TECHNISCHE
UNIVERSITÄT
DARMSTADT

Pattern-basierte Prozessbeschreibung und -unterstützung

Ein Werkzeug zur Unterstützung von Prozessen zur Anpassung von E-Learning-Materialien

Vom Fachbereich
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertationsschrift

von

Dipl.-Inform. Birgit Zimmermann

geboren am 22. August 1975 in Freiburg im Breisgau

Tag der Einreichung: 25.08.2008
Tag der Disputation: 27.11.2008

Vorsitzender: Prof. Dr.-Ing. Jürgen Stenzel
Erstreferent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Jörg M. Haake

Darmstadt 2008
Hochschulkennziffer D-17

Kurzfassung

Die Erstellung hochwertiger Lernressourcen für das E-Learning ist eine zeitaufwendige und kostenintensive Arbeit. Daher ist es wünschenswert, existierende Lernressourcen wiederzuverwenden. Doch oft entspricht der neue Einsatzkontext nicht exakt dem Kontext, für den eine Lernressource ursprünglich erstellt wurde. Soll beispielsweise ein Kurs, der für Unternehmen A erstellt wurde in Unternehmen B wiederverwendet werden, so können sich die Anforderungen an das Layout oder die Terminologie unterscheiden. In diesem Fall ist es notwendig, die Lernressource an den neuen Einsatzkontext anzupassen.

Die Erstellung und Anpassung von Lernressourcen wird überwiegend von Domänenexperten (Experten in der jeweiligen Inhaltsdomäne einer Lernressource) durchgeführt. Sie verfügen jedoch meist nur über geringe Expertise bezüglich der notwendigen Anpassungsprozesse. Sie sind in dieser Hinsicht eher Laien. Anpassungsprozesse sind für sie daher oft nur schwer durchzuführen. Das gilt insbesondere, da die Anpassung von Lernressourcen eine komplexe Aufgabe ist, die viel Spezialwissen verlangt. Zum einen werden Lernressourcen in einer Vielzahl von Dokumentenformaten erstellt. Zum anderen gibt es viele verschiedene Arten von Anpassungen (u. a. Gestaltung, Sprache, Didaktik …) und verschiedene Werkzeuge für deren Durchführung. Damit Laien die Anpassungsprozesse effizient und fehlerfrei durchführen können, besteht die Zielsetzung, sie geeignet durch ein Software-Werkzeug zu unterstützen.

Das zur Durchführung der Anpassungsprozesse notwendige Spezialwissen besitzen Personen, die Experten in der Ausführung dieser Prozesse sind. Um eine Software zur Unterstützung der Anpassungsprozesse zu entwickeln, ist es wichtig, das Wissen dieser Experten in die Entwicklung einfließen zu lassen. Sie wissen, wie die Prozesse durchgeführt werden und welche Funktionalitäten zur Unterstützung benötigt werden.

Jedoch ist die Einbeziehung von Prozessexperten bei der Erstellung von Software zur Prozessunterstützung in der Praxis nicht immer gegeben. Üblicherweise werden die Experten zwar zu den von ihnen durchgeführten Prozessen befragt, Entwurf und Erstellung der Software werden aber von Software-Designern und Entwicklern vorgenommen. Die Software spiegelt im Resultat daher häufig das Verständnis dieser Personen von den zu unterstützenden Prozessen wieder. Dieses weicht aber oftmals davon ab, wie die Prozesse von Prozessexperten verstanden und durchgeführt werden.

Die vorliegende Arbeit stellt ein durchgängiges Konzept vor, das es erlaubt, Prozessexperten direkt in die Entwicklung von Software zur Prozessunterstützung einzubinden. So kann ein Werkzeug zur Unterstützung von Anpassungsprozessen erstellt werden, das das Wissen der Prozessexperten beinhaltet und nachfolgend andere Nutzer, nämlich die Laien, bei der Durchführung der Prozesse unterstützt.

Dazu wurde ein dreistufiges Konzept entwickelt, das im ersten Schritt Prozessexperten die Möglichkeit bietet, die von ihnen durchgeführten Anpassungsprozesse so zu beschreiben, dass basierend auf diesen Beschreibungen im zweiten Schritt ein Software-Prototyp erstellt werden kann. Dieser dient im dritten Schritt als Basis zur Entwicklung eines einsatzfähigen Werkzeuges zur Unterstützung von Anpassungsprozessen.

Im ersten Schritt des vorgestellten Konzeptes beschreiben Prozessexperten die Anpassungsprozesse. Dazu wird eine Pattern-basierte Notationsform verwendet. Diese bietet den Vorteil, dass sie aufgrund der Verwendung natürlicher Sprache einfach verständlich und somit auch für Prozessexperten ohne Kenntnisse in der Modellierung von Prozessen leicht erlernbar ist. Patterns haben sich auch in anderen Gebieten bewährt, um das Wissen von Experten in der Durchführung bestimmter Tätigkeiten zu dokumentieren. Ein im Rahmen der Dissertation konzipiertes Eingabewerkzeug unterstützt die Prozessexperten beim Anlegen der Pattern-artigen Prozessbeschreibungen.

Das Eingabewerkzeug wandelt die Angaben der Prozessexperten in eine maschinenlesbare XML-Darstellung um. Diese dient im zweiten Schritt als Eingabe für ein Werkzeug, das es den Prozessexperten erlaubt, aus den von ihnen erstellten Prozessbeschreibungen einen Software-Prototyp zu generieren. Anhand dieses Prototyps können die Prozessexperten überprüfen, ob die Anpassungsprozesse korrekt abgebildet und unterstützt werden. Entspricht ein Prototyp nicht den Vorstellungen der Prozessexperten, können die Prozessbeschreibungen verändert und erneut ein Prototyp generiert werden. Dieses Vorgehen kann so oft wiederholt werden, bis der Prototyp den Vorstellungen der Prozessexperten entspricht.

Der so entstandene Prototyp spiegelt das Verständnis der Prozessexperten von Anpassungsprozessen wieder. Er kann einem Entwickler zur Verfügung gestellt werden. Dieser kann im dritten Schritt des Konzeptes den Prototyp durch Zufügen automatisierter Funktionen zu einem einsatzfähigen Werkzeug zur Unterstützung von Laien bei der Durchführung von Anpassungsprozessen erweitern. Im Rahmen der Dissertation wurden Methoden entwickelt, die es dem Entwickler im Allgemeinen erlauben eine konsistente Erweiterung des Prototyps um Automatisierungsfunktionen vorzunehmen. Des Weiteren wurde durch die Implementierung von Automatisierungsfunktionen für verschiedene Typen von Anpassungsprozessen ein Unterstützungswerkzeug erstellt.

Das fertige Werkzeug führt weitere Personen durch die Anpassungsprozesse und gibt Hilfestellungen, die auf dem Wissen der Prozessexperten beruhen. Wo sinnvoll möglich, werden automatisierte Funktionen angeboten, die den Anwendern Arbeit abnehmen. So werden auch Personen, die keine Prozessexperten für Anpassungsprozesse sind, bei der Anpassung existierender Lernressourcen an neue Einsatzkontexte unterstützt.

Die Evaluation der im Rahmen der Dissertation entwickelten Konzepte und Werkzeuge mittels eines Benutzertests zeigt, dass das entwickelte Konzept Prozessexperten in die Lage versetzt, die von ihnen durchgeführten Prozesse zu beschreiben und daraus eigenständig Prototypen zu generieren, die die beschriebenen Prozesse abbilden. Weiterhin wurde in einem zweiten Test gezeigt, dass das mit diesem Ansatz entwickelte Werkzeug zur Unterstützung von Anpassungsprozessen die Durchführung der Anpassungsprozesse verbessert: Es werden weniger Fehler gemacht, die Durchführung wird deutlich beschleunigt und die Benutzerzufriedenheit ist höher als bei einem herkömmlichen Vergleichswerkzeug.

Abstract

Creating high quality E-Learning material is a time and cost consuming task. Re-using existing material could reduce these costs. But often a one-to-one reuse of the existing material is not possible, as the new scenario of usage differs to a certain degree from the original usage scenario. If, for example, a learning resource created for company A has to be reused in company B, it is likely that the layout and the terminology have changed. In this case it is necessary to adapt the learning resource to the new usage scenario.

Predominantly, learning resources are created by domain experts, who have expertise in the domain the learning resource deals with. But often these persons are not experts in adapting the learning resources to new usage scenarios. Hence, it is difficult for them to perform the adaptation processes. There are several reasons for this: On the one hand there are many file formats used within learning resources. On the other hand there are many different kinds of adaptations (like layout, language, didactics etc.). Thus, adapting learning resources is a complex task. To allow novices to perform the adaptation processes efficiently and without errors, it is necessary to offer them an appropriate tool support.

Special knowledge is needed to perform adaptation processes. Thus, to develop software for supporting adaptation processes, it is important to integrate the knowledge of experts in performing adaptation processes into the development process, as they know how the processes are performed and which functionalities are required to support users in performing the processes.

However, in reality experts in performing adaptation processes are barely involved in the design process of new software. Usually, process experts are consulted for the processes they perform, but the design and the development of the software are done by software designers and developers. As a result, the software often reflects the understanding of these persons of the processes, which in many cases differs from how a process expert understands and performs the processes.

In this thesis, a concept is proposed that allows experts in performing adaptation processes to be involved more directly into the development process of software for supporting adaptation processes. Thus a tool can be developed, that contains the knowledge of process experts and supports other persons in performing the processes.

The developed concept consists of three steps, where the first step offers process experts the option to describe the adaptation processes. Relying on this description, a software prototype can be created in the second step. In the third step , this prototype serves as a basis to develop a functional tool for supporting adaptation processes.

For the description of the adaptation processes a pattern based notation formalism is applied in the first step. This formalism has the advantage of using natural language. Thus, it is easy to understand and easy to learn even for process experts, who have no knowledge of process modelling. Moreover, patterns have been proven suitable to document expert knowledge on how to perform certain tasks. Additionally, an input tool has been developed that supports process experts in creating the pattern-like process descriptions.

The input tool creates a machine-readable XML-representation of the descriptions provided by the process experts. In the second step this representation serves as input for a second tool that allows process experts to generate a software prototype based on their process descriptions. With this prototype it is possible to check, if the adaptation processes are supported in a correct and desired way. If a prototype does not meet the process understanding of the experts, the process descriptions can be adapted and a new prototype can be generated. This proceeding can be repeated until the prototype meets the expectations of the process experts.

The prototype developed based on this proceeding reflects the understanding of experts of the adaptation processes. Thus, it serves as a valuable basis for further development. In the third step it can be passed on to a developer, who can add automated functionalities to the prototype. Thereby, a fully functional tool to support adaptation processes is developed. This tool guides other persons through the adaptation processes and gives them support, based on the knowledge of experts. Whenever possible, automated functionalities are offered, to serve the convenience of the user. As a result, even unskilled persons, who are not experts in performing adaptation processes, are supported in adapting existing learning resource to new usage scenarios.

As evaluation of the concepts and tools developed in this thesis, a user test has been conducted. It has shown that the concepts presented here enable process experts, to describe the processes they perform, and to create a prototype, which represents these processes. In addition it has been shown in a second user test that the developed tool for adaptation processes, which has been created based on the discussed concepts, enhances the support for adaptation processes: Users make less errors, the adaptation takes less time and the satisfaction of the users is higher compared to users working with a conventional tool.

Danksagung

Die vorliegende Dissertation entstand während meiner Tätigkeit als Doktorandin für SAP Research und das Fachgebiet Multimedia Kommunikation (KOM) an der Technischen Universität Darmstadt. Beiden danke ich herzlich für die Unterstützung.

Ganz besonders möchte ich mich bei Prof. Dr.-Ing. Ralf Steinmetz für die Betreuung dieser Arbeit bedanken. Das kreative und hilfsbereite Umfeld an dem vom ihm geleiteten Fachgebiet KOM hat sehr zum Gelingen dieser Arbeit beigetragen. Weiterhin gilt mein Dank dem Zweitbetreuer meiner Arbeit, Prof. Dr.-Ing. Jörg M. Haake.

Beim Zustandekommen dieser Dissertation hat mich eine Reihe von Personen unterstützt. Hier sind zum einen die Studenten zu nennen, die bei mir ihre Studien- und Diplomarbeiten geschrieben haben: Alexander, Cheng und Michael. Besonders danken möchte ich in diesem Zusammenhang Safar Buchholz, die durch ihre Studienarbeit einen wertvollen Beitrag bei der Durchführung der Evaluation dieser Arbeit geleistet hat. Außerdem gilt mein Dank meinen Hiwis Lars und Judith.

Weiterhin danke ich meinen Kollegen in der Knowledge Media Gruppe, hier ganz besonders Dr. Christoph Rensing für die fabelhafte Unterstützung während der gesamten Zeit meiner Promotion. Weiterhin danke ich den Kollegen, die mit mir am Content Sharing Projekt gearbeitet haben: Marek, Sonja und Tomas. An dieser Stelle seien auch meine Zimmerkollegen Doreen und Tomas genannt, mit denen ich eine sehr schöne Zeit verbracht habe.

Auch bei SAP Research habe ich am CEC Darmstadt ein fabelhaftes Umfeld vorgefunden und sehr gute Unterstützung. Dafür danke ich insbesondere dem Leiter des CEC Darmstadt, Dr. Knut Manske.

Allen Personen, die mir beim Korrekturlesen dieser Arbeit geholfen haben: Vielen Dank. Hier sind zu nennen: Andreas, Christian, Christoph, Marek, Sonja und Stefan.

Sehr dankbar bin ich auch meinem Vater und meiner Mutter, sowie Marianne und Björn für all das Vertrauen und die Unterstützung, die sie mir mein Leben lang gegeben haben.

Von ganzem Herzen danke ich außerdem Matthias, der durch seine emotionale Unterstützung einen großen Teil zum Gelingen dieser Arbeit beigetragen hat.

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Befragt man Anwender von Software, so hört man immer wieder die Klage, dass viele Programme nicht oder nur fehlerhaft die Ausführung von Prozessen unterstützen. Ein Beispiel hierfür findet sich bei der Unterstützung der Durchführung von Anpassungsprozessen. Diese Prozesse dienen der Anpassung bereits existierender Lernressourcen für das E-Learning an neue Einsatzkontexte.

Während einer Expertenumfrage zu Anpassungsprozessen [Zi+06] haben die Befragten mehrfach geäußert, dass keine Programme existieren, die alle bei den Prozessen benötigten Arbeitsabläufe geeignet unterstützen. Vielmehr müssen für die Anpassung von Lernressourcen in der Regel Werkzeuge, die zur Erstellung von Ressourcen entwickelt wurden, genutzt werden. Doch diese Werkzeuge sehen keine Unterstützung von Anpassungsprozessen vor. Zudem bedingen sie einen hohen manuellen Aufwand bei der Durchführung dieser Prozesse. Diese Problematik verstärkt sich dadurch, dass die Lernressourcen oftmals eine Reihe unterschiedlicher Formate beinhalten. Somit müssen viele verschiedene Werkzeuge genutzt werden, um die Anpassungsprozesse durchzuführen. Jemandem, der eine Anpassung durchführen möchte, stehen gegebenenfalls nicht alle benötigten Werkzeuge zur Verfügung, oder er hat nicht die Kenntnisse, um alle Werkzeuge bedienen zu können.

