu generieren. Des Weiteren kann durch diese Schulungen gezielt auf die Wünsche und Bedürfnisse der Anwender eingegangen

[8] vgl. [Leh94] / 80
[9] vgl. [Rup87] / 18
[10] vgl. [Leh94] / 83

werden.

Die Beschäftigung mit der Frage wer die Dokumentation erstellt und wer sie erstellen sollte, bringt folgende Aspekte für Probleme oder Konflikte zu tage. Das Dokumentieren ist eine scheinbar unproduktive Arbeit. Das Verfassen von Benutzerhandbüchern wird als zweitrangig angesehen. Die Motivation für die Abschlußarbeiten an einem Projekt, zu denen auch die Erstellung des Benutzerhandbuches gehört, ist nicht mehr im nötigen Ausmaß vorhanden. Diese Arbeiten werden daher an die unteren Hirarchiestufen delegiert. Es herrscht of Zeit- und Personalmangel. Es ist sehr schwierig, komplexe Zusammenhänge mit einfachen Worten verständlich zu machen[11].

2.2.2 Nutzung und Verbreitung der Software

Eine nähere Betrachtung des Faktors „Nutzung und Verbreitung der Software“ führt zwangsläufig zu der Frage welche Zielgruppen durch die Softwaredokumentation erreicht werden sollen[12]. Diese Benutzergruppen üben einen direkten Einfluss auf die Dokumentationsgestaltung aus, da sie die Ausführlichkeit und den Detaillierungsgrad der Erklärungen der Dokumentation bestimmen, sowie den Schreibstil bzw. die äußere Form der Ausführungen beeinflussen. Erst durch die Kenntnis der Zielgruppe ist es möglich den Informationsbedarf der Nutzer während des produktiven Betriebs der Software zu antizipieren und somit Inhalt, Umfang und Gestaltung der Handbücher festzulegen[13]. Eine solche Gruppierung kann jedoch nur schwerpunktmäßig erfolgen, da starke Überschneidungen zwischen den einzelnen Benutzergruppen existieren. Dieser mehrdimensionale und selten objektiv zu betrachtende Sachverhalt sollte innerhalb einer konkreten Zielgruppenanalyse perspektivisch und zielorientiert betrachtet werden. Zielgruppen können entweder indirekt aus den Arten und Formen der Softwaredokumentation ermittelt werden, da eine zielgerichtete Dokumentation in entsprechenden Bezeichnungen ihren Niederschlag findet, als auch direkt über die Adressaten der Dokumentation.

2.2.3 Abgrenzung der Benutzergruppen

Diese direkte Abgrenzung von Benutzergruppen kann nur anhand von Merkmalen, die größeren Gruppen von Benutzern zu eigen sind bzw. ihnen unterstellt werden können, durchgeführt werden. Eine allgemeine Charakterisierung der Benutzer ist für diesen Zweck nicht erforderlich, vielmehr müssen Arbeitsaufgaben der Zielgruppen ermittelt

[11] vgl. [Leh94] / 80
[12] vgl. [Rup87] / 18
[13] vgl. [Leh94] / 38

werden um eine zielgerichtete Dokumentationsenwicklung zu ermöglichen. Diese Arbeitsaufgaben können charakterisiert werden durch Tätigkeiten, welche der Benutzer im Arbeitsprozess ausführt, Arbeitsobjekte, auf die sich diese Tätigkeiten beziehen, und Arbeitsabläufe, welche Abfolgen wechselnder Tätigkeiten an unterschiedlichen oder gleich bleibenden Arbeitsobjekten sind.

"Tätigkeiten sind abgrenzbare Handlungsabläufe, die im betrachteten Zusammenhang als elementare Bausteine in Arbeitsabläufen zur Bewältigung bestimmter Aufgaben mit dem zu beschreibenden Produkt vorkommen."[14]

Unter den Arbeitsobjekten werden alle konkreten oder abstrakten Objekte verstanden, welche in den Tätigkeiten als Einheit beschrieben werden. Diese können unterschiedlich komplex sein. Im Rahmen der Definition von Arbeitsobjekten muss zwischen realen Objekten und ihrer Repräsentation im Softwaresystem unterschieden, sowie die Entsprechung zwischen beiden festgelegt werden, damit diese Eingang in das Benutzerhandbuch finden können.

Arbeitsabläufe fassen hingegen eine Menge von Einzeltätigkeiten zusammen um mit ihrer Hilfe komplexe Aufgaben lösen zu können. In diesem Zusammenhang ist es von großer Bedeutung die Arbeitsabläufe innerhalb des Produktivsystemes genau zu erfassen, die sie ausführenden Stellen zu ermitteln und so eine zielgerichtete Dokumentation, welche direkt auf die ausführenden Adressaten ausgerichtet ist, zu erstellen[15].

Weitere Aspekte die es bei der Abgrenzung von Benutzergruppen zu beachten gilt, sind die Kenntnisse und Voraussetzungen der Softwarebenutzer, welche sich in Art und Umfang quantifizieren lassen. So kann die Gruppe der Produktanwender in Bezug auf ihre Qualifikationen sowohl homogen als auch heterogen aufgebaut sein[16]. Dies gilt bei der Dokumentationsgestaltung zu beachten, um Anfänger, Fortgeschrittene und Spezialisten entsprechend ihrer Fähigkeiten bei der Ausführung ihrer Arbeitsabläufe zu unterstützen[17]. Ein weiterer Faktor, welcher sich jedoch nur in einem geringen und schlecht zu erfassenden Umfang auf die Zielgruppenanalyse auswirkt, ist die Motivation der Benutzer das Softwareprodukt zu verwenden.

Die Informationen über die Benutzer des Programms muss, theoretisch betrachtet, der Auftraggeber liefern. Diese können jedoch durch weitere Quellen ergänzt werden. Das Marketing beziehungsweise der Vertrieb sollte Vorstellungen und Untersuchungen über die Zielgruppe des Softwareproduktes liefern. Des weiteren sollte die Schulungsabteilung in der Lage sein, aufgrund von Erfahrungen mit früheren Produkten, Hinweise auf die Merkmale der Benutzer geben.

[14] [Rup87] / 35
[15] vgl. [Rup87] / 36
[16] vgl. [Sch90]
[17] vgl. [Bal00] / 641

2.2.4 Arten von Benutzergruppen

In Zuge dieser Analyse kann grob in drei Benutzergruppen unterschieden werden. Die erste Guppe setzt sich aus den Anwendern und Interessenten des Softwaresystems zusammen[18]. Für diese ist vorrangig die Benutzerdokumentation, welche ausreichend Informationen über das Produkt enthalten sollte um die Systemanwender bei der funktionsgerechten Handhabung der Software zu unterstützen, von Bedeutung. Die zweite Gruppe besteht aus dem Entwicklungs- und Wartungspersonal, welches Daten über den Aufbau und die interne Funktionsweise des Systems benötigt, da auf Basis dieser Informationen Systemänderungen und Erweiterungen durchgeführt werden. Die letzte Gruppe stellt das Management dar, welches sich je nach Anwender- oder Entwicklerperspektive für Informationen aus organisatorischer, kalkulatorischer und führungstechnischer Sicht interessiert. Diese Einteilung kann bei Bedarf um externe Partner erweitert werden, da sich die Aktivitäten bei der Softwareentwicklung nicht auf das eigene Unternehmen beschränken müssen. Unter externen Partnern wird beispielsweise eine Ausgliederung der Softwareentwicklung, der Einsatz externer Mitarbeiter oder Dienstleistungen für oder durch andere Unternehmen verstanden.

[18] vgl. [Wal90]

2.3 Arten der Softwaredokumentationen

Im Rahmen der Softwareentwicklung fällt eine Vielzahl an Dokumenten, wie Beschreibungen, Anleitungen, Instruktionsblätter usw. an, so dass eine Klassifikation nur schwer möglich ist [19]. Dieser Aspekt wird noch verstärkt, da in der Praxis viele Formen der Dokumentation als überflüssig angesehen werden, wozu ihre oft mangelhafte Qualität sicherlich einen entscheidenden Beitrag geleistet haben dürfte.

2.3.1 Funktionen der Softwaredokumentation

Um dennoch eine Klassifikation zu ermöglichen, ist es notwendig die Hauptfunktionen der Softwaredokumentation zu betrachten um im Anschluss eine Systematisierung der Dokumentationsarten zu gestatten. Im Folgenden wird in drei Hauptfunktionen unterschieden.

Die erste Funktion ist die Anleitungsfunktion der Dokumentation [20]. Auf ihrer Basis wird die Kauf- und Einsatzentscheidung getroffen. Sie umfasst nicht nur die Arbeitsanleitung, zur Fehlerbehandlung, Unterbrechungsbehandlung oder zur Datenarchivierung, sondern bildet ebenfalls die Arbeitsgrundlage zur Vermeidung von Mehrfachentwicklungen sowie zur Softwarepflege, welche Änderungen, Erweiterungen und die Fehlersuche beinhaltet.

Bei der zweiten Hauptfunktion handelt es sich um die Kontroll- und Nachweisfunktion. Diese kommt gegenüber internen und externen Prüfern zum Tragen, wie es z.B. im Kontext der Produkthaftung oder einer ISO-9001-Zertifizierung notwendig ist. Des Weiteren ermöglicht sie umfangreiche Projektfortschrittskontrollen.

Die Kommunikationsfunktion bildet die dritte und letzte Funktion der Softwaredokumentation[21]. Ihre Hauptaufgabe ist es eine einheitliche Kommunikationsbasis zu schaffen, um eine effiziente Verständigung zwischen allen an der Softwareentwicklung beteiligten Personen zu erreichen. Diese Funktion wird in einer ganzheitlichen Betrachtung des Softwareprojektes um Auftragnehmer bzw. Auftraggeber sowie um Käufer und Verkäufer erweitert. Dies dient dem Ziel die Wiederverwendbarkeit der Software zu steigern und eine umfassende Transparenz und Vergleichbarkeit der Programme auf dem Softwaremarkt zu realisieren.

Die unterschiedlichen Dokumentationsfunktionen müssen bei der inhaltlichen Gestaltung der Softwaredokumentation Berücksichtigung finden. Dokumentationsteile und

[19] vgl. [Leh94] / 6
[20] vgl. [Sch02] / 5
[21] vgl. [Sch02] / 7

einzelne Gliederungspunkte sollen hinsichtlich der beabsichtigten Verwendung und dem daraus resultierenden unterschiedlichen Leserkreis differenziert gewichtet werden. Um eine solche Gewichtung zu ermöglichen, wird im nachfolgenden Abschnitt eine Systematisierung der Dokumentationsbereiche vorgenommen um sie im Anschluss mit den Zielgruppen der Dokumentation zu verbinden.

2.3.2 Dokumentationsformen und -bereiche

In der Literatur gehen die meisten Autoren nur kurz auf die Arten der Dokumentation ein bzw. präsentieren eine exemplarische Aufzählung von Handbüchern, jedoch wird nur selten eine systematische Darstellung vorgenommen, für welche sich mehrere Kriterien anbieten. Auf einer allgemeinen Ebene kann zwischen direkter und indirekter Dokumentation unterschieden werden. Im Rahmen der direkten Dokumentation werden Objekte beliebiger Art verbal und unmittelbar beschrieben. Oft wird direkt zur Bedienung und Benutzung, z.B. durch das Benutzerhandbuch, angeleitet. Die indirekte Dokumentation stellt im weiteren Sinne eine Meta-Dokumentation dar. Als Beispiel kann die Literaturdokumentation genannt werden. Sie dient der Erschließung von dokumentarischen Daten und der Unterstützung bei der Beantwortung spezifischer Fragen. In der Praxis findet eine Trennung zwischen diesen beiden Dokumentationsformen nur selten statt, so dass die meisten Handbücher sowohl Systeminformationen als auch Benutzeranleitungen enthalten.

In einer speziellen Betrachtung kann diese allgemeine Sichtweise weiter in verschiedene Dokumentationsbereiche untergliedert werden. Im Kontext dieser Betrachtung soll hier in 5 Bereiche unterschieden werden.

Die Projektmanagementdokumentation, welche auch als Projektablaufdokumentation bezeichnet wird, beinhaltet Unterlagen für die Initiierung, Planung und Steuerung eines Projektes. Dieser Dokumentationsbereich umfasst sowohl den Projektauftrag inklusive aller Anforderungsdefinitionen als auch den kompletten Schriftwechsel sowie Sitzungsprotokolle. Des Weiteren fließen Dokumente aus Reviewsitzungen, Fortschritts- und Kostenberichten, usw. in diese Unterlagen ein. Häufig findet man noch ein Projekthandbuch vor, in dem alle erforderlichen Daten für das gesamte DV-Projekt systematisiert sind, welches einen übergeordneten Rahmen bildet. Dieser Rahmen beinhaltet in umfangreichen Softwareprojekten normalerweise auch Planungen der Projektleitung, wie Ressourcen-, Zeit-, Kosten- und Personalpläne. Solche Vorgaben sind vorrangig im Entwicklungsprozess von Individualsoftware von großer Bedeutung, um innerhalb der Vorstudie kalkulatorisch realisierbare Anforderungen zu identifizieren, als auch in direkten Verhandlungen mit den Auftraggebern den personellen Aufwand und die damit verbundenen Kosten zu quantifizieren.

Die Unterlagen der Organisations- und Systemdokumentation umfassen alle Dokumente die während der Projektdurchführung anfallen und die sich auf das zukünftige Softwaresystem beziehen. Beinhaltet werden beispielsweise Konzepte, Entwurfsergebnisse und Analyseergebnisse. In einem weiteren Sinne können auch Schnittstellen- und Transferprotokolldefinitionen für den Datenaustausch mit Fremdsystemen zu diesem Dokumentationsbereich gezählt werden, als auch allgemeine Architekturvorgaben sowie Integrationsdokumentationen für Drittsoftware wie Applikationsserver oder Datenbankengines. Diese Unterlagen bilden die Basis für nachfolgende Implementierungen und der direkten Ausgestaltung des Softwaresystems in der Entwicklungs- bzw. Programmierungsphase.

In die Programmdokumentation sind der Programmierauftrag und die Programmiervorgaben integriert. Von den Programmierern erstellte Hilfsdokumente, wie Pseudocode oder Ablaufpläne, die Auflistung des gesamten Quelltextes, die Testdaten sowie die Ergebnisse der Abnahmetests stellen weitere Aspekte dieses Dokumentationsbereiches dar. Da die erstellte Software, während ihres Produktlebenszyklus, erfahrungsgemäß häufigen und oft umfangreichen Änderungen unterliegt, ist es notwendig diesen Wandel in einer nachvollziehbaren Weise mit zu dokumentieren.

