

Bachelorarbeit

Entwicklung template-basierter Eingabemasken für SE-orientierte Wiki-Systeme

Autor: Ingo Göppinger
(Matrikelnummer: 0221994)

Betreuer: Prof. Dr. Peter Kaiser (HS Mannheim)
Dipl.-Inform. Jörg Rech (Fraunhofer IESE)

Hochschule Mannheim
Windeckstr. 110
68163 Mannheim

Studiengang Informatik Bachelor

Erklärung zur erstellten Arbeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig und ohne Hilfe Dritter angefertigt zu haben. Gedanken und Zitate, die ich aus fremden Quellen direkt oder indirekt übernommen habe, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und wurde bisher nicht veröffentlicht.

Ich erkläre mich damit einverstanden, dass die Arbeit durch das Fraunhofer Institut für Experimentelles Software Engineering der Öffentlichkeit zugänglich gemacht wird.

Mannheim, 28.02.2007

Unterschrift

Inhaltsverzeichnis

Abkürzungsverzeichnis	5
Allgemeine Hinweise zum Dokument	5
1. Einleitung	6
1.1 Allgemeines.....	6
1.2 Motivation.....	6
1.3 Aufgabenstellung.....	6
1.4 Ausgangssituation.....	7
1.5 Wesentliche Ergebnisse	7
1.6 Aufbau des Dokuments	8
2. State-of-the-art	9
2.1 Wiki.....	9
2.2 MediaWiki.....	10
2.3 XML	10
2.4 CSS.....	10
2.5 Standardisierte Dokumente in der Softwareentwicklung.....	10
3. Templates	12
3.1 Beschreibung der Quellen.....	12
3.1.1 IEEE.....	12
3.1.2 ANSI.....	12
3.1.3 Volere.....	12
3.1.4 Gang of Four.....	12
3.1.5 ReadySet.....	13
3.2 Beschreibung der Templates.....	13
3.2.1 ANSI/IEEE Std 830-1993 – Software Requirement Specification.....	13
3.2.2 Volere Requirement Specification Template.....	14
3.2.3 IEEE Std 829 - Testfallspezifikation.....	15
3.2.4 Architektur und Entwurfsmuster - Gang of Four Template.....	16
3.2.5 Change Request.....	16
3.2.6 ReadySET.....	17
3.2.6.1 ReadySET – Testcase.....	17
3.2.6.2 ReadySET - Use Case.....	17
3.3 Spezifikation in XML.....	18
3.3.1 ANSI/IEEE Std 830-1993.....	18
3.3.2 Volere Requirement Specification Template.....	20
3.3.3 ANSI/IEEE Std 829 - Testfallspezifikation.....	22
3.3.4 Architektur und Entwurfsmuster - Gang of Four Template.....	24
3.3.5 Change Request.....	25
3.3.6 ReadySET - Use Case.....	25
3.3.7 ReadySET – Testcase.....	26
4. Workflow	27

5. Realisierung	28
5.1 Anforderungen.....	28
5.1.1 Verwendung von Templates zur Erstellung von Eingabemasken.....	28
5.1.2 Speichern eines Artikels, erstellt mit einem Template.....	29
5.1.3 Editieren von bereits bestehenden Artikeln.....	29
5.1.3.1 Bearbeiten mit altem Template.....	29
5.1.3.2 Bearbeiten mit anderer Version des verwendeten Templates.....	29
5.1.3.3 Bearbeiten eines Artikels mit anderem Template.....	30
5.1.3.4 Bearbeiten eines Artikels ohne Eingabemaske.....	30
5.1.3.5 Bearbeiten eines Abschnitts eines Artikels.....	30
5.1.4 Vorschaufunktion beim Editieren.....	30
5.1.5 Layout der Formulare unter Verwendung von CSS.....	30
5.1.6 Layout der Ausgabemasken.....	31
5.2 Architektur.....	33
5.3 Implementierung.....	33
5.3.1 TemplateParser.php.....	34
5.3.2 IESEHooks.php.....	34
5.4 Test.....	34
6. Validierung	35
7. Zusammenfassung der Ergebnisse	36
7.1 Einblicke in die Applikation.....	36
7.2 Verwendung der Software.....	37
7.2.1 Vorlage erstellen.....	37
7.2.1.1 Von der Vorlage zur XML-Spezifikation.....	38
7.2.2 Vorlage nutzen.....	39
7.2.3 CSS-Spezifikation.....	39
8. Ausblick	41
9. Literaturverzeichnis	42
10. Glossar	43

Abkürzungsverzeichnis

CSS	Cascading Style Sheets
CVS	Concurrent Versions System
DIN	Deutsches Institut für Normung
DOM	Document Object Model
GOF	Gang of Four
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
ISO	Internationale Organisation für Normung
PHP	Hypertext Preprocessor
SAX	Simple API for XML
SOP	Software Organisation Plattform
XML	Extensible Markup Language

Allgemeine Hinweise zum Dokument

Begriffe, die im Glossar (Kapitel 10) erläutert sind, werden kursiv geschrieben. Das Glossar ist alphabetisch geordnet.

Beispiel:
Auszeichnungssprache

Zitate oder Gedanken anderer Personen werden entsprechend kenntlich gemacht. Die verwendeten Kürzel können im Literaturverzeichnis (Kapitel 9) nachgeschlagen werden. Dort findet man jeweils den Autor, den Titel des Buches und das Erscheinungsjahr.

Beispiel:
[SE-IS-01]
steht für das Buch „Software Engineering“ von Ian Sommerville aus dem Jahr 2001.

Internetseiten werden an den entsprechenden Stellen ausgeschrieben und mit dem Datum versehen, an dem die Seite das letzte Mal abgerufen wurde.

1. Einleitung

1.1 Allgemeines

Softwareentwicklung ist inzwischen zu einem komplexen Prozess geworden, an dem meist viele Personen beteiligt sind und der viele verschiedene Phasen beinhaltet. Die Arbeiten, die im Laufe eines solchen Projekts anfallen, müssen in geeigneter Weise koordiniert und dokumentiert werden.

1.2 Motivation

Um Softwareentwickler und Projektleiter bei Ihrer Arbeit zu unterstützen und um die Arbeit innerhalb von Projektteams einfacher zu gestalten, werden meist viele verschiedene Tools verwendet. Hierfür ist es notwendig, dass sich jedes Projektmitglied mit der eingesetzten Software auseinandersetzt, sie kennen lernt und weiß an welcher Stelle bzw. mit welcher Software er die Information, die er gerade benötigt oder bearbeiten will, finden kann. Besser wäre es, wenn man die Tätigkeiten, die im Bereich Software-Entwicklung anfallen, mittels eines Tools – einer „Software Organisations Plattform“ – unterstützen könnte.

Nicht nur bei der Terminplanung oder bei der Kommunikation innerhalb der Projektteams sollte so ein Tool Unterstützung bieten. Auch bei der Erstellung typischer Dokumente, die für die Softwareentwicklung wichtig sind, z.B. bei der Sammlung von Anforderungen, der Erstellung von Testfällen oder dem Dokumentieren von Änderungsanforderungen, sollte ein solches Tool hilfreich sein. Für diese Dokumente gibt es verschiedene Vorgaben und *Normen*, die von verschiedenen Instituten und Organisationen definiert werden. Diese *Normen* sollten in einer solchen Software berücksichtigt werden. Es sollte möglich sein, bereits gespeicherte Vorlagen zu nutzen, neue hinzuzufügen oder alte Vorlagen zu ändern. Das Aussehen einer Vorlage sollte nach eigenen Wünschen anpassbar sein.

1.3 Aufgabenstellung

Aufgabe ist es template-basierte Eingabe- und Ausgabemasken für ein *WIKI-System* zu entwickeln, das speziell in der Softwareentwicklung eingesetzt wird. Zu diesem Zweck sollen unter anderem Templates für Anforderungen, Testfälle und Use Cases erstellt werden. Die Templates richten sich dabei nach Vorgaben und *Standards* wie sie beispielsweise von IEEE definiert worden sind.

Die Arbeit lässt sich in drei Bereiche unterteilen. Zunächst müssen die entsprechenden Vorlagen gefunden und ausgewählt werden. Diese Vorlagen werden dann in XML spezifiziert. Danach soll das *MediaWiki* so erweitert werden, dass diese Vorlagen im XML-Format als *Artikel* gespeichert und bei der Erstellung eines neuen *Artikels* ausgewählt werden können. Die Auswahl einer Vorlage soll bewirken, dass eine Eingabemaske geladen wird, die die Struktur des Dokuments, also der Vorlage, widerspiegelt. Die Eingabemasken sollen Elemente wie Textboxen, Selectboxen oder Radiobuttons enthalten. Es wird nicht für jedes Dokument eine eigene Eingabemaske geschrieben, sondern die Spezifikation in XML soll genügen,

entsprechende Eingabemasken zu generieren. So soll es später den Nutzern möglich sein, weitere Vorlagen in XML zu spezifizieren und zu speichern, ohne dass es nötig ist Eingabemasken in HTML zu definieren. Lediglich für das Erscheinungsbild werden zusätzlich noch CSS-Dateien bzw. CSS-Artikel im *Wiki* abgelegt, die einem Template, genauer gesagt einer Version eines Templates, zugeordnet werden. Für eine Vorlage ist es notwendig zwei verschiedene CSS-Spezifikationen zu hinterlegen, eine für das Erscheinungsbild der Ausgabemaske und eine für die Eingabemaske. Im Kapitel „Workflow“ werden die Anforderungen an das System und die Möglichkeiten, die der Nutzer durch das System bekommt, noch genauer erläutert.

1.4 Ausgangssituation

Das Fraunhofer Institut IESE in Kaiserslautern hat bereits eine eigene Software-Organisation-Plattform (SOP) im Einsatz, die ständig weiterentwickelt wird. Die Basis bildet das MediaWiki. Dabei handelt es sich um die selbe Wikiengine, die auch für die bekannte Online-Encyclopedie „Wikipedia“ verwendet wird.

Für die Entwicklung template-basierter Eingabe- und Ausgabemasken werden zwei verschiedene Ansätze verfolgt.

Der erste Ansatz nutzt neu angelegte Datenbanktabellen, wie z.B. die Tabelle „templatecontents“, in der die Inhalte von Templates abgelegt werden. Das Problem bei diesem Ansatz ist offensichtlich. Um Templates abzulegen, muss ein Nutzer direkt in der Datenbank Änderungen vornehmen bzw. seine Vorlagen speichern und bearbeiten. Außerdem wird hierbei keine Versionierung durch das *Wiki* vorgenommen.

Der zweite Ansatz entspricht dem, was schon in der Aufgabenstellung erläutert wurde. Vorlagen und CSS-Dateien sollen hierbei als normale *Wikiartikel* abgelegt und bei Bedarf überarbeitet werden können. Die Versionierung erfolgt automatisch, wie es bei gewöhnlichen *Wikiartikeln* der Fall ist.

Der mit dieser Arbeit umgesetzte zweite Ansatz, hat klare Vorteile gegenüber dem ersten Ansatz. Vorlagen werden automatisch versioniert und können einfach überarbeitet werden. Der Benutzer kann seine Vorlagen und somit auch seine Eingabemasken sehr leicht erstellen. Es ist lediglich notwendig die Vorlage in XML zu spezifizieren und als *Artikel* abzulegen. Um Elemente wie Radiobuttons oder Selectboxen zu erzeugen, muss sich der Benutzer nur an einfache Regeln bei der Definition der Vorlagen in XML halten. Auf diese Regeln wird im Laufe der Arbeit noch näher eingegangen.

1.5 Wesentliche Ergebnisse

Bei der Recherche der Vorlagen fiel auf, dass sich die meisten Autoren von Fachbüchern der Informatik, auf Quellen wie IEEE, ANSI, ISO oder DIN beziehen. Auch Unternehmen orientieren sich an diesen Quellen. Meistens werden die Vorlagen nicht 1:1 umgesetzt, sondern den Anforderungen entsprechend abgeändert.