Personen, die tagtäglich die Anpassungsprozesse durchführen und somit über eine entsprechende Expertise verfügen, sind in der Durchführung von Anpassungsprozessen und in der Nutzung der Werkzeuge trainiert. Sie haben das nötige Prozesswissen, um diese Programme auch ohne geeignete Unterstützung bei der Prozessdurchführung zu bedienen. Aber Neulinge oder Personen, die die Aufgaben nur unregelmäßig durchführen, werden nicht ausreichend unterstützt. Daher ist es wünschenswert, besser geeignete Programme zur Unterstützung dieser Prozesse zu entwickeln.

Um eine solche Software zur Unterstützung der Anpassungsprozesse zu entwickeln, ist es wichtig, das Wissen der Personen, die sich in der Ausführung der Prozesse sehr gut auskennen (die sogenannten *Prozessexperten*), in die Entwicklung von Unterstützungswerkzeugen einfließen zu lassen. Diese Personen können am besten beurteilen, welche Funktionalitäten zur Unterstützung der Durchführung von Prozessen benötigt werden. Weiterhin ist es wünschenswert, dass das Wissen der Prozessexperten nicht nur bei der Erstellung der Software berücksichtigt wird, sondern z.B. durch Hilfestellungen auch Bestandteil der Software ist und dadurch explizit an andere Personen weitergegeben werden kann. Diese Forderung gilt nicht nur für die Unterstützung der Anpassungsprozesse, sondern auch allgemein. Eine durchgängige Einbindung der Prozessexperten in die Analyse- und Designphase fordern van der Aalst und van Hee: „The way in which the development process is carried out should correspond with this by involving the “users” as much as possible in the design of processes and systems.” [AH02, S.212] Und Fowler sagt, dass „effektive Modelle nur durch Leute erstellt werden können, die den Problembereich genau verstehen“ [Fo96, S.4].

Der Einbezug von Prozessexperten ist in der Praxis bei der Erstellung von Software zur Prozessunterstützung nicht immer der Fall. Bei der klassischen Software-Entwicklung beginnt man im Allgemeinen mit einer Analysephase, in deren Verlauf Prozessexperten befragt werden, um zu ermitteln, wie die Prozesse durchgeführt werden und wie eine sinnvolle Unterstützung aussehen soll. In dieser Phase werden zudem Anforderungen an die zu erstellende Software sowohl von Benutzerseite als auch von Systemseite gesammelt. Software-Designer erstellen in der Design-Phase basierend auf den gesammelten Informationen Modelle der Prozesse, die während der Entwicklungsphase implementiert werden [Ro87]. (Das genaue Vorgehen hängt vom gewählten Vorgehensmodell ab, beispielsweise Software-Entwicklung basierend auf dem Wasserfallmodell, dem Spiralmodell, oder Extreme Programming.) Es gibt keine allgemein akzeptierte, einfache Methode, die bei der Entwicklung von Prozessunterstützungswerkzeugen Experten unmittelbar in die Modellierung der Prozesse einbezieht. Die Experten müssten sich vielmehr in die existierenden, komplexen Software-Entwicklungs- und Prozessmodellierungsmethoden einarbeiten. Daher kann es bei dieser Art der Software-Entwicklung zu Missverständnissen zwischen den Prozessexperten und den Software-Designern kommt [Ro01].

Die Software spiegelt im Resultat daher häufig das Verständnis eines Software-Designers von den Prozessen wieder [Mü07]. Dieses weicht aber oftmals davon ab, wie die Prozesse von Prozessexperten verstanden und durchgeführt werden. Um dem entgegen zu wirken, sollte die Einbindung der Prozessexperten diesen die Möglichkeit bieten, ihr Wissen so zur Verfügung zu stellen, dass es später direkt in die Entwicklung einfließen kann. Dazu ist ein gut verständliches Konzept, das seitens der Prozessexperten kein langwieriges Training verlangt, notwendig.

1.2 Ziele, Ansatz und Beiträge der Arbeit

Ziel der vorliegenden Arbeit ist es, die in der Motivation genannten Herausforderungen zu adressieren und erstens ein Werkzeug zur Unterstützung von Laien und Experten in der Durchführung der Anpassungsprozesse zu entwickeln, zu erproben und zu evaluieren. Dabei soll zweitens das Ziel verfolgt werden, die Experten unmittelbar und ohne spezielle Modellierungskenntnisse in die Entwicklung des Werkzeuges einbeziehen zu können. So soll gewährleistet werden, dass das Werkzeug eine korrekte Prozessunterstützung realisiert, die eine schnellere und weniger fehlerhafte Anpassung ermöglicht.

Die Arbeit verfolgt somit zwei Hauptziele:

1. Die Entwicklung eines Konzeptes und eines Frameworks für einen einfach handhabbaren Prozessbeschreibungsformalismus als Basis für die Erstellung eines Werkzeugs zur Prozessunterstützung von Laien und Experten
2. Die Verbesserung der Unterstützung bei der Durchführung von Anpassungsprozessen

Das Konzept zur Beschreibung von Anpassungsprozessen soll dabei folgende Anforderungen berücksichtigen:

- Das Konzept soll Prozessexperten ohne Modellierungskenntnisse eine Möglichkeit bieten, ihr Wissen über Anpassungsprozesse ohne die Hilfe von Software-Designern und / oder -Entwicklern formalisiert zu beschreiben.

- Das so zur Verfügung gestellte Wissen soll als Basis der Entwicklung eines Werkzeugs zur Unterstützung von Anpassungsprozessen dienen und muss daher alle hierfür benötigten Informationen beinhalten.
- Der verwendete Formalismus soll von Prozessexperten wie von Software-Designern und -Entwicklern ohne aufwändige Einarbeitung zu erlernen sein.
- Der Formalismus soll Software-Designern und -Entwicklern erlauben, gewohnte Methoden und Werkzeuge zur Software-Entwicklung zu verwenden.
- Das Konzept muss geeignet sein, ein Werkzeug zu entwickeln, das Personen durch die Anpassungsprozesse führt.

Die Verbesserung der Unterstützung bei der Durchführung von Anpassungsprozessen soll durch ein Werkzeug realisiert werden, das durch die Prozesse führt und eine schnellere und weniger fehleranfällige Anpassung von Lernressourcen ermöglicht.

Der in der Arbeit verwendete Ansatz basiert auf der Verwendung von Patterns als Prozessbeschreibungen, da Patterns eine bereits in anderen Bereichen erprobte Möglichkeit zum Festhalten von Expertenwissen sind. Sie bieten zudem eine natürlichsprachliche Notation und sind somit leicht verständlich. Basierend auf dem Ansatz der Verwendung von Patterns stellt die vorliegende Arbeit zur Erreichung der Ziele ein Konzept zur automatisierten Wizard-Erzeugung basierend auf Prozessbeschreibungen in Form einer Pattern-artigen Notation vor.

Um den Anforderungen gerecht zu werden und die Ziele der Arbeit zu erreichen, leistet die vorliegende Arbeit folgende Beiträge:

- Es wurde ein *übergreifendes Konzept zur Erstellung eines Werkzeugs zur Unterstützung von Anpassungsprozessen unter direkter Einbindung von Prozessexperten* entwickelt, das auch für Prozessexperten einfach handhabbar ist und eine dem Expertenwissen entsprechende Unterstützung der Anpassungsprozesse ermöglicht. Dieses Konzept sieht drei Teilschritte vor:
  - Beschreibung der Prozesse in Form von Patterns, wozu ein Konzept, eine Pattern-Notation und ein Werkzeug entwickelt wurden,
  - Generierung eines funktionsfähigen Werkzeugs in Form eines Wizards zur Prozessunterstützung. Die Generierung erfolgt auf Basis der Patterns, wozu ein Konzept und ein Werkzeug entwickelt wurden,
  - Erweiterung des generierten Wizards z.B. durch Automatisierungen von einzelnen Prozessteilschritten, wozu ein Konzept entwickelt wurde.
- Basierend auf dem Konzept und mit Hilfe der Werkzeuge wurde *ein voll funktionsfähiges Unterstützungswerkzeug für Anpassungsprozesse entwickelt*. Dabei wurden zudem *automatisierte Funktionen zur Anpassung von Lernressourcen entworfen und umgesetzt*. Dieses Werkzeug dient der Prüfung des übergreifenden Konzeptes und der zuvor genannten Werkzeuge und weiterhin unmittelbar den Nutzern in der Durchführung der Anpassungsprozesse.
- Es wurden *zwei Benutzerstudien durchgeführt*, zum einen, um das Konzept und

die Werkzeuge zur Prozessbeschreibung und Wizard-Generierung von verschiedenen Personengruppen testen zu lassen, und zum anderen, um den Nutzen des Anpassungswerkzeuges zu evaluieren.

- Da das in der Motivation genannte Problem unzureichender Prozessunterstützung auch bei Software zur Unterstützung anderer Prozesse zu finden ist, wurde anhand weiterer Prozesse getestet, ob *sich das Konzept zur Prozessbeschreibung und Wizard-Generierung auch für andere Prozesse einsetzen lässt*.

1.3 Aufbau der Arbeit

Die Arbeit ist folgendermaßen aufgebaut: Zuerst wird in Kapitel 2 das Anwendungsszenario „Anpassung existierender Lernressourcen an veränderte Einsatzszenarien“ der Arbeit vorgestellt. Dabei wird aufgezeigt, wie die Prozesse zur Anpassung existierender Lernressourcen aufgebaut sind und welche Anforderungen sich an ein Werkzeug in Form eines Wizards zur Unterstützung dieser Prozesse ergeben.

Danach werden in Kapitel 3 bestehende Software-Entwicklungsmethoden auf Ihre Eignung zur Erstellung eines solchen Wizards unter direktem Einbezug von Anpassungsexperten untersucht. Die Anforderungen an ein Konzept zur Erzeugung eines Wizards zur Unterstützung von Anpassungsprozessen werden im Detail betrachtet. Ein neues, aus drei Schritten bestehendes, Konzept wird vorgestellt, das diesen Anforderungen gerecht wird.

Die Beschreibung der drei Schritte ist Gegenstand der folgenden Kapitel: Die Erstellung der Prozessbeschreibungen in Form von Patterns und der zugrunde liegende Formalismus werden in Kapitel 4 erläutert. Die automatisierte Prototyp-Erstellung wird in Kapitel 5 vorgestellt. Im sechsten Kapitel wird erläutert, wie der automatisch erzeugte Prototyp als Basis für eine Weiterentwicklung zu einem Wizard für die Unterstützung von Anpassungsprozessen verwendet werden kann. Dazu wird gezeigt, wie mit dem in dieser Dissertation vorgeschlagenen Konzept ein Prototyp für die Unterstützung von Anpassungsprozessen erstellt wurde. Weiterhin wird erläutert, wie basierend auf dem so erstellten Prototyp ein einsatzfähiges Werkzeug zur Unterstützung von Anpassungsprozessen realisiert wurde.

Der Ansatz, der in dieser Arbeit vorgestellt werden soll, wurde für Anpassungsprozesse entwickelt. Doch das damit adressierte Problem ungeeigneter Prozessunterstützung lässt sich auch bei anderen Prozessen finden. In Kapitel 7 wird deswegen untersucht, ob sich der Ansatz auf andere Prozesse übertragen lässt.

Kapitel 8 stellt zwei durchgeführte Evaluationen vor, mittels derer geprüft wurde, ob sich das übergreifende Konzept zur Generierung von Wizards auf Basis von Patterns von Benutzern mit verschiedenstem Vorwissen in Bezug auf Prozessmodellierung anwenden lässt und ob und inwieweit das in Kapitel 6 vorgestellte Anpassungswerkzeug eine Unterstützung der Anpassungsprozesse erlaubt.

Anschließend an die Evaluation werden die Konzepte und weiteren Beiträge der Arbeit zusammengefasst und es wird ein Ausblick über noch offene Fragestellungen und mögliche Anschlussarbeiten gegeben.

2 Anpassung existierender Lernressourcen an veränderte Einsatzszenarien

Die Anpassung bereits existierender E-Learning Materialien an veränderte Einsatzkontexte ist sehr komplex. Daher sind Programme wünschenswert, die Laien wie Experten geeignet bei der Durchführung der hierzu benötigten Prozesse unterstützen. In der Motivation wurde bereits erwähnt, dass sich bei einer Benutzerumfrage zu Anpassungsprozessen [Zi+06] gezeigt hat, dass Experten in der Durchführung der Anpassungsprozesse derartige Programme vermissen. Ein Hauptziel der vorliegenden Arbeit ist es, ein Programm zu entwickeln, das dieses Problem löst. Daher wird in Kapitel 2 ein detaillierter Überblick über die Anpassung bereits existierender E-Learning Materialien an veränderte Einsatzszenarien gegeben.

2.1 Szenariobeschreibung und Definitionen

Die Bedeutung von E-Learning hat in den letzten Jahren stark zugenommen. Viele Firmen bieten ihren Mitarbeitern die Möglichkeit, sich mittels elektronischer Lernmaterialien [WH98] fortzubilden. Auch im Hochschulsektor werden immer öfter, insbesondere zur Ergänzung der traditionellen Präsenzlehre, E-Learning Methoden eingesetzt [28]. Für kleinere Firmen, die sich teure individualisierte Fortbildungen nicht leisten können oder wollen, ist E-Learning ein Weg, den Mitarbeitern dennoch berufliche Weiterbildung zu ermöglichen.