Die Unterlagen für die Systembedienung, vorranging für den Einsatz in Mehrbenutzersystemen, werden im Rahmen der Rechenzentrumsdokumentation zusammengefasst. Hierbei ist eine zielgruppenbezogene Fokussierung auf Programmaspekte wie Systemstart, Datensicherung oder Wiederanlaufverhalten nach einen Systemausfall zu finden. Dieser Dokumentationsaspekt ist vorangig bei Unternehmenssoftware wie SAP/R3 oder Bankeninformations- und -steuerungssystemen zu finden, gerät jedoch bei massenorientierter Standardsoftware, wie MS-Office oder Grafikbearbeitungsprogrammen, zunehmend in den Hintergrund.

Den letzten der fünf zu betrachtenden Dokumentationsbereiche stellt die Benutzerdokumentation dar. Diese hat die Funktion den Benutzern, in den entsprechenden Fachabteilungen, nötige Hilfestellung im Umgang mit dem System bzw. der Software zu geben. Je nach Kontext oder Produkt können auch Hinweise zur Implementierung bzw. Installation des Produktes enthalten zu sein, sowie Schulungs- oder Trainingsunterlagen integriert werden. Den Hauptaspekt stellt sicherlich die Bedienungsanleitung dar, welche direkt die Anleitungsfunktion der Dokumentation erfüllt.

Diese Dokumentationsbereiche können nun in einer Matrix den voraussichtlichen Zielgruppen zugeordnet werden. Aus dieser Zuordnung lassen sich die erforderlichen Handbücher und ihre Inhalte ableiten. In Abbildung 1 wird eine solche Präzisierung von Inhalten der Dokumentationsbereiche aufgezeigt.

Projektdok.	Technische Dokumentation		
	Entwicklungsdok.	Produktdok.	Benutzerdok.
Dokumente zur Projektabwicklung	*Aufgabenbeschreibung*	*Übersichtsbeschreibung*	*Betriebsunterlagen*
- Projektbeschreibung - Projektorganisation - Verantwortlichkeiten - Einsatzplanung - Zeitplanung - Kostenplanung - Dokumentationsplanung - Hilfsmittelaufstellung - Entwicklungsrichtlinien - Entscheidungsrichtlinien - Fortschrittsberichte - Abschlussberichte - Kostenschätzungen	- Aufgabenstellung - Anforderungen - Situationsanalyse - Problemanalyse - Prozessbeschreibung - Studien - Lösungsbeschreibung - Entwurfsschritte - Entwurfsalternativen - Entwurfsentscheidungen - Testplanung - Simulationen - Testeinrichtungen - IDE-Dokumentationen	- Systembeschreibung - Funktionsweise - Programmbeschreibung - Programmumgebung - Programmstruktur - Programmablauf - Datenmodell - Leistungskenndaten - Hardwarebeschreibung - Anlagenübersicht - Gerätebeschreibungen - Aufbaupläne - Vernetzungspläne	- Benutzerhandbuch - Bedienungsanleitung - Operatorenhandbuch - Wartungsunterlagen - Installationsanleitung - Fehlerbeschreibung - Schulungsunterlagen - Einführungen - Ausbildungspläne - Trainingsunterlagen - Vertriebsunterlagen - Leistungsbeschreibung - Lieferumfang - Konkurrenzvergleich - Referenzen

Abbildung 1: Dokumentationsbereiche (vgl. [Sch85])

2.3.3 Dokumentationsbezug

Eine weitere Gliederungsmöglichkeit ergibt sich, wenn nicht einzelne Dokumentationsbereiche, sondern der Bezug der Dokumentation betrachtet wird. In einer solchen Analyse kann in die zielgruppenorientierte Dokumentation, welche beispielsweise das Benutzerhandbuch oder das Operatorhandbuch umfasst, die ereignisorientierte Dokumentation, die sich exemplarisch aus Grobkonzept sowie dem Handbuch Projektorganisation zusammensetzen kann, und in die objektbezogene Dokumentation mit ihren Job-Handbüchern, Datei- und Datenbankdokumentationen, Testdokumentationen usw. unterschieden werden. Hierbei ist es selbstverständlich möglich, entsprechend den Gegebenheiten eines Softwareprojektes, kombinierte Formen zu verwenden.

Im Rahmen einer solchen zielbezogenen Untersuchung bietet es sich an, die Dokumentation in die Bereiche projektbezogene Dokumentation und produktbezogene Dokumentation zu gliedern. Die Grundlage für diese Unterscheidung bildet die Tatsache, dass die Softwareentwicklung meist in Form von Projekten erfolgt, welche als Ergebnis ein oder mehrere Produkte hervorbringen. Auf dieser Basis erscheint es notwendig, die Dokumentation differenziert für den Prozess und das Ergebnis der Programmerstellung zu betrachten.

Die Projektdokumentation kann hierbei in zwei Hauptkategorien unterschieden werden. Diese bilden die Projektüberwachung und die Systementwicklung. Die Dokumentation der Projektüberwachung bezieht sich direkt auf die Projektorganisation und schließt somit die Betrachtung der Faktoren Personal, Zeit, Kosten und Ressourcen mit ein. Sie umfasst all jene Dokumente, welche sich konkret auf die zu entwickelnde

Software beziehen, wobei eine genaue Abgrenzung nicht immer möglich ist, da Teile der Dokumente nach Projektabschluss in die Produktdokumentation aufgenommen werden können. Inhalte dieser Hauptkategorie sind beispielsweise:

- Projektstandards
- Übersicht über Projektdokumente
- Pflichtenheft und Funktionsbeschreibung
- Projektspezifischer Phasenplan
- Terminplan, Kostenplan, Kapazitätsplan, Personalplan
- Checklisten und Prüfergebnisse
- Freigabemitteilung
- Projektberichte
- Angebote
- Verträge

Die *Produktdokumentation* setzt sich idealtypisch aus folgenden Komponenten zusammen:

- Design- und Entwurfsdokumentation
- Quellprogramm
- Systemdokumentation
- Testdokumentation
- Wartungsdokumentation

Allen Gliederungskonzepten ist gemein, dass die einzelnen Systematisierungspunkte, je nach Art und Zielrichtung der zu erstellenden Software aber auch hinsichtlich der internen und externen Verwendung der Dokumentation, sehr unterschiedlich ausgeprägt sein können. Des Weiteren wird die zeitliche Dimension der Softwaredokumentation vollständig vernachlässigt. Der Prozess des Dokumentierens ist jedoch keine zeitlich begrenzte Tätigkeit im Rahmen der Softwareentwicklung, noch entsteht dabei ein stabiles unveränderliches Ergebnis. Er stellt vielmehr einen Abbildungsprozess dar, welcher den gesamten Softwarelebenszyklus umfasst. Aus diesem Grund wird in dem nachfolgenden Kapitel der Versuch einer Gliederung der Dokumentation hinsichtlich des Projektverlaufes bzw. Phasen der Softwareentwicklung unternommen.

2.4 Einordnung in die Phasen der Softwareentwicklung

Die Organisation der Dokumentation ist, insbesondere in großen Unternehmen, kein endgültiger Vorgang. Vielmehr unterliegt der Dokumentationsprozess von Software- zu Softwareprojekt sich ständig ändernden Rahmenbedingungen, an welche dieser stetig angepasst und, durch diesen Faktor bedingt, aktualisiert werden muss.

Um den Ablauf der Dokumentationserstellung aufgrund dieser Gegebenheiten dennoch verstehen, sowie formal und institutionell festlegen zu können, ist es notwendig sich mit denjenigen Abläufen der Softwareentwicklung näher auseinander zu setzen, welche allen Softwareprojekten der Unternehmung gemein sind. Von diesen Aspekten ausgehend, kann sowohl die zeitliche Dimension der Softwaredokumentationserstellung näher betrachtet, als auch die Organisationseinheiten, welche dem Dokumentationsumfeld direkt oder indirekt angehören, einzelnen Phasen der Erstellung der Softwaredokumentation zugeordnet werden[22].

2.4.1 Phasenmodelle

Der Prozess der Softwareentwicklung ist zumeist durch einen festgelegten organisatorischen Rahmen definiert, wobei dieser Rahmen durch so genannte Phasen- bzw. Vorgehensmodelle beschrieben wird.[23] Hierbei legt das Phasenmodell fest, welche Aktivitäten der Programmentwicklung in welcher Reihenfolge durchgeführt werden und welche Personen und Ereignisse an diesen Vorgängen beteiligt sind. Unter einer Phase wird in diesem Kontext eine genau festgelegte Menge zusammengefasster Aktivitäten verstanden. Diese Aktivitäten werden durch klar definierte Rollen bzw. Organisationseinheiten, unter der Berücksichtigung von Methoden, Richtlinien, Konventionen, Checklisten, Mustern, usw., ausgeführt und erzeugen ein definiertes Ergebnis.

Bei der Auswahl des zu betrachtenden Phasenmodells ist es von wesentlicher Bedeutung zwischen zwei Grundtypen zu unterscheiden. Den ersten Grundtyp bildet das evolutionäre Phasenmodell, welches die stufenweise Entwicklung eines Softwareproduktes abbildet. Diese code-getriebenen Entwicklung ist dadurch charakterisiert, dass zu Projektbeginn noch nicht alle Anforderungen an die zu entwickelnde Software bekannt sind. Aus diesem Grund wird in einem ersten Schritt der Systemkern des Programms entwickelt und dieser iterativ und somit Schritt für Schritt um weitere Komponenten erweitert. Treten im Projektverlauf jedoch Anforderungen zu tage

[22] vgl. [Rup87] / 25
[23] vgl. [Bal00] / 55

denen die bisher entwickelten Systemkomponenten nicht genügen können bzw. verändern sich bereits bekannte Anforderungen, kann dies schnell zu einer kompletten Neuentwicklung des gesamten Softwaresystems führen[24].

Der zweite Typ, das inkrementelle Vorgehensmodell, versucht diesem Effekt entgegenzuwirken bzw. ihm gänzlich vorzubeugen. Hierbei findet zu Beginn des Softwareprojektes eine vollständige Erfassung der Anforderungen, an das zu erzeugende Produkt, statt. Diese funktionsgetriebene Softwareentwicklung bietet eine hohe Anforderungsstabilität und somit große Sicherheit hinsichtlich Programmänderungen in der Entwicklung, da das komplette System bereits zu Beginn bekannt ist und Änderungen und Erweiterungen des Gesamtsystems nur inkrementell erfolgen[25].

In den nachfolgenden Betrachtungen wird daher nur der zweite Typ der genannten Vorgehensmodelle behandelt (siehe Abbildung 2)[26].

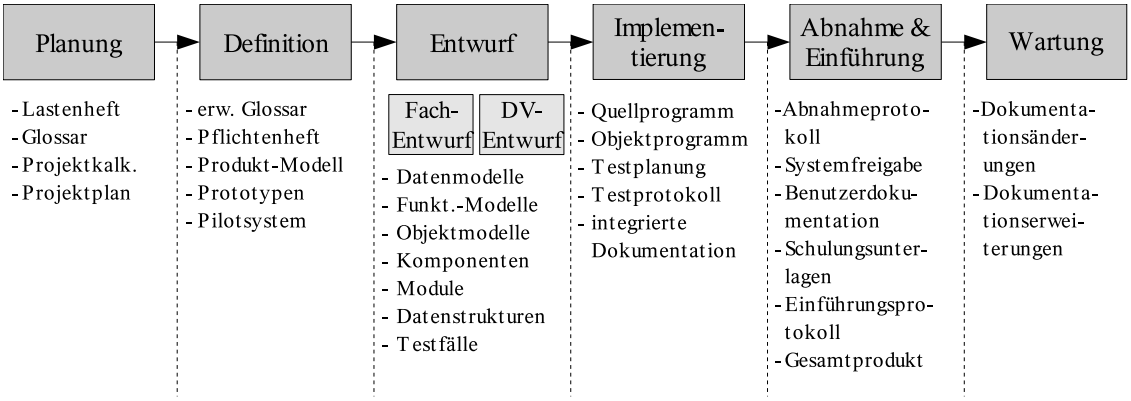

Abbildung 2: Das inkrementelle Phasenmodell

2.4.2 Planungsphase

Die erste Phase des inkrementellen Vorgehensmodell bildet die Planungsphase, welche sich mit der Voruntersuchung des zu realisierenden Softwaresystems beschäftigt, und somit auch als Durchführbarkeitsuntersuchung bezeichnet werden kann[27]. Das Ziel dieser vorbereitenden Prozesse ist die fachliche, ökonomische und personelle Durchführbarkeit der Produktentwicklung zu untersuchen und im weiteren Projektverlauf zu gewährleisten. Dies umfasst sowohl Aktivitäten hinsichtlich der Produktauswahl, welche durch Marktanalysen, Trendstudien, Kundenanfragen, Vorentwicklungen sowie der Untersuchung aktueller Forschungsergebnisse gekennzeichnet sind, als auch der direkten Voruntersuchung des zu realisierenden Produktes. Hierbei ist in der

[24] vgl. [Bal00] / 56ff
[25] vgl. [Bal00] / 58
[26] vgl. [Mer98] / 145ff
[27] vgl. [Bal00] / 58

Praxis vorrangig eine Fokussierung auf Ist-Analysen, der Aufnahme von Hauptanforderungen, Hauptfunktionen und Hauptdaten verbunden. Im Anschluss an diese vorgelagerten Maßnahmen erfolgt die eigentliche Durchführbarkeitsuntersuchung[28], die sich mit der fachlichen , also der softwaretechnischen Realisierbarkeit, und der personellen Durchführbarkeit auseinandersetzt. Diese Aktivitäten können den aktuellen Erfordernissen entsprechend um Wirtschaftlichkeitsrechnungen, Risikoprüfungen sowie Aufwands- und Terminschätzungen erweitert werden.

Im Verlauf dieser Phase werden wichtige vorbereitende Dokumente des Softwareprojektes generiert. So entstehen nicht nur das Lastenheft und das Glossar sondern es werden auch essentielle projektunterstützende Pläne wie die Projektkalkulation und der Projektplan erstellt. Um dies zu ermöglichen müssen sowohl die Auftraggeber, der Projektleiter als auch Anwendungsspezialisten an dieser Phase beteiligt werden. Insofern die Zielgruppe der zu erstellenden Software einen größeren Kundenkreis umfasst, kann es auch, in dieser frühen Projektphase, nötig sein, Vertriebs- und Marketingspezialisten in die Entscheidungen mit einzubeziehen um notwendige Marketingdokumente für Messen, Ausschreibungen, usw. zu erzeugen. Abschließend kann zu dieser Phase angemerkt werden, dass keine einheitliche Auffassung über den Umfang und Ablauf der Tätigkeiten gibt und die involvierten Personengruppen von Unternehmen zu Unternehmen variieren.