Die Anforderungen an die Software konnten erfüllt werden. Templates können als normale *Artikel* abgelegt werden und für andere *Artikel* als Vorlage verwendet werden. Entsprechend der Spezifikation in XML wird die Eingabemaske bei der Auswahl eines Templates generiert. Ist zu einem Template zusätzlich noch ein *Artikel* mit CSS-Eigenschaften hinterlegt, so werden diese verwendet, um den *Artikel* oder die Eingabemaske entsprechend anzuzeigen.

1.6 Aufbau des Dokuments

Im zweiten Kapitel werden zunächst einmal Techniken und Methoden vorgestellt, die für diese Arbeit von Bedeutung sind. Hierbei wird der aktuelle Wissensstand wiedergegeben. Im dritten Kapitel werden die Vorlagen, sowie die entsprechenden Quellen beschrieben. Am Ende des Kapitels findet man als Ergebnis die Templates spezifiziert in XML. Im vierten Kapitel wird noch einmal der genaue Workflow erläutert, also welche Möglichkeiten die Software dem Benutzer bietet. Im darauf folgenden Kapitel wird geschildert, wie die Anforderungen realisiert wurden. Hier wird kurz auf die Implementierung und das Testen eingegangen. Das Kapitel enthält keine vollständige Spezifikation aller Funktionen und auch kein vollständiges Design. Es soll lediglich einen groben Einblick in die Realisierung bieten. Für die Beschreibung der neuen Funktionen gibt es ein zusätzliches Dokument. Im sechsten Kapitel wird beschrieben, wie die Validierung des Produkts durchgeführt wurde. Danach folgt eine Zusammenfassung der Ergebnisse. Am Ende der Arbeit folgen dann noch die Kapitel „Ausblick“, „Literaturverzeichnis“ und „Glossar“. Im Kapitel „Ausblick“ werden mögliche Folgearbeiten beschrieben, die auf dieser Arbeit aufbauen können.

2. State-of-the-art

2.1 Wiki

Den meisten Menschen, die schon einmal nach Informationen im Internet geschaut haben, dürfte die Seite von wikipedia (<http://de.wikipedia.org>) bekannt sein. Hierbei handelt es sich um eine öffentliche Enzyklopädie, in der es möglich ist, eigene Beiträge zu veröffentlichen oder bereits vorhandene Beiträge zu bearbeiten. Auf sehr einfache Art und Weise wird hier ermöglicht, sein Wissen mit anderen zu teilen.

Durch einfache Mechanismen kann den *Artikeln* ein strukturiertes Aussehen gegeben werden, z.B. durch Erstellen von Überschriften oder Tabellen. Außerdem hat man die Möglichkeit Verknüpfungen zu anderen *Artikeln* anzulegen. Normalerweise benötigt man Kenntnisse in HTML, um Internetseiten anzulegen. Aber um in einem *Wiki* eine Seite zu erstellen, braucht man keinerlei Wissen über HTML, da sich Elemente wie Überschriften, Aufzählungen oder Links auf einfache Art und Weise erzeugen lassen. Hierfür gibt es eine eigene, einfach gehaltene *Auszeichnungssprache* – die *Wiki Markup Language*.

Wikis ermöglichen aber nicht nur, dass jeder einen *Artikel* editieren oder anlegen kann, sondern jeder *Artikel* wird auch noch versioniert. Das bedeutet, dass man jederzeit auf ältere Artikelversionen zurückzugreifen kann. Sollte es einmal nötig sein eine Version eines *Artikels* gegen eine vorherige Version auszutauschen, so ist dies ebenfalls kein Problem. Änderungen, die an einem *Artikel* vorgenommen werden, lassen sich später leicht nachvollziehen.

Wikipedia ist nur eines von vielen *Wikis* im Internet aber mit Sicherheit das Bekannteste. In vielen Bereichen werden inzwischen *Wikis* eingesetzt um Informationen verfügbar zu machen und zu sammeln. So gibt es z.B. ein Projekt um einen vollständigen, aktuellen, verlässlichen und weltweiten Reiseführer zu erstellen unter <http://www.wikitravel.org>. Auch hier sind die Informationen in einem *Wiki* abgelegt. Unter <http://www.jurawiki.de> findet man eine freie Kommunikations- und Kooperations-Plattform für Juristen und juristisch Interessierte. *Wikis* sind also in völlig unterschiedlichen Gebieten einsetzbar.

In der Softwareentwicklung werden *Wikis* sehr häufig eingesetzt. Gerade in diesem Bereich ist der Austausch von Informationen essentiell. Ein gutes Beispiel stellt hier die Seite <http://linuxwiki.de> dar. LinuxWiki bietet eine Plattform für alle deutsch sprechenden GNU-, Linux-, OpenSource- und FreeSoftware-Interessierten an.

In Webanwendungen werden auch immer häufiger *Wikis* integriert. Unter <http://trac.edgewall.org/> findet man zum Beispiel das Trac Project. Trac ist eine Software, die für Softwareentwickler geschrieben wurde. Es unterstützt Entwickler bei der Planung und Koordination von Projekten. Durch die Wikifunktionalität ist es möglich, das Tool auch als Kommunikationsplattform zu nutzen.

2.2 MediaWiki

Der Kern des Wikipedia und auch vieler anderer *Wikis* besteht aus dem *MediaWiki*. Man spricht auch von der Wiki-Engine. Das *MediaWiki* ist sozusagen ein *Wiki*, in dem es noch keine *Artikel* gibt. In ihm sind die Funktionalitäten enthalten, die ein *Wiki* anbieten muss. Das *MediaWiki* ist in PHP geschrieben und der Sourcecode kann kostenlos heruntergeladen, verwendet und auch modifiziert werden. Unter der Internetadresse <http://svn.wikimedia.org/doc/> findet man einigermaßen gut dokumentiert, wie das *MediaWiki* aufgebaut ist und welche Funktionen es gibt. Unter <http://meta.wikimedia.org> findet man zusätzliche Informationen, wie man das *MediaWiki* nach seinen Vorstellungen anpassen und erweitern kann.

Es gibt auch noch viele andere Wikis. Beschreibungen kann man zum Beispiel unter <http://www.wikimatrix.org> finden. Dort wird einem auch die Möglichkeit geboten Wikis miteinander zu vergleichen. Da die Basis des SOP das *MediaWiki bildet*, war es im Rahmen dieser Arbeit nicht notwendig, verschiedene Wikis miteinander zu vergleichen.

2.3 XML

Mit XML kann so ziemlich jede Art von Information in so genannten XML-Dokumenten gespeichert und organisiert werden. Da es sich um einen offenen *Standard* handelt, ist es weder an ein bestimmtes Unternehmen noch an eine bestimmte Software gebunden. XML stellt Methoden bereit um die Qualität eines Dokuments zu überprüfen. Darunter fallen Regeln für die Syntax, Überprüfung nach Datentypen, eine interne Link-Überprüfung und einiges mehr.
(vgl. [XML-ETR-04])

2.4 CSS

Cascading Style Sheets ist eine deklarative *Stylesheet*-Sprache für strukturierte Dokumente. Sie wird vor allem zusammen mit HTML und XML eingesetzt. CSS soll dabei festlegen, wie ein besonders ausgezeichneter Inhalt dargestellt werden soll.
(vgl. [XML-ETR-04])

2.5 Standardisierte Dokumente in der Softwareentwicklung

Viele Unternehmen und Organisationen befassen sich mit dem Thema *Standards*. Für fast alles gibt es Richtlinien oder Empfehlungen, wie etwas gebaut, erstellt oder aussehen sollte. *Standards* sollen dazu dienen, auf Bewährtem aufzubauen, brauchbare Teile zu verwenden und je nach Bedarf zu erweitern. *Standards* entwickeln sich mit der Zeit, wenn sich eine Art und Weise etwas zu machen gegenüber anderen Methoden durchsetzt.

Standardisierte oder genormte Dokumente sind so strukturiert, dass es dem Leser leichter fällt, sie zu lesen und zu verstehen. Das Suchen einer bestimmten Information wird durch das Muster, nach dem die Dokumente konzipiert sind, vereinfacht. Durch die Nutzung von standardisierten Dokumenten wird zusätzlich

sicher gestellt, dass man keine wichtigen Informationen beim Erstellen eines Dokuments vergisst. Ein standardisiertes Dokument beinhaltet alle wichtigen Teile für ein Problem.

In der Softwareentwicklung gibt es eine ganze Reihe von wichtigen Dokumenten und Teile von Dokumenten. Angefangen beim Pflichtenheft, über einzelne Use Cases bis hin zu Testfällen. Für Teilbereiche solcher Dokumente gibt es wiederum *Standards*. So orientieren sich z.B. viele Unternehmen bei der Erstellung eines Pflichtenheftes an der von IEEE standardisierten „Software Requirements Specification“ um ihre Anforderungen zu dokumentieren. Es ist nicht zwingend erforderlich, dass man sich genau an einen *Standard* hält, sondern meistens orientieren sich Firmen lediglich an ihnen und modifizieren sie je nach Bedarf, da nicht immer alle Kapitel notwendig sind. An die wesentlichen Punkte einer Spezifikation sollte man sich allerdings halten. (vgl. [SE-IS-01])

Durch die Nutzung solcher Richtlinien wird nicht nur die Arbeit der Entwicklerteams einfacher, sondern gleichzeitig ermöglicht man auch dem Kunden, dass er sich in einem Dokument leichter zurecht findet. Da diese *Standards* von den meisten Firmen eingehalten werden, kennt sich auch der Kunde im Normalfall mit dieser Art Dokument aus. Das Pflichtenheft stellt in der Softwareentwicklung die Vertragsgrundlage dar. Wenn die darin enthaltenen Anforderungen zum einen klar strukturiert sind und zum anderen einem bekannten Muster folgen, wird es für den Kunden einfacher zu erkennen, ob verstanden wurde was er von der Software erwartet.

IEEE gehört zu den renommierten Instituten. Weitere *Normen* und *Standards* werden z.B. durch *ISO* oder *DIN* festgelegt. Auf welche Quelle man zurückgreift bleibt jedem selbst überlassen, allerdings sollte man sich eben an den Bekanntesten und Renommierten orientieren und danach auswählen, was am geeignetsten ist.

In den meisten Unternehmen werden solche standardisierten Dokumente bereits verwendet. Meist werden diese Vorlagen dann an einer zentralen Stelle abgelegt, auf die alle Mitarbeiter Zugriff haben. Durch firmeninterne Regelungen wird dann vereinbart, dass man sich an diese Muster halten soll. Der Nutzen würde schließlich verloren gehen, wenn jeder seine eigenen modifizierten *Standards* verwendet.

Häufig werden solche Dokumente und Vorlagen in *Repositorys* abgelegt. Solche Systeme ermöglichen es, Dokumente zu versionieren und erleichtern die gemeinschaftliche Arbeit. Wer ein Dokument bearbeiten will, kann sich die entsprechende Datei herunterladen, bearbeiten und wieder in das *Repository* einfügen. So werden Dokumente oft mit Microsoft Word, Open Office Writer oder anderen Schreibprogrammen erstellt und die entsprechenden Dateien in einem *Repository* abgelegt. CVS und Subversion sind die bekanntesten *Repositorys*, die derzeit für solche Zwecke genutzt werden.