Als Lernmaterialien werden im E-Learning sogenannte Lernressourcen eingesetzt. Diese werden folgendermaßen definiert:

> *„Eine **Lernressource** ist jede digital verfügbare Ressource, die im E-Learning genutzt wird.“* [Re+08]

Die Erstellung hochwertiger Lernressourcen für das E-Learning ist eine zeitaufwendige und kostenintensive Aufgabe. Eine Wiederverwendung bereits existierender Lernressourcen wäre deswegen wünschenswert. Das Ermöglichen einer derartigen Wiederverwendung existierender Lernressourcen ist derzeit eine wichtige Forschungsfrage [DH03]. Wiederverwendung wird definiert als:

> *“**Re-Use** of Learning Resources is every kind of use of existing Learning Resources, which are already used in a certain context.*
>
> *A Learning Resource may be re-used in learning or teaching without any modification. A Learning Resource may be re-used in Authoring by Aggregation. A Learning Resource may be re-used in Re-Authoring.”* [Re+05]

Doch diese Wiederverwendung scheitert oft daran, dass sich der neue Einsatzkontext von dem Einsatzkontext unterscheidet, für den eine Lernressource ursprünglich erstellt wurde. In diesem Fall ist eine einfache Wiederverwendung nicht möglich. Stattdessen ist es notwendig, die existierende Lernressource so zu ändern, dass sie im neuen Einsatzkontext verwendet werden kann. Erschwerend kommt hinzu, dass oft nicht eine gesamte Lernressource wiederverwendet werden kann, sondern nur Teile daraus. Diese Teile müssen aus der ursprünglichen Lernressource herausgelöst und zu einer neuen

Lernressource kombiniert werden. Ein Effekt, der sich dann oft ergibt, ist der sogenannte Mosaik-Effekt [IRC03]. Dieser resultiert daraus, dass die kombinierten Teile sowohl inhaltlich als auch rein visuell unterschiedlich aufgebaut sind, beispielsweise aufgrund unterschiedlicher Vorgaben hinsichtlich des verwendeten Vokabulars oder unterschiedlicher Stylesheets. Um die Wiederverwendung einer existierenden Lernressource in einem neuen Einsatzkontext zu ermöglichen und den Mosaik-Effekt zu vermeiden, ist eine Anpassung der Ressource an den neuen Einsatzkontext notwendig.

Das Aufbereiten bereits existierender Lernressourcen mit dem Ziel, eine Wiederverwendung zu ermöglichen, wird in [Re$^+$05, Re$^+$08] als Repurposing bezeichnet. Es ist folgendermaßen definiert:

> *"**Repurposing** is the transformation of a Learning Resource to suit a new learning or teaching context. This means especially that the Learning Resource is transformed to suit a new learning objective or a new target group, which is different from the learning objective or target group the Learning Resource was created for."* [Re$^+$05]

Repurposing setzt sich aus drei Teilaufgaben zusammen (vergleiche Abbildung 1): *Modularisierung*, *Anpassung* und *Aggregation* [Re$^+$08]. *Modularisierung* bezeichnet hierbei das Zerteilen von Lernressourcen in kleinere, einzeln wieder verwendbare Einheiten, die sogenannten Module. *Aggregation* ist das Zusammenfügen mehrerer Module zu einer größeren Lerneinheit. „Die *Anpassung* umfasst die Änderung einer Lernressource hinsichtlich genau einer Dimension des Kontextes, in dem die geänderte Lernressource eingesetzt werden soll." [Re$^+$08] Kontextdimensionen sind zum Beispiel Sprache, Layout oder Terminologie. Hinsichtlich der unterschiedlichen Dimensionen gibt es verschiedene Anpassungsarten, z.B. Anpassung an eine veränderte Terminologie oder Anpassung an ein verändertes Corporate Design. Zur Ausführung einer Anpassungsart wird ein Anpassungsprozess durchgeführt.

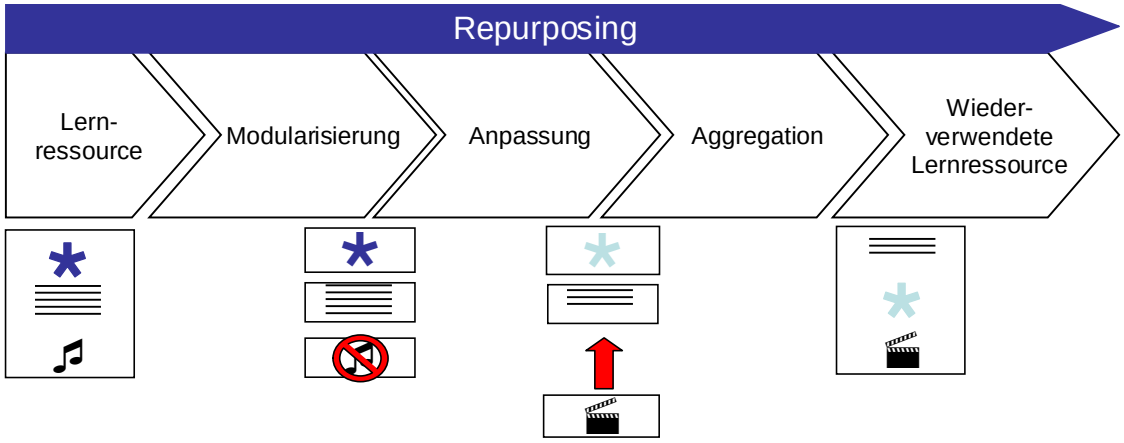


Abbildung 1: Aufbau des Repurposing [Be$^+$05].

In der vorliegenden Arbeit wird eine der drei Teilaufgaben des Repurposing detailliert betrachtet: Die Anpassung existierender Lernressourcen an neue Einsatzkontexte. Die beiden anderen Teilaufgaben werden in [Me08] vorgestellt.

Die Durchführung von Anpassungen geschieht in Form von Prozessen. Deswegen wird an dieser Stelle das Prozessverständnis der vorliegenden Arbeit geklärt.

Abhängig von den verschiedenen Anwendungsgebieten, in denen Prozesse auftreten, gibt es eine Vielzahl unterschiedlichster Prozessdefinitionen [Mü07]. In Merriam Webster's Dictionary [15] findet sich eine grundlegende Definition eines Prozesses als:

> „*a series of actions or operations conducing to an end*"

In der Informatik werden Prozesse verstanden als [CS93]:

> *„eine Folge von Aktionen in einem Zustandsraum"*

Beide Definitionen betonen, dass es sich bei einem Prozess um eine Abfolge von Aktionen handelt. Vahs [Va03] grenzt den Prozessbegriff dahingehend ein, dass er die Zielgerichtetheit von Prozessen betont. Er definiert Prozesse folgendermaßen:

> *„Ein Prozess wird demnach als eine Verkettung … von Tätigkeiten verstanden, mit denen bestimmte Ziele verfolgt werden."* [Va03, S. 206]

Dieses Verständnis von Prozessen findet auch bei Anpassungsprozessen, wie sie in der vorliegenden Arbeit betrachtet werden, Verwendung.

Für Autoren sind die Prozesse zur Anpassung an neue Einsatzkontexte von Lernressourcen oft nur schwer durchzuführen. Das hat verschiedene Gründe. Zum einen werden Lernressourcen in einer Vielzahl von Dokumentenformaten (z.B. HTML, XML, Flash …) erstellt. Dies ist insbesondere bei multimedialen Lernressourcen der Fall, die nach [SN02] aus statischen und kontinuierlichen Medien bestehen und dem entsprechend unterschiedliche Dateiformate beinhalten. Bei der Anpassung muss man Kenntnisse zur Bearbeitung aller Formate haben, die eine Lernressource enthält. Außerdem benötigt man die zur Bearbeitung des jeweiligen Formates geeigneten Werkzeuge, die man auch beherrschen muss. Zum anderen gibt es viele Dimensionen, hinsichtlich derer eine Anpassung erfolgen kann (u. a. Gestaltung, Sprache, Didaktik …). Dadurch sind verschiedene Arten von Anpassungen notwendig. Viele Autoren haben aber nicht die Kenntnisse, um die für die verschiedenen Anpassungsarten benötigten Prozesse perfekt durchzuführen. [Hö+05]

Erschwerend kommt hinzu, dass es in vielen Fällen nicht reicht, nur eine Art von Anpassung vorzunehmen. Vielmehr ist es notwendig, mehrere Anpassungsprozesse durchzuführen, um ein wirklich gutes Resultat zu erzielen. So müssen beispielsweise bei der Wiederverwendung einer Lernressource, die für Firma A erzeugt wurde, in einer anderen Firma B meist mindestens zwei Anpassungsprozesse durchgeführt werden: Das Layout muss an die Vorgaben hinsichtlich des Corporate Designs der Firma B angepasst werden und die Terminologie muss hinsichtlich der in Firma B verwendeten Terminologie verändert werden.

Die Anpassung von Lernressourcen ist demnach eine komplexe Aufgabe. Um sie durchführen zu können, ist ein hohes Maß an Wissen notwendig. Dieses steht Personen zur Verfügung, die Experten in der Ausführung von Anpassungsprozessen sind. Sie werden in der vorliegenden Arbeit als *Prozessexperten* bezeichnet. Der Begriff Prozessexperte ist in dieser Arbeit folgendermaßen definiert:

> ***Prozessexperten*** *sind Personen, die mit einem bestimmten Prozess aufgrund oftmaliger Anwendung und Wiederholung vertraut sind. Sie beschäftigen sich bereits seit längerer Zeit schwerpunktmäßig mit diesem Prozess und haben daher Spezialwissen zu diesem Prozess, das sie anderen Personen zur Verfügung stellen können.*

Annahme bei dieser Definition ist, dass die Prozessexperten ihr Wissen zu dem von ihnen ausgeführten Prozess reflektieren und ausdrücken können.

Bei Prozessexperten handelt es sich beispielsweise um Übersetzer, Pädagogen, Designer. Doch in der Realität müssen die Anpassungsprozesse oftmals von Autoren durchgeführt werden, die Experten in den jeweiligen Inhalten der Lernressourcen sind. Man bezeichnet diese Personen als *Domänenexperten*, da sie Experten in der Wissensdomäne sind, die Thema des jeweiligen Kurses ist. In den meisten Fällen sind Domänenexperten nicht gleichzeitig auch Prozessexperten in den verschiedenen Anpassungsprozessen. Daher ist es sinnvoll, sie bei der Durchführung von Anpassungsprozessen mit einem Werkzeug zu unterstützen, das sie durch die Prozesse führt und ihnen bei der Durchführung hilft, so dass auch sie ein zufriedenstellendes Resultat erzielen können.

2.2 Analyse der Anpassungsprozesse

Grundlage eines Werkzeugs zur Unterstützung bei der Durchführung von Anpassungsprozessen ist das Wissen darüber, welche Arten von Anpassungen verwendet werden und wie die entsprechenden Anpassungsprozesse durchzuführen sind. Um dies zu ermitteln, wurde im Rahmen der vorliegenden Arbeit eine Bedarfsanalyse durchgeführt. Diese bestand aus drei Schritten:

1. Theoretische Überlegungen
2. Analyse verwandter Arbeiten
3. Benutzerbefragung

2.2.1 Verschiedene Arten von Anpassungen

Im ersten Schritt wurde, basierend auf eigenen Erfahrungen, ermittelt, welche Anpassungsarten durchgeführt werden können. Als zweites wurden verwandte Arbeiten (siehe Abschnitt 2.5) im Hinblick auf die darin genannten Anpassungsarten durchsucht. Ergebnis der beiden ersten Schritte war eine Liste von fünfzehn Anpassungsarten:

- Die Anpassung an ein verändertes Lernziel (z.B. wenn das gleiche Thema behandelt wird, aber mit einem anderen Schwerpunkt)
- Die Anpassung an eine andere Zeitdauer (z.B. Teile aus einer Lernressource herauslösen, um aus einem E-Learning Kurs von drei Stunden Dauer einen 30 minütigen Überblickskurs zu generieren)
- Die Anpassung an einen veränderten Schwierigkeitsgrad (z.B. durch zusätzliche Erklärungen und Übungen die Lernressource etwas leichter machen)

- Die Anpassung an eine veränderte Lernstrategie der Kursteilnehmer (z.B. zu Beginn Fragen stellen, um Vorwissen zu aktivieren, oder Aufgaben zum Trainieren des Problemlöseverhaltens zufügen)
- Die Anpassung an eine andere Sprache (Übersetzung)
- Die Anpassung an eine veränderte Terminologie (z.B. begründet durch eine andere branchenspezifische Fachsprache)
- Die Anpassung an unterschiedliche Bildschirmauflösungen (z.B. bei Verwendung unterschiedlich großer Bildschirme zum Betrachten einer Lernressource)
- Die Anpassung an die zum Betrachten verwendeten Endgeräte (z.B. bei Verwendung von PC, mobilen Kleinrechnern, Handy…)
- Die Anpassung an unterschiedliche Bandbreiten, die zum Herunterladen der Ressourcen zu Verfügung stehen (z.B. bei Modem, ISDN, DSL, …)
- Die Anpassung an einen anderen Interaktionsgrad der Kursteilnehmer (z.B. steigt bei Hinzufügen von Simulationen, bei denen der Lernende selbst aktiv werden muss, der Interaktionsgrad)
- Die Anpassung zum Erstellen einer Druckversion (z.B. wenn eine zusätzliche Druckversion benötigt wird)
- Die Anpassung an eine andere semantische Dichte (Definition nach [Ie02]: „Die semantische Dicht ist der Grad der Prägnanz eines Lernobjektes.“ Z.B. verringert das Zufügen zusätzlicher Erläuterungen zu eventuell verwendeten Fachbegriffen die semantische Dichte eines Kurses.)
- Die Umwandlung in andere Formate (z.B. von HTML nach PDF)
- Die Anpassung an veränderte Vorgaben hinsichtlich des Corporate Designs (z.B. wenn sich die Design-Richtlinien ändern)
- Die Anpassung an die Bedürfnisse von Behinderten (z.B. wenn auch Farbenblinde unter den Lernenden sind)

Die folgenden beiden Abbildungen zeigen ein Beispiel für die Durchführung einer Anpassung an veränderte Vorgaben hinsichtlich des Corporate Designs. Abbildung 2 zeigt einen Kursseite vor der Durchführung der Anpassung, Abbildung 3 danach.

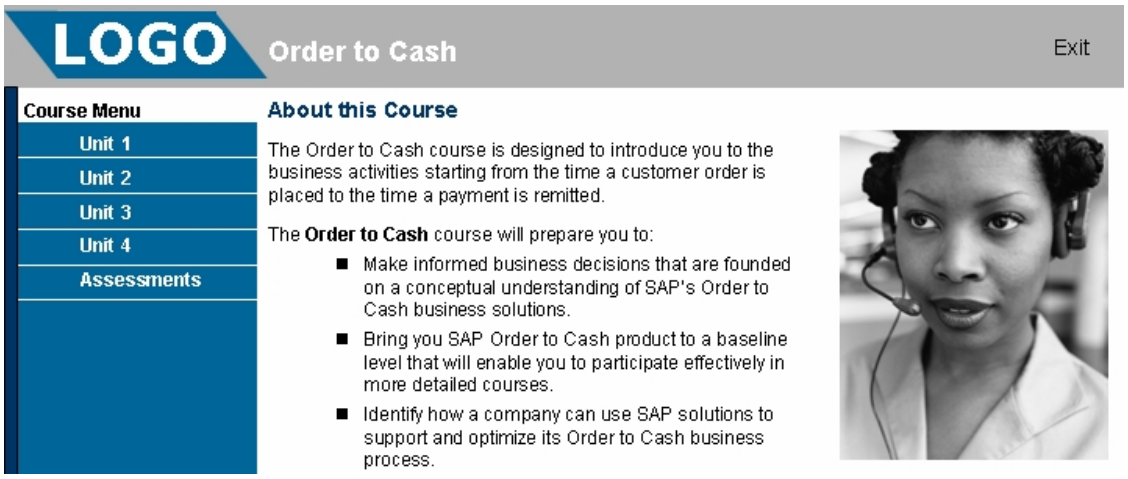


Abbildung 2: Kursseite vor der Anpassung.