2.4.3 Definitionsphase

Die Definitionsphase stellt den zweiten Abschnitt des gewählten Vorgehensmodell dar. Ihre Aufgabe ist es, aus den zu ermittelnden Anforderungen an das Softwareprodukt ein vollständiges, konsistentes und eindeutiges Produktmodell zu erstellen[29]. Unter Anforderungen werden in der Softwareentwicklung die qualitativen und quantitativen Eigenschaften des zu erstellenden Produktes verstanden. Eine eindeutige Abgrenzung einzelner Anforderungen ist in der Praxis nur schwer möglich, da die Eigenschaften des abzubildenden Systems häufig vage, verschwommen, unzusammenhängend, unvollständig und widersprüchlich sein können. Um dennoch eine annähernd vollständige Erfassung der Anforderungen zu ermöglichen, müssen diese in einem ersten Schritt ermittelt und beschrieben werden. In den nachfolgenden Aktivitäten werden sie analysiert und als fachliche Lösungen ausmodelliert. So können einzelne Anforderungen im Rahmen eines abgeschlossenen Systems animiert, simuliert und ausgeführt werden, um sie in einem abschließenden Prozess für die Produktentwicklung zu verabschieden.

Dieser gesamte Vorgang kann als ein systematisch und iterativ durchgeführter Prozess verstanden werden, bei dem, von den allgemeinen Eigenschaften ausgehend, eine

[28] vgl. [Bal00] / 60
[29] vgl. [Bal00] / 98ff

sich immer stärker auf die speziellen Produktattribute fokussierende Anforderungsermittlung durchgeführt wird, was dem klassischen "Top-Down-Ansatz" entspricht. Da diese Phase für den weiteren Projektverlauf äußerst signifikant ist, wird im Rahmen der Softwaredokumentation eine umfangreiche Erweiterung bestehender Dokumente, wie z.B. dem Glossar, vorgenommen, sowie eine Vielzahl projektbegleitender Arbeiten erstellt. In diesem Kontext sind insbesondere das Pflichtenheft sowie das aus dieser Phase resultierende Produkt-Modell zu nennen. Des Weiteren können umfangreiche Dokumentationen im Rahmen prototypischen Oberflächen- und Pilotsystemerstellung entstehen.

An solch einem umfangreichen Prozess sind verschiedene Personen beteiligt sein, welche auch gleichzeitig die Adressaten der resultierenden Dokumente darstellen. So ist es sinnvoll neben den eigentlichen Auftraggebern, den späteren Benutzern des Systems und den, der Projektorganisation zugehörigen, Systemanalytikern auch die entsprechenden Fachabteilungen, technische Redakteure, Software-Ergonomen, sowie gegebenenfalls die Marketingabteilung, in die Aktivitäten der Definitionsphase einzubinden um so eine optimale Anforderungsanalyse zu garantieren.

2.4.4 Entwurfsphase

Die der Definitionsphase nachgestellte Entwurfsphase hat das Ziel, aus den resultierenden Produktanforderungen, eine softwaretechnische Lösung zu schaffen und somit die, in einem nachgelagerten Entwicklungsprozess zu realisierende, Softwarearchitektur zu gestalten. Unter Beachtung dieser Zielfunktion ist es notwendig weitere produktbezogene Einflussfaktoren, wie z.B. zu erwartende Einsatzbedingungen der Software, auftretende Umgebungs- und Randbedingungen sowie nichtfunktionale Produktanforderungen, unter dem Gesichtspunkt der gegebenen Qualitätsrichtlinien zu ermitteln[30]. Diese von Unternehmens-, Datenbankadministratoren und Systemanalytikern unterstützen vorgelagerten Aktivitäten beeinflussen die Entscheidungsfindung von Softwarearchitekten und -designern hinsichtlich des Softwareentwurfs direkt, da sie wichtige Informationen über die Robustheit, Interaktivität, Komplexität, usw. der zu erstellenden Software liefern, und münden in der abschließenden Definition einer konkreten Softwarearchitektur.

Basierend auf diesen Vorarbeiten können in einem zweiten Schritt einzelne Systemkomponenten, inklusive ihrer Schnittstellen, in ihrem Funktions- und Leistungsumfang definiert, sowie assoziative Beziehungen zwischen ihnen modelliert werden. Die aus diesen Untersuchungen resultierenden Ergebnisse können in einen Fach- und DV-Entwurf untergliedert werden. Der Fachentwurf steuert der Projektdokumentation

[30] vgl. [Bal00] / 686ff

hierbei Daten-, Funktions- und Objektmodelle bei, während der DV-Entwurf neben den Beziehungen zwischen Systemkomponenten und auf der nächst höheren Abstraktionsebene Modulen auch Dokumente über die logische und physische Datenstruktur enthält, sowie potentielle Testfälle umfasst.

2.4.5 Implementierungsphase

Die Aufgaben der vierten, auch Implementierung genannten, Phase bestehen aus der konkreten Entwicklung einzelner Systemkomponenten, um aus, durch den Softwareentwurf vorgegebenen, Spezifikationen Leistungen in Form eines oder mehrere Programme zu erbringen[31]. Neben Aktivitäten wie der Konzeption und Umsetzung von Datenstrukturen und Algorithmen, sowie der Strukturierung des Programms ist es hinsichtlich der Projektdokumentation nötig, alle Problemlösungen umfangreich zu dokumentieren sowie Angaben über das Laufzeitverhalten einzelner Komponenten zu tätigen. Auf Basis dieser Dokumente sind umfangreiche Systemtests und eine spätere Verifikation des Softwareproduktes, bezüglich der eingangs erarbeiteten Produktanforderungen möglich.

Die von Programmieren, Anwendungsspezialisten und Algorithmenkonstrukteuren erstellten Quell- und Objektprogramme stellen, bezogen auf die softwaretechnische Anforderungsumsetzung, wichtigsten Ergebnisse dieser Phase dar. Hinsichtlich der Systemdokumentation sind jedoch eine integrierte Softwaredokumentation, bestehend aus Kurzbeschreibungen, Verwaltungsinformationen und kommentiertem Quellcode, sowie Testplanungen und -protokolle von herausragender Bedeutung, da sie die Grundlage für spätere Erweiterungs- und Wartungsarbeiten bilden und detailliert einzelne funktionale Aspekte des Produktes belegen.

2.4.6 Einführungsphase

Das aus der Implementierungsphase resultierende Softwareprodukt, kann nun in einem vorletzten Schritt des inkrementellen Vorgehensmodells in den entsprechenden Fachabteilungen der Kunden installiert werden. Dieser Prozess wird auch als Einführungsphase bezeichnet, deren Ziel neben der Abnahme in der Einführung und Inbetriebnahme des fertig gestellten Softwareproduktes liegt und welche somit in eine Abnahme- und Einführungsphase untergliedert werden kann.

In der Abnahmephase[32] werden, neben der Übergabe des Gesamtproduktes und der zugehörigen Dokumentation, umfangreiche Stress-, Belastungs- und Abnahmetests am

[31] vgl. [Bal00] / 1064ff
[32] vgl. [Bal00] / 1086ff

Produkt durchgeführt um so die ordnungsgemäße Erfüllung aller gestellten Anforderungen zu überprüfen und die Integration in die Umgebungs- und Rahmenbedingungen des Softwareeinsatzes zu realisieren. Dieser Aspekt wird in der Einführungsphase weiterführend betrachtet, da während dieser nicht nur notwendige Datenmigrationen von Alt- und Fremdsystemen ausgeführt werden, sondern auch umfangreiche Parallel- und Versuchsläufe der erstellten Software stattfinden. Dies ermöglicht notwendige Schulungen der Benutzer an laufenden Systemen, ohne produktive Abläufe während der Inbetriebnahme des Produktes zu beeinflussen.

Aus dieser Phase resultieren neben dem Gesamtprodukt und der abschließenden Systemfreigabe wichtige Dokumentationsbestandteile wie Abnahme- und Einführungsprotokolle, System- und Benutzerdokumentationen sowie Schulungsunterlagen, deren Zielgruppen sich aus Adressaten wie beispielsweise den Aufragnehmern und -gebern, dem Projektleiter, potentiellen Endanwendern, dem Schulungspersonal, dem Einführungspersonal, verschiedenen Rechtsberatern, usw. zusammensetzen. Diese Phase stellt den Abschluss der eigentlichen Softwareentwicklung dar und ermöglicht den Übergang in den zweiten Abschnitt des Produktlebenszyklus, der Produktivsetzung des Systems.

2.4.7 Wartungsphase

Die Wartungsphase bildet den letzten Abschnitt des hier betrachteten Phasenmodells[33]. Ihre Aufgaben bestehen aus dem Anpassen des Produktes an veränderte Umweltbedingungen, der Erweiterung um neue Anforderungen sowie dem Beheben von Fehlern im täglichen Betrieb. Dies kann neben Anpassungen, Änderungen und Erweiterungen auch Optimierungen, Leistungsverbesserungen, Stabilisierungen und Korrekturen umfassen. Hierbei ist die Einbeziehung von vorliegenden Systemdokumentationen unumgänglich, um den involvierten Zielgruppen, wie System- und Datenbankadministratoren, Anwendungsspezialisten und Beratern, die Arbeit am bestehenden System zu erleichtern. Alle in dieser Phase auftretenden Wartungsarbeiten müssen der System- und Benutzerdokumentation hinzugefügt werden, so dass sie bei zukünftigen Änderungen berücksichtigt werden können.

Eine eindeutige Zuordnung der im Projektverlauf anfallenden Dokumente zu einzelnen Phasen der Softwareentwicklung ist in der Praxis nicht immer möglich, da viele Dokumente über mehrere Abschnitte des betrachteten Vorgehensmodells erstellt, erweitert und bearbeitet werden. Trotz der Bemühungen des inkrementellen Phasenmodells Änderungen in den Anforderungen der zu realisierenden Software zu vermeiden, treten

[33] vgl. [Bal00] / 1090ff

in realen Projekten häufig Anforderungsänderungen auf, da nicht immer alle technischen und fachlichen Realisierungsmöglichkeiten in der Planungsphase erfasst werden können bzw. durch Kommunikationsverluste zwischen Auftraggeber und -nehmer eine oft ungenügende Funktionsdefinition vorgenommen wird. Im Rahmen dieser Änderungen und Anpassungen ist es nötig die korrespondierenden Dokumente zu aktualisieren bzw. zu ergänzen. Dies kann eine Fehlerquelle während der Erstellung der Softwaredokumentation implizieren, da die Gefahr besteht das diese Aktualisierungen, bedingt durch Termindruck und unzureichende personelle Zuordnung, nicht in einem erforderlichen Umfang realisiert werden.

2.5 Dokumentationsqualität

Für eine umfassende Betrachtung der Qualität einer Softwaredokumentation, ist es unumgänglich in einem ersten Schritt den Begriff Qualität und deren Spezialform, die Softwarequalität, zu definieren um so in einer nachfolgenden Analyse die Dokumentationsqualität abzuleiten.

2.5.1 Definitionen

Unter dem Begriff Qualität kann im weiteren Sinne die Gesamtheit aller Eigenschaften und Merkmale eines Produktes oder einer Tätigkeit verstanden werden, welche sich auf deren Eignung für die Erfüllung gegebener Anforderungen bezieht[34].

Eine auf die Softwareentwicklung ausgerichtete Betrachtung definiert Softwarequalität in einem engeren Sinn als die Gesamtheit der Merkmale und Merkmalswerte eines Softwareproduktes, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen.

Der Begriff Qualität lässt sich bezüglich der Softwaredokumentation nur sehr schwierig objektiv definieren[35]. Die verschiedenen Zielgruppen des Produktes neigen zu einer stark subjektiven Beurteilung der Dokumentation, bedingt durch unterschiedliche Systemkenntnisse, allgemeiner DV-Erfahrung und heterogenen Wissensstand. Dieser Effekt wird durch die Zugehörigkeit der Endanwender zu verschiedenen Benutzergruppen und der Differenzierung in produkt- und prozessbezogene Betrachtung verstärkt, so dass eine kurze und prägnante Darstellung der Dokumentationsqualität eines Softwaresystems nicht möglich ist.

Auf der Basis dieser Erkenntnisse können für eine qualitative Beurteilung der Dokumentationsmerkmale, angelehnt an die Definition der Softwarequalität, ausschließlich allgemeine und spezielle Anforderungen an eine Softwaredokumentation herangezogen werden um anhand dieser den Erfüllungsgrad der Dokumentationsziele zu ermitteln.

[34] vgl. [Bal98] / 257
[35] vgl. [Leh94] / 117

2.5.2 Dokumentationsmerkmale

Dokumentationsmerkmale können, auf einer allgemeinen Ebene, in die Inhaltliche Güte, die Organisationsgüte und in die Formatgüte unterteilt werden, welche die Dokumentationsqualität direkt positiv oder negativ beeinflussen.

Unter der inhaltlichen Güte versteht man die Breite und Tiefe, mit der der Inhalt der Dokumentation abgehandelt wird[36]. Hierbei spielen die inhaltliche Vollständigkeit und Ausgewogenheit der einzelnen Abschnitte, hinsichtlich ihres Detaillierungsgrad, eine wesentliche Rolle. Des Weiteren gilt es unnötige Redundanzen zu vermeiden, sowie auf Ausnahmen und Sonderfälle zu verweisen. Dies dient dem Zweck, die inhaltliche Konsistenz der Ausführungen zu sichern.

Die Organisationsgüte vereint sowohl Form als auch Reihenfolge der getätigten Darstellungen und umfasst im weiteren Sinne vorhandene Gliederungen und Orientierungsmöglichkeiten der Darlegungen. Klare Strukturen und eine konsequente Gliederungen erhöhen die formale Qualität der Dokumentation hierbei ebenso, wie Indizes, Nummerierungen sowie aussagekräftige Überschriften und Zusammenfassungen.

Das dritte und letzte Dokumentationsmerkmal stellt die Formatgüte dar. Unter diesem Aspekt wird der ordnungsgemäße und themenbezogene Gebrauch geeigneter Sprachmittel verstanden. Dieses Merkmal kann durch Illustrationen, in Form von Abbildungen, Tabellen, Diagrammen, usw., erweitert werden.

2.5.3 Dokumentationsanforderungen

Um zu vermeiden, dass die dargestellten Dokumentationsmerkmale als allgemeine isolierte Anforderungen betrachtet werden, wurden durch die Deutsche Gesellschaft für Qualität (DGQ) detailliertere Anforderungen an eine Softwaredokumentation erarbeitet um somit eine genauere Beschreibung und Bewertung der Dokumente zu ermöglichen[37]. Hierbei wurden sieben Anforderungen herausgearbeitet.