3. Templates

3.1 Beschreibung der Quellen

Bei der Recherche von geeigneten Quellen zur Erstellung der XML-Templates gibt es Kriterien. Die Vorlagen sollen möglichst weit verbreitet sein und häufig eingesetzt werden. Dann kann man davon ausgehen, dass sie auch nützlich und erprobt sind. Die bekanntesten Institute sind IEEE, ANSI, ISO und DIN. Im Bereich Anforderungsanalyse gibt es außerdem noch eine bekannte Sammlung von Hilfsmitteln und Materialien unter dem Namen Volere. Für die Beschreibung von Entwurfsmustern wird meist das Gang of Four Template verwendet. Eine weitere gute Sammlung von Vorlagen für Dokumente findet man außerdem unter <http://readysset.tigris.org>. Es ist sinnvoll in erster Linie auf diese Quellen zurückzugreifen. Sucht man nach weiteren Vorlagen, stellt man fest, dass es sich meist um leicht modifizierte Varianten der bereits genannten Quellen handelt.

Im folgenden werden die ausgewählten Quellen kurz vorgestellt. Die Auswahl der Vorlagen erfolgte nach Absprache mit dem Fraunhofer Institut IESE.

3.1.1 IEEE

„IEEE Software Engineering Standards“ werden in der Industrie eingesetzt, um den Gewinn in der Software-Entwicklung zu maximieren. Die Standards decken eine ganze Reihe von Bereichen wie Informatik, Qualitätsmanagement oder Projektmanagement ab. In ihnen werden Terminologien, Prozesse, Messmethoden, Dokumentvorlagen und einiges mehr beschrieben.

(vgl. <http://standards.ieee.org/software> Datum 05.01.07)

3.1.2 ANSI

ANSI ist das nationale Standardisierungsgremium der USA. Es entwickelt und publiziert *Standards*. ANSI ist nicht gewinnorientiert, regierungsunabhängig und wird von mehr als 1000 Gewerbeorganisationen, Berufsvereinigungen und Firmen unterstützt. ANSI ist der amerikanische Vertreter und stimmberechtigtes Mitglied bei ISO.

(vgl. <http://www.itwissen.info/definition/lexikon> Datum: 11.01.07)

3.1.3 Volere

Volere ist eine Sammlung von Hilfsmitteln und Materialien rund um das Thema Anforderungsanalyse im Softwareentwicklungsprozess. Die Volere-Templates können kostenfrei heruntergeladen und verwendet werden.

(vgl. <http://www.volere.de> und <http://de.wikipedia.org/wiki/Volere> Datum: 14.02.07)

3.1.4 Gang of Four

Mit „Gang of Four“ sind die vier Autoren des Buches „Design Patterns - Elements of Reusable Object-Oriented Software“ gemeint. Das Buch gilt als Standardwerk in der Softwareentwicklung im Bereich Entwurfsmuster.

3.1.5 ReadySet

ReadySET ist ein Open Source Projekt, dessen Ziel es ist, Templates für Dokumente zur Verfügung zu stellen, die in der Softwareentwicklung immer wieder benötigt werden.

(vgl. <http://readysset.tigris.org/> Datum: 14.02.07)

3.2 Beschreibung der Templates

3.2.1 ANSI/IEEE Std 830-1993 – Software Requirement Specification

Die Software Requirement Specification ist ein von IEEE veröffentlichter *Standard* zur Spezifikation von Software. (vgl. [IEEESC-94])

Im folgenden kann man den Aufbau sehen, wie er von IEEE definiert worden ist:

1. Introduction (Einleitung)
 - 1.1.Purpose (Zweck des Dokuments)
 - 1.2.Scope (Ziel des Softwareprodukts)
 - 1.3.Definitions, acronyms, and abbreviations
(Erläuterungen zu Begriffen und / oder Abkürzungen)
 - 1.4.References (Verweise auf sonstige Ressourcen oder Quellen)
 - 1.5.Overview (Übersicht: Wie ist das Dokument aufgebaut?)

2. Overall description (Allgemeine Beschreibung des Softwareprodukts)
 - 2.1.Product perspective
(Produkt Perspektive zu anderen Softwareprodukten)
 - 2.1.1.System interfaces
 - 2.1.2.User interfaces
 - 2.1.3.Hardware interfaces
 - 2.1.4.Software interfaces
 - 2.1.5.Communications interfaces
 - 2.1.6.Memory constraints
 - 2.1.7.Operations
 - 2.1.8. Site adaptation requirements
 - 2.2.Product functions (Produktfunktionen - Zusammenfassung)
 - 2.3.User characteristics (Benutzermerkmale)
 - 2.4.Constraints (Einschränkungen für den Entwickler)
 - 2.5.Assumptions and dependencies (Annahmen und Abhängigkeiten)
 - 2.6.Appportioning of requirements

3. Specific requirements (Spezifizierte Anforderungen)
 - 3.1.External interfaces
 - 3.2.Functions
 - 3.3.Performance requirements
 - 3.4.Logical database requirements
 - 3.5. Design constraints
 - 3.5.1. Standards compliance
 - 3.6.Software system attributes (Anforderungen an Performance)
 - 3.6.1.Reliability

- 3.6.2.Availability
 - 3.6.3.Security
 - 3.6.4.Maintainability
 - 3.6.5.Portability
 - 3.7.Organizing the specific requirements (Qualitätsanforderungen)
 - 3.7.1.System mode
 - 3.7.2.User class
 - 3.7.3.Objects
 - 3.7.4.Feature
 - 3.7.5.Stimulus
 - 3.7.6.Response
 - 3.7.7.Functional hierarchy
 - 3.8.Additional comments (Sonstige Anforderungen)
4. Supporting information
- 4.1.Table of contents and index
 - 4.2.Appendixes

3.2.2 Volere Requirement Specification Template

Das „Volere Requirement Specification Template“ dient als Basis für die Spezifikationen der Anforderungen. Das Template enthält für jeden Anforderungstyp einen eigenen Abschnitt. (vgl. [VRST-JSR-06])

Der Aufbau des Templates sieht folgendermaßen aus:

1. Project Drivers (Projekttreiber)
 - 1.1.The Purpose of the Project (Der Zweck des Projekts)
 - 1.2.The Client, the Customer, and Other Stakeholders
(Kunden und andere Beteiligte/Betroffene)
 - 1.3.Users of the Product (Nutzer des Produkts)
2. Project Constraints (Randbedingungen)
 - 2.1.Mandated Constraints (Vorgegebene Randbedingungen für das Projekt)
 - 2.2.Naming Conventions and Definitions
(Namenskonventionen und Definitionen)
 - 2.3.Relevant Facts and Assumptions (Relevante Fakten und Annahmen)
3. Functional Requirements (Funktionale Anforderungen)
 - 3.1.The Scope of the Work (Arbeitsumfang)
 - 3.2.The Scope of the Product (Abgrenzung des Produkts)
 - 3.3.Functional and Data Requirements
(Anforderungen an Funktionen und Daten des Produkts)
4. Nonfunctional Requirements (Nicht-funktionale Anforderungen)
 - 4.1.Look and Feel Requirements
 - 4.2.Usability and Humanity Requirements (Benutzbarkeitsanforderungen)
 - 4.3.Performance Requirements
(Performance / Durchsatz / Kapazität / Sicherheit)
 - 4.4.Operational and Environmental Requirements
(Operationelle Anforderungen)

- 4.5.Maintainability and Support Requirements
(Wartungs- und Portierungsanforderungen)
- 4.6.Security Requirements (Zugriffsschutzanforderungen)
- 4.7.Cultural and Political Requirements
(Kulturelle und politische Anforderungen)
- 4.8.Legal Requirements (Rechtliche Anforderungen)

- 5. Project Issues (Projektangelegenheiten)
 - 5.1.Open Issues (Offene Punkte)
 - 5.2.Off-the-Shelf Solutions (Fertiglösungen)
 - 5.3.New Problems (Neue Probleme)
 - 5.4.Tasks (Aufgaben)
 - 5.5.Migration to the New Product (Inbetriebnahme und Migration)
 - 5.6.Risks (Risiken)
 - 5.7.Costs (Kosten)
 - 5.8.User Documentation and Training
(Benutzerdokumentation und Schulung)
 - 5.9.Waiting Room (Wartezimmer)
 - 5.10.Ideas for Solutions (Lösungsideen)

3.2.3 IEEE Std 829 - Testfallspezifikation

Die Testfallspezifikation (IEEE Std 829) beschreibt wie ein Testfall aufgebaut und beschrieben werden kann.

(vgl. [STD-JT-02])

- 1. Test-case-specification identifier (Testfallspezifikation ID)
- 2. Test items (Testobjekte)
- 3. Input specifications (Eingaben)
- 4. Output specifications (Ausgaben)
- 5. Environmental needs (Umgebung)
 - 5.1.Hardware
 - 5.2.Software
 - 5.3.Other Requirements (Sonstige Anforderungen)
- 6. Special procedural requirements (Besonderheiten)
- 7. Intercase dependencies (Abhängigkeiten)

3.2.4 Architektur und Entwurfsmuster - Gang of Four Template

Das Gang of Four Template ist ein Schema um Entwurfsmuster zu beschreiben. (vgl. [DP-EG-95] und <http://hillside.net/patterns/writing/GOFtemplate.htm> Datum: 14.02.07)

Das Template ist folgendermaßen aufgebaut:

1. Pattern Name (Name und Klassifikation des Musters)
2. Intent (Zweck)
3. Also Known As (Synonyme für das Muster)
4. Motivation (Hintergründe für den Einsatz des Musters)
5. Applicability (Anwendbarkeit)
6. Structure (Beschreibung der Struktur des Musters)
7. Participants (Klassen, die an dem Muster beteiligt sind)
8. Consequences (Konsequenzen; Vor und Nachteile)
9. Implementation
(Implementierung; Tipps zur Implementierung, Warnungen vor Fehlern)
10. Sample Code and Usage (Beispielcode)
11. Known Uses (Praxiseinsatz; Wo wird das Muster bereits eingesetzt?)
12. Related Patterns (Querverweise zu anderen Mustern)

3.2.5 Change Request

Da es bei den bereits beschriebenen Quellen keinerlei *Standard* für Änderungsanforderungen gab, musste hier auf eine weitere Quelle zurückgegriffen werden. Eine brauchbare Quelle stellt das Buch „Automating the Change Management Process with Electronic Contracts“ von A. Keller dar. Keller beschreibt den Aufbau von Change Requests (Änderungsanforderungen) folgendermaßen:

1. Type (Änderungstyp)
2. ID (eindeutige ID zum identifizieren)
3. Deadline(Termin)
4. Priority (Priorität der Änderung)
5. Customer (Kunde direkt/indirekt)
6. Abstract (Beschreibung der Änderung und Grund der Änderung)
7. Related documents (Relevante Dokumente)

(vgl. [CMP-AK-05])

3.2.6 ReadySET

Im folgenden werden zwei Muster aus dem Angebot des ReadySET vorgestellt. Wie bereits erwähnt handelt es sich bei ReadySET um eine Sammlung von Dokumentenvorlagen für die Softwareentwicklung.

3.2.6.1 ReadySET – Testcase

Der Aufbau eines Testcase wird nach ReadySET folgendermaßen beschrieben:

1. unique-test-case-id
2. Purpose (Zweck des Tests)
3. Precondition
(Welchen Zustand muss das System haben?)
4. Test Data (Liste mit Variablen und ihre Werte für diesen Testfall)
5. Steps (Beschreibung der einzelnen Schritte in diesem Testfall)
6. Notes and Questions (Notizen und Fragen)

(vgl. <http://readysset.tigris.org> Datum: 14.02.07)

3.2.6.2 ReadySET - Use Case

Der Aufbau eines Use Case sieht folgendermaßen aus:

1. Summary (Zusammenfassung)
2. Priority (Priorität)
3. Use Frequency (Wie häufig wird das Feature eingesetzt?)
4. Direct Actors (Wer nutzt es?)
5. Stakeholders
6. Precondition (Vorbedingungen)
7. Main Success Scenario (Beschreibung des erfolgreichen Ablaufs)
8. Alternative Scenario Extensions (Alternativen, abhängig von Zuständen)
9. Notes and Questions (Notizen und Fragen)

(vgl. <http://readysset.tigris.org> Datum: 14.02.07)

3.3 Spezifikation in XML

In diesem Kapitel werden nacheinander die beschriebenen Dokument-Vorlagen in ihrer XML-Repräsentation gezeigt. Die Darstellung entspricht bereits der Version, die man auch in das System einpflegen kann. Texte, die zwischen einzelnen Abschnitten oder Sektionen stehen, entsprechen den späteren default-Werten der entsprechenden Textboxen.