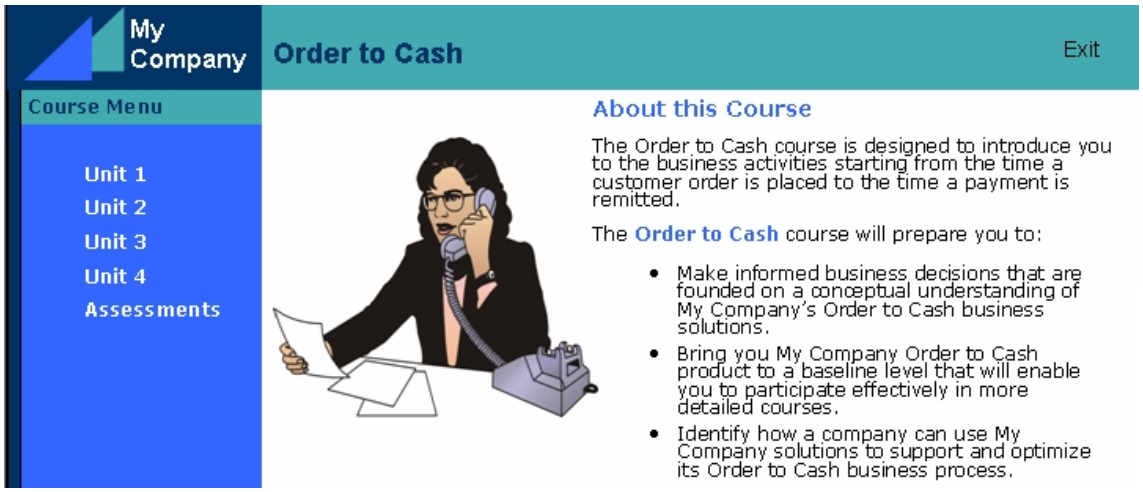


Abbildung 3: Kursseite nach der Anpassung.

Weiterhin hat sich in den beiden ersten Schritten der Bedarfsanalyse gezeigt, dass es drei Kategorien gibt, hinsichtlich derer die verschiedenen Anpassungsarten klassifiziert werden können:

Es gibt *gestalterische Anpassungsarten*, die vor allem eine Änderung des Aussehens der Lernressourcen hervorrufen. Hierzu zählt zum Beispiel die Anpassung an veränderte Design-Vorgaben. Dann gibt es *inhaltliche Anpassungsarten*, die sich vor allem auf den Inhalt auswirken, wie eine Anpassung an eine veränderte Terminologie. Und zum Schluss gibt es *technische Anpassungsarten*, die vor allem aufgrund technischer Gegebenheiten durchgeführt werden müssen, wie eine Anpassung an unterschiedliche Download-Bandbreiten. Tabelle 1 zeigt diese Zuordnung.

Gestaltung	Inhalt	Technologie
• Corporate Design • Bedürfnisse von Behinderten • Druckbarkeit	• Lernziel • Zeitdauer • Schwierigkeitsgrad • Lernstrategie • Übersetzung • Terminologie • Interaktionsgrad • Semantische Dichte	• Bildschirmauflösung • Endgeräte • Bandbreiten • Formatumwandlung

Tabelle 1: Kategorien von Anpassungen [ZRS06a].

Viele Anpassungsarten lassen sich nicht eindeutig einer Kategorie zuordnen, sondern wirken sich auf mehrere Kategorien aus. So verändert zum Beispiel eine Anpassung an die Bedürfnisse von Behinderten meist den Inhalt und das Aussehen einer Lernressource. In der Tabelle ist jeweils die Zuordnung gezeigt, die am stärksten ausgeprägt ist.

Basierend auf den Resultaten der beiden ersten Schritte der Bedarfsanalyse wurde im dritten Schritt eine Benutzerbefragung durchgeführt. Ziel der Befragung war zum einen, zu überprüfen, welche der in den beiden ersten Schritten ermittelten Anpassungsarten von Personen durchgeführt werden, die sich mit der Erstellung und Anpassung von Lernressourcen beschäftigen, und wie sie dabei vorgehen. Zum anderen sollte herausgefunden werden, ob diese Personen Anpassungsarten in der Liste vermissen oder ob die Liste als vollständig betrachtet werden kann.

Insgesamt wurden 15 Personen aus acht verschiedenen Firmen befragt. Alle Personen beschäftigen sich in ihrer täglichen Arbeit hauptsächlich damit, Lernressourcen zu erstellen und bestehende Ressourcen an veränderte Bedingungen anzupassen. Sie sind also Experten in der Durchführung der zur Anpassung benötigten Prozesse.

Das Ziel der Befragung war nicht, eine quantitative Aussage über verschiedene Anpassungsarten und -prozesse zu erhalten, sondern, wie zuvor bereits dargestellt, einen Überblick über den Bedarf an verschiedenen Anpassungsarten und das Vorgehen bei deren Durchführung in verschiedenen Unternehmen zu erhalten. Deswegen war es ausreichend, eine verhältnismäßig geringe Anzahl von Personen zu befragen.

Die Befragung bestand aus zwei Teilen:

1. Ein Fragebogen, der vorab an die Teilnehmer versandt wurde. Er diente primär der Ermittlung eingesetzter Formate und durchgeführter Anpassungsprozesse.
2. Ein Telefoninterview, das basierend auf den zum Fragebogen erhaltenen Antworten durchgeführt wurde. Grundlage des Interviews bildeten Fragen, die den Teilnehmern zur Vorbereitung vorab zugesandt worden waren.

In den nachfolgenden Abschnitten werden die Ergebnisse der Befragung vorgestellt.

2.2.2 Von Experten durchgeführte Anpassungsarten

Die Befragten wurden gebeten, für die 15 Arten von Anpassungen anzugeben, ob sie die jeweilige Anpassungsart „oft“, „selten“ oder „nie“ durchführen. Wie in Abbildung 4 zu sehen ist, wurden die Angaben „oft“, „selten“ und „nie“ teilweise ähnlich häufig genannt. Es ist lediglich ein schwacher Trend zu erkennen, dass Übersetzung, Anpassung hinsichtlich eines veränderten (Corporate) Designs sowie Anpassung an veränderte Lernziele besonders oft vorkommen. Die Anpassungsarten hinsichtlich verwendeter Endgeräte und Barrierefreiheit werden am seltensten ausgeführt. Doch das wird sich in der Zukunft vermutlich ändern, da beide Aspekte zunehmend an Bedeutung gewinnen.

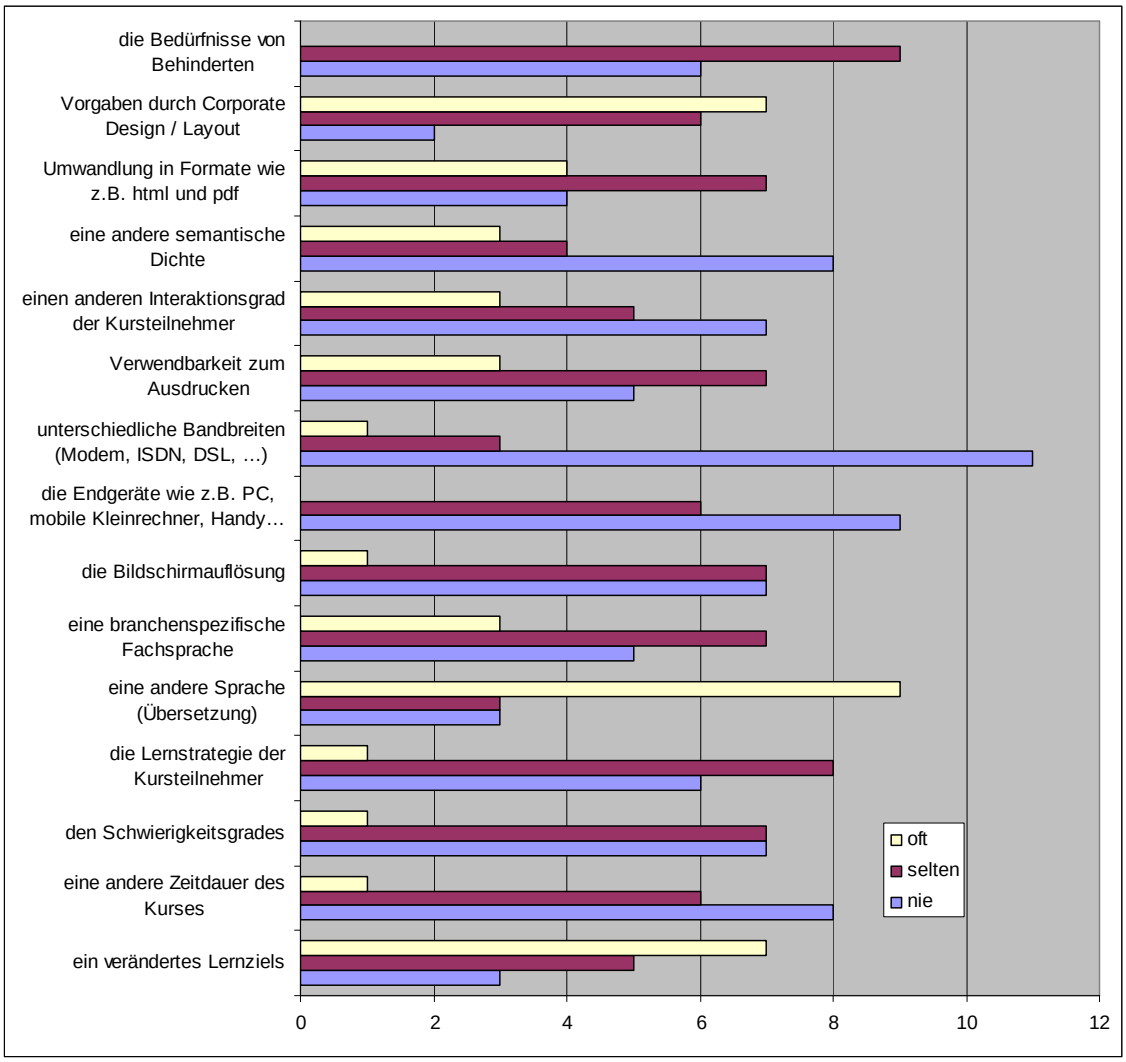


Abbildung 4: Durchgeführte Anpassungsarten.

Es gibt also keine Anpassungsart, die so gut wie nie ausgeführt wird. Andererseits gibt es auch keine Anpassungsart, die von allen Befragten besonders häufig ausgeführt wird.

Die Befragten wurden auch gebeten, zu prüfen, ob die Liste der 15 Anpassungsarten alle Anpassungsarten enthält, die sie in ihrer Arbeit durchführen. Keiner der Befragten

gab bei der Beantwortung an, eine Anpassungsart in der Liste zu vermissen. Das bedeutet jedoch nicht, dass die Liste somit vollständig ist. Es lässt sich aber sagen, dass zumindest für die betrachtete Zielgruppe die Liste vollständig zu sein scheint.

2.2.3 Strukturierte und unstrukturierte Anpassungsprozesse

Neben der Ermittlung, welche Anpassungsarten durchgeführt werden, war ein weiteres wichtiges Ziel der Befragung, herauszufinden, wie Prozessexperten bei der Durchführung der Anpassungsprozesse vorgehen. Daher wurden die Experten im Rahmen des Interviews gebeten, zu beschreiben, wie sie bei Anpassungsprozessen vorgehen.

Alle Prozessexperten beschrieben drei Anpassungsprozesse, die anhand der beantworteten Fragebögen ermittelt und den Befragten vorab mitgeteilt wurden. Jeder Anpassungsprozess wurde aufgrund dieses Vorgehens mehrfach beschrieben. Bei den Beschreibungen wurden nur Personen berücksichtigt, die den jeweiligen Anpassungsprozess regelmäßig durchführen. Außerdem wurde berücksichtigt, dass mindestens eine Beschreibung von einer Person kam, die den beschriebenen Anpassungsprozess oft durchführt. Somit wurde sichergestellt, dass die befragten Personen auch tatsächlich über das nötige Expertenwissen verfügten. Die Interviews wurden aufgezeichnet, die Tonmaterialien wurden verschriftlicht und anschließend ausgewertet.

Für jeden Anpassungsprozess wurde eine übergreifende Beschreibung erstellt, die alle zugehörigen Prozessbeschreibungen aus den Einzelinterviews zu einer Beschreibung zusammenfasste. Dabei stellte sich ein Hauptergebnis des Interviews heraus: Es gibt eine Reihe von Anpassungsprozessen, die von allen Personen ähnlich ausgeführt werden. Diese Anpassungsprozesse werden sehr strukturiert und regelbasiert durchgeführt. Für die restlichen Anpassungsprozesse hat sich dagegen ergeben, dass sie basierend auf Erfahrungen durchgeführt werden und nur wenigen oder gar keinen Regeln folgen.

Basierend auf diesem Ergebnis wurden zwei Kategorien von Anpassungsprozessen definiert:

1. Die erste Kategorie beinhaltet die Anpassungsprozesse, die von den Befragten in ähnlicher Art und Weise durchgeführt werden und die auf bestimmten Vorgehensregeln beruhen. Im Folgenden werden diese Anpassungsprozesse als *strukturierte, regelbasierte Anpassungsprozesse* bezeichnet. Im Hinblick auf ein Unterstützungswerkzeug lässt sich sagen, dass sie sich gut vollständig oder zumindest teilweise durch Automatisierung unterstützen lassen.
2. Die zweite Kategorie besteht aus den Anpassungsprozessen, die von den befragten Personen teilweise recht unterschiedlich ausgeführt werden. Grund hierfür ist, dass ihre Durchführung auf Erfahrungen basiert. Daher ist es auch schwer, für diese Anpassungsprozesse eine automatisierte Unterstützung anzubieten. Im günstigsten Falle lassen sie sich teilautomatisiert unterstützen. Diese Anpassungsprozesse werden im Folgenden als *unstrukturierte, erfahrungsbasierte Anpassungsprozesse* bezeichnet.