Die Änderbarkeit der Softwaredokumentation, hat das Ziel Erweiterungen und Anpassungen unter der Prämisse eines möglichst geringem Aufwand sicherzustellen.

Die Aktualität gewährleistet hingegen, dass die Ausführungen dem letzten Stand entsprechen und somit zwischen der neusten Version der Software und ihrer Beschreibung keine Diskrepanzen bestehen.

Unter der Eindeutigkeit und Einheitlichkeit wird die durchgängige Verwendung gleicher Begriffe und Methoden verstanden. Um dies zu ermöglichen, kann es notwendig

[36] vgl. [Wal90] / 101
[37] vgl [fQe91] / 61f

sein unternehmensweit geltende Richtlinien und Formulierungen zu erarbeiten, um widersprüchliche Aussagen in den Ausführungen der Dokumentation zu vermeiden.

Die Anforderung der Identifizierbarkeit verlangt eine eindeutige Ansprechbarkeit von Teildokumenten um die fachliche Zuordnung zwischen der Dokumentation und der zu beschreibenden Produktkomponente aufrecht zu erhalten. Berücksichtigt werden in diesem Kontext auch so genannte Meta- oder Verwaltungsdaten der Ausarbeitungen, wie beispielsweise der Name des Autors, Änderungs- und Erstellungsdaten sowie der aktuelle Bearbeitungsstand in Form einer Versionierung.

Der Aspekt der Normenkonformität befasst sich mit der Einhaltung von Vorschriften, welche der Erstellung der Dokumentation einen regulatorischen Rahmen bieten. Dies kann sich sowohl auf firmeninterne als auch auf allgemein geltende (branchenübliche, nationale, internationale) Standards beziehen.

Des Weiteren muss die Verständlichkeit der Ausführungen sichergestellt werden. Da potentielle Leser die getroffenen Aussagen und Informationen verstehen sollen, ist es notwendig diese auf die Zielgruppen und das Benutzerumfeld abstimmen.

Die Vollständigkeit erhebt den Anspruch, dass alle von der jeweiligen Zielgruppe benötigten Informationen vollständig in die Dokumentation aufgenommen werden und kann in zwei Aspekte untergliedert werden. Die formale Vollständigkeit stellt sicher, dass alle Bestandteile, welche in den entsprechenden Verzeichnissen, wie Inhalts- oder Abbildungsverzeichnis, aufgeführt sind auch so in den eigentlichen Dokumenten vorkommen. Eine inhaltliche Vollständigkeit ist hingegen erst dann gegeben, wenn neben der Form auch alle Sachverhalte und Komponenten, die Gegenstand der Dokumentation sind, vollständig beschrieben werden.

Die letzte von der DGQ geforderte Eigenschaft stellt die Widerspruchsfreiheit, also das Nichtvorhandensein gegensätzlicher Aussagen, dar. Hierbei gilt es zu beachten, dass es neben inhaltlichen Widersprüchen innerhalb der Dokumentation auch zu Diskrepanzen zwischen den getroffenen Aussagen und den abzubildenden Sachverhalten kommen kann.

Um diese Formalisierungen in einen standardisierten Rahmen zu überführen, wurden verschiedene Normen und Richtlinien, mit dem Ziel der Bewertung und Verbesserung der Dokumentationsqualität, geschaffen. In den nachfolgenden Ausführungen sollen exemplarisch drei der wichtigsten Standardisierungsbemühungen betrachtet werden, ohne dabei einen Anspruch auf Vollständigkeit zu erheben.

2.5.4 ISO 9000

In einer rückblickenden Betrachtung der Softwareentwicklung ist eine Konzentration auf die konstruktive Verbesserung von Software-Produkten zu erkennen. Auf dieser Basis wurden Qualitätsmodelle erstellt, welche ausschließlich auf Messungen an Zwischenprodukten und am Endprodukt basieren[38].

Praxiserfahrungen zeigen jedoch, dass die Qualität des Softwareproduktes wesentlich durch die Qualität des Erstellungsprozesses beeinflusst wird. Dieser Ansatz wird durch die Familie der ISO 9000 Normen forciert, welche einen allgemeinen, übergeordneten, organisatorischen Rahmen, zur Qualitätssicherung von materiellen und immateriellen Produkten, bezogen auf ein Auftraggeber-Lieferanten Verhältnis festlegt. Im Rahmen der Softwareentwicklung stellt die ISO 9000-3 die für diesen Kontext relevante Norm dar.

Da der Dokumentationsprozess einen wichtigen Aspekt in der Erstellung von Software darstellt, kann die ISO 9000-3 äquivalent auf diese Vorgänge angewandt werden[39]. Innerhalb der Norm wird kein spezielles Vorgehensmodell vorgeschrieben, jedoch bestimmte Anforderungen formuliert. Der Prozess der Softwareentwicklung und somit auch die Dokumentationserstellung muss in spezifischen Phasen stattfinden, wobei die Vorgaben für jede Phase konkret festgelegt sind. Da die Einordnung der Dokumentation in die Phasen der Softwareentwicklung bereits in Kapitel 2.5. ausführlich behandelt wurde, soll an dieser Stelle nicht auf die konkreten Aktivitäten und Ergebnisse dieser Vorgänge eingegangen werden.

Dieses inkrementelle Vorgehen wird direkt in das Qualitätsmanagement und das Qualitätssicherungssystem eingebunden. Um dies zu ermöglichen, müssen Festlegungen in Form von Regeln, Praktiken und Übereinkommen aufgestellt werden, um das geschaffene Qualitätsmanagement- und Qualitätssicherungssystem wirksam einsetzen zu können. Die ISO-Norm spezifiziert hierfür diverse unterstützende Tätigkeiten, wie beispielsweise Messungen und Verbesserungen am Produkt und innerhalb des Softwareerstellungsprozesses sowie eine übergeordnete Lenkung der Dokumentenerstellung und fordert darüber hinaus die Integration des Qualitätsmanagements in die gesamte Unternehmens- und damit auch Projektorganisation. Dies verpflichtet nicht nur die Geschäftsführung zur Qualität, sondern schafft ein innerbetriebliches Qualitätsbewusstsein aller Mitarbeiter. Durch einen sich ständig wiederholenden Zertifizierungsprozess wird zudem sichergestellt, dass dem Qualitätsmanagement, auch über einen längeren Zeitraum, die notwendige Aufmerksamkeit und Ressourcen zur Verfügung stehen.

[38] vgl. [Bal98] /328f
[39] vgl. [Bal98] / 330ff

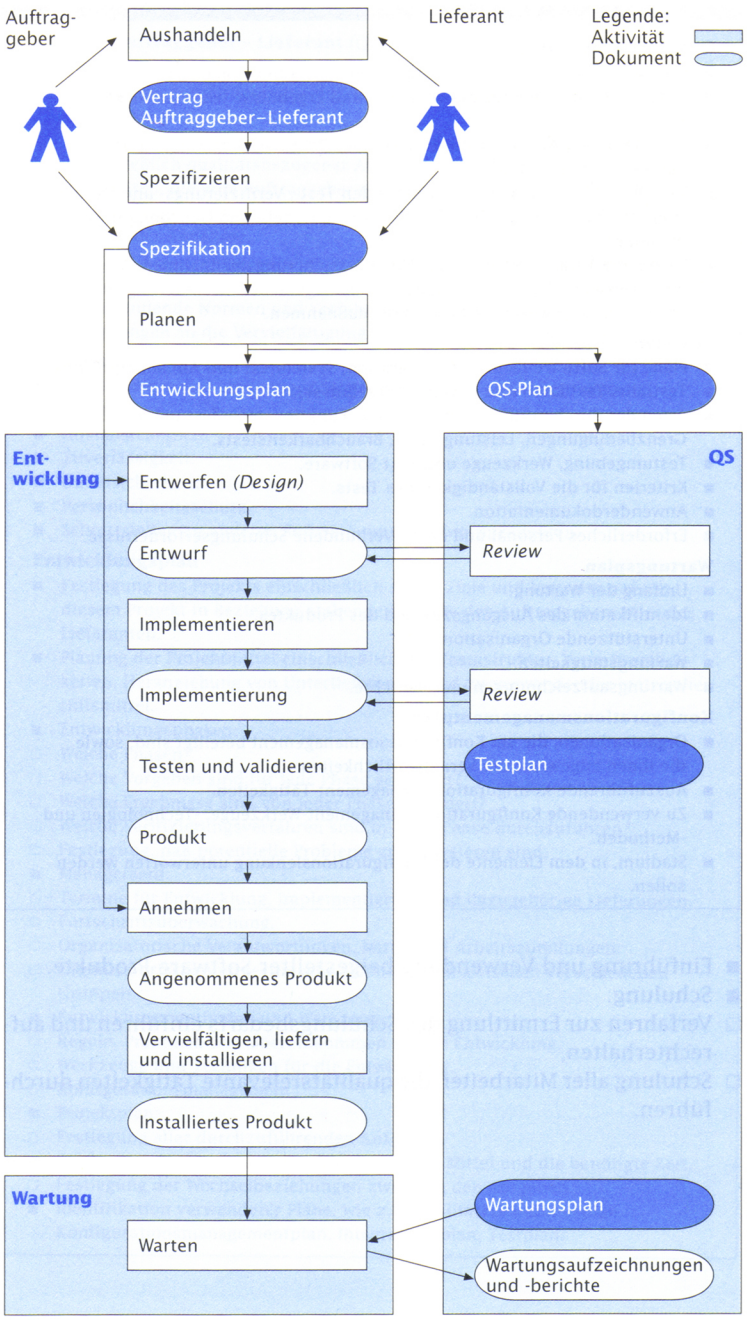

Abbildung 3: ISO 9000-3 - Aktivitäten und Dokumente

Der ISO 9000-3 Ansatz wirkt sich durch seine prozessorientierte Betrachtung, im Rahmen der Softwaredokumentation, fast ausschließlich auf den Vorgang der Dokumentenerstellung aus. Dies garantiert auf der einen Seite, dass die Softwaredokumentation in einem durch die Qualitätssicherung geforderten Umfang von den zuständigen Organisationseinheiten erstellt wird, jedoch können keine Aussagen über die inhaltliche und formale Qualität der erstellten Ausführungen getroffen werden. Dieser Aspekt wird durch die wesentlich ältere DIN 66230 aufgegriffen.

2.5.5 DIN 66230

Die DIN 66230 Norm geht nicht auf die Vorgehensweise der Softwareentwicklung ein, sondern beschreibt die zur Anwendung der Software erforderlichen Angaben, welche zusammen fassend als Programmdokumentation bezeichnet werden[40]. In diesem formalisierten Rahmen kann von zwei zu untersuchenden Ebenen ausgegangen werden, dem Programm und dem Programmbaustein. Die DIN 66230 ist darüber hinaus jedoch auch auf alle anderen Ebenen der zu erstellenden Software anwendbar, was beispielsweise das Programmsystem, das einzelne Programm sowie mögliche Unterprogramme einschließen kann. Aus diesem Grund ist für die Anwendung der Norm eine rekursive Betrachtung vorgesehen. Die Bezeichnung der einzelnen Programmebenen sind sinngemäß einzusetzen, wie etwa Programmsystem und Programm durch Programmbaustein und Unterprogramm.

Die Porgammdokumentation stellt die Voraussetzung für die Anwendung der beschriebenen Programme dar und bildet darüber hinaus die Basis für Einsatzentscheidungen sowie einer zweckdienlichen und ökonomischen Installation. Des Weiteren unterstützt sie Fehlerbeseitigungen, Wartungsarbeiten und Schulungen am Zielsystem. Die DIN-Norm legt ausdrücklich eine mögliche Gestaltung der Programmdokumentation fest. So sind neben einer minutiösen Gliederung der Meta-Informationen des Programms, wie etwa den Programmkenndaten, bestehend aus Programmbezeichnungen, Deskriptoren, Aufgaben, dem Gerätebedarf, der Programmgröße, dem Programmbedarf, den benötigten Programmiersprachen, den Betriebsarten, Dateien, Konventionen, Zuständigkeiten, Unterlagen und sonstigen Angaben, ebenfalls die genauen Aufgaben, und deren Lösungsbeschreibungen, Funktionen sowie den Aufbau des Programms strukturiert festgehalten. Des Weiteren sind der Programmablauf, die dafür benötigten Daten, die Anwendungsgrenzen, die Datensicherung, Anwendungsbeispiele, Installierung und Tests sowie umfangreiche Ausführungen für den Betrieb der Software festgelegt. Eine solch ausführliche Aufzählung soll exemplarisch den Umfang, sowie die stark regulatorische Wirkung dieser Norm aufzeigen. Dieser Aspekt wird durch die weiterführende Untergliederung der dargestellten Programmdokumentation in das Anwendungs- und das datenverarbeitungstechnische Handbuch verstärkt.

Das Anwendungshandbuch hat die Aufgabe Informationen für die sinnvolle Verwendung des Programms bereitzustellen. Durch die Software lösbare Aufgaben, sowie die den Lösungsansätzen zugrunde liegenden Theorien, werden in diesem Handbuch beschrieben. Darüber hinaus können auch die konkreten Datenspezifikationen Eingang in die Ausführungen finden.

[40] vgl. [DIN81]

Das datenverarbeitungstechnische Handbuch regelt hingegen die Umgebungsbedingungen für den erfolgreichen Betrieb des Softwaresystems. Aus diesem Grund enthält es umfangreiche Spezifikationen zur Installation, dem Betrieb und der Pflege aller zugehörigen Programme. Es bietet, in einer erweiterten Fassung, außerdem klare Regularien für zu erfolgende Wartungsarbeiten, Fehlerbeseitigungen und im Produktivbetrieb auftretende Fehler.

2.5.6 IEEE 1063

Der am 20.12.2001 veröffentlichte internationale Standard IEEE 1063 (Software-Benutzerdokumentation) hat es sich zum Ziel gesetzt die veraltete Fassung von 1987 zu ersetzen und um zeitgemäße Richtlinien für die qualitative Gestaltung von elektronischen Dokumentationen und Online-Hilfen zu erweitern[41]. In diesem Rahmen enthält der geschaffene Standard umfangreiche Empfehlungen hinsichtlich Struktur, Inhalt und Form von Software-Benutzerdokumentationen. Um diese Richtlinien in einer hochwertigen Ausarbeitung umsetzen zu können, ist es notwendig die strukturellen, inhaltlichen und formalen Anforderungen, welche die IEEE 1063 an die Softwaredokumentation stellt, zu betrachten.