Ein Abschnitt mit folgendem Aufbau

```
<section name="Purpose">  
    Identify the product whose software requirements are specified in this  
    document, including the revision or release number. Describe the  
    scope of the product that is covered by this SRS, particularly if this  
    SRS describes only part of the system or a single subsystem.  
</section>
```

würde also folgendermaßen ausgewertet werden:

Der Name dieser Sektion (<section name="Purpose">) wird später zu einer Überschrift (Purpose). Der Text zwischen dem Anfangs-Tag <section> und End-Tag </section> wird später als default-Wert in einer Textbox unter der Überschrift auftauchen. Darauf wird im weiteren Verlauf der Arbeit näher eingegangen.

3.3.1 ANSI/IEEE Std 830-1993

```
<?xml version="1.0" encoding="UTF-8" ?>  
<template name="IEEE830" type="TOC">  
    <chapter name="Introduction">  
        <section name="Purpose">  
            Identify the product whose software requirements are specified  
            in this document, including the revision or release number.  
            Describe the scope of the product that is covered by this SRS,  
            particularly if this SRS describes only part of the system or a  
            single subsystem.  
        </section>  
        <section name="Document_Conventions">  
            Describe any standards or typographical conventions that were  
            followed when writing this SRS, such as fonts or highlighting  
            that have special significance. For example, state whether  
            priorities for higher-level requirements are assumed to be  
            inherited by detailed requirements, or whether every  
            requirement statement is to have its own priority.  
        </section>  
        <section name="Intended_Audience_and_Reading_Suggestions">  
            Describe the different types of reader that the document is  
            intended for, such as developers, project managers, marketing  
            staff, users, testers, and documentation writers. Describe what  
            the rest of this SRS contains and how it is organized. Suggest a
```

sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

</section>

<section name="Product_Scope">

Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.

</section>

<section name="References">

List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.

</section>

</chapter>

<chapter name="Overall_Description">

<section name="Product_Perspective" />

<section name="Product_Functions" />

<section name="User_Classes_and_Characteristics" />

<section name="Operating_Environment" />

<section name="Design_and_Implementation_Constraints" />

<section name="User_Documentation" />

<section name="Assumptions_and_Dependencies" />

</chapter>

<chapter name="External Interface Requirements">

<section name="User_Interfaces" />

<section name="Hardware_Interfaces" />

<section name="Software_Interfaces" />

<section name="Communications_Interfaces" />

</chapter>

<chapter name="System_Features">

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

</chapter>

<chapter name="Other_Nonfunctional_Requirements">

<section name="Performance_Requirements" />

<section name="Safety_Requirements" />

<section name="Security_Requirements" />

<section name="Software_Quality_Attributes" />

<section name="Business_Rules" />

</chapter>

```

<chapter name="Other_Requirements">
    Define any other requirements not covered elsewhere in the SRS.
    This might include database requirements, internationalization
    requirements, legal requirements, reuse objectives for the project, and
    so on. Add any new sections that are pertinent to the project.
</chapter>
</template>

```

3.3.2 Volere Requirement Specification Template

```

<?xml version="1.0" encoding="UTF-8" ?>
<template name="Volere_Requirements_Specification">
  <chapter name="Project_Drivers">
    <section name="Purpose_of_the_Project">
      Purpose of the Project
    </section>
    <section name="Client,_Customer_and_other_Stakeholders">
      The Client, the Customer, and Other Stakeholders
    </section>
    <section name="Users_of_the_Product">
      Users of the Product
    </section>
  </chapter>
  <chapter name="Project_Constraints">
    <section name="Mandated_Constraints">
      Mandated Constraints
    </section>
    <section name="Naming_Conventions_and_Definitions">
      Naming Conventions and Definitions
    </section>
    <section name="Relevant_Facts_and_Assumptions">
      Relevant Facts and Assumptions
    </section>
  </chapter>
  <chapter name="Functional_Requirements">
    <section name="Scope_of_the_Work">
      The Scope of the Work
    </section>
    <section name="Scope_of_the_Product">
      The Scope of the Product
    </section>
    <section name="Functional_and_Data_Requirements">
      Functional and Data Requirements
    </section>
  </chapter>
  <chapter name="Nonfunctional_Requirements">
    <section name="Look_and_Feel_Requirements">

```

```

        Look and Feel Requirements
    </section>
    <section name="Usability_and_Humanity_Requirements">
        Usability and Humanity Requirements
    </section>
    <section name="Performance_Requirements">
        Performance Requirements
    </section>
    <section name="Operational_and_Environmental_Requirements">
        Operational and Environmental Requirements
    </section>
    <section name="Maintainability_and_Support_Requirements">
        Maintainability and Support Requirements
    </section>
    <section name="Security_Requirements">
        Security Requirements
    </section>
    <section name="Cultural_and_Political_Requirements">
        Cultural and Political Requirements
    </section>
    <section name="Legal_Requirements">
        Security Requirements
    </section>
</chapter>
<chapter name="Project_Issues">
    <section name=" Open Issues">
        Issues that have been raised and do not yet have a
        conclusion.
    </section>
    <section name="Off-the-Shelf_Solutions">
        List of existing products that should be investigated as
        potential solutions. Reference any surveys that have been
        done on these products.
    </section>
    <section name="New_Problems">
        New Problems
    </section>
    <section name="Tasks">
        Tasks
    </section>
    <section name="Migration_to_the_New Product">
        Migration to the New Product
    </section>
    <section name="Risks">
        All projects involve risk—namely, the risk that something will go
        wrong.
    </section>
    <section name="Costs">
        Costs
    </section>
</chapter>

```

```

<section name="User_Documentation_and_Training">
    User Documentation and Training
</section>
<section name="Waiting_Room">
    Requirements that will not be part of the next release. These
    requirements might be included in future releases of the
    product
</section>
<section name="Ideas_for_Solutions">
    Ideas for Solutions
</section>
</chapter>
</template>

```

3.3.3 ANSI/IEEE Std 829 - Testfallspezifikation

```

<?xml version="1.0" encoding="UTF-8" ?>
<template name="IEEE_829_Test_Case_Specification" type="TestCase">
  <item name="Test Case Specification Identifier">
    Test Case Specification Identifier
  </item>
  <item name="Test Items">
    Describe features and conditions tested
  </item>
  <item name="Input Specifications">
    <subitem name="Data_Names">
      Data_Names
    </subitem>
    <subitem name="Ordering">
      Ordering
    </subitem>
    <subitem name="Values">
      with tolerances or generation procedures
    </subitem>
    <subitem name="States">
      States
    </subitem>
    <subitem name="Timing">
      Timing
    </subitem>
  </item>
  <item name="Output Specifications">
    <subitem name="Data_Names">
      Data Names
    </subitem>
    <subitem name="Ordering">
      Ordering
    </subitem>
  </item>
</template>

```

```
</item>
<item name="Environmental Needs">
  <subitem name="Hardware">
    Hardware
  </subitem>
  <subitem name="Software">
    Software
  </subitem>
  <subitem name="Other">
    Other
  </subitem>
</item>
<item name="Special_Procedural_Requirements" />
  Special Procedural Requirements
</item>
<item name="Inter-Case_Dependencies" />
  Inter-Case Dependencies
</item>
</template>
```

3.3.4 Architektur und Entwurfsmuster - Gang of Four Template

```
<?xml version="1.0" encoding="UTF-8" ?>
<template name="GOF">
  <section name="Pattern_name">
    Name of Pattern
  </section>
  <section name="Intent">
    Intent of this pattern.
  </section>
  <section name="Also_known_as">
    synonym of pattern name
  </section>
  <section name="Motivation">
    motivation to use this pattern
  </section>
  <section name="Applicability">
    What are the situations in which the design pattern can be applied?
    What are examples of poor designs that the pattern can address?
    How can you recognize these situations?
  </section>
  <section name="Structure">
    description of the pattern's structure
  </section>
  <section name="Participants">
    The classes and objects participating in the design pattern and their
    responsibilities.
  </section>
  <section name="Consequences">
    How does the pattern support its objectives? What are the trade-offs
    and results of using the pattern? What aspect of system structure
    does it let you independently?
  </section>
  <section name="Implementation">
    What pitfalls, hints, or techniques should you be aware of when
    implementing the pattern? Are there language-specific issues?
  </section>
  <section name="Sample_Code_and_Usage">
    Code fragments that illustrate how you might implement the pattern.
  </section>
  <section name="Known_Uses">
    Examples of the pattern found in real systems.
  </section>
  <section name="Related_Patterns">
    What design patterns are closely related to this one? What are the
    important differences? With which other patterns should this one be
    used?
  </section>
</template>
```

3.3.5 Change Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<template name="Change_Request">
  <section name="Type">
    Change Request Type
  </section>
  <section name="Change_Request_ID">
    unique ID to identify the Change request
  </section>
  <section name="Deadline">
    Deadline date
  </section>
  <radio name="Priority" options="Essential;Expected;Desired;Optional"
  default="Essential">
</radio>
  <section name="Customer">
    Customer
  </section>
  <section name="Abstract">
    description and cause for this change
  </section>
  <section name="Related_documents">
    Other documents relevant for this change request
  </section>
</template>
```

3.3.6 ReadySET - Use Case

```
<?xml version="1.0" encoding="UTF-8" ?>
<template name="Readysset_Usecase">
  <section name="Summary">
    Summary of this UseCase
  </section>
  <radio name="Priority" options="Essential;Expected;Desired;Optional"
  default="Essential">
</radio>
  <radio name="Use_Frequency"
  options="Always;Often;Sometimes;Rarely;Once " default="Always">
</radio>
  <section name="Direct_Actors">
    Direct Actors
  </section>
  <section name="Stakeholders">
    Stakeholders
  </section>
```

```

<section name="Precondition">
    Precondition
</section>
<section name="Main_Success_Scenario">
    description of successful process
</section>
<section name="Alternative_Scenario_Extensions">
    Alternative Scenario Extensions
</section>
<section name="Notes_and_Questions">
    Notes and Questions
</section>
</template>

```

3.3.7 ReadySET – Testcase