Tabelle 2 zeigt die Zuordnung der fünfzehn ermittelten Anpassungsarten zu den beiden Anpassungsprozesskategorien.

strukturierte, regelbasierte Anpassungsprozesse	**unstrukturierte, erfahrungsbasierte Anpassungsprozesse**
Anpassung an eine andere Sprache (Übersetzung)	Anpassung an ein verändertes Lernziel
Anpassung an eine branchenspezifische Fachsprache (Terminologie)	Anpassung an eine andere Zeitdauer
Anpassung an eine veränderte Bildschirmauflösung	Anpassung an einen anderen Schwierigkeitsgrad
Anpassung an die zum Betrachten verwendeten Endgeräte	Anpassung an eine andere Lernstrategie
Anpassung an unterschiedliche Bandbreiten	Anpassung an eine andere semantische Dichte
Anpassung zum Erzeugen einer Druckversion	Anpassung an einen anderen Interaktionsgrad
Umwandlung in andere Formate	
Anpassung an ein geändertes (Corporate) Design	
Anpassung hinsichtlich Barrierefreiheit	

Tabelle 2: Zuordnung Anpassungsarten zu Anpassungsartenkategorien.

Für die strukturierten, regelbasierten Anpassungsprozesse ergaben sich aus der Befragung detaillierte Vorgehensbeschreibungen. Für die unstrukturierten, erfahrungsbasierten Anpassungsprozesse gibt es keine derartige Beschreibung. Aber es stellte sich heraus, dass es eine Reihe von Tipps und Tricks gibt, die gesammelt werden konnten. Auch diese stellen für andere Personen eine wertvolle Hilfe dar, da sie helfen, Fehler zu vermeiden.

2.2.4 Aufbau von Anpassungsprozessen

Ein wichtiges Merkmal eines Prozesses ist sein Aufbau. Allen in 2.1 genannten Prozessdefinitionen ist gemeinsam, dass sich die beschriebenen Prozesse aus Schritten zusammensetzen, die in den Definitionen als Aktionen oder Tätigkeiten bezeichnet werden. In der Analyse des Aufbaus von Anpassungsprozessen hat sich gezeigt, dass sich auch Anpassungsprozesse aus einer Reihe von Schritten zusammensetzen. Betrachtet man beispielsweise den Prozess der Anpassung an ein verändertes Corporate Design so stellt man diverse Schritte fest: Man muss unter anderem grafische Elemente, die nicht mehr passen, ersetzen. Oder man muss überzählige grafische Elemente löschen. Gegebenenfalls muss man auch Änderungen nach Vorgabe eines Style-Leitfadens durchführen, um zum Beispiel Fonts oder Hintergrundfarben anzupassen.

Betrachtet man diese Schritte genauer, so bestehen sie aus kleineren Schritten. Das Ersetzen grafischer Elemente beispielsweise verlangt, dass man alle in einer Lernressource

vorhandenen grafischen Elemente ermittelt. Dann ist zu entscheiden, welche Elemente ausgetauscht werden müssen. Es ist festzulegen, durch welche neuen Elemente die ursprünglichen Elemente ersetzt werden sollen etc.

Die größeren Schritte sind deutlich abstrakter als die kleineren. Die größeren Schritte fassen die kleineren zusammen. Im Folgenden werden die größeren Schritte, die der logischen Gruppierung mehrerer kleinerer Schritte dienen, als *Prozessschritte* bezeichnet.

Prozessschritte sind also die logischen Schritte, die ausgeführt werden müssen, um einen Anpassungsprozess durchzuführen. Sie bestehen aus einer Aneinanderreihung von kleineren Schritten und beschreiben den Ablauf dieser kleineren Schritte. Kleinere Schritte können dabei erneut Prozessschritte sein. Sie können aber auch so klein sein, dass sie sich nicht weiter unterteilen lassen. Diese atomaren Schritte sollen im Folgenden als *atomare Unterschritte* bezeichnet werden.

Prozessschritte sind folgendermaßen definiert:

> ***Prozessschritte*** *sind die Schritte, aus denen sich ein Prozess zusammensetzt. Sie dienen der logischen Gruppierung kleinerer Schritte und beschreiben deren Ablauflogik. Dabei kann es sich um weitere Prozessschritte oder um atomare Unterschritte handeln.*

Atomare Unterschritte sind definiert als:

> ***Atomare Unterschritte*** *sind Prozessschritte, die sich nicht weiter unterteilen lassen.*

Aus programmiertechnischer Sicht können sich atomare Unterschritte aus diversen Funktionen zusammensetzen, aber aus Anwendersicht bilden sie eine nicht weiter unterteilbare Einheit. Beispielsweise ist das Ersetzen eines Bildes aus Sicht des Benutzers ein atomarer Unterschritt. Programmiertechnisch setzt sich dieser Unterschritt aber in manchen Formaten aus mehreren Funktionen zusammen (z.B. Löschen des alten und Einfügen eines neuen Bildes).

Weiterhin werden atomare Unterschritte dadurch gekennzeichnet, dass an ihnen genau zwei Akteure beteiligt sind: Derjenige, von dem die Aktion des Unterschrittes ausgeht, und derjenige, auf den sie sich richtet (vergleiche [Ac98]). Es kann auch vorkommen, dass beide Akteursrollen von demselben Akteur wahrgenommen werden. Das ist unter anderem der Fall bei Unterschritten, wo von einem Akteur eine Entscheidung verlangt wird. Ein Beispiel hierfür wäre: „Welche grafischen Elemente sollen gelöscht werden?“ Im Gegensatz dazu sind an einem Unterschritt zum Löschen eines grafischen Elementes zwei Akteure beteiligt: Der Benutzer, der das Löschen veranlasst, und das System, durch das die Löschung vorgenommen wird.

Es gibt eine Reihe atomarer Unterschritte, die in mehreren Prozessschritten verwendet werden. Beispielsweise wird die Ermittlung grafischer Elemente sowohl beim Austauschen oder Löschen grafischer Elemente gebraucht, als auch beim Überprüfen der korrekten Größe aller verwendeten grafischen Elemente.

Somit ergibt sich für Anpassungsprozesse der in Abbildung 5 gezeigte hierarchische Aufbau.

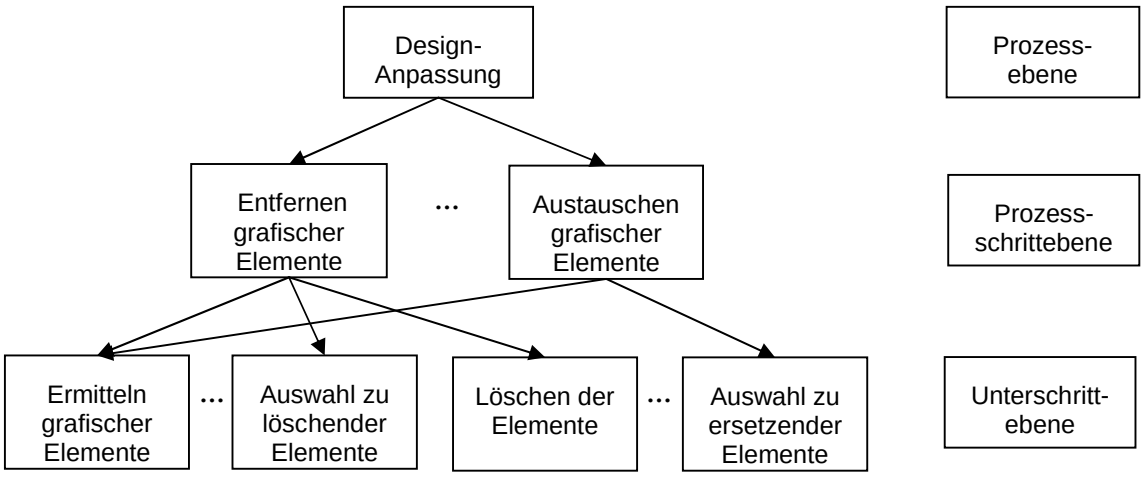


Abbildung 5: Hierarchie des Design-Anpassungsprozesses.

In der Betriebswirtschaftslehre werden Tätigkeiten in Bezug auf ihren sogenannten Rang in zwei Gruppen unterteilt: Entscheidungen und Ausführungen [He92]. Um eine Entscheidung treffen zu können, bedarf es einer Reihe von Informationen, die ermittelt werden müssen, bevor man die Entscheidung treffen kann. Auch viele Ausführungen benötigen vor ihrer Durchführung Informationen, die zuvor ermittelt werden müssen. Deswegen werden in dieser Arbeit drei Arten von atomaren Unterschritten unterschieden: Ermittlungen, Entscheidungen und Ausführungen.

Ermittlungen dienen der Beschaffung von Informationen. Beispielsweise: Die Ermittlung aller in einer Lernressource verwendeten Bilder.

Entscheidungen werden von der Person, die den Prozess durchführt, basierend auf ermittelten Informationen getroffen. Beispielsweise: Die Entscheidung, welche Bilder gelöscht werden müssen.

Ausführungen entsprechen Tätigkeiten. Hierbei wird eine Veränderung des Zustandes vorgenommen. Beispielsweise: Die Ausführung des Löschens der Bilder.

Wie bereits gesagt, reicht es oft nicht, nur einen Anpassungsprozess durchzuführen. In vielen Fällen müssen mehrere Anpassungsprozesse durchgeführt werden, um das gewünschte Resultat zu erzielen. Dementsprechend gibt es Anpassungsprozesse, die es wahrscheinlich machen, dass nach ihrer Durchführung weitere Anpassungsprozesse durchgeführt werden müssen. Beispielsweise kann eine Anpassung an ein verändertes (Corporate) Design bedeuten, dass eine Lernressource in einer anderen Firma eingesetzt werden soll. In diesem Fall sollte auch geprüft werden, ob sich die Terminologie geändert hat und gegebenenfalls angepasst werden muss.

2.2.5 Weitere Merkmale der Anpassungsprozesse

Neben dem hierarchischen Prozessaufbau weisen Anpassungsprozesse noch eine Reihe weiterer Merkmale auf. Diese sind:

1. Oft ist nur ein Autor für die Durchführung aller Anpassungsprozesse zuständig. Das bedeutet, dass Anpassungsprozesse von einer Person durchgeführt werden können. Es kann aber auch vorkommen, dass mehrere Personen an der Ausführung beteiligt sind. Aber es gibt immer eine Person, die die Hautpausführende ist und / oder die Aufgaben der anderen koordiniert.
2. Es wurde bereits erläutert, dass Spezialwissen nötig ist, um bei der Durchführung der Anpassungsprozesse ein optimales Resultat zu erzielen. In der Realität kommt es aber immer wieder vor, dass auch Laien die Prozesse durchführen müssen. Diese können ebenfalls zufrieden stellende Resultate erzielen, wenn sie geeignet unterstützt werden.
3. Die Struktur der Anpassungsprozesse ist strikt definiert und lässt sich bereits vor der Ausführung des Prozesses vollständig beschreiben. Es ist also möglich, anzugeben, aus welchen Prozessschritten sich ein Anpassungsprozess zusammensetzt, in welcher Reihenfolge diese durchlaufen werden und welchen Bedingungen die Ausführung unterliegt.
4. Die Befragung der Prozessexperten hat zu einer Reihe von Prozessbeschreibungen geführt. Diese wurden im Rahmen der vorliegenden Dissertation weiter verbessert. Dadurch wurden Beschreibungen erstellt, die es erlauben, dass auch Personen ohne Expertenkenntnisse die Prozesse ausführen können (vergleiche Abschnitt 4.2). Dabei hat sich gezeigt, dass Experten beschreiben können, wie sie die Anpassungsprozesse durchführen, so dass auch Laien anhand der Beschreibung die Prozesse durchführen können.
5. Anpassungsprozesse werden regelmäßig durchgeführt, da sie immer dann benötigt werden, wenn existierende Lernressourcen in veränderten Einsatzszenarien verwendet werden sollen. Somit ist der Aufwand für die Erstellung eines Werkzeugs, das bei der Durchführung der Prozesse unterstützt, gerechtfertigt.

Das dritte Merkmal besagt, dass es sich bei Anpassungsprozessen um strikt definierbare, klar strukturierte Prozesse handelt, die zur Laufzeit keinen Änderungen unterliegen. Im Gegensatz zu diesen Prozessen gibt es die sogenannten AdHoc-Prozesse, die sich nicht schon vor der Ausführung vollständig definieren lassen, da sie Schritte enthalten, die sich zur Laufzeit ändern können [HW99]. In der vorliegenden Arbeit wird ein Ansatz zur Unterstützung der klar strukturierten Anpassungsprozesse vorgestellt. Ein System, das kooperative AdHoc-Prozesse basierend auf einem Hypermedia System unterstützt, wird in [HW99] und [WHW02] vorgestellt.

2.3 Anforderungen an ein Unterstützungswerkzeug für Anpassungsprozesse

Wie bereits erläutert, sind Anpassungsprozesse komplexe Prozesse, für deren Durchführung eine Unterstützung durch ein Werkzeug wünschenswert wäre. Die Expertenbefragung hat ergeben, dass derzeit kein geeignetes Werkzeug zur Unterstützung der Anpassungsprozesse existiert. Deswegen wurden die Experten gebeten, ihre Wünsche und Anforderungen an ein derartiges Werkzeug zu beschreiben. In diesem Abschnitt werden die Ergebnisse der Benutzerbefragung untersucht.

2.3.1 In Lernressourcen verwendete Formate

Ein Werkzeug, das bei der Anpassung von Lernressourcen unterstützen soll, muss mit diesen Lernressourcen arbeiten. Bereits im Rahmen der beiden ersten Schritte der Bedarfsanalyse hat sich gezeigt, dass Lernressourcen üblicherweise aus einer Vielzahl von Dateien bestehen, die eine Reihe von Formaten beinhalten können.

In der Benutzerbefragung sollte herausgefunden werden, welche Formate in der Zielgruppe für ein Unterstützungswerkzeug eingesetzt werden. Dabei hat sich herausgestellt, dass textbasierte Formate wie das von Microsoft Word verwendet DOC-Format oder PDF am häufigsten bei der Erstellung von Lernressourcen eingesetzt werden. Als zweites folgen HTML und Flash, danach PowerPoint und Formate mit Soundunterstützung. Am seltensten setzen die Befragten Animationen und Videos sowie XML-basierte Formate ein. Allerdings haben viele der Befragten geäußert, dass sie für die Zukunft planen, auf XML-Formate umzusteigen. Abbildung 6 zeigt die Häufigkeitsverteilung der Antworten.

Im Rahmen des Content Sharing Projektes [4] wurde eine ähnliche Befragung durchgeführt. Hierbei wurden 17 Personen befragt. Die dabei ermittelten Daten entsprechen dem eben vorgestellten Ergebnis.