Die strukturellen Dokumentationsanforderungen[42] beziehen sich auf den internen Aufbau der Ausführungen. Dieser kann in eine logische und physikalische Struktur eingeteilt werden. Unter der logischen Struktur sind in diesem Kontext bestimmte Themen, inhaltlich von einander abgegrenzte Bereiche, zu verstehen, welche durch die physikalische Struktur repräsentiert werden. Der physikalische Aufbau zeichnet sich durch eine bestimmte Menge an Seiten bzw. Screens aus, wobei eine logische Einheit durch mehrere dieser Screens dargestellt werden kann. Inhaltlich sollte die Produktdokumentation in eine anleitende Dokumentation, welche an den Aufgaben der Benutzer orientiert und didaktisch aufgebaut sein sollte, und in die Referenzdokumentation, die den wahlfreien Zugriff auf Informationen ermöglicht, untergliedert sein. Des Weiteren wird eine Anreicherung der Ausarbeitungen mit identifizierenden Angaben, Inhalts- und Abbildungsverzeichnissen und Einführungskapiteln vorgeschlagen, um eine sachgerechte Verwendung der Dokumente zu erleichtern. Insbesondere sollten Sicherheitshinweise vor der anleitenden Dokumentation und den entsprechenden Arbeitsabläufen platziert werden.

Die für eine technische Dokumentation geltenden Grundsätze der Vollständigkeit und Richtigkeit können in einem gleichen Maße für anleitende und referenzierende Informationen angewandt werden[43]. Die Erreichung dieser Anforderungen wird beispielsweise

[41] vgl. [Grü02] / 1
[42] vgl. [Grü02] / 1f
[43] vgl. [Grü02] / 2f

durch Beispiele und Abbildungen unterstützt, vor allem wenn diese kontextsensitiv in die Applikation integriert sind. Neben aufgabenorientierten "Schritt für Schritt" Anleitungen muss die Dokumentation darüber hinaus strategische Informationen für den Gebrauch der Software, z.B. in Form von Workflows oder Prozessbeschreibungen, bieten. Routinetätigkeiten, wie Installation und Deinstallation oder der Umgang mit der grafischen Benutzeroberfläche, sollten an einer zentralen Stelle der Ausführungen einmalig beschrieben sein und gegebenenfalls um Arbeitsabläufe und Tutorials ergänzt werden. Die detaillierte Auflistung aller Softwarebefehle und Fehlermeldungen erfolgt in der Referenzdokumentation und beinhaltet alle bekannten Probleme und deren Lösungsmöglichkeiten, sowie Supportinformationen für weiter gehende Hilfestellungen.

Im Zentrum der formalen Dokumentationsanforderungen[44] steht der Begriff der Einheitlichkeit. In diesem Rahmen ist darunter nicht nur die einheitliche Formatauszeichnung für Bedienelemente, Prozesse und Sicherheitshinweise zu verstehen, sondern auch die terminologische Einheitlichkeit der einzelnen Ausarbeitungen. Um dies zu gewährleisten ist das Dokumentationsformat so zu wählen, dass die Informationen zu jedem Bedienzeitpunkt einsehbar sind und der Anwender in seinen Arbeitsabläufen durch eine funktionsspezifische Online-Hilfe unterstützt wird. Des Weiteren sind die elektronische Dokumente um gedruckte Ausführungen, wie Systemanforderungen, Installationsanleitungen und der Anleitung für den Zugriff auf die Online-Dokumentation zu ergänzen. Der IEEE 1063 Standard stellt darüber hinaus konkrete Anforderungen an die Typographie, die Orientierungshilfen (z.B. Suchfunktionen) und an die Navigationselemente der Dokumentation.

Wie deutlich zu erkennen ist, versucht der IEEE 1063 Standard allgemein gültige Dokumentationsmerkmale und -anforderungen, im Rahmen elektronischer Ausführungen, umzusetzen und aktuelle Gestaltungsempfehlungen für die Erstellung von qualitativ hochwertigen Softwaredokumentationen bereit zu stellen.

[44] vgl. [Grü02] / 3f

3 Dokumentationsunterstützung

Wie in Kapitel 2 aufgezeigt, stellt die Softwaredokumentation einen sehr umfangreichen Prozess dar. Die starke Integration der Dokumentationstätigkeiten in alle Phasen der Softwareentwicklung und die hohe Anzahl der involvierten Personen, erschweren eine zentrale Koordination, Verwaltung und Kontrolle der Dokumentationserstellung. Durch heterogene Eingangsdaten und dem damit verbundenen Abstimmungstätigkeiten zwischen einzelnen Autoren der Softwaredokumentation können sowohl umfangreiche Reibungsverluste sowie ein erhöhter Erstellungsaufwand auftreten. Darüber hinaus erschwert die Einhaltung vorgegebener Qualitäts- und Dokumentenstandards, unter der Prämisse der Erstellung differenzierter kundenbzogener Ausgabeformate, eine integrative und homogene

In den folgenden Ausführungen soll anhand moderner dokumentationsunterstützender Werkzeuge aufgezeigt werden, wie diese auftretenden Schwierigkeiten minimiert und somit eine Entlastung der an der Dokumentationserstellung beteiligten Personen erreicht werden kann. Dies ist beispielsweise durch eine umfassende softwaretechnische Unterstützung, wie der Verwendung zentraler Repositories, einer umfassender Inline-Quellkodedokumentation oder der automatischen Erzeugung verschiedener Zielformate möglich. Hierbei soll die Umsetzung dieser Techniken anhand zweier standardisierter und ausgereifter Werkzeuge untersucht und die Verwendung dieser Programme erläutert werden.

3.1 Doxygen

3.1.1 Historie

Das Programm Doxygen wurde 1997 für die Dokumentationsunterstützung großer Softwareprojekte entwickelt. Ursprünglich nur für den Einsatz von C und C++ unter dem Betriebssystem Linux konzipiert, zeichnen sich aktuelle Versionen durch ein hohes Maß an Portabilität und einer umfassenden Unterstützung weiterer Betriebssysteme, wie Windows, MacOS und Unix aus[45].

3.1.2 Motivation

Aus der ursprünglichen Motivation heraus, die Hochsprache C und C++ um dokumentationsunterstützende Sprachkonstrukte zu erweitern, entstand ein komplexes Framework, welches nicht nur verwandte Sprachen, wie Java, C# oder IDL integriert,

[45] vgl. [vH04a]

sondern drüber hinaus auch diverse Ausgabeformate in der Dokumentationserstellung unterstützt. Durch vielfältige Konfigurationsmöglichkeiten und umfassender Kompatibilität zu JavaDoc in der Version 1.1, KDoc, Doc++, COCOON und der oft in größeren Open-Source-Projekten anzutreffenden Qt-Dokumentation wurde ein Werkzeug geschaffen welches, über die reine API-Dokumentation hinaus, umfangreiche Möglichkeiten für die Erstellung der Quelltextdokumentation im Rahmen der übergreifenden Programmdokumentation bietet[46].

3.1.3 Eigenschaften

Eine Betrachtung der Eigenschaften dieses Dokumentationswerkzeuges sollte in zwei Stufen erfolgen, wobei in einem ersten Schritt die Basisfunktionalität des Programms betrachtet und nachfolgend auf die erweiterten Funktionalitäten eingegangen wird[47].

Um den Dokumentationsaufwand den individuellen Projekterfordernissen anzupassen, kann der Programmierer die Kommentierung des Quelltextes auf einer sehr niedrigen Stufe, beispielsweise in Form reinen Textes, vornehmen oder diesen mit feinen Abstufungen um HTML-Tags oder spezielle Doxygen-Kommandos anreichern. Dieses Vorgehen kann iterativ auf ganze Domänen, Module, Dateien und Klassen angewandt werden, wobei in diesen Quelldokumenten nicht nur sprachabhängige Konstrukte wie Datenzusammenstellungen (structs), Strukturvorlagen (templates), Funktionen, Variablen usw. um beschreibende Kommentare erweiterbar sind, sondern ganze Klassen- und Schnittstellenbeschreibungen vor ihrer eigentlichen Definition oder in separaten Beschreibungsdateien dokumentiert werden können. Doxygen kann während der Dokumentationserstellung zwischen verschiedenen Sichtbarkeitsstufen des Quelltextes, wie öffentlich (public), geschützt (protected) und privat (private), unterscheiden und somit konfigurationsabhängig verschiedene Abstufungen der Dokumentation erstellen. Dies ist im Rahmen der Einbeziehung unternehmensexterner Entwicklungskapazitäten von großer Bedeutung, da auf diesem Weg reine Schnittstellendokumentationen, ohne die Einbeziehung interner privater Funktionalitäten, realisierbar sind. Hinsichtlich der Ausgabeformate kann in der Dokumentenerstellung zwischen „on-line“ und „off-line“ Formaten gewählt werden. Neben Dokumenttypen wie HTML und UNIX „man-pages“ sowie einer RTF (Rich Text Format) und LaTeXAusgabe stehen navigierbare Formate wie PDF, HTML und komprimiertes HTML in Form des Windows-Hilfeformates zur Verfügung. Darüber hinaus kann die Dokumentation in der formatunabhängigen Beschreibungssprache XML generiert werden um eine optimale maschinelle Weiterverarbeitung zu gewährleisten.

[46] vgl. [Gro03]
[47] vgl. [vH04b]

Über diese Basisfunktionalitäten hinaus kann der Entwickler einen vollständigen C-Präprozessor nutzen, um eine ordnungsgemäße Auflösung aller abhängigen Programmbestandteile zu gewährleisten. Auf dieser Basis ist eine umfangreiche Makro-Sprache nutzbar, welche die Umsetzung wieder kehrender Aufgaben erleichtert sowie sprach- bzw. laufzeitbezogene Dokumentationskonstrukten realisiert. Das Resultat des Dokumentationsprozesses kann automatisiert um Klassendiagramme, Abhängigkeitsgraphen, Kollaborationsdiagramme, graphische Klassenhierarchien, usw. erweitert werden. Dokumentationen anderer Softwareprojekte sind referenziell einbindbar, unabhängig von ihrem aktuellen Speicherort. Des Weiteren beinhaltet Doxygen eine schnelle und rang-basierte Suchmaschine zum Auffinden von Zeichenketten und Wörtern in Klassen und Funktionsdokumentationen.

3.1.4 Interne Funktionsweise

Um die einzelnen Aspekte der Dokumentationserstellung unter Verwendung des Werkzeugs Doxygen zu beschreiben, ist es notwendig die interne Funktionsweise des Programms abzubilden. Anhand einer solchen Beschreibung kann detailliert aufgezeigt werden, welche Schritte notwendig sind um aus einem kommentierten Quelltext eine verständliche und navigierbare Programmdokumentation zu erzeugen.

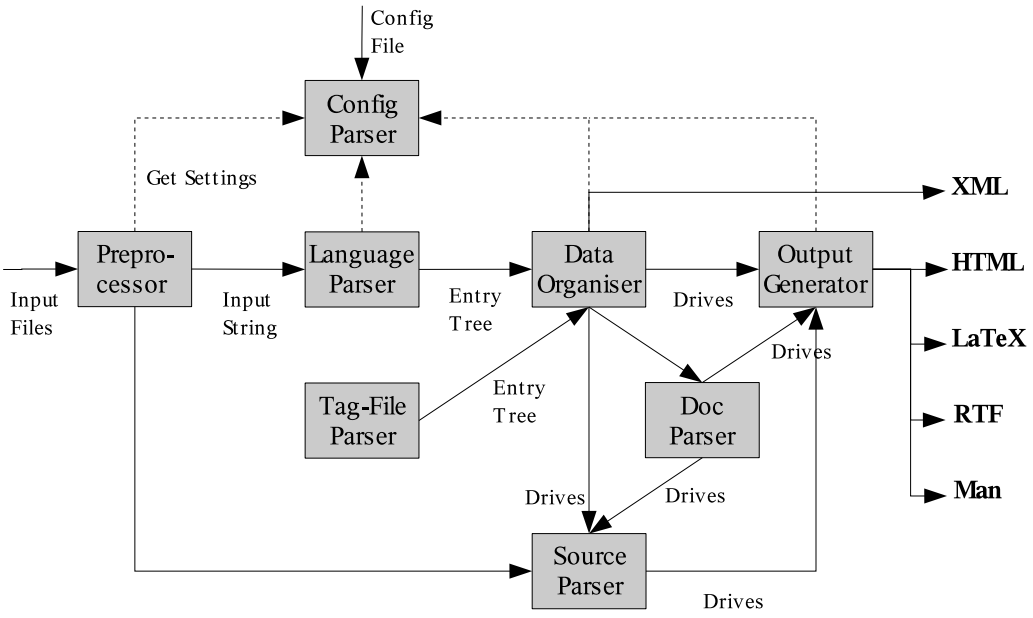

Abbildung 4: Doxygen - interne Funktionsweise

Wie in Abbildung 4 zu sehen ist, wird während des Programmstarts die projektabhängige Konfigurationsdatei eingelesen. Aus den in ihr enthaltenen Angaben werden die zu verarbeitenden Quelldateien ermittelt und der dem eigentlichen Generierungsprozess vorangestellte Präprozessor konfiguriert. In Abhängigkeit der verwendeten Programmiersprache werden die Eingangsdaten geparst, programmspezifische Dokumentationsbefehle umgesetzt und eine objektbasierte Datenstruktur erzeugt. Ausgehend von diesem Objektbaum können in einem abschließenden Schritt die gewünschten Ausgabeformate generiert werden. Die für die Dokumentationsgenerierung relevanten Schritte, bestehen wie aufgezeigt aus der Konfiguration, dem Einlesen und Verarbeiten der Quellen und dem Erzeugen der gewünschten Ausgabestruktur. Aus diesem Grund werden die nachfolgenden Ausführungen diese drei Schritte näher beschäftigten.

3.1.5 Konfiguration

Zentraler Bestandteil der Konfiguration des Programms Doxygen ist die Konfigurationsdatei des Werkzeuges. Diese liegt in einem formfreien ASCII-Format vor und kann

sowohl maschinengestützt über eine grafische Benutzeroberfläche, dem Doxywizard, als auch händisch mit einem Texteditor bearbeitet werden.