```

<?xml version="1.0" encoding="UTF-8" ?>
<template name="Readysset_Testcase">
    <section name="Unique_Testcase_Id">
        unique-test-case-id
    </section>
    <section name="Purpose">
        Short sentence or two about the aspect of the system is being tested.
        If this gets too long, break the test case up or put more information
        into the feature descriptions.
    </section>
    <section name="Precondition">
        Assumptions that must be met before the test case can be run. E.g.,
        "logged in", "guest login allowed", "user testuser exists".
    </section>
    <section name="Test_Data">
        List of variables and their possible values used in the test case. You
        can list specific values or describe value ranges. The test case should
        be performed once for each combination of values. These values are
        written in set notation, one per line.
    </section>
    <section name="Steps">
        Steps to carry out the test.
    </section>
    <section name="Notes_and_Questions">
        Notes and Questions
    </section>
</template>

```

4. Workflow

In diesem Kapitel wird genau beschrieben, wie ein Benutzer das System künftig nutzen kann. Nacheinander werden hierbei alle Möglichkeiten erläutert, die der Benutzer hat.

Ein Benutzer kann ein Template anlegen und es als normalen *Wikiartikel* speichern. Das Template wird im XML-Format geschrieben. Beim Anlegen bzw. beim Editieren eines anderen *Artikels* ist es möglich das Template auszuwählen. Nach Auswahl des Templates wird eine entsprechende Eingabemaske angezeigt, die Elemente wie Textboxen, Selectboxen oder Radiobuttons enthält. Die einzelnen Felder enthalten bereits default-Werte. Für Textfelder sind dies Erklärungen, was in die jeweiligen Felder eingetragen werden soll. Für Selectboxen bzw. Radiobuttons sind dies Werte, die man im XML bereits als default definiert hat. Die Ausgabe der Formulare kann mittels CSS angepasst werden. Diese CSS-Eigenschaften werden ebenfalls in einem separaten *Artikel* gespeichert. Während ein *Artikel* bearbeitet wird, ist es jederzeit möglich eine Vorschau anzusehen, die zeigt wie der *Artikel* nach dem Speichern aussieht. Wenn der Benutzer einen *Artikel* speichert, der ein Template verwendet, so wird dieser *Artikel* automatisch einer entsprechenden Kategorie, die den Namen des verwendeten Templates trägt, zugeordnet.

Möchte der Benutzer einen *Artikel* editieren, der mit Hilfe eines Templates erstellt wurde, so wird zunächst die Eingabemaske geladen, die auch beim Anlegen verwendet wurde. Das heißt, es wird die selbe Version wie beim letzten Editieren genutzt. Die Eingabefelder werden mit den gespeicherten Daten gefüllt. Es werden also nicht die default-Werte verwendet, wie es beim Anlegen der Fall war. Der Benutzer hat jedoch die Möglichkeit, das Template oder die Version des Templates zu wechseln. Bei einem Wechsel bleiben die Daten erhalten. Zum einen wird versucht, anhand der Überschriften bereits passende Felder für die Daten zu finden und zum anderen werden die kompletten, gespeicherten Daten in ein zusätzliches Feld geschrieben, das als temporärer Speicher fungiert. Der Benutzer kann dann die Daten aus diesem Feld in die entsprechenden neuen Felder kopieren.

Neben der Möglichkeit das Template zu wechseln, ist es auch möglich, die Eingabemaske abzuschalten. Das heißt, dass der Benutzer dann alle Daten in normalem Wikimarkup in einer Textbox hat und auf normale Art und Weise editieren kann. Außerdem ist es möglich, nur einen Teil eines *Artikels* – also einen Abschnitt - zu bearbeiten. Für das Layout der Ausgabemasken wird ebenfalls CSS verwendet. Für jedes Template kann man einen entsprechenden *Artikel* anlegen, der zum einen die CSS-Eigenschaften für die Eingabe und zum anderen die CSS-Eigenschaften für die Ausgabe beinhaltet. Ein solcher *Artikel* kann mehrere Definitionen enthalten. Dies hängt davon ab, ob es mehrere Versionen eines Templates gibt. Jede dieser Version sollte eine eigene CSS-Spezifikation erhalten.

Neue Versionen eines Templates entstehen dann, wenn der Benutzer ein bestehendes Template bearbeitet, z.B. wenn er einen neuen Abschnitt hinzufügt, einen Abschnitt entfernt oder eine Überschrift eines Abschnitts ändert.

5. Realisierung

In diesem Kapitel wird beschrieben, wie die Anforderungen erfüllt wurden. Allerdings werden hier nicht alle Details der Implementierung erläutert. Es wird lediglich dargestellt, an welchen Stellen in das MediaWiki bzw. in die derzeitige Version des SOP eingegriffen wurde. Außerdem wird ein grober Aufbau des Systems gegeben. Hierbei wird allerdings nur der Teil näher erläutert, der für die Erweiterung von Bedeutung ist. Im Unterkapitel Test wird beschrieben, wie das System getestet wurde. Es enthält ebenfalls kein vollständiges Testdokument.

Im Unterkapitel Anforderungen werden noch einmal die Anforderungen, die auch schon im Kapitel Workflow und im Kapitel Aufgabenstellung beschrieben wurden feiner untergliedert und beschrieben.

5.1 Anforderungen

5.1.1 Verwendung von Templates zur Erstellung von Eingabemasken

Der Benutzer kann Templates, die in XML definiert sind, als *Wikiartikel* speichern. Wenn der Benutzer einen neuen *Artikel* anlegt, kann er aus einer Selectbox das Template auswählen, was dazu führt, dass eine Eingabemaske nachgeladen wird. Die einzelnen Elemente enthalten bereits Werte. Bei Textboxen sind dies Erklärungen, was jeweils eingetragen werden soll. Bei anderen Elementen (z.B. Radiobox) sind dies default-Werte, die im XML zu finden sind.

Diese Anforderung lässt sich noch feiner untergliedern. Um ein Template zu speichern, reichen die vorhandenen Funktionen des *MediaWiki* bereits aus. Ein Template kann wie ein normaler *Artikel* erstellt werden, dessen Inhalt dann aus XML, wie man es im Kapitel Templates findet, besteht. Um den *Artikel* als Template zu kennzeichnen kann man ihn der Kategorie Template zuordnen. Auch diese Funktionalität bietet das *MediaWiki* bereits.

Die erste neue Anforderung ergibt sich daraus, dass dieser XML-Inhalt analysiert werden muss. Das System muss das XML aus der Datenbank, genauer gesagt aus einem *Artikel* laden, die einzelnen Elemente erkennen und auswerten können. Hierbei ist zu beachten, dass einzelne Sektionen Untersektionen enthalten können. Auch diese können wiederum weitere Untersektionen enthalten.

Die Analyse des XML dient dazu die Eingabemasken zu erstellen. Das bedeutet, dass das System den einzelnen Elementen eines XML-Konstrukts typische HTML-Formularelemente zuzuordnen muss. Sektionsnamen werden dabei zu Überschriften, die allerdings auch der Struktur entsprechen müssen. Das heißt, dass Kapitel Unterkapitel enthalten können – eben entsprechend den Untersektionen im XML. Für die Abgrenzung dieser Kapitel müssen verschiedenen Überschriftgrößen verwendet werden. Bestimmte Tags im XML sollen dazu dienen, Elemente wie Radiobuttons oder Selectboxen zu kennzeichnen. Trifft das System bei der Analyse des XML auf einen derartigen Tag müssen die entsprechenden HTML-Eingabeelemente generiert werden.

5.1.2 Speichern eines Artikels, erstellt mit einem Template

Der Text, der gespeichert wird, enthält normales Wikimarkup. Es sollen keine zusätzlichen Informationen über die Elemente der Eingabemaske abgespeichert werden. Wenn man den gespeicherten Text in der Datenbank ansieht, soll es sich um einen Textblock handeln, der normales Wikimarkup enthält. Lediglich zu welcher Kategorie von Templates der *Artikel* gehört wird zusätzlich in den Textblock geschrieben. Die Form entspricht aber auch hier der gewöhnlichen Wikifunktionalität. Wichtig wird dies vor allem dann, wenn sich der Benutzer entscheidet einen *Artikel* zu bearbeiten, ohne die Eingabemaske zu verwenden. Die Information, welches Template und welche Version verwendet wurde, muss zusätzlich an anderer Stelle gespeichert werden. Dies ist notwendig um bei späterem Bearbeiten eines *Artikels* die richtige Eingabemaske zu laden. Außerdem ist diese Information auch wichtig, um die Ausgabe entsprechend darzustellen. Darauf wird auch noch im Kapitel 5.1.6 eingegangen.

5.1.3 Editieren von bereits bestehenden Artikeln

Wird ein bereits gespeicherter *Artikel* erneut editiert, so soll der Benutzer verschiedene Möglichkeiten haben, die im folgenden erläutert werden.

5.1.3.1 Bearbeiten mit altem Template

Wenn der Benutzer einen *Artikel* erneut editiert, wird zunächst die Eingabemaske geladen, die er auch beim Anlegen des *Artikels* verwendet hat. Die Felder sind dann mit den bereits gespeicherten Werten, die der *Wikiartikel* enthält, gefüllt. Er kann seine Änderungen durchführen und den *Artikel* speichern.

5.1.3.2 Bearbeiten mit anderer Version des verwendeten Templates

Der Benutzer kann aus einer Selectbox, in der alle Versionen des verwendeten Templates aufgeführt sind, ein anderes wählen. Dies führt dazu, dass die Eingabemaske geladen wird, die zu dieser Version des Templates gehört. Die gespeicherten Daten werden entsprechend den Überschriften zugeordnet. Die gesamten Daten werden zusätzlich in ein Bemerkungs-Feld kopiert. Dies ist wichtig, falls in der neu gewählten Version des Templates bestimmte Abschnitte umbenannt oder weg gefallen sind. Die Daten dürfen nicht durch einen Wechsel der Version verloren gehen. Das zusätzliche Feld dient aber lediglich als temporärer Speicher und wird beim Speichern des *Artikels* nicht gesichert. Der Benutzer muss die alten Inhalte also zunächst in die Abschnitte des neu verwendeten Templates kopieren. Falls gegenüber der vorher verwendeten Version des Templates Felder hinzugekommen sind, so werden diese mit den default-Werten gefüllt, die aus dem XML kommen. Dieser Text wird zusätzlich farblich markiert. Dadurch wird es für den Benutzer einfacher zu erkennen, welche Felder noch ausgefüllt werden müssen.

5.1.3.3 Bearbeiten eines Artikels mit anderem Template

In einer zusätzlichen Selectbox werden alle vorliegenden Templates angezeigt. Wählt der Benutzer eines dieser Templates, so wird die entsprechende Eingabemaske geladen. Mit den gespeicherten Informationen wird genauso verfahren, wie bei einem Versionswechsel (siehe 5.1.3.2). Die Version des Templates ist hierbei immer die aktuellste, vorliegende Version.

5.1.3.4 Bearbeiten eines Artikels ohne Eingabemaske

Wie bereits erklärt, wird beim Editieren eines *Artikels* zunächst die Eingabemaske geladen, die auch beim Anlegen verwendet wurde. Jedoch hat der Benutzer die Möglichkeit diese Eingabemaske abzuschalten. Dafür gibt es eine dritte Selectbox mit der man das Formular an- oder ausschalten kann. Schaltet der Benutzer das Formular ab, so erhält er eine Textbox, in die der gesamte Inhalt des *Artikels* geschrieben wird. Der Text entspricht hier der normalen Wikimarkup-Form.

5.1.3.5 Bearbeiten eines Abschnitts eines Artikels