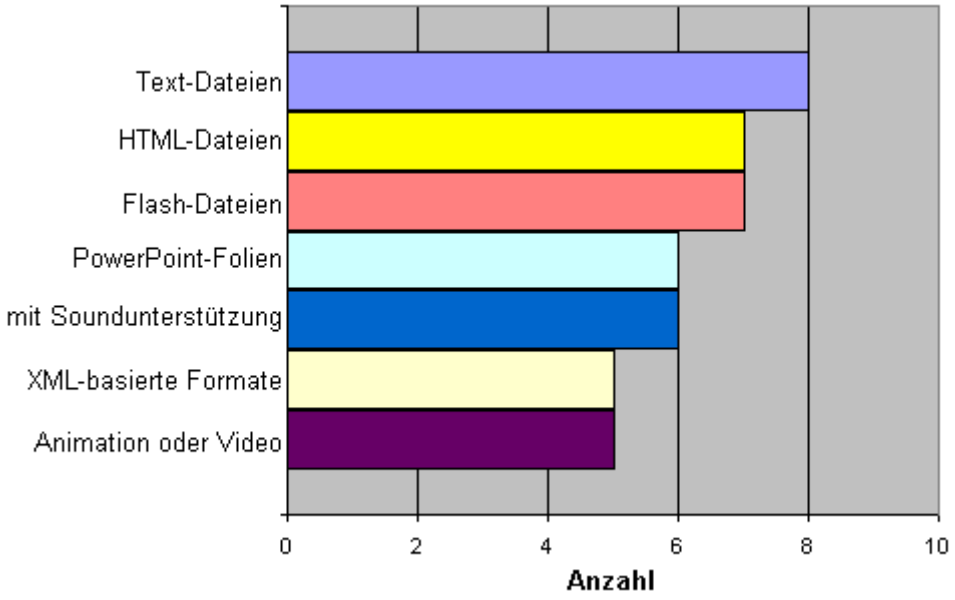


Abbildung 6: Häufigkeit der verwendeten Formate.

Die Benutzerbefragung hat außerdem gezeigt, dass entsprechend den vielen verschiedenen Formaten auch eine Reihe verschiedener Werkzeuge zur Erstellung und Bearbeitung von Lernressourcen eingesetzt werden.

Folgerung: Lernressourcen bestehen oft aus einer Unmenge von Dateien, die eine Reihe verschiedener Formate enthalten. Dementsprechend werden auch diverse unterschiedliche Werkzeuge zum Erstellen und Bearbeiten von Lernressourcen eingesetzt. Möchte man eine bereits existierende Lernressource anpassen, so muss man alle

enthaltenen Dateien einzeln anpassen. Dazu benötigt man Werkzeuge, die die darin verwendeten Formate bearbeiten können. Doch jemandem, der eine Lernressource anpassen möchte, stehen gegebenenfalls nicht alle benötigten Werkzeuge zur Verfügung. Oder er verfügt nicht über die nötigen Kenntnisse, um diese Werkzeuge bedienen zu können.

Daher wäre für ein Werkzeug zur Unterstützung von Anpassungsprozessen anzustreben, dass es alle verschiedenen Formate unabhängig vom Erstellungswerkzeug Formatübergreifend bearbeiten kann. Außerdem sollte eine Dateigrenzen-übergreifende Bearbeitung ganzer Lernressourcen möglich sein, so dass der Anwender nicht alle Dateien einzeln öffnen muss. Ein derartiges Werkzeug würde es ermöglichen, dass sich der Anwender vollständig auf die Anpassung konzentrieren kann, da er sich nicht um die verschiedenen Formate oder Dateien und Dateigrenzen kümmern muss. Außerdem wäre eine Format- und Dateigrenzen-übergreifende Bearbeitung schneller, da nicht alle Dateien und Werkzeuge geöffnet werden müssten.

2.3.2 Konzept für ein Unterstützungswerkzeug

Basierend auf den Ergebnissen der beiden ersten Schritte der Bedarfsanalyse wurde ein Konzept für ein Werkzeug zur Unterstützung der Anpassungsprozesse entwickelt. Dieses wurde den Prozessexperten im Rahmen des Interviews vorgestellt. Die Befragten wurden gebeten, das Konzept zu bewerten und Wünsche und Anregungen an ein Anpassungswerkzeug mitzuteilen.

Folgendes Konzept wurde den Interviewteilnehmern zugesandt, bevor das Interview durchgeführt wurde:

> *„Bitte bewerten Sie das im Folgenden beschriebene generische Vorgehen eines Programms, das Sie beim Durchführen der Anpassung von Lernressourcen unterstützt:*
>
> *Lernmaterialien liegen sowohl als gesamte Kurse als auch in Form kleinerer logischer Einheiten (sog. Module) vor. Kurse setzen sich aus mehreren Modulen zusammen, die auch separat angepasst werden können.*
>
> 1. *Schritt: Auswahl des anzupassenden Kurses / Moduls*
> 2. *Schritt: Auswahl der gewünschten Anpassung aus einer Reihe von Anpassungsarten (z.B. Übersetzung, Anpassung an Bildschirmgröße, Anpassung des Layouts)*
> 3. *Schritt: Das Programm prüft, ob eine Unterstützung der gewählten Anpassungsart möglich ist.*
>
>   - *Falls keine automatisierte Unterstützung möglich ist, bekommen Sie eine Anleitung zur manuellen Durchführung des Anpassungsprozesses.*
>   - *Falls die Anpassungsart nur teilweise unterstützt wird, so wird Ihnen mitgeteilt, für welche Teile eine Unterstützung möglich ist und welche Teile Sie manuell durchführen müssen. Sie können dann wählen, welche der möglichen Teilschritte automatisiert ausgeführt werden sollen.*

Für diese startet ein Dialog, der Sie schrittweise durch den Anpassungsprozess führt. Für jeden Schritt bekommen Sie das Ergebnis präsentiert, das Sie annehmen oder verwerfen können. Einzelne Anpassungsschritte können auf Wunsch ausgelassen werden.

- *Falls für alle Teilschritte eine automatisierte Unterstützung möglich ist, so startet ein Dialog, der Sie durch den Anpassungsprozess führt. Hierbei können Sie wählen, welche der einzelnen Schritte durchgeführt werden sollen. Der Anpassungsprozess kann im Gesamten oder schrittweise vorgenommen werden. Sie bekommen jeweils das Ergebnis präsentiert, das Sie annehmen oder verwerfen können.*

Es ist immer möglich, sich für die einzelnen Schritte des Anpassungsprozesses eine Anleitung für deren manuelle Durchführung anzeigen zu lassen."

Die Befragten sollten einschätzen, wie sinnvoll ein derartiges Vorgehen bei einem Programm wäre, das sie bei der Durchführung verschiedener Anpassungsprozesse unterstützen soll. Sechs Personen bewerteten das Konzept als sehr sinnvoll, weitere sechs als sinnvoll. Eine Person empfand das Werkzeug als weniger sinnvoll. Eine Person sagte aus, dass ein derartiges Werkzeug für sie nicht sinnvoll wäre, da diese Person keine Tätigkeiten durchführt, die sich automatisieren lassen. Eine Person war sich nicht sicher, wie sie das Konzept bewerten sollte.

Folgerung: Das vorgestellte Konzept wird von den Befragten überwiegend positiv bewertet. Die vorgestellte Vorgehensweise scheint somit für die Zielgruppe geeignet.

2.3.3 Weitere Wünsche an ein Unterstützungswerkzeug für Anpassungsprozesse

Im letzten Teil des Interviews wurden die Teilnehmer gefragt, ob sie Anregungen haben, wie das Programm verbessert werden kann, und ob es weitere Funktionalitäten gibt, die berücksichtigt werden sollen.

Von den Befragten wurden einerseits sehr spezielle Wünsche geäußert, andererseits aber auch sehr grundlegende Dinge, wie zum Beispiel der Wunsch nach einfacher Bedienbarkeit und Schnelligkeit. Auch Versionierung und Protokollierung der gemachten Änderungen wurden mehrfach genannt. Weiterhin wurde ein Rollen- und Berechtigungskonzept gewünscht, sowie die Möglichkeit, Reviews und Workflows zu verwenden, für den Fall, dass mehrere Personen am Arbeitsprozess beteiligt sind. Außerdem wurde explizit erwähnt, dass die Manipulation verschiedener Formate innerhalb eines Werkzeugs gewünscht ist. Weiterhin soll das Werkzeug Benutzer in Abhängigkeit ihres Kenntnisstandes unterstützen, also einen Laien- und einen Expertenmodus vorsehen. Außerdem soll das Werkzeug Anwender darauf hinweisen, was im Anschluss an eine Anpassung noch manuell nachgearbeitet werden muss, oder wo es Beziehungen zwischen verschiedenen Anpassungsarten gibt, die berücksichtigt werden müssen. Besonders wichtig ist auch, dass die Befragten keine vollständige Automatisierung erwarten, sondern nur eine Automatisierung der Prozesse bzw. Prozessschritte wünschen, die sich nach dem aktuellen Stand der Technik sinnvoll automatisieren lassen.

Folgerung: Die Wünsche der Befragten sind sehr unterschiedlich. Je nach geplantem Einsatzkontext variieren sie. Deswegen ist ein Anpassungswerkzeug so zu gestalten, dass es an die Anforderungen der jeweiligen Benutzergruppe angepasst werden kann.

2.3.4 Zusammenfassung: Anforderungen an ein Unterstützungswerkzeug für Anpassungsprozesse

Ein Werkzeug zur Unterstützung bei der Durchführung von Anpassungsprozessen muss auf den Ergebnissen der Bedarfsanalyse beruhen. Nachfolgend werden diese Ergebnisse zusammengefasst und daraus Anforderungen an ein Anpassungswerkzeug hergeleitet:

- Die Bedarfsanalyse hat ergeben, dass Lernressourcen üblicherweise aus mehreren Dateien bestehen, die meist verschiedene Formate beinhalten.

  Anforderung 1: Ein Anpassungswerkzeug muss in der Lage sein, Format- und Dateigrenzen-übergreifend zu arbeiten. Nur so lässt sich sicherstellen, dass Nutzer mit einem Werkzeug alle (oder zumindest alle gängigen) Formate bearbeiten können und nicht jede Datei einzeln bearbeiten müssen.

- Möchte man existierende Lernressourcen an veränderte Einsatzkontexte anpassen, so sind oft mehrere Arten von Anpassungen notwendig.

  Anforderung 2: Das Werkzeug soll Unterstützung für alle Anpassungsprozesse anbieten. Dadurch wird ermöglicht, dass Anwender nicht mehrere Werkzeuge verwenden müssen, um die Gesamtanpassung durchzuführen.

- Es gibt Anpassungsarten die oft dazu führen, dass eine andere Anpassungsart im Anschluss ausgeführt werden muss.

  Anforderung 3: Das Werkzeug soll auch die Beziehungen zwischen verschiedenen Anpassungsarten berücksichtigen und dem Nutzer mitteilen.

- Verschiedene Anpassungsprozesse können in unterschiedlichem Maße durch Werkzeuge unterstützt werden. Das ist zum einen dadurch bedingt, dass aufgrund des derzeitigen technologischen Standes nicht alle Anpassungsprozesse in gleichem Maße automatisiert werden können. Zum anderen hat die Auswertung der Benutzerbefragung ergeben, dass es zwei Gruppen bezogen auf die Durchführung von Anpassungsprozessen gibt: die strukturierten, regelbasierten Anpassungsprozesse und die unstrukturierten, erfahrungsbasierten Anpassungsprozesse. Zur ersten Gruppe zählen vor allem gestalterische Anpassungsarten sowie technische Anpassungsarten. Die Durchführung der Anpassungsprozesse dieser Gruppe können gut (teil-)automatisiert unterstützt werden. Die zweite Prozessgruppe enthält dagegen hauptsächlich inhaltliche Anpassungsarten. Sie lassen sich deutlich schwerer automatisiert unterstützten.

  Anforderung 4: Das Werkzeug soll nicht nur automatisierte Unterstützung anbieten, sondern auch Hinweise und Hilfestellungen bei der Durchführung von Anpassungsprozessen geben. Die Hilfestellungen müssen auf dem Wissen von Prozessexperten beruhen, da diese wissen, wie die Prozesse durchzuführen sind.

Anforderung 5: Der Anwender muss bereits zu Beginn der Anpassung darauf hingewiesen werden, zu welchem Grad die verschiedenen Anpassungsprozesse unterstützt werden. So lässt sich vermeiden, dass erst nach mehreren Schritten klar wird, dass ein Anpassungsprozess nur manuell durchführbar ist.

- Im Interview haben sich die Prozessexperten gewünscht, dass ein Anpassungswerkzeug den Kenntnisstand der Nutzer berücksichtigen kann. Dadurch kann jeder Anwender die für ihn geeignete Form der Unterstützung erhalten.

  Anforderung 6: Das Anpassungswerkzeug soll sowohl einen Experten- als auch einen Laienmodus vorsehen.

Neben diesen sehr speziellen Anforderungen an ein Werkzeug zur Unterstützung von Anpassungsprozessen gibt es weitere, eher allgemeine Anforderungen, die auch für andere Werkzeuge zur Prozessunterstützung gelten.

2.4 Allgemeine Anforderungen an ein Prozessunterstützungswerkzeug

2.4.1 Ein Wizard als Unterstützungswerkzeug für Anpassungsprozesse

Das zu entwickelnde Anpassungswerkzeug soll Anwender bei der Durchführung der Anpassungsprozesse unterstützten. Dabei sollen insbesondere Personen berücksichtigt werden, die die Anpassungen zwar gelegentlich ausführen müssen, die aber keine Experten in der Durchführung der Anpassungsprozesse sind. Die meisten Autoren von E-Learning Materialien fallen in diese Personengruppe.

Da die Anpassungsprozesse sich aus Prozessschritten und atomaren Unterschritten zusammensetzen, muss ein Unterstützungswerkzeug alle diese Schritte benennen und Anwender Schritt für Schritt durch den gesamten Prozess führen.

Sogenannte *Wizards* sind eine in der Informatik weitverbreitete Lösung für die schrittweise Führung von Personen ohne Expertenkenntnisse durch einen Prozess. Laut [FWB06] sollen sie dann eingesetzt werden, wenn Nicht-Experten eine komplexe Aufgabe bestehend aus einer Reihe von Unteraufgaben durchführen müssen, die sie selten durchführen. Der Nicht-Experte kennt zwar das Ziel, das er erreichen will, weiß aber nicht unbedingt, welche Schritte ihn dahin führen. Die Schritte können geordnet werden und können in Abhängigkeit zu einander stehen. Beispielsweise kann ein Schritt von einer Entscheidung in einem vorigen Schritt abhängen. [FWB06]

Damit ergibt sich für Wizards folgende Definition:

> ***Wizards*** *sind Software-Programme, die Benutzer Schritt für Schritt durch komplexe Aufgaben führen, auch wenn die Benutzer keine speziellen Vorkenntnisse mitbringen.*

Der von [FWB06] beschriebene Anwendungsfall für Wizards trifft auch in dieser Arbeit zu: Das Unterstützungswerkzeug soll Benutzer schrittweise durch einen Anpassungsprozess führen, der sich aus einer Reihe von Prozessschritten zusammensetzt. Die Benutzer kennen dabei das Ziel des Anpassungsprozesses, aber sie sind nicht zwingend Prozessexperten. Das Werkzeug soll dem Benutzer anzeigen, welches Ziel erreicht

werden soll und welche Prozessschritte dazu nötig sind. Deswegen soll das zu erstellende Werkzeug in Form eines Wizards realisiert werden.