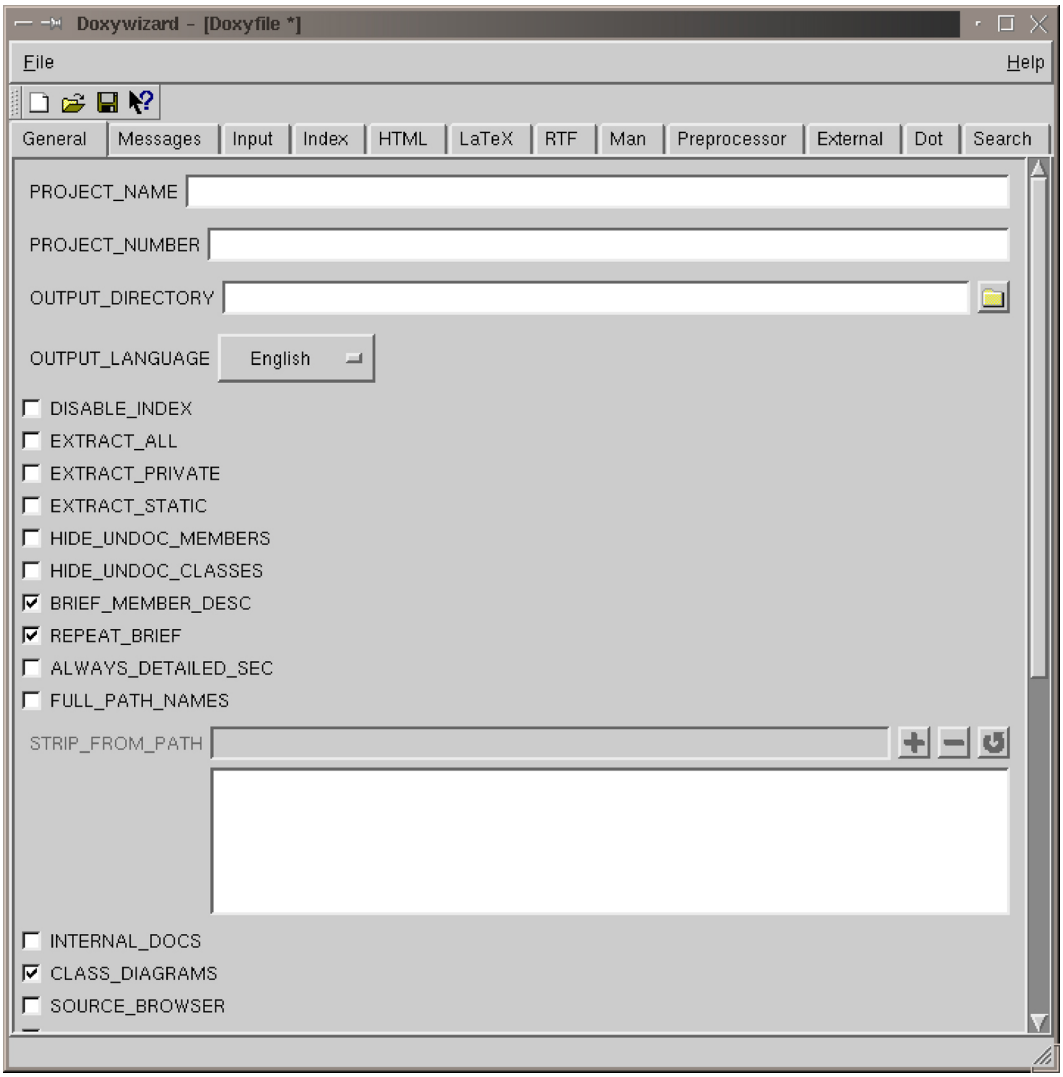

Abbildung 5: Doxygen - Doxy-Wizard

Die Konfigurationsdatei besteht aus einer Liste von Zuweisungen, welche dem Muster Tag-Name (TAG_NAME), in Großbuchstaben geschrieben, gefolgt von einem Gleichheitszeichen und dem eigentlichen Wert bzw. einer Liste zu verarbeitender Werte, entsprechen. Die zu übergebenen Werte dürfen keine Leerzeichen enthalten, es sei denn sie werden durch doppelte Anführungszeichen begrenzt. Tritt ein Schlüssel-Wert-Paar mehrfach in der Datei auf, überschreibt der letzte Eintrag alle vorherigen. Des Weiteren können ebenfalls Umgebungsvariablen des Betriebssystems verwendet werden.

Die Konfigurationsoptionen können in verschiedene Kategorien unterteilt werden. Diese werden in den nachfolgenden Ausführungen genannt und kurz beschrieben, wobei aufgrund der Fülle von Einstellungsmöglichkeiten nur einige wenige vorgestellt werden.

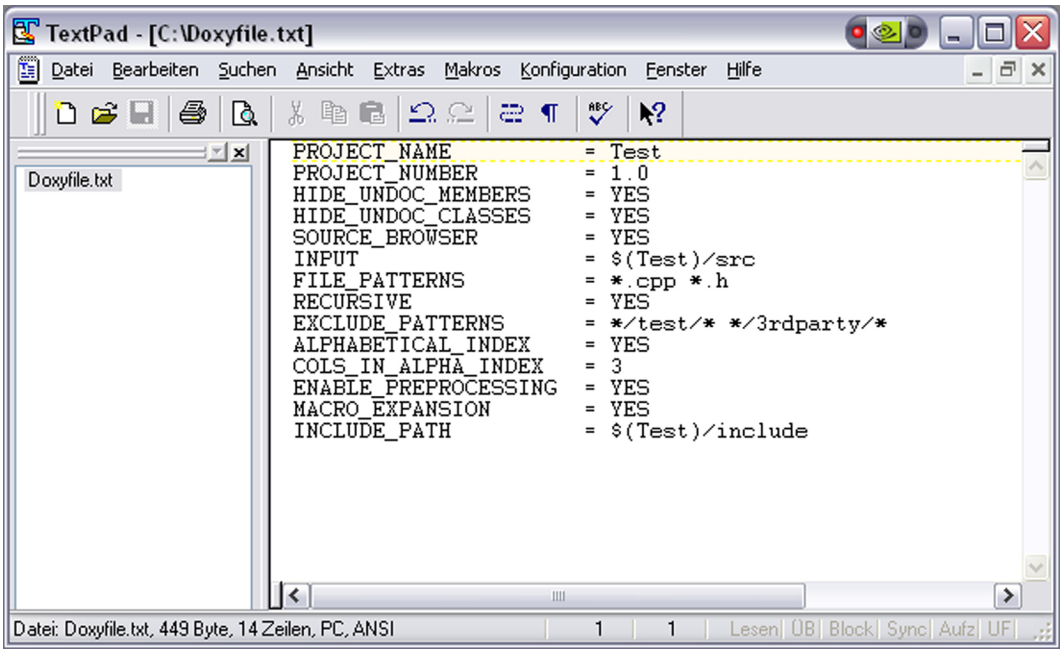

Abbildung 6: Doxygen - Konfigurationsdatei

Die Gruppe der projektrelevanten Einstellungen beinhaltet Parameter um sowohl projektübergreifende Daten, wie beispielsweise den Projektnamen, die Projektversion, den Ausgabepfad, die Ausgabesprache, als auch globale Optionen, wie Platzhalter, den Aufbau der Eingangsdaten, zu berücksichtigende Elemente und Optimierungen des Ausgabeformates einzustellen.

Unter den Build-Optionen sind diejenigen Einstellungen zu verstehen, welche sich direkt auf den Erstellungsprozess der Dokumentation beziehen. Hierbei kann festgelegt werden, welche Teile des Quellcodes einbezogen werden sollen und welche Sichtbarkeitsebenen im Ergebnis zur Verfügung stehen. Des Weiteren kann die Konfiguration, hinsichtlich der auszugebenden Warnungen und Fehler, den individuellen Bedürfnissen angepasst werden.

Die Gruppe der eingabebezogenen Optionen verwaltet Parameter über die in die Dokumentation einfließenden Daten. Es werden vielfältige Möglichkeiten geboten, bestimmte Dateien und Dateizusammenstellungen in das Dokumentationsprojekt einzubeziehen bzw. auszuschließen. Darüber hinaus besteht die Möglichkeit automatisiert alphabetische und Quellen-Indizes generieren zu lassen.

Die ausgabebezogenen Einstellungen beinhalten alle Parameter für die Konfiguration der zu erstellenden Dokumentationsformate, unterteilt in typspezifische Untergruppen. Dies ist notwendig, da beispielsweise HTML-Dokumente andere Formatanforderungen, als LaTeXoder RTF-Dokumente stellen. So können für eine HTML-Ausgabe detailliert die Kopf- und Fußzeilendaten festgelegt werden, wohingegen für eine LaTeX-basierte Ausgabe umfangreiche Einstellungen hinsichtlich des Seitenlayoutes, der zu

erstellenden Indizes und des zu verwendenden Zielformates getätigt werden.

Über diese bereits beschriebenen Optionen hinaus, bietet Doxygen Konfigurationsmöglichkeiten für die Einstellung des Präprozessors, das Einbinden externer Dokumente über so genannte „Tag-Files“. Falls ein externes Graphentool, wie „Graphviz“ verwendet wird oder die Option zur Erstellung einer Volltextsuche auf PERL-Basis aktiviert ist, können hierfür ebenfalls detaillierte Feineinstellungen getroffen werden.

3.1.6 Dokumentationsprozess

Für die Erstellung einer anwenderfreundlichen Programmdokumentation, ist es notwendig den Quellcode der Software mit beschreibenden Kommentaren anzureichern um nicht nur die Struktur und Abhängigkeiten der Programmbestandteile, sondern auch die Funktionsweise einzelner Module verständlich darzulegen. Aus diesem Grund verfügen aktuelle Programmiersprachen wie C, C++, C# oder Java jeweils über eine spezifische Dokumentationssyntax. Da diese jedoch von einer Programmiersprache zur anderen unterschiedlich umfangreich ausfällt, definiert Doxygen einen einheitliche Standard für die Quellcodedokumentation. Die verschiedenen Bestandteile einer Klassendokumentation sollen im Folgenden näher betrachtet werden.

Die Dokumentation des Klassenkopfes beinhaltet üblicherweise eine Beschreibung der Funktionen und des Aufbaus des abgebildeten Objektes, sowie Metainformationen für die Quellcode-Verwaltung. Die Metainformationen bestehen üblicherweise aus dem Autor der Klasse, dem Erstellungsdatum, der Versionsnummer, vorgenommenen Änderungen seit der Erstellung, Urheberschutzbemerkungen sowie aus Referenzen auf, mit dieser Klasse in Beziehung stehenden, Programmbestandteilen.

```
/**
 * /brief Kurzbeschreibung der Testklasse
 *
 * Dies ist die Langbeschreibung der Testklasse.
 *
 * @author NJ
 * @date   27.02.2004
 */
public class Test {
```

Abbildung 7: Doxygen - Beispieldokumentation eines Klassenkopfes

Wie in Abbildung 7 zu sehen ist, ist diese Art der Beschreibung stark an JavaDoc angelehnt. Im Gegensatz zu dieser Dokumentationssprache können die Ausführungen in Doxygen in eine Kurz- und Langfassung der Beschreibung gegliedert werden.

Member-Variablen, Konstanten und andere Konstrukte einer Klasse, können prinzipiell genauso wie diese dokumentiert werden. Sprachabhängig können weitere Befehle, wie z.B. \struct, \union, \enum, \file, \package u.v.a. genutzt werden.

```
/** Name der Testklasse */
private String name = "Test";
```

Abbildung 8: Doxygen - Beispieldokumentation von Member-Variablen

Des Weiteren besteht in Doxygen die Möglichkeit HTML-Elemente für Strukturbeschreibungen zu nutzen. Auf diese Weise können nicht nur Listen und Beispiele erzeugt, sondern auch globale Formatierungen überschreiben werden.

Die Syntax der Methodendokumentation entspricht ebenfalls dem bisher verwendeten Mustern. Ergänzt wird diese Beschreibungsstruktur um funktionsspezifische Aspekte, wie übergebene Parameter, dem Rückgabewert oder auftretenden Ausnahmen.

```
/**
 * Konstruktor der Testklasse.
 *
 * @param name der Name der Testklasse
 */
public Test( String name ) {
  setName( name );
}

/**
 * Setzt den Namen der Testklasse.
 *
 * @param name der Name der Testklasse
 */
public void setName( String name ) {
  this.name = name;
}

/**
 * Gibt den Namen der Testklasse zurück.
 *
 * @return den Namen der Testklasse
 */
public String getName() {
 return name;
}
```

Abbildung 9: Doxygen - Beispieldokumentation von Methoden

Falls es die Funktionsbeschreibung der Methoden erfordert, können über eine LaTeX-ähnliche Formelbeschreibungssprache mathematische Ausdrücke in die Dokumentation eingebunden werden. Für die Nutzung dieser Eigenschaft wird ein LaTeX-Compiler sowie ein PostScript-Konverter benötigt, um die Formeln in Form von Bitmaps in die Ausgabedateien einzubinden.

Über die hier vorgestellten Konstrukte hinaus, bietet Doxygen weitere Möglichkeiten die zu erstellende Dokumentation zu strukturieren. Diese Möglichkeiten umfassen neben dem Einbinden von Grafiken und Diagrammen und der Einbindung externer Dokumente über so genannte Tag-Files auch das logische Gruppieren der Quelltextbestandteile innerhalb der Ausgabedokumente.

3.1.7 Ausgabeformate

Das Werkzeug Doxygen bietet die Möglichkeit verschiedene Ausgabeformate der Dokumentation zu generieren, welche in „on-line“ und „off-line“ Formate unterschieden werden. Unter „on-line“ Formaten sind Dokumente zu verstehen, welche Navigationselemente in Form von Hyperlinks enthalten bzw. interpretierbare Beschreibungssprachen darstellen. In diesem Rahmen unterstützt Doxygen HTML und komprimiertes HTML, welches z.B. im Windows-Hilfesystem Verwendung findet. So genannte „offline“ Formate bezeichnen Dokumenttypen, deren Darstellung system- und beschreibungsunabhängig ist. Dies umfasst hierbei Formate wie Text, Man-Pages, RTF und LaTeX. Durch die Unterstützung des LaTeX-Formates wurde indirekt eine Möglichkeit geschaffen viele weitere Dokumenttypen zu erzeugen, da LaTeX-Dokumente unter Verwendung spezieller Compiler in verschiedene Formate, wie beispielsweise PDF (Portable Document Format), überführt werden können. Falls sich diese Optionen der Dokumentationsgenerierung als nicht ausreichend erweist, kann die Dokumentation im XML-Format generiert werden. Dieses sowohl für Menschen als auch Maschinen gut lesbare Format kann beispielsweise unter Zuhilfenahme von XSLT beliebige Ausgabedaten generieren, welche wiederum den Ausgangspunkt einer weiteren Verarbeitung, z.B. mit DocBook, darstellen können.

Doxygen stellt ein mächtiges Werkzeug zur Unterstützung der Softwaredokumentation dar. Durch die Fülle an Konfigurations- und Integrationsmöglichkeiten, bietet sich dem Entwickler die Möglichkeit auf die projektspezifischen Rahmenbedingungen abgestimmte Programmdokumentation zu erstellen. In diesem Rahmen kann die formale Güte der Dokumentation wesentlich gesteigert, sowie durch konsequenten Einsatz eine inhaltlich vollständige und zielgruppenorientierte Softwarebeschreibung erreicht werden. Dies erhöht die Dokumentationsqualität in einem nicht unerheblichen Maße.