Wenn man sich einen *Wikiartikel* ansieht, so findet man hinter jedem Abschnitt einen Bearbeitungs-Link, der mit „bearbeiten“ gekennzeichnet ist. Der Benutzer hat die Möglichkeit durch einen Klick auf einen dieser Links eine Sektion zu bearbeiten. Dies bedeutet, dass das entsprechende Eingabeelement (Textbox, Selectbox usw.) geladen wird und der gespeicherte Inhalt angezeigt wird. Der Benutzer hat die Möglichkeit, gezielt nur diesen Bereich zu bearbeiten, ohne dass er den Rest des Formulars angezeigt bekommt.

5.1.4 Vorschaufunktion beim Editieren

Beim Anlegen eines *Artikels*, beim Überarbeiten eines *Artikels* oder eines bestimmten Abschnitts eines *Artikels*, hat der Benutzer die Möglichkeit eine Vorschau anzusehen. Unter der Vorschau wird erneut die entsprechende Eingabemaske angezeigt, gefüllt mit den zuletzt eingegebenen Daten.

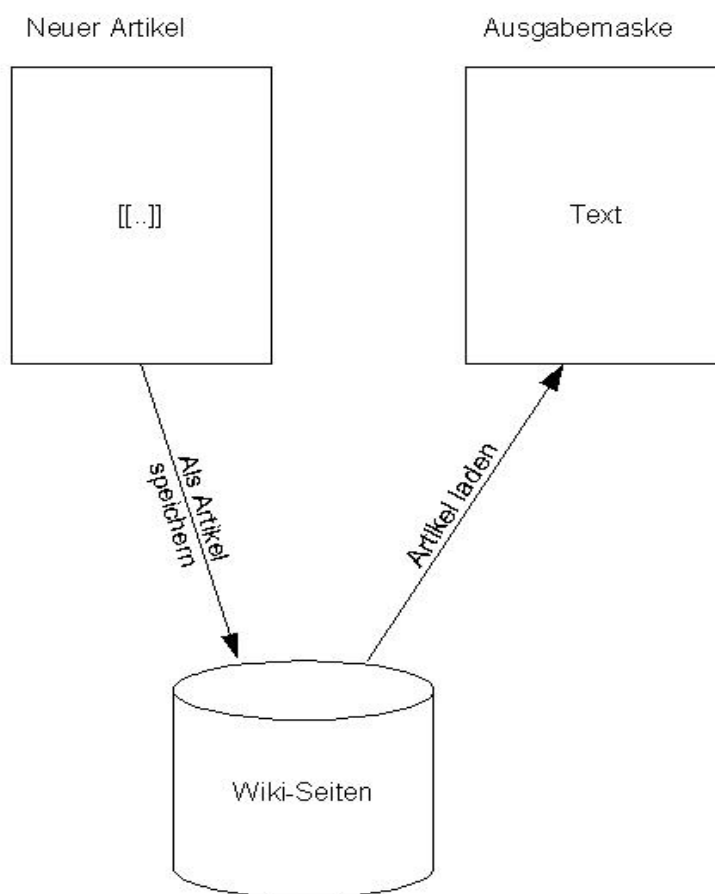
5.1.5 Layout der Formulare unter Verwendung von CSS

Solange es zu einem Template kein CSS gibt, werden die Elemente eines Formulars einfach untereinander in linearer Form aufgeführt. Das bedeutet ein Element folgt auf das nächste. Um die Position von Elementen zu ändern, hat der Benutzer die Möglichkeit CSS-Artikel abzuspeichern. Zu einer Version eines Templates gibt es eine passende Definition der Ausgabemaske. Damit hat der Benutzer verschiedene Möglichkeiten. Er kann Positionen ändern und so zum Beispiel zwei Abschnitte nebeneinander setzen. Oder er kann die Größe der Elemente festlegen. Natürlich ist es nicht notwendig, dass alle Textboxen die gleiche Größe haben. Ist zum Beispiel klar, dass ein Abschnitt genau zwei Zeilen enthalten wird, so kann man dies über die Größeneigenschaften, die man im CSS hinterlegt, regeln.

5.1.6 Layout der Ausgabemasken

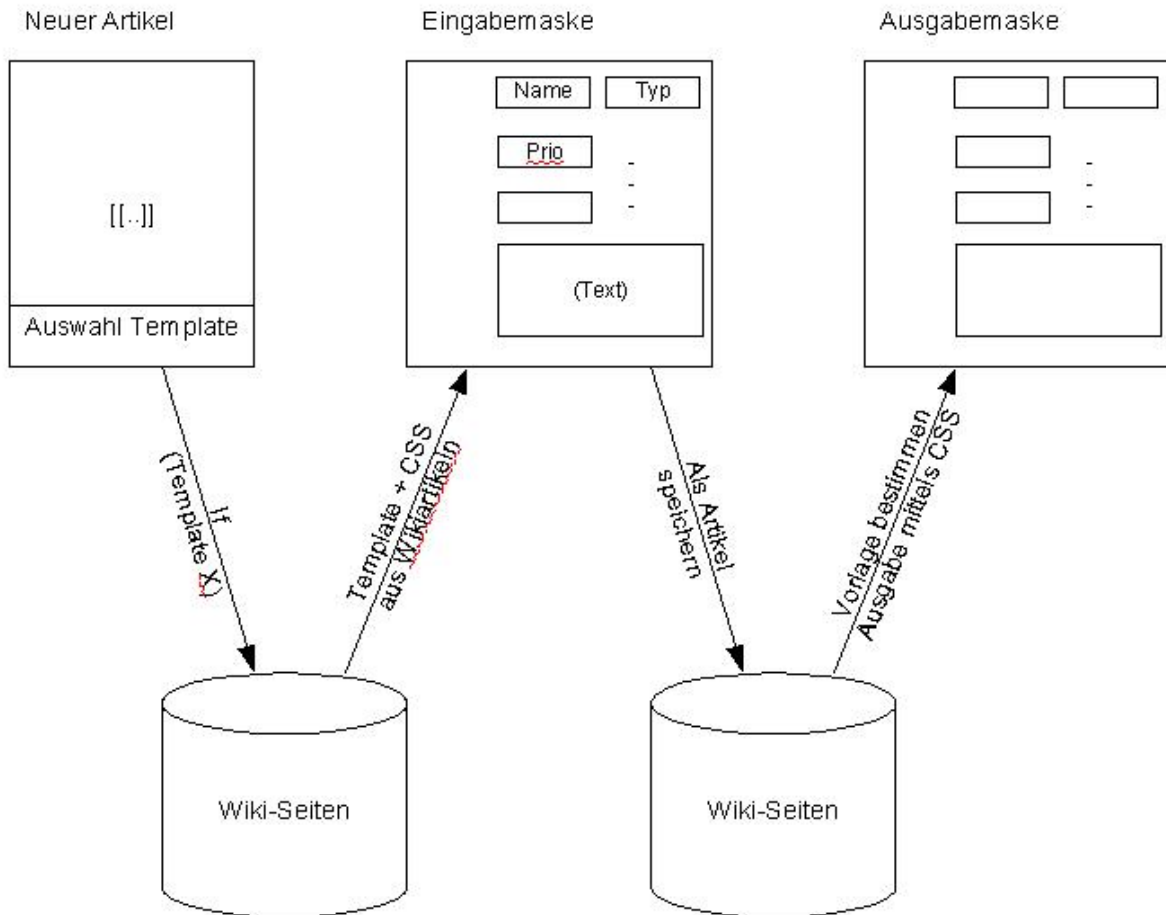
Entsprechend der Eingabemasken hat der Benutzer auch die Möglichkeit die Positionen seiner Abschnitte bei der Ausgabe, also bei der Anzeige eines *Artikels* zu beeinflussen. Genauso wie CSS-Eigenschaften für eine Version eines Templates abgelegt werden können, können auch Informationen für die Ausgabe abgelegt werden. So kann man die Ausgabe entsprechend der Eingabe anpassen.

Die folgende Grafik zeigt die Funktionsweise des *MediaWikis* vor der Erweiterung:



Im ursprünglichen MediaWiki können Artikel mittels *Wikimarkup* erstellt werden. Der Text kann in eine Textbox eingetragen werden. Danach wird der Inhalt als Wikiartikel in der Datenbank gespeichert. Bei der Ausgabe wird der Artikel aus der Datenbank geladen und angezeigt. CSS wird hierbei nicht verwendet. Die Inhalte werden einfach untereinander angezeigt. Das Wikimarkup aus der Datenbank wird bei der Ausgabe in HTML umgewandelt, um Überschriften, Listen oder normalen Text zu erzeugen.

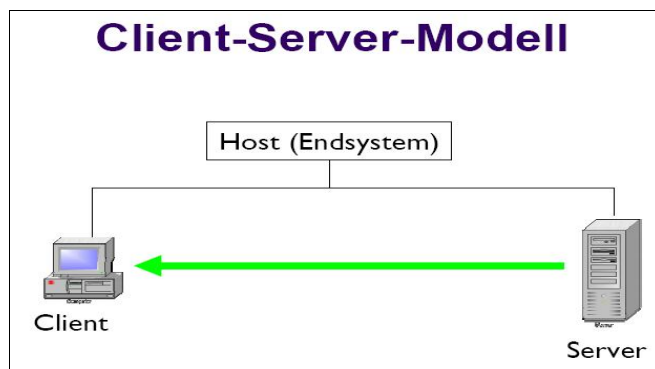
Die folgende Grafik zeigt die Funktionsweise des *Mediawiki* nach der Erweiterung:



Wird ein neuer Artikel bearbeitet, kann ein Template gewählt werden. Das Template und eine passende CSS-Spezifikation sind in der Datenbank als normale Wikiartikel gespeichert. Die Auswahl des Templates führt zum Laden der entsprechenden Eingabemaske. Diese wird mittels CSS gelayoutet. Es wird eine strukturierte Eingabe ermöglicht. Nach dem Editieren wird der Text als normaler Wikiartikel in der Datenbank gespeichert, dessen Inhalt aus Wikimarkup besteht. Beim Anzeigen eines Artikels wird erkannt, welche Vorlage verwendet wurde und das entsprechende CSS für die Ausgabe geladen.

5.2 Architektur

Die gesamte Anwendung ist konzipiert nach dem Client-Server Prinzip. Der Server, auf dem die Anwendung läuft und auf dem die Datenbank installiert ist, wird durch den Client „angesprochen“ und bekommt die fertigen Informationen zurück. Als Benutzerschnittstelle auf Client-Seite dient ein Internbrowser.



Quelle Abbildung: [SYS-MF-05]

5.3 Implementierung

Durch die Anforderungen war es notwendig an zwei verschiedenen Bereichen, in das bestehende System einzugreifen. Zum einen beim Erstellen und Editieren eines *Artikels* und zum anderen beim Anzeigen eines *Artikels*. Die Klasse *EditPage.php* kommt beim Editieren eines *Artikels* zum Einsatz. Daher war diese Klasse der erste Ansatzpunkt, um mit der Programmierung einzusteigen. Da es im MediWiki normalerweise keine besondere Funktionalität für XML-Inhalte gibt, war hierfür eine zusätzliche Klasse (*TemplateParser.php*) notwendig. Diese Klasse wird im folgenden noch etwas genauer erläutert. Um die *EditPage.php* nicht zu sehr ändern zu müssen, erfolgen in dieser Klasse lediglich Aufrufe zusätzlicher Funktionen/Erweiterungen, die in der Klasse *IESEHooks.php* gespeichert sind. Auf die Funktionsweise von Hooks und somit der Möglichkeit Erweiterungen im Allgemeinen zu schreiben, wird in dieser Arbeit nicht eingegangen. Allerdings findet man unter

http://meta.wikimedia.org/wiki/Write_your_own_MediaWiki_extension

eine gute Beschreibung rund um dieses Thema. Da im Laufe der Zeit schon andere Erweiterungen am Fraunhofer Institut entwickelt worden sind, existierte diese Klasse bereits.

Für die Ausgabe der *Artikel* musste an einer weiteren Stelle eingegriffen werden. Die Änderungen betrafen zum einen die Klasse *Parser.php* und zum anderen die Klasse *Linker.php*. Beide Klassen arbeiten zusammen, um die Ausgabe der *Artikel* zu steuern.

Es wurde versucht, so wenig wie möglich an den Basisklassen des MediaWikis zu ändern. Sollte eine neue Version des MediaWiki erscheinen, so können die Änderungen einfacher eingebettet werden.

5.3.1 TemplateParser.php

Diese Klasse übernimmt das Analysieren und Erstellen der Eingabemasken und daher ist es sinnvoll, etwas genauer auf diese Klasse einzugehen. Es gibt bereits fertige *XML-Parser* für php. Die zwei bekanntesten Parser sind DOM und SAX. Da in einem vorherigen Projekt bereits die Entscheidung für den DOM-Parser gefallen war, wurde dieser auch weiterhin verwendet. Bis zur Version 4.0 von PHP waren die Funktionen des DOM-Parsers bereits integriert, danach wurden sie allerdings gestrichen. Um diese Funktionalitäten jedoch nutzen zu können, wurde die zusätzliche Klasse `domxml-php4-to-php5.php` hinzugefügt und eingebunden. Der Parser wird genutzt um die Inhalte eines XML-Dokuments verfügbar zu machen. Es gibt verschiedene Methoden, mit denen man die einzelnen Elemente bekommen oder deren Typ bestimmen kann. Diese Funktionalität wurde in der Klasse `TemplateParser.php` verwendet. Je nach Typ werden entsprechende Eingabeelemente erzeugt oder Werte ausgelesen um sie wiederzugeben. So zum Beispiel beim Bestimmen der default-Werte für Text- und Selectboxen. Eine Auflistung der Funktionen kann man dem zusätzliche Dokument „Design und Implementierung“ entnehmen.

5.3.2 IESEHooks.php

Diese Klasse stellt das Bindeglied zwischen der Klasse `EditPage.php` und `TemplateParser.php` dar. In der Klasse `EditPage.php` werden immer nur Funktionen der Klasse `IESEHooks.php` aufgerufen. Sie enthält einige wichtige Funktionen, wie z.B. das Erkennen, ob ein *Artikel* mit einem Template erstellt wurde oder eine Funktion zum Speichern eines *Artikels*. Außerdem ruft sie die benötigten Funktionen aus der Klasse `TemplateParser.php` auf.

5.4 Test

Alle neu geschriebenen Funktionen sind einzeln überprüft worden. Für diesen Zweck wurde eine zusätzliche Funktion geschrieben, die die Ergebnisse einer Funktion in eine Textdatei schreibt. Die daraus resultierenden Werte wurden dann mit den Soll-Werten verglichen. Auf diese Weise war es nicht notwendig störende Ausgaben auf dem Bildschirm auszugeben und es war auch möglich, die Ausgaben in die Textdatei im Programmcode stehen zu lassen, ohne den Ablauf der Applikation zu beeinflussen. Für die Überprüfung wurden zunächst Eingabewerte festgelegt und bestimmt, welches Ergebnis eine Funktion liefern muss. Mit Hilfe der Ausgabe in die Datei wurde dann die Überprüfung vorgenommen. Die Eingabewerte wurden nicht willkürlich gewählt, sondern zusätzlich zu Standardwerten wurden Werte gewählt, die zu Problemen führen könnten. Es wurden also *White-Box-Tests* verwendet.

Natürlich sind nicht alle Funktionen auf diese Art und Weise testbar. Viele Anforderungen betreffen schließlich die Optik der Applikation, so dass man hier nur testen kann, indem man die Applikation benutzt und sich das Ergebnis auf dem Bildschirm ansieht. Getestet wurde auch gegen die Anforderungen, d.h. es wurde überprüft ob alle Anforderungen aus der Aufgabenstellung erfüllt wurden. Getestet wurde zum einen während der Implementierung und zum anderen nach der Fertigstellung der Software. Für die Tests wurden die beiden Browser Firefox und Internet Explorer verwendet.

6. Validierung

Schon während der Implementierung wurden die Ergebnisse regelmäßig mit dem Betreuer am Fraunhofer Institut IESE besprochen um sicher zu gehen, dass alle Anforderungen richtig verstanden wurden.

Die Software, inklusive der Erweiterungen, wurden nach der Fertigstellung an das Fraunhofer Institut IESE übergeben. Dort wurde die Applikation auf einem Testserver installiert und getestet. Auf diesen Testserver haben alle Mitarbeiter des Fraunhofer Institut IESE Zugriff.

Wenn die Erweiterung auf dem Testserver ausreichend getestet wurde, wird sie in die aktuelle Version des SOP integriert.

7. Zusammenfassung der Ergebnisse

Die Anforderungen an das System konnten alle erfüllt werden. Eine Reihe wichtiger Vorlagen wurde im Laufe der Arbeit recherchiert, in XML spezifiziert und in das System eingepflegt.

Die Erweiterung des MediaWiki hat dazu geführt, dass die Vorteile einer formularbasierten und somit strukturierten Eingabe mit den Vorteilen, die ein Wiki bietet, kombiniert wurden.

7.1 Einblicke in die Applikation

Die folgende Abbildung zeigt einen Ausschnitt, wie Eingabemasken mit der neuen Erweiterung aussehen. Der Ausschnitt stammt aus der Eingabemaske, die generiert wird, wenn man die Vorlage „ReadySET UseCase“ verwendet. Die Vorlage enthält neben Textboxen auch Radiobuttons, die es dem Benutzer ermöglichen aus einer bestimmten Menge von Antwortmöglichkeiten eine Auswahl zu treffen. Mittels CSS wurde die Textbox in ihrer Höhe und Breite verändert.

The screenshot shows a web form titled "READYSET UseCase (REVISIONNUMBER: 6678)". It features a "Summary" section with a rich text editor toolbar and a text area containing the text "Summary of this UseCase". Below this are two sections with radio button options: "Priority" with options "Essential", "Expected", "Desired", and "Optional"; and "Use Frequency" with options "Always", "Often", "Sometimes", "Rarely", and "Once".

Abbildung: Eingabemaskenausschnitt

Die Auswahl eines Templates erfolgt über Selectboxen. In der folgenden Abbildung sieht man alle drei Möglichkeiten, die der Benutzer hat:

1. Wahl eines anderen Templates
2. Wahl einer anderen Version
3. Abschalten der Eingabemaske

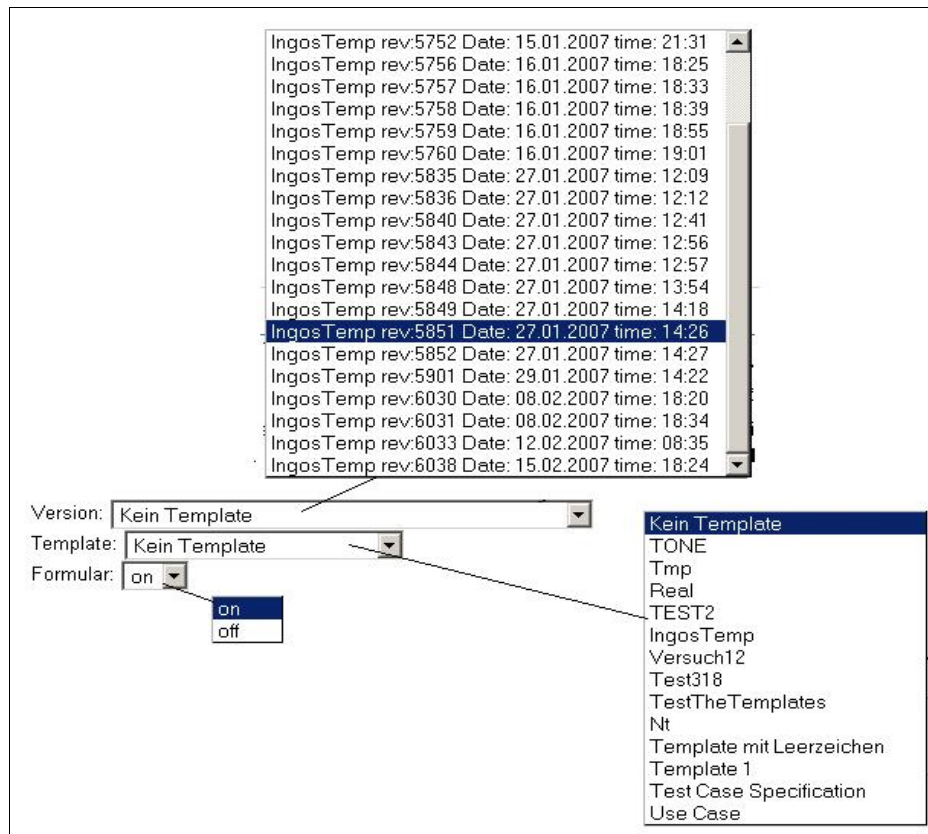


Abbildung:Auswahl Template, Version oder Formular ausschalten

7.2 Verwendung der Software

In diesem Kapitel wird erklärt, wie ein Benutzer die Erweiterung nutzen kann.

7.2.1 Vorlage erstellen

Wie bereits erläutert, können Vorlagen wie normale Artikel erstellt werden. Der Inhalt des Artikels besteht aus der XML-Spezifikation eines Templates. Ganz am Ende eines Artikels muss der Benutzer noch folgende Zeile hinzufügen:

```
[[Category:Template]]
```

Dies ist nötig, um ein Template der Kategorie Template zuzuordnen. Dies dient dazu, dass das System dieses Template künftig als Vorlage erkennt und zur Verfügung stellt.

7.2.1.1 Von der Vorlage zur XML-Spezifikation

Am folgenden Beispiel wird erklärt, wie man von einer Vorlage zu einer passenden XML-Spezifikation gelangt

Vorlagenbeschreibung:

1. Summary (Zusammenfassung)
2. Priority (Priorität)
3. Use Frequency (Wie häufig wird das Feature eingesetzt?)
4. Direct Actors (Wer nutzt es)
5. Stakeholders
6. Precondition (Vorbedingungen)
7. Main Success Scenario (Beschreibung des erfolgreichen Ablaufs)
8. Alternative Scenario Extensions (Alternativen, abhängig von Zuständen)
9. Notes and Questions (Notizen und Fragen)

Die Vorlage enthält neun verschiedenen Abschnitte. Priority und Use Frequency sind besondere Abschnitte. Hier sollte aus einer vorgegebenen Menge von Werten ausgewählt werden können. Alle anderen Abschnitte sollen mit Text beschrieben werden.

Um einen Abschnitt zu spezifizieren benötigt man einen Anfangs- und einen passenden End-Tag. Zwischen die beiden Tags schreibt man den default-Wert, der in der Eingabemaske in der entsprechenden Textbox angezeigt werden soll. Dieser default-Wert ist außerdem notwendig, damit erkannt wird, an welcher Stelle Textboxen erstellt werden sollen. Steht hier kein default-Wert für die Textbox, so wird lediglich eine Überschrift erstellt.

Beispiel:

```
<section name="Summary">
    Summary of this UseCase
</section>
```

Um eine Gruppe mit Radiobuttons zu erzeugen, nutzt man ein spezielles radio-Tag. Auch hier muss man das Ende mit einem entsprechenden End-Tag kennzeichnen.

Beispiel:

```
<radio name="Priority" options="Essential;Expected;Desired;Optional"
default="Essential">
</radio>
```

Mit `options="Essential;Expected;Desired;Optional"` werden die vier Auswahlmöglichkeiten festgelegt, die der Benutzer hat und durch `default="Essential"` wird der default-Wert für die Ausgabemaske gesetzt. Für Selectboxen kann man analog vorgehen. Man verwendet allerdings anstelle des radio-Tags ein select-Tag.

Alle XML-Spezifikationen in Kapitel 3.3 sind nach diesem Schema erstellt worden.

7.2.2 Vorlage nutzen

Wenn der Benutzer einen neuen Artikel bearbeitet, kann er aus einer Selectbox ein Template wählen. Danach wird die entsprechende Eingabemaske geladen. Bearbeitet er einen bereits gespeicherten Artikel, stehen ihm die drei Selectboxen zur Verfügung, die bereits in Kapitel 7.1 beschrieben sind.

7.2.3 CSS-Spezifikation

Für jede Version eines Templates sollten zwei Spezifikationen gespeichert werden, eine für die Ausgabe und eine für die Eingabe. Alle Spezifikationen werden in einen Artikel geschrieben. Der Artikel trägt den Namen des Templates plus den Zusatz CSS.

Beispiel: Template: Readyset UseCase
 CSS-Artikel: Readyset UseCaseCSS

Zunächst wird im Artikel festgelegt, welche Version eines Templates spezifiziert wird und ob es sich um die Spezifikation für die Ein- oder Ausgabe handelt. Dazu erstellt man eine Überschrift in *Wikimarkup*, die folgendermaßen aussehen muss:

Ausgabemaske:

= Versionsnummer Formular =

Eingabemaske:

= Versionsnummer Output =

Beispiel:

= 6651 Formular =

Die Versionsnummer eines Templates erhält man, indem man einen Artikel bearbeitet und ein Template auswählt. Ganz am Anfang des Artikels steht das verwendete Template und die Versionsnummer.

Danach folgt die eigentliche CSS-Spezifikation.

Für die Eingabemasken gelten folgende Regeln:

1. Die Überschriften sind jeweils einer Klasse zugeordnet, deren Namen sich aus dem Titel des Abschnitts und dem Zusatz *_head* zusammensetzt. Leerzeichen müssen allerdings bei der Spezifikation durch einen Unterstrich ersetzt werden.
2. EditToolbars, die vor jeder Textbox zu finden sind, sind einer Klasse zugeordnet, deren Namen sich aus der darüber liegenden Überschrift plus dem Zusatz *Toolbar* zusammensetzt..
3. Textboxen werden einer Klasse zugeordnet, die jeweils den Namen der Überschrift trägt. Leerzeichen werden durch einen Unterstrich ersetzt.

4. Elemente wie Radiobuttons oder Selectboxen sind einer Klasse zugeordnet, deren Namen sich aus der Überschrift des Abschnitts und den Präfix Elementname_ zusammensetzt. Elementname steht hier z.B für radio oder select.
5. Die Spezifikation in CSS wird umschlossen durch style-Tags.
<style type="text/css"> ... </style>
6. Positionsangaben müssen immer relativ sein.

Beispiel:

```
<style type="text/css">
  .Summary_head{position:relative; margin-left:120px; margin-right:350px;}
  .SummaryToolbar{position:relative; margin-left:120px; margin-right:350px;}
  .Summary{position:relative; margin-left:120px; margin-right:350px;}
  .Priority{position:relative; margin-left:120px; margin-right:350px;}
  .radio_Priority{position:relative; margin-left:120px; margin-right:350px;}
</style>
```

Für die Ausgabemasken gelten folgende Regeln:

1. Ein kompletter Abschnitt wird einer Klasse zugeordnet, deren Namen sich jeweils aus der Überschrift des Abschnitts und dem Zusatz View zusammensetzt.
2. Die Spezifikation in CSS wird umschlossen durch style-Tags.
<style type="text/css"> ... </style>
3. Positionsangaben müssen immer relativ sein.

Beispiel:

```
<style type="text/css">
  .PriorityView{position:relative; margin-left:120px;}
  .SummaryView{position:relative; margin-left:120px;}
</style>
```

Durch die CSS-Spezifikation können Positionen und Größen von Elementen beeinflusst werden

8. Ausblick

Eine weitere Aufgabe könnte sein, einen XML-Editor in das SOP zu integrieren, der zum einen das Erstellen der XML-Artikel einfacher macht und zum anderen eine Validierung des XML übernimmt. Dies könnten z.B. Überprüfungen sein, ob es zu einem Start-Tag auch immer ein passendes End-Tag gibt. Komfortabel wäre es, wenn dieser Editor eine Vorschaufunktion anbieten würde, die zeigt, wie die generierte Eingabemaske am Ende aussehen würde. Hierfür könnte man auf die neuen Funktionen, die im Laufe dieser Arbeit entstanden sind, zurückgreifen.

Auch für das CSS könnte man sich einen entsprechenden Editor vorstellen. Am besten wäre es, wenn man direkt nach dem Erstellen eines Templates noch die Möglichkeit geboten bekommt, einen passenden CSS-Artikel anzulegen bzw. bei einer Änderung eines Templates sollte der entsprechende CSS-Artikel geladen werden, damit man ihn editieren kann. Die benötigten Informationen zum Template könnten dann direkt übernommen werden.

9. Literaturverzeichnis

Die folgende Liste enthält alle Bücher, Papers oder Arbeiten anderer Personen, auf die im Laufe der Arbeit Bezug genommen wurde.

<u>Kürzel:</u>	<u>Buchbeschreibung:</u>
[WWC-AE-05]	Wiki Web Collaboration; Anja Ebersbach, Markus Glaser, Richard Heigl; 2005
[IEEESC-94]	IEEE Standards Collection Software Engineering; 1994
[PHP5-MZ-06]	Jetzt lerne ich PHP 5; Matt Zandstra; 2006
[XML-ETR-04]	Einführung in XML (deutsche Übersetzung); Erik T. Ray; 2004
[VCS-BCS-06]	Version Control with Subversion; Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato; 2006
[PEAA-MF-03]	Patterns für Enterprise Application-Architekturen (deutsche Übersetzung); Martin Fowler; 2003
[REUM-CR-06]	Requirements-Engineering und Management (deutsche Übersetzung); Chris Rupp; 2006
[SE-IS-01]	Software Engineering; Ian Sommerville; 2001
[VRST-JSR-06]	Volere Requirements Specification Template; James & Suzanne Roberts; 2006
[STD-JT-02]	Standard for Software Test Documentation (Paper); Juha Taina; 2002
[DP-EG-95]	Design Patterns - Elements of Reusable Object-Oriented Software; Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; 1995
[CMP-AK-05]	Automating the Change Management Process with Electronic Contracts ; A. Keller; 2005
[SYS-MF-05]	Einführung in Computernetzwerke (Folien zur Vorlesung „Systemsoftware“ WS04/05); Miriam Föllner; 2005

10. Glossar

Artikel

Ein Artikel (Wikiartikel) entspricht einer Seite in einem *Wiki*. Artikel können durch den Benutzer angelegt und/oder editiert werden. Jeder Artikel wird automatisch versioniert.

Auszeichnungssprache

Eine Markup-Sprache ist eine Menge von Symbolen, die im Text des Dokuments platziert werden können, um die einzelnen Teile dieses Dokuments zu benennen und sie voneinander abzugrenzen. [XML-ETR-04]

Eine Auszeichnungssprache (engl. Markup Language, Abk. ML) dient zur Beschreibung der Daten und teilweise des Verfahrens, das zur Bearbeitung dieser Daten nötig ist. Ursprünglich dienten die Auszeichnungen im Text als Anweisungen für die Setzer aus der Druckindustrie. Bei einer Auszeichnungssprache werden die Eigenschaften, Zugehörigkeiten und Verfahren von bestimmten Wörtern, Sätzen und Abschnitten eines Textes beschrieben bzw. zugeteilt, meist indem sie mit Tags markiert werden. [..]Auch das Wikipedia-Projekt hat eine eigene Auszeichnungssprache (Wikitext). Diese Formate erleichtern die Erstellung von Formatierungen oder auch Tabellen, für die ansonsten Kenntnisse in HTML nötig wären.

(vgl. <http://de.wikipedia.org/wiki/Auszeichnungssprache> Datum: 26.12.06)

CVS

CVS (Concurrent Versions System) ist eine Software, die sich um die Versionierung von Dateien kümmert. Mit diesem Tool kann man die Versionsgeschichte von Quellcode-Dateien und Dokumenten aufzeichnen.

(vgl. <http://www.nongnu.org/cvs> Datum: 11.01.07)

DIN

Der eingetragene Verein versteht sich als eine Art "runder Tisch", an dem Hersteller, Verbraucher, Handwerk, Handel, Dienstleistungsunternehmen, Wissenschaftler, technische Überwachungs-Organisationen und der Staat zusammen sitzen, um den Stand der Technik zu ermitteln und in *Normen* festzuhalten.

(<http://www.quality.de/lexikon/din.htm> Datum: 14.02.07)

ISO

Die Internationale Organisation für *Normung* (ISO) ist die internationale Vereinigung von Normungsorganisationen und erarbeitet internationale *Normen* in verschiedenen Bereichen.

(vgl. http://de.wikipedia.org/wiki/Internationale_Organisation_für_Normung Datum: 04.01.07)

MediaWiki

MediaWiki ist eine Wiki-Software (eine sogenannte Wiki-Engine), die ursprünglich für die freie Enzyklopädie Wikipedia entwickelt wurde. Mittlerweile wird sie auch für verschiedene andere Projekte der gemeinnützigen Wikimedia-Stiftung und, da sie für jeden frei verfügbar ist, auch für eine Vielzahl anderer Projekte im Internet oder in Intranets verwendet. Sie ist unter der GPL lizenziert und in der Programmiersprache PHP5 geschrieben. Zum Speichern der Inhalte nutzt MediaWiki die relationale Datenbank MySQL. Neben Wikipedia und ihren Wikimedia-Schwesterprojekten setzen heute zahlreiche Organisationen, Firmen und Institutionen MediaWiki ein. (vgl. [WWC-AE-05] und <http://de.wikipedia.org/wiki/Mediawiki> Datum: 03.12.06)

Norm

Eine Norm bezeichnet allgemein einen Namen oder einen Regelfall - geplant, verordnet oder sich infolge ungeplanter Prozesse ergebend [...]. (<http://de.wikipedia.org/wiki/Norm> Datum: 18.12.07)

Repository

Repositorys werden unter anderem zum Versionsmanagement verwendet. Beim CVS etwa werden z. B. Quellcode-dateien oder andere Textdateien aus dem Repository „ausgecheckt“, d. h. auf den Rechner eines Programmierers geladen. Nach der Bearbeitung werden die geänderten Dateien wieder in das Repository „eingchecked“, wobei die Veränderung protokolliert wird. (vgl. <http://de.wikipedia.org/wiki/Repository>)

Standard

Ein Standard ist eine vergleichsweise einheitliche/vereinheitlichte, weithin anerkannte und meist auch angewandte (oder zumindest angestrebte) Art und Weise, etwas herzustellen oder durchzuführen, die sich gegenüber anderen Arten und Weisen durchgesetzt hat. (<http://de.wikipedia.org/wiki/Standards>; Datum: 02.01.07)

Stylesheet

Ein Stylesheet ist am ehesten mit einer Formatvorlage zu vergleichen. Grundidee hierbei ist die Trennung von Information (Daten) und Darstellung. Das Stylesheet interpretiert die zugewiesenen Daten (Text, Tabellen, Grafiken etc.) und formatiert sie (z.B. für die Bildschirmausgabe) entsprechend den vorgegebenen Regeln. Mit Stylesheets ist in höherem Maße eine Arbeitsteilung möglich, als das früher z. B. bei HTML und eingebetteten Formatierungsbefehlen möglich war. (vgl. <http://de.wikipedia.org/wiki/Stylesheet> Datum: 18.02.07)

White-Box-Test

Der Begriff White-Box-Test bezeichnet eine Methode des Software-Tests, bei der die Tests mit Kenntnissen über die innere Funktionsweise des zu testenden Systems entwickelt werden. Im Gegensatz zum Black-Box-Test benötigt man für diese Art von Tests Kenntnisse über den Quellcode.

Wiki

Ein Wiki ist eine web-basierte Software, die es allen Lesern von Seiten erlaubt, den Inhalt einer Seite mit Hilfe eines Browsers zu ändern. Dadurch wird das Wiki zu einer einfach zu nutzenden Plattform, die es ermöglicht gemeinsam an Texten zu arbeiten. [vgl. WWC-AE-05]

Wikiartikel

(siehe „Artikel“)

Wiki Markup Language

Einfach gehaltene *Auszeichnungssprache* um Wiki-Artikeln ein strukturiertes Aussehen verleihen zu können. Es ist möglich Überschriften, Links, Aufzählungen oder Tabellen zu erstellen.

WIKI-System

(siehe „Wiki“)

XML-Parser

XML-Parser analysieren XML-Dokumente und stellen die darin enthaltenen Informationen (also Elemente, Attribute usw.) der weiteren Verarbeitung zur Verfügung.

Danksagung

Ich bedanke mich bei Herrn Jörg Rech (Fraunhofer IESE Kaiserslautern), der mir die Chance für diese Arbeit gegeben und mich bei der Bearbeitung unterstützt hat, sowie bei Prof. Dr. Peter Kaiser (HS Mannheim), der meine Bachelorarbeit betreute.

Außerdem möchte ich mich noch bei meiner Familie, besonders bei meinen Eltern sowie bei meiner Freundin bedanken, die mich immer unterstützt haben.