Ein Wizard setzt sich aus einer Abfolge von Dialogen zusammen, die den Benutzer Schritt für Schritt durch den Prozess führen [We01], wobei unter Dialogen Fenster in einem Computer Programm verstanden werden, die dem Nutzer die Auswahl von Optionen oder die Dateneingabe ermöglichen [SP05].

Bekannte Beispiele für Wizards, die im Deutschen auch als Assistenten bezeichnet werden, sind u. a. die Programme, die Anwender bei der Ausführung diverser Aufgaben unter den Microsoft Betriebssystemen unterstützen, wie das Einrichten eines neuen Druckers (Abbildung 7). Ein weiteres Beispiel sind die Wizards von OpenOffice, wie der Wizard zum Erstellen einer neuen Datenbank in OpenOffice Base (Abbildung 8). All diesen Wizards ist gemeinsam, dass sie Anwender schrittweise durch eine Aufgabe führen, wobei die Anwender keine Expertenkenntnisse benötigen.

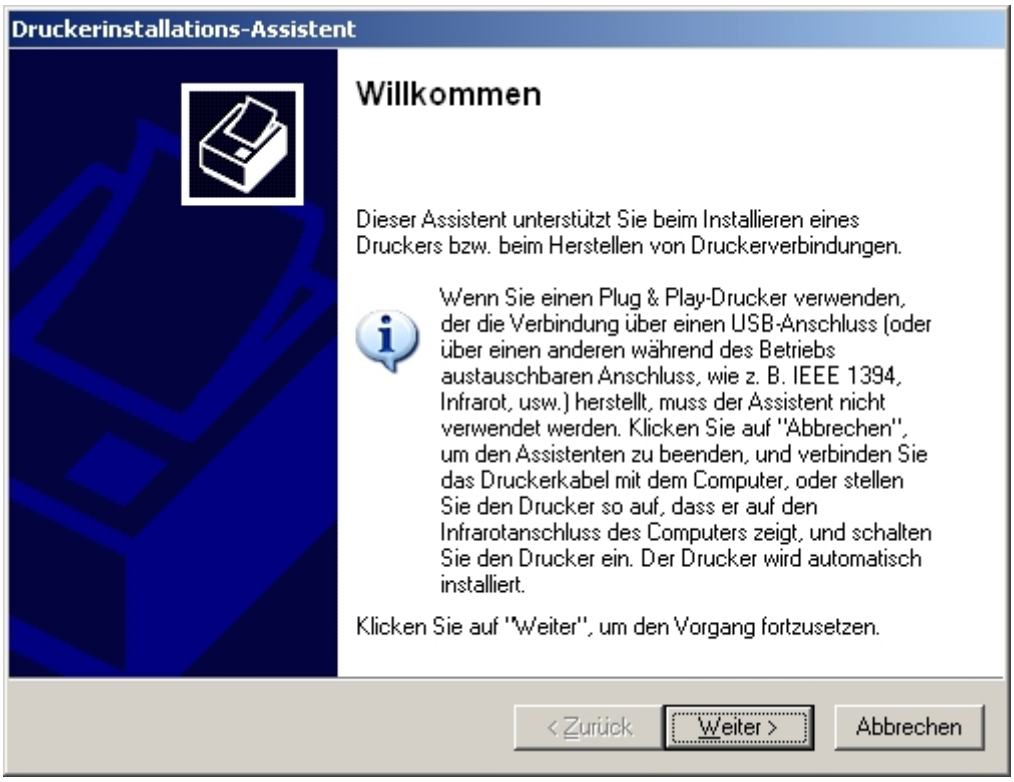


Abbildung 7: Wizard zum Einrichten eines neuen Druckers unter Microsoft XP.

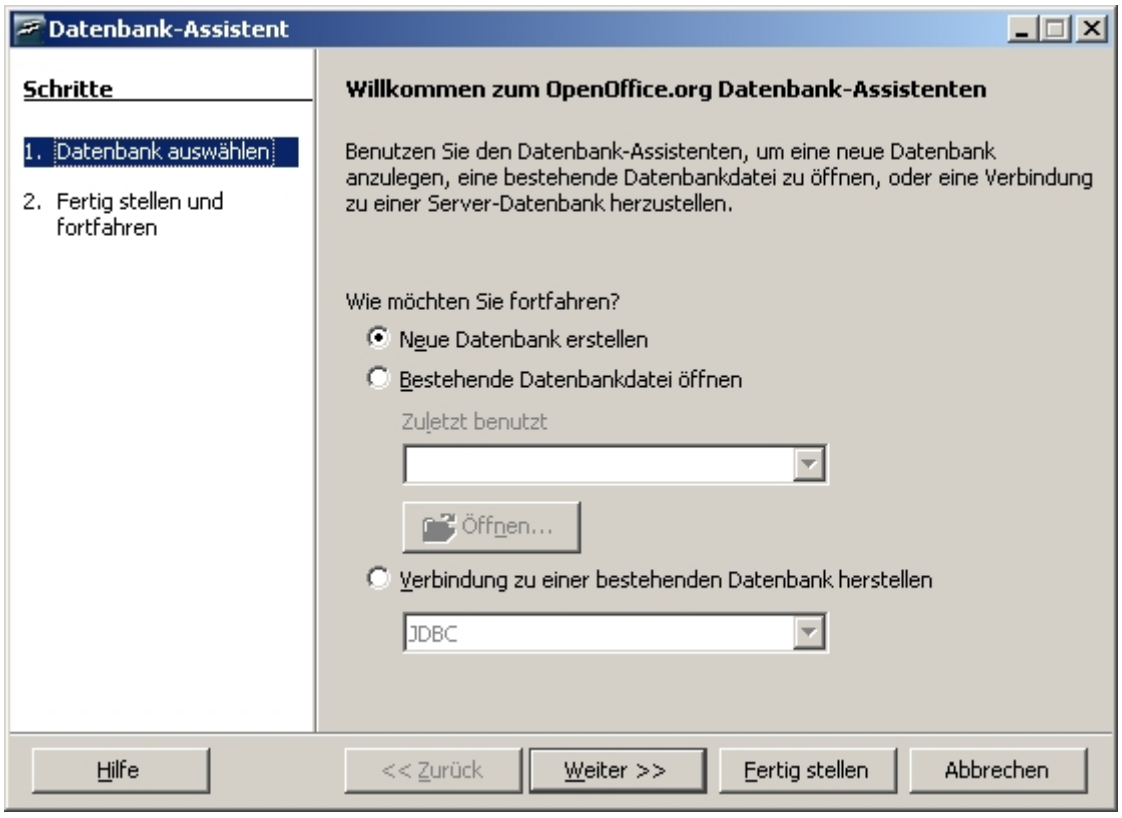


Abbildung 8: Wizard zur Erstellung von Datenbanken in OpenOffice Base.

Der für die Prozessunterstützung zu erstellende Wizard muss Anwender durch die Anpassungsprozesse führen. Dazu muss er zum einen den Prozessaufbau widerspiegeln. Das heißt, er muss alle in einem Anpassungsprozess benötigten Prozessschritte und atomaren Unterschritte, deren Reihenfolge sowie die Vorbedingungen für deren Ausführung berücksichtigen. Zum anderen müssen zu allen Bestandteilen eines Anpassungsprozesses Informationen zur Durchführung gegeben werden. Prozessexperten können am besten beschreiben, wie die Anpassungsprozesse aufgebaut sind und wie sie durchgeführt werden müssen.

Wie in Abschnitt 2.3 gefordert, soll der Wizard, wo immer das sinnvoll möglich ist, den Anwender entlasten, indem er Aufgaben automatisiert durchführt. Wo dies nicht sinnvoll möglich ist, soll der Wizard Hilfestellungen zur Prozessdurchführung geben. Damit Anwender nicht erst im Laufe der Ausführung des Wizards feststellen müssen, dass bestimmte Funktionen nicht automatisiert unterstützt werden, sind sie bereits direkt nach dem Starten des Wizards darüber zu informieren, in wie weit ein Prozess automatisiert unterstützt wird.

Da sich die Möglichkeiten der automatisierten Unterstützung ständig erweitern, sollte es möglich sein, den Wizard um weitere automatisierte Funktionalitäten zu erweitern, falls für eine bisher nicht automatisiert unterstützte Funktion entsprechende Unterstützung möglich wird.

2.4.2 Anforderungen an die Gestaltung eines Wizards

Es gibt eine Reihe von Arbeiten und Standards, die sich mit Anforderungen an die Gestaltung von Anwendungssoftware beschäftigen. So werden in der DIN EN ISO 9241 „Ergonomie der Mensch-System-Interaktion“ [Di95] von 1995 Anforderungen an eine ergonomische Gestaltung der Arbeit mit dem Computer gegeben. Teil 10 dieser Norm benennt „Grundsätze der Dialoggestaltung“. Er wurde 2006 durch den Teil 110 [Di06] ersetzt, der eine Aktualisierung der ursprünglichen Richtlinien darstellt.

Teil 110 nennt sieben Grundsätze zur Gestaltung von Dialogen. Da sich Wizards aus einer Folge von Dialogen zusammensetzen, werden die Grundsätze in der vorliegenden Arbeit auf Wizards übertragen. Daraus ergeben sich folgende sieben Anforderungen an die Gestaltung von Wizards:

- Aufgabenangemessenheit
  Ein Wizard ist aufgabenangemessen, wenn er Benutzer dabei unterstützt, ihre Arbeitsaufgaben zu erledigen.
- Selbstbeschreibungsfähigkeit
  Dem Benutzer muss jederzeit klar sein, welche Aufgabe er gerade mit dem Wizard durchführt, wie diese auszuführen ist und welche Möglichkeiten zur Unterstützung der Wizard bietet.
- Steuerbarkeit
  Der Benutzer soll entscheiden können, in welcher Geschwindigkeit er den Wizard durchlaufen will. Außerdem muss er jederzeit im Wizard vor und zurück gehen können.
- Erwartungskonformität
  Der Wizard soll in einer Art und Weise reagieren, die für den Benutzer vorhersehbar ist.
- Fehlertoleranz
  Der Wizard sollte Fehler, die oft gemacht werden, im Vorfeld vermeiden, beispielsweise indem in Zahleneingabefelder keine Buchstaben eingegeben werden können. Wurde doch ein Fehler gemacht, muss der Wizard eine Möglichkeit bieten, diesen zu erkennen und zu korrigieren.
- Individualisierbarkeit
  Der Wizard soll an die Bedürfnisse der Benutzer angepasst werden können.
- Lernförderlichkeit
  Der Wizard soll den Benutzer beim Erlernen der Nutzung anleiten.

In Kapitel 6 wird erläutert, wie mit dem in dieser Dissertation vorzustellenden Ansatz ein Werkzeug erstellt wurde, das den in Abschnitt 2.3 und Abschnitt 2.4 genannten Anforderungen gerecht wird.

2.5 Verwandte Arbeiten

Die Wiederverwendung existierender Lernressourcen wird derzeit in vielen Arbeiten behandelt. Ein Grund hierfür ist die aufwendige Erzeugung neuer Lernressourcen. Sie führt zu dem Wunsch, bereits existierende Lernressourcen wiederverwenden zu können. Im Folgenden wird ein Überblick über verwandte Arbeiten gegeben, die sich mit Wiederverwendung und / oder Anpassung existierender Lernressourcen beschäftigen.

Zum einen sind in diesem Zusammenhang die verschiedenen Arbeiten zur Definition, Gestaltung und Metadatenauszeichnung von Lernobjekten (Learning Objects) [Ie02, Wi02] als wiederverwendbare Lernressourcen zu nennen und zum anderen Arbeiten zur technischen Infrastruktur, die für eine Wiederverwendung notwendig sind, wie Lernobjektarchive [3, 13, 26, 11] oder Systeme zur automatischen Komposition von Lernressourcen zu größeren Einheiten [RFP05]. Diese Arbeiten bilden die Grundlage dafür, dass eine Wiederverwendung überhaupt erst möglich ist.

Wichtig bei der Wiederverwendung existierender Lernressourcen ist es, diese finden zu können [EH98]. Deswegen stellt Steinacker [St02] einen Ansatz vor, der es ermöglicht, Lernressourcen modular in einem zentralen Repository zu speichern und, dank geeigneter Metadatenauszeichnung, die in der jeweiligen Situation benötigten Ressourcen zu finden. Dabei können einzelne Teile genauso wie komplette Kurse gefunden werden. Diese können zum Lernen oder auch als Grundlage einer Wiederverwendung genutzt werden.

Auch die Arbeiten von Seeberg [Se02] und El Saddik [ES01] betonen die Wichtigkeit modular aufgebauter Lernressourcen für die Wiederverwendung. Beide beschäftigten sich mit adaptiven Lernressourcen, die sich an die Bedürfnisse der Nutzer anpassen. Dazu entwickelten sie das Multibook System. Dabei handelt es sich um ein Lehrsystem, das multimediale Ressourcen nutzt und dadurch die Möglichkeit bietet, sich an die Präferenzen der Nutzer anzupassen. Damit dies möglich ist, müssen die Lernressourcen modular und kontextfrei aufgebaut sein [Se04].

Alle bisher vorgestellten Arbeiten verlangen die Erstellung von modularen, aggregierbaren Lernressourcen als Grundlage für eine Wiederverwendung. In der Praxis findet diese aber selten statt. Beispiele für Arbeiten, die sich dieses Problems annehmen, finden sich bei Verbert et al. [VD07, VOD08], die mit dem ALOCoM Framework eine Möglichkeit vorstellen, Lernressourcen zu disaggregieren und die so entstandenen modularen Lernressourcen anschließend „on-the-fly“ zu neuen Einheiten zu aggregieren.

ALOCoM basiert auf einer Ontologie, die als generisches Inhaltsmodell verwendet wird. Basierend auf dieser Ontologie werden Lernressourcen in Komponenten zerlegt. Diese können dann zu neuen Lernressourcen zusammengesetzt werden. Derzeit unterstützt ALOCoM den Import der Präsentationsformate von PowerPoint und OpenOffice, sowie mit dem Reload Editor [22] erstellter Lernressourcen, die als IMS Content Packages [11] oder SCORM Content Packages [25] abgelegt werden. Ein Microsoft Word PlugIn erlaubt außerdem die Wiederverwendung von Wikipedia-Komponenten in Textdokumenten. Andere Formate können nicht bearbeitet werden. Auch eine Format- und Dateigrenzen-übergreifende Bearbeitung der Lernressourcen ist nicht möglich.

ALOCoM legt den Fokus auf Dekomposition und Aggregation von Lernressourcen. Eine Anpassung an veränderte Einsatzszenarien wird nicht berücksichtigt.