3.2 DocBook

3.2.1 Historie

DocBook ist eine Sprache zur Erstellung technischer Dokumentationen und basiert auf der Strukturierung des Inhaltes. Aus diesem Grund stützt sich DocBook ursprünglich auf die Auszeichnungssprache SGML, jedoch wird in aktuellen Versionen vermehrt die Sprache XML verwendet. Das 13 Jahre alte DocBook-Format wurde 1991 als ein gemeinsames Projekt des O'Reilly-Verlages und HAL Computer Systems entwickelt, mit dem Ziel den Austausch von UNIX-Dokumentationen zu verbessern. Im Jahr 1998 traten diese Entwicklungspartner in die „Organization for the Advancement of Structured Standards" (OASIS) ein, um die industrielle Standardisierung des Werkzeuges DocBook voranzutreiben. In den folgenden Jahren schlossen immer mehr bekannte Firmen, wie beispielsweise Novell und Digital, diesem Komitee an um bei der Entwicklung des DocBook-Standards mitzuwirken[48].

3.2.2 Eigenschaften

DocBook kann im weiteren Sinne als eine lose Sammlung aus Beschreibungs- und Transformationsdateien bezeichnet werden, mit dem Ziel verschiedene Formen von Dokumenten wie Hilfesysteme, Webseiten, Bücher, FAQ's, Artikel, Bedienungsanleitungen u.v.a. zu erstellen. Hierbei ist die Dokumentationssyntax des DocBook-Systems in einer umfangreichen DTD (Document Type Definition) abgelegt und bietet dem Softwareentwickler ein Repository mit über 400 Anweisungen (Tags). Diese gegebene Flexibilität kann durch das Erweitern der vorhandenen DTD um eigene Dokumentdefinitionen weiter verstärkt werden. Die Ausgabeerzeugung kann ebenfalls direkt beeinflusst werden, da sie vollständig auf eine Verarbeitung von XSL-Transformationsbeschreibungen ausgelegt ist, und somit die Generierung individueller Formate ermöglicht. Diese Zusammenstellung aus Beschreibungs- und Transformationsvorlagen garantiert eine vollständige Unabhängigkeit hinsichtlich des eingesetzten Betriebssystems und der, für den Erstellungsprozess, verwendeten Programme[49].

3.2.3 Interne Funktionsweise

Die einzelnen Abschnitte der Dokumentationsgenerierung mit DocBook gliedern sich im Wesentlichen in 3 Phasen. In einem ersten Schritt muss das Quelldokument in der Sprache XML durch den Entwickler erzeugt oder aus anderen Formaten generiert werden. Nach dieser Tätigkeit wird die Gültigkeit des fertigen Quelldokumentes anhand

[48] vgl. [Tri02]
[49] vgl. [Wal03]

der DocBook-DTD validiert, um die Erstellung eines Objektbaumes durch den XML-Parser zu ermöglichen. In der letzten Phase durchläuft die abgebildete Dokumentstruktur den XSL-Transformationsprozessor um in das gewünschte Ausgabeformat abgebildet zu werden[50].

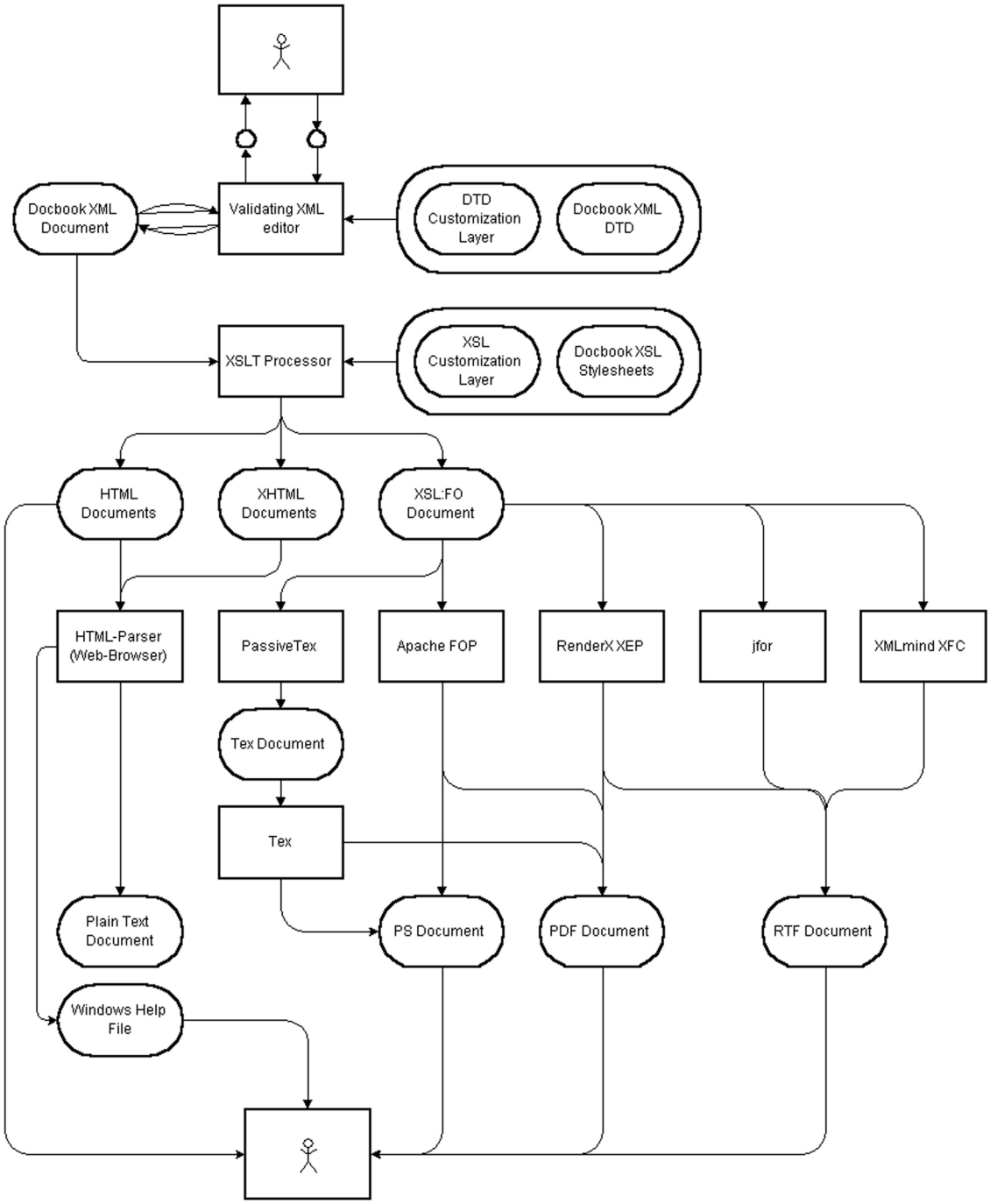

Abbildung 10: DocBook - interne Funktionsweise

[50] vgl. [Wal03]

3.2.4 Konfiguration

Da DocBook kein Programm im eigentlichen Sinne darstellt, verfügt es auch nicht über eine zentrale Konfigurationsdatei oder entsprechende Oberflächen um Einstellungen zu tätigen. Vielmehr stellt die verwendete DTD eine Art Konfigurationsmechanismus des Werkzeuges dar, über die benutzerdefinierte Anpassungen vorgenommen werden können. Die einzelnen Ebenen und Bestandteile sind in der Abbildung 11 aufgelistet und sollen in den folgenden Ausführungen kurz beschrieben werden[51].

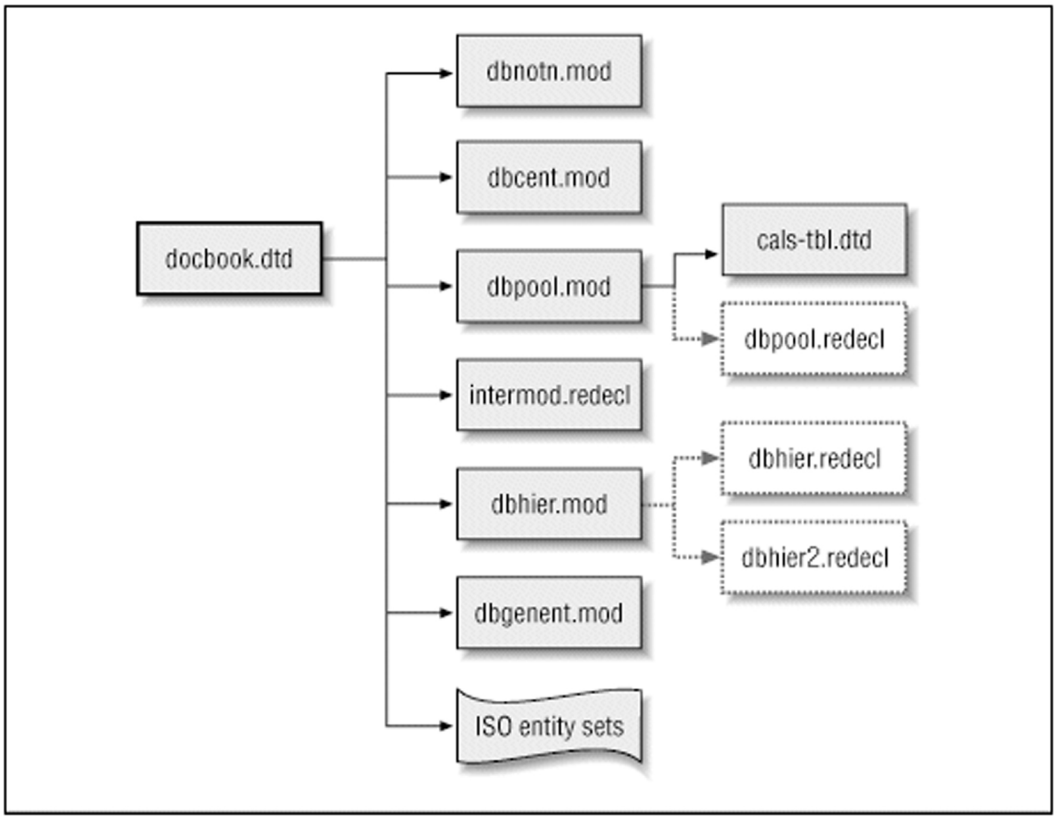

Abbildung 11: DocBook - Document Type Definition

Die Datei „docbook.dtd“ stellt die Haupttreiberdatei dar, in welcher alle einzubindenden Module deklariert werden. Sie enthält darüber hinaus Referenzen auf weitere Hauptmodule, wie in Abbildung 11 dargestellt.

Unter dem Modul „dbhier.mod“ wird die gesamte Hierarchie des DocBook-Systems zusammengefasst. Somit sind in ihr alle Befehle, welchen den Aufbau des DocBook-Dokumentationsformates betreffen, enthalten. Änderungen an dieser Datei wirken sich direkt auf die Hauptstruktur der zu erzeugenden DoxyGen-Dokumente aus, so dass dies der idealen Platz für individuelle Umbauten des Dokumentaufbaus ist.

Die DTD „dbpool.mod“ enthält den Informationspool. Dies bedeutet, dass die in dieser Datei enthaltenen Elemente den Inhalt des Dokumentes, beispielsweise die Zitierwei-

[51] vgl. [Tri02]

se, eingebettete Elemente sowie das Inhalts- und Literaturverzeichnis, beschreiben. Änderungen sollten an dieser Datei nur vorgenommen werden, wenn die Struktur um selbsterstellte Elemente, z.B. gesondert darzustellende Zitate oder Quelltexte, erweitert werden soll.

Die Datei „dbnotn.mod“ bildet die eigentliche Dokumentationsnotation ab und kann somit um ergänzende Anweisungen erweitert bzw. aus Gründen der Handhabbarkeit gekürzt werden.

Das Modul „dbcent.mod“ legt fest, welche Zeichen durch das DocBook-Format abgebildet werden können und enthält vorkonfigurierte ISO-Zeichensätze.

Das Dokument „dbgenent.mod“ enthält eine Sammlung der wichtigsten DocBook-Elemente. Somit stellt sie die zentrale Stelle für Erweiterungen, beispielsweise in Form von Firmenlogos, Adresselementen oder individuellen Textbausteinen, dar.

Die letzte Konfigurationsmöglichkeit bildet die Datei „cals-tbl.dtd“. Diese enthält Deklarationen für die Tabellendarstellung in DocBook. Der hier definierte Tabellenstandard lehnt sich an das CALS-Tabellenmodell an, welches durch das „United States Department of Defense“ entwickelt und standardisiert wurde. Aus diesem Grund sollten Änderungen, welche die Abbildung tabellarischer Daten betreffen, in der Datei „dbpool.mod“ getätigt werden, um diese standardisierte Definition nicht zu verändern.

```
Die meisten Element-Deklarationen können an dieser Stelle überladen werden

<!ENTITY % orig-pool "-//OASIS//ELEMENTS DocBook Information Pool V3.1//EN">
%orig-pool;

Die Dokument-Hierarchie-Elemente können an dieser Stelle überladen werden

<!ENTITY % orig-hier "-//OASIS//ELEMENTS DocBook Document Hierarchy V3.!//EN">
%orig-hier;

Neue bzw. abgeänderte Elemente können an dieser Stelle definiert bzw. überladen werden

<!ENTITY % orig-notn "-//OASIS//ELEMENTS DocBook Notations V3.1//EN">
%orig-notn;

Zeichensätze können an dieser Stelle definiert und geändert werden

<!ENTITY % orig-cent "-//OASIS//ELEMENTS DocBook Character Entities V3.1//EN">
%orig-cent;


Tp-Level Elemente können an dieser Stelle definiert bzw. überladen werden

<!ENTITY % orig-gen "-//OASIS//ELEMENTS DocBook Additional General Entities V3.1//EN">
%orig-gen;
```

Abbildung 12: DocBook - Beispiel einer DocBook-DTD

Wie in den vorhergehenden Ausführungen aufgezeigt, bestehen vielfältige Möglichkeiten DocBook den individuellen Projekt- und Dokumentationsanforderungen anzupassen. Dies ermöglicht es beispielsweise, durch das Einbinden eigener DTD-Dateien, externe Quellformate, wie Doxygen-XML, in DocBook zu integrieren und mit geeigneten XSL-Transformationsbeschreibungen in die gewünschten Ausgabeformate zu konvertieren.

3.2.5 Dokumentationsprozess

Der eigentliche Dokumentationsprozess mit DocBook gestaltet sich relativ einfach, da die komplette Erstellung in XML erfolgt. Für diesen Prozess können sowohl einfach Textbearbeitungsprogramme wie VI, EMACS, Notepad oder TextPad verwendet werden, als auch spezielle XML-Editoren wie XMLMind oder XML-Spy. Spezialisierte XML-Editoren bieten den großen Vorteil, dass die komplette Struktur der erzeugten Quelldokumente als ein Objektbaum abgebildet wird, durch den es sich wesentlich leichter navigieren lässt als durch eine „flache“ Textdatei. Des Weiteren bieten diese Programme Möglichkeiten die erzeugten Textbausteine gegen die DocBook-DTD auf ihre Gültigkeit hin zu validieren.

DocBook kann in drei Typen von Quelldokumenten unterscheiden[52]. Für kurze Artikel, technische Notizen oder FAQ's eignet sich das Format „article“. Sollen hingegen größere Dokumente, wie Bücher, Handbücher, Bedienungsanleitungen oder andere Nachschlagewerke erzeugt werden, sollte der Typ „book“ verwendet werden, da dieser eine Gliederung in Kapitel und Unterabschnitte erlaubt. Eine dritte Möglichkeit bildet das Format „set“, welches als eine Sammlung von Büchern verstanden werden kann. Diese Dreiteilung dient in erster Linie der Einschränkung von unnötigen Formatbeschreibungen während des Dokumentationsprozesses. So ist es für einen kurzen Artikel nicht nötig einzelne Kapitel, wie es die Vorlage „book“ verlangt, zu definieren. Darüber hinaus können während des Transformationsprozesses der Ergebniserstellung bestimmte Gestaltungselemente in Abhängigkeit des verwendeten Dokumenttyps unterschiedlich erzeugt werden, z.B. verschiedene Ränder für „book“ und „article“.

Jedes DocBook-Dokument beginnt mit der Definition der XML-Version und dem verwendeten Zeichensatz, wie in Abbildung 13 dargestellt.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
                      "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
```

Abbildung 13: DocBook - Header eines DocBook-Dokumentes

Nachfolgend wird der verwendete Dokumenttyp festgelegt, wobei an dieser Stelle nur auf den Typ „book“ eingegangen wird. Diesem folgt mindestens ein Titel (title) oder einem Satz an Informationen über das Buch (bookinfo) und eine beliebig lange Liste an weiteren Buchbestandteilen, wie Inhaltsverzeichnissen (toc), Vorworten (preface), Kapiteln (chapter), Glossars (glossary), Stichwortverzeichnissen (index) und Anhängen (appendix). Diese Elemente sind bis auf den Titel und die Buchinformationen optional.

[52] vgl. [Tri02]

```
<book lang="de">

  <!-- Buchinformationen                                          -->
  <bookinfo>

    <!-- Titel des Buches                                        -->
    <title>
      Buchtitel
    </title>

  </bookinfo>

  <!-- Inhaltsverzeichnis                                         -->
  <toc />
```

Abbildung 14: DocBook - Einleitung eines DocBook-Dokumentes

Wie in Abbildung 14 zu sehen, wurde dem Element „book“ der Parameter „lang“ mit dem Wert „de“ zugeordnet. Es handelt sich hierbei um ein Universalattribut, welches in jedem Element vorkommen kann. In diesem Fall bedeutet es, dass das Buch in deutscher Sprache verfasst ist. Dies ist besonders geeignet um fremdsprachige Teile in einem Dokument zu markieren.

Diesen einleitenden Bestandteilen eines Buches folgen die eigentlichen Kapitel (chapter) inklusive ihren jeweiligen Titeln (title) und den entsprechend ausformulierten Absätzen (para).

```
  <!-- Kapitel                                                    -->
  <chapter>

    <!-- Titel des Kapitels                                      -->
    <title>
      Titel des ersten Kapitel
    </title>

    <!-- Abschnitt des Kapitels                                  -->
    <para>
      Text des ersten Kapitel
    </para>

  </chapter>

</book>
```

Abbildung 15: DocBook - Kapitel eines DocBook-Dokumentes

Das hier aufgeführte Beispiel eines DocBook-Dokumentes kann in der Praxis um diverse zusätzliche Strukturelemente erweitert werden. So stehen dem Autor der Dokumentation beispielsweise Gestaltungselemente wie Listen, Tabellen, Beispiele und Abbildungen, Fußnoten, Referenzen und u.v.a. zur Verfügung. Darüber hinaus können die zu verarbeitenden Quellen profiliert werden, worunter die zielgruppenorientierte Erstellung eines Dokumentes zu verstehen ist. Hierfür wird in den Ausführungen definiert, welcher Abschnitt für welche Benutzergruppe sichtbar sein soll. Um die entsprechenden Elemente zu markieren, stehen eine Reihe von Attributen zur Verfügung, nach denen im Verarbeitungsstadium gefiltert werden kann. Diese Profilierung kann nach den Gesichtspunkten Hardwarearchitektur (arch), Betriebssystem (os), Sicherheitsebene (security), Benutzererfahrung (userlevel) und Zwischenhändler (vendor).

3.2.6 Ausgabeformate

Bedingt durch die Konvertierung der XML-Eingangsdaten, unter Verwendung von XSLT, in beliebige Zielformate, steht dem Entwickler ein fast unbeschränktes Spektrum an Konvertierungsmöglichkeiten offen[53]. In Kombination mit den DocBook-DTD Dateien wird häufig das standardisierte Paket DocBook-XSL-Stylesheets ausgeliefert, welches bereits Transformationsskripte für die Formate HTML, XHTML, komprimiertes HTML (Windows-Hilfe), Java-Help, iSilo, Plucker und XSL:FO, der Vorstufe für eine Generierung von PS- und PDF-Dateien, bereitstellt. Diese können unter Verwendung eines beliebigen XSLT-Prozessor, wie z.B. Xalan oder Saxon, auf die Quell-XML-Dateien angewandt werden. Im Rahmen dieser umfangreichen Transformationsmöglichkeiten firmieren unter diversen OpenSource-Dachorganisation, wie beispielsweise SourceForge, Projekte welche sich die Konvertierung des DocBook-Standards in andere Fremdformate zum Ziel gesetzt haben. Durch diese Bemühungen stehen momentan z.B. Stylesheets für eine Konvertierung in LaTeX, StarWriter oder in das RTF-Format zur Verfügung, welche frei nutzbar sind.

Wie den vorhergehenden Ausführungen zu entnehmen ist, steht dem Softwareentwickler bzw. technischem Autor mit DocBook ein mächtiges und individuell konfigurierbares Werkzeug zur Verfügung. Voraussetzung für einen erfolgreichen ergebinsorientierten Einsatz ist die umfassende Kenntnis der durch DocBook angebotenen Dokumentationselemente und ein gewisses Basiswissen der Auszeichnungssprache XML, sowie der damit verbundenen Sprachen DTD und XSLT. Diesen hohen Einarbeitungsaufwand rechtfertigt jedoch die beschriebene Flexibilität des Werkzeuges, wie der Freiheit in der Wahl des Zielformates, der maßgeschneiderten Dokumentationserstellung für unterschiedliche Zielgruppen und Systeme sowie der freien Gestaltung individueller Dokumentationsstrukturen.

[53] vgl. [Wal03]

4 Fazit

Mit Hilfe der vorliegenden Arbeit wird aufgezeigt, dass in der Fachwelt keine einheitliche Auffassung des Dokumentationsbegriffs im Allgemeinen und der Softwaredokumentation im Speziellen vorliegt. Aus diesem Grund wurde das Umfeld der Softwaredokumentation unter der Berücksichtigung verschiedener Einflussfaktoren näher betrachtet und seine Bestandteile gegenüber der Umwelt abgegrenzt. Dies ermöglichte es die einzelnen an der Dokumentationserstellung beteiligten Organisationseinheiten festzulegen und ihre Beteiligung an einzelnen Aspekten der Dokumentation aufzuzeigen. Durch die Analyse allgemeiner und spezieller Merkmale der Benutzer einer Softwaredokumentation konnten zudem einzelne Zielgruppen definiert und gegeneinander abgegrenzt werden. In einem weiteren Schritt wurden anhand der zu erfüllenden Funktionen einzelne Dokumentationsarten festgelegt und mit den ermittelten Zielgruppen in Relation gesetzt. Diese gewonnenen Erkenntnisse ermöglichten eine genaue Einordnung der Dokumentationsbestandteile, der beteiligten Organisationseinheiten und der angesprochenen Zielgruppen in die Phasen der Softwareentwicklung.

Durch die genaue Darlegung des Dokumentationsprozesses konnten in einer weiteren wissenschaftlichen Betrachtung die Anforderungen an die Ergebnisse dieses Ablaufes definiert werden. Dies und die Einbeziehung verschiedener Dokumentationsstandards ermöglichten eine genaue Abbildung der qualitativen Anforderungen an die einzelnen Dokumentationsbestandteile.

Die auf diesem Wege erworbenen Erkenntnisse belegen eindeutig, dass der Dokumentationsprozess, durch seine hohe Durchdringung der Softwareentwicklung, nur in seiner Gesamtheit beschrieben werden kann. Dies und die projektspezifischen Dokumentationsanforderungen erschweren eine eindeutige Definition der anfallenden Dokumentationsarten und der zu ihrer Erstellung notwendigen Tätigkeiten. In diesem Kontext ist es zwar möglich Rahmen qualitativer Anforderungen, z.B. durch die IEEE 1063, zu erstellen, dessen Bestandteile müssen jedoch den individuellen Anforderungen entsprechend inhaltlich gefüllt und angepasst werden.

Um den Aufwand dieser umfangreichen Arbeiten weitestgehend zu minimieren, wurden in Abschnitt 3 der Ausführungen technische Werkzeuge und Hilfsmittel aufgezeigt, die es den an der Softwareentwicklung beteiligten Personen ermöglichen einen Teil der Dokumentationserstellung zu automatisieren. Durch die Verwendung dieser modernen Technologien ist es möglich ein unternehmens- und projektbezogenes Dokumentationsframework zu erstellen, welches auf der einen Seite die individuellen Gestaltungs- und Qualitätsanforderungen erfüllt und auf der anderen Seite durch eine Modularisierung einzelne Dokumentationsbestandteile den entsprechenden Organisationseinheiten zuordnet. Dies und die Nutzung verschiedener Zielformate machen eine

redundante Datenhaltung überflüssig, wodurch direkte positive Auswirkungen hinsichtlich der Dokumentationsqualität zu beobachten sind. Darüber hinaus wird auf diesem Weg eine zielgruppenbezogene Dokumentationserstellung möglich.

Die dargestellten Hilfsmittel entlassen die an der Softwareentwicklung beteiligten Personen jedoch nicht aus der Pflicht inhaltlich hochwertige Dokumentationen zu erzeugen. Es kann lediglich ein unterstützendes Rahmenwerk geboten werden, dass eine Konzentration auf die thematischen Aspekte der Dokumentation erleichtert. Dies und ein übergreifendes Qualitätsmanagement, wie das ISO9000-Modell, können die Softwaredokumentation zu einem integralen und hochwertigen Bestandteil der eigentlichen Software werden lassen und so den Wert des Produktes für den Kunden als auch für die eigene Unternehmung wesentliche steigern.

Literatur

[Bal98] Helmut Balzert. *Lehrbuch der Software-Technik Band 2.* Spektrum Akademischer Verlag, Heidelberg-Berlin, 1998.

[Bal00] Helmut Balzert. *Lehrbuch der Software-Technik Band 1.* Spektrum Akademischer Verlag, Heidelberg-Berlin, 2000.

[DIN81] *DIN 66230.* Deutsches Institut Fur Normung E.V., 1981.

[Doc04] *DocBook.* Wikipedia, 2004. http://www.docbook.org/wiki/moin.cgi/DocBook.

[fQe91] Deutsche Gesellschaft für Qualität e.V. *Software-Qualitätssicherung.* TÜV Rheinland, Köln, 1991.

[Grü02] Gertrud Grünwied. *Unterstützung für Online-Hilfe-Redatkion durch die IEEE-Norm.* Wilken GmbH, Ulm, 2002. http://www.tekom.de/resourcen/pdf/ceb02_02k.pdf.

[Gro03] Chemnitzer Linux User Group. *Dokumentation - Die Achillesferse des leidenschaftlichen Softwareentwicklers.* 2003. http://www.clug.de/vortraege/doxygen/index.html.

[Gru86] Bruno Grupp. *EDV-Projekte richtig dokumentieren.* Beuth-Verlag, Berlin, 1986. DGQ-NTG-Schrift Nr. 12-51.

[Leh94] Franz Lehner. *Software Dokumentation und Messung der Dokumentationsqualität.* Carl Hanser Verlag - München, 1994.

[Mer98] Mertens. *Grundzüge der Wirtschaftsinformatik.* Springer, Heidelberg, 1998.

[Rau83] Sova Rautenberg. *Dokumentation computergestützter Informationssysteme.* Saur, K.G - München, 1983.

[Rey02] Eric Reymond. *DocBook Demystification HOWTO.* 2002. http://cvsview.tldp.org/index.cgi/*checkout*/LDP/howto/docbook/DocBook-Demystification-HOWTO/.

[Rup87] Walter Rupietta. *Benutzerdokumentation für Softwareprojekte.* BI Wissenschaftsverlag, Mannheim - Wien - Zürich, 1987. Herausgegeben von Helmut Balzert.

[Sch85] H.-J. Scheibl. *Wie dokumentiere ich ein DV-Projekt?* Sindelfingen, 1985.

[Sch90] Hoffmann Schlummer. *Erfolgreich beschreiben - Praxis des Technischen Redakteurs.* Berlin-Offenbach, 1990.

[Sch02] Greif / Schrepf. *Richtlinie für die Softwaredokumentation.* Physikalisch Technische Bundesanstalt, 2002. http://www.berlin.ptb.de/8/83/831/swq/doc/drl/drl.pdf.

[Tri02] Lars Trielof. *Dokumentationen mit DocBook-XML.* 2002. http://trieloff.net/docbook/tutorial/.

[vH04a] Dimitri van Heesch. *DoxyGen - Introduction.* 2004. http://www.stack.nl/ dimitri/doxygen/.

[vH04b] Dimitri van Heesch. *DoxyGen - Manual.* 2004. http://www.stack.nl/ dimitri/doxygen/manual.html.

[Wal90] Ernest Wallmüller. *Software - Qualitätssicherung in der Praxis.* Hanser Fachbuch, 1990.

[Wal03] Norman Walsh. *DocBook - The Definitive Guide.* O'Reilly & Associates, Inc., 2003.