Ähnlich zu den Arbeiten von Verbert et al. sind die Arbeiten im SCORE Projekt [24, Ka+03, At+03]. Hier werden existierende Lernressourcen von Autoren mit Hilfe einer Ontologie, die die Domäne der Lernressourcen beschreibt, in sogenannte Lernatome zerlegt. Zwischen den einzelnen Lernatomen können zwei Arten von Beziehungen definiert werden *isSubtopicOf* und *isPrerequisiteFor*. Zusätzlich zur inhaltlichen Beschreibung der Lernatome durch die Domänenontologie werden ihnen sogenannte *ResourceTypes* zugeordnet, die ihre Funktion beschreiben, wie Example, Definition etc.

Die auf diese Art erzeugten Lernatome werden in einem Repository gespeichert. Bei der Erzeugung neuer Lernressourcen kann auf die bereits existierenden Lernressourcen zugegriffen werden. Die inhaltliche Domänenontologie hilft beim Finden geeigneter Themen. Didaktische Templates unterstützen beim Definieren der Kursstruktur. Anhand der Auszeichnung mit ResourceTypes können Lernatome vom richtigen ResourceType in die Kursstruktur eingefügt werden. SCORE setzt eine manuelle Bearbeitung der Lernressourcen durch den Autor voraus. Dies ist ein hoher Aufwand. Weiterhin wird außer Acht gelassen, dass das Zusammenfügen von Lernatomen aus verschiedenen Lernressourcen zu einem Bedarf an Anpassungen führen kann.

Bei SCORE wie auch bei vielen anderen Arbeiten zur Wiederverwendung wird vorausgesetzt, dass die Lernressourcen kontextfrei gestaltet werden. Didaktisch sinnvoll aufbereitete Lernressourcen müssen aber gerade den Kontext des Lerners berücksichtigen. Baumgartner [Ba04] spricht daher vom „Reusability of Objects and Instruction Paradox“, das sich aus dem Widerspruch ergibt, dass zur Wiederverwendung eine Kontextfreiheit wünschenswert ist, der Lernende aber an seinen Kontext gebunden ist.

Da nur wenige kontextfreie Lernressourcen existieren, müssen Lernressourcen vor einer Wiederverwendung erst an den neuen Einsatzkontext angepasst werden. Das ResourceCenter [HRS05, Hö05] stellt ein Konzept und dessen Umsetzung in einem Werkzeug dar, das dennoch eine Wiederverwendung von Lernressourcen ermöglicht. Dieser Ansatz basiert auf dem sogenannten *Authoring by Aggregation*, bei dem Lernressourcen durch Aggregation einzelner Teilkomponenten erstellt werden. Die mit dem ResourceCenter erzeugten Lernressourcen setzen sich aus einzeln erstellten Modulen zusammen. Diese Module können nach ihrer Erstellung in verschiedene Lernressourcen eingebunden werden. Das ResourceCenter ermöglicht die Erstellung sowie die spätere Aggregation von Lernmodulen. Hörmann weist auf die Wichtigkeit von Anpassungen hin, sieht aber im ResourceCenter nur eine Möglichkeit zur manuellen Anpassung vor.

Meyer [MRS07, Me08] greift die Arbeiten von Hörmann auf. Er stellt darüber hinaus in seinen Arbeiten eine Möglichkeit vor, die es erlaubt, auch Lernressourcen wiederzuverwenden, die nicht - wie beim ResourceCenter nötig - in modularer Form vorliegen. Er hat zu diesem Zweck ein Werkzeug entwickelt, das es Anwendern erlaubt, Lernressourcen je nach aktuellem Bedarf flexibel in Module unterschiedlicher Granularität aufzuteilen. Diese Module können als eigenständige SCORM-Einheiten exportiert werden. Die so entstandenen Module lassen sich durch Aggregation wieder zu größeren Einheiten und kompletten Kursen zusammenfügen. Das Werkzeug von Meyer setzt die beiden zum Repurposing benötigten Teilprozesse der Modularisierung und Aggregation um.

Obrenovic et al. [OSS04] stellen einen Ansatz zur Wiederverwendung von Multimedia-Dokumenten unter Verwendung eines XML-basierten Zwischenformates vor. In dieses Zwischenformat werden Multimedia-Formate transformiert und von diesem Zwischenformat wieder in die spezifischen Multimedia-Formate zurückverwandelt. Zur Modellierung von Repurposing wird eine Ontologie verwendet. Diese Arbeit stellt einen guten Ausgangspunkt für eine Wiederverwendung inklusive Anpassung dar, stellt aber derzeit kein Repurposing-Werkzeug zu deren Durchführung zur Verfügung.

Ansätze, die sich vertieft mit einer Anpassung beschäftigen, berücksichtigen nur ausgewählte Formate und Anpassungsdimensionen. SMIL (Synchronized Multimedia Integration Language) [27, Bu01, Bu03] ist z.B. ein vom W3C verabschiedeter XML-basierter Sprachstandard für zeitsynchronisierte, multimediale Materialien. Es fordert eine klare Trennung zwischen Inhalt und Struktur und ermöglicht die Beschreibung des Verhaltens verschiedener Multimedia-Objekte sowie des Layouts von Präsentationen. SMIL berücksichtigt die Anpassung an die Übertragungsbandbreite sowie die Anpassung an verschiedene Bildschirmauflösungen und Farbtiefen. Weitere Anpassungsarten sind aber nicht berücksichtigt. Außerdem ist es mit SMIL nur möglich, Materialien zu bearbeiten, die auch in SMIL erstellt wurden.

Es gibt auch eine Reihe kommerzieller Werkzeuge, die ausgewählte Anpassungen ermöglichen. Doch bei kommerziellen Werkzeugen wird meist genau eine Anpassungsart unterstützt und oftmals ist es auch nur möglich, ein Format zu bearbeiten. So erlaubt SYSTRAN [29] beispielsweise automatisierte Übersetzung für verschiedenste Sprachen und verschiedenste Inhaltsformate.

Ein kommerzielles Werkzeug, das diverse Anpassungsarten unterstützt ist TT Knowledge Force von Team Training Solutions [30]. Dieses Werkzeug erlaubt den Einsatz von Wissensressourcen in verschiedenen Einsatzgebieten, indem es beispielsweise die Übersetzung von Standardanweisungstexten unterstützt oder das Verwenden verschiedener Masken, um das Layout an die jeweiligen Vorgaben anzupassen. Allerdings handelt es sich bei TT Knowledge Force um eine Toolsuite, die nicht flexibel erweiterbar ist. Außerdem werden nur vorgegebene Formate berücksichtigt.

Keiner der vorgestellten Ansätze ermöglicht es, dass Autoren von Lernressourcen Format- und Anpassungsarten-übergreifend bei der Durchführung aller Anpassungsarten unterstützt werden. Die einzelnen Werkzeuge decken immer nur Teilaspekte ab, so dass ein Anwender mehrere Werkzeuge bedienen muss, möchte er alle eventuell notwendigen Anpassungsprozesse durchführen. Das verlangt eine hohe Kompetenz seitens der Anwender. Mit dem in der vorliegenden Arbeit vorzustellenden Ansatz zur Erstellung eines Unterstützungswerkzeuges wurde ein Anpassungswerkzeug erstellt, das in der Lage ist, Format- und Anpassungsarten-übergreifend zu unterstützen.

2.6 Zusammenfassung

In diesem Kapitel wurde das Anwendungsszenario der vorliegenden Arbeit vorgestellt: Die Prozesse, die notwendig sind, um bereits existierende Lernressourcen an neue Einsatzkontexte anzupassen. Zuerst wurde Anpassung als Bestandteil des Repurposing betrachtet. Ausgehend von einer Bedarfsanalyse wurden dann die zur Durchführung der

Anpassung notwendigen Prozesse detailliert vorgestellt. Dabei hat sich gezeigt, dass für die betrachtete Zielgruppe fünfzehn Anpassungsarten existieren, die sich drei Kategorien von Anpassungsarten zuordnen lassen: gestalterische Anpassungsarten, inhaltliche Anpassungsarten und technische Anpassungsarten.

Zur Ermittlung der Anpassungsprozesse, die von Experten durchgeführt werden, bestand ein Teil der Bedarfsanalyse aus einer Benutzerumfrage. Ein wichtiges Ergebnis dabei war, dass es zwei Gruppen bezüglich der Durchführung von Anpassungsprozessen gibt: strukturierte, regelbasierte Anpassungsprozesse, die von allen Durchführenden in ähnlicher Weise vorgenommen werden, und unstrukturierte, erfahrungsbasierte Anpassungsprozesse, die teilweise recht unterschiedlich ausgeführt werden. Im Hinblick auf eine Werkzeugunterstützung lässt sich sagen, dass die Gruppe der strukturierten, regelbasierten Anpassungsprozesse sich zumeist gut automatisiert unterstützen lassen, während es bei den unstrukturierten, erfahrungsbasierten Anpassungsprozessen oft schwer ist, mehr als nur eine teilweise Unterstützung zu geben. Wichtig ist jedoch, dass für alle Anpassungsprozesse Hilfestellungen, wie Tipps und Tricks, zur Verfügung gestellt werden können.

Anpassungsprozesse werden oft auch von Laien durchgeführt. Damit diese ein gutes Resultat erzielen können, brauchen sie Unterstützung. Basierend auf der Benutzerumfrage wurden Anforderungen an ein Werkzeug zur Unterstützung der Anpassungsprozesse ermittelt. Dabei wurden sowohl anpassungsprozessspezifische Anforderungen als auch allgemeine Anforderungen berücksichtigt. Abschließend wurden verwandte Arbeiten untersucht und in Bezug zur vorliegenden Arbeit gestellt.

Um ein Werkzeug entwickeln zu können, das den in diesem Kapitel genannten Anforderungen gerecht wird, ist das Wissen von Prozessexperten notwendig. Deswegen wird im folgenden Kapitel erläutert, wie Prozessexperten in die Erstellung eines Werkzeuges zur Unterstützung von Anpassungsprozessen eingebunden werden können.

3 Konzept einer Pattern-basierten Prozessbeschreibung und -unterstützung

Wie bereits erläutert, ist eines der Hauptziele dieser Arbeit, Prozessexperten stärker in die Entwicklung eines Werkzeuges zur Unterstützung von Anpassungsprozessen einzubinden. Dieses Werkzeug soll insbesondere Personen, die keine Prozessexperten sind, bei der Prozessdurchführung unterstützen. Ein Merkmal der in Kapitel 2 vorgestellten Anpassungsprozesse ist, dass ihre Durchführung auf Spezialwissen basiert. Möchte man Laien bei der Durchführung dieser Prozesse unterstützen, so müssen sie das notwendige Wissen zur Verfügung gestellt bekommen. Daher muss ein Werkzeug, das - wie im zweiten Hauptziel der Arbeit gefordert - die Prozessunterstützung verbessert, auf dem Wissen von Prozessexperten beruhen. Dabei ist sicherzustellen, dass das Werkzeug das Verständnis der Prozessexperten und nicht das der Software-Designer widerspiegelt.

Es gilt also, ein Konzept zu entwickeln, das es Experten in der Durchführung der Anpassungsprozesse erlaubt, ihr Prozesswissen zu beschreiben und dadurch für die Entwicklung zur Verfügung zu stellen. Da Prozessexperten oftmals keine Spezialisten in den verschiedenen Möglichkeiten zur Prozessbeschreibung und -modellierung sind, gilt es, einen Formalismus zum Erfassen des Wissens und zum Beschreiben der Anpassungsprozesse zu finden, der leicht verständlich und schnell erlernbar ist. Weiterhin sollte der Formalismus das Wissen so sammeln, dass es auch für andere Personen wertvoll ist und als Unterstützung bei der Prozessdurchführung verwendet werden kann. Basierend auf diesem dokumentierten Wissen ist ein Werkzeug zu erstellen, das Laien bei der Durchführung der beschriebenen Anpassungsprozesse unterstützt.

Wie van der Aalst und van Hee [AH02] betonen, ist die frühe und stetige Einbindung der Personen, die den zu unterstützenden Prozess kennen, in den Entwicklungsprozess sehr wichtig. Die im Rahmen der Anforderungsanalyse für Anpassungsprozesse in Kapitel 2 geführten Gespräche mit Prozessexperten haben gezeigt, dass das oft nicht der Fall ist und dass existierende Software in vielen Fällen nicht dem Prozessverständnis der Personen entspricht, die diese Prozesse täglich ausführen.

Weiterhin hilft die frühzeitige Erstellung von Prototypen Prozessexperten dabei, eine Vorstellung von der späteren Software zu entwickeln. Doch bei vielen existierenden Ansätzen, die Prototypen einsetzen, erstellen Entwickler die Prototypen, basierend auf Gesprächen mit Prozessexperten. Hierbei kann es zu Missverständnissen kommen oder Wissen kann verloren gehen. Auch bei der Veranschaulichung des späteren Produktes durch Prototypen können Missverständnissen auftreten. Ein durch den Prozessexperten selbst entwickelter Prototyp könnte dem entgegen wirken, da der Prozessexperte kontrollieren könnte, dass der Prototyp exakt den gewünschten Prozess widerspiegelt. Deswegen soll der in dieser Arbeit zu entwickelnde Ansatz darauf beruhen, den Prozessexperten selbst zu ermöglichen, einen derartigen prototypischen Wizard zu erstellen.

Im vorigen Kapitel wurden Anpassungsprozesse vorgestellt, die durch das zu entwickelnde Konzept unterstützt werden sollen. In diesem Kapitel wird erläutert, wie ein Werkzeug erstellt werden kann, das auf Expertenwissen beruht und bei der Durchführung der Anpassungsprozesse unterstützt. Dazu werden zuerst verschiedene Vorgehensweisen zur Erstellung von Software im Allgemeinen betrachtet. Anschließend wird

ein Konzept vorgestellt, das es ermöglicht, basierend auf dem Wissen von Prozessexperten ein Werkzeug zur Unterstützung von Anpassungsprozessen zu erstellen, das den bereits genannten Anforderungen gerecht wird.

3.1 Vorgehensmodelle zur Softwareerstellung

Im Software Engineering gibt es bereits eine Vielzahl von Modellen, die sich mit Vorgehensweisen bei der Erstellung von Software beschäftigen. Nachfolgend wird ein Überblick über einige dieser Vorgehensmodelle gegeben.

3.1.1 Wasserfallmodell

Ein klassisches Vorgehen bei der Erstellung von Software ist das Wasserfallmodell [Ro87]. Es teilt die Herstellung von Software in verschiedene aufeinanderfolgende Phasen ein: Zuerst werden die Anforderungen an eine neue Software spezifiziert. Dabei werden software- wie systemseitige Anforderungen erhoben. Die Anforderungen werden analysiert. Danach wird in der Design-Phase die neue Software entworfen. Diese wird implementiert, getestet und ggf. verbessert oder verfeinert. Dann wird sie ausgeliefert und in Betrieb genommen (siehe Abbildung 9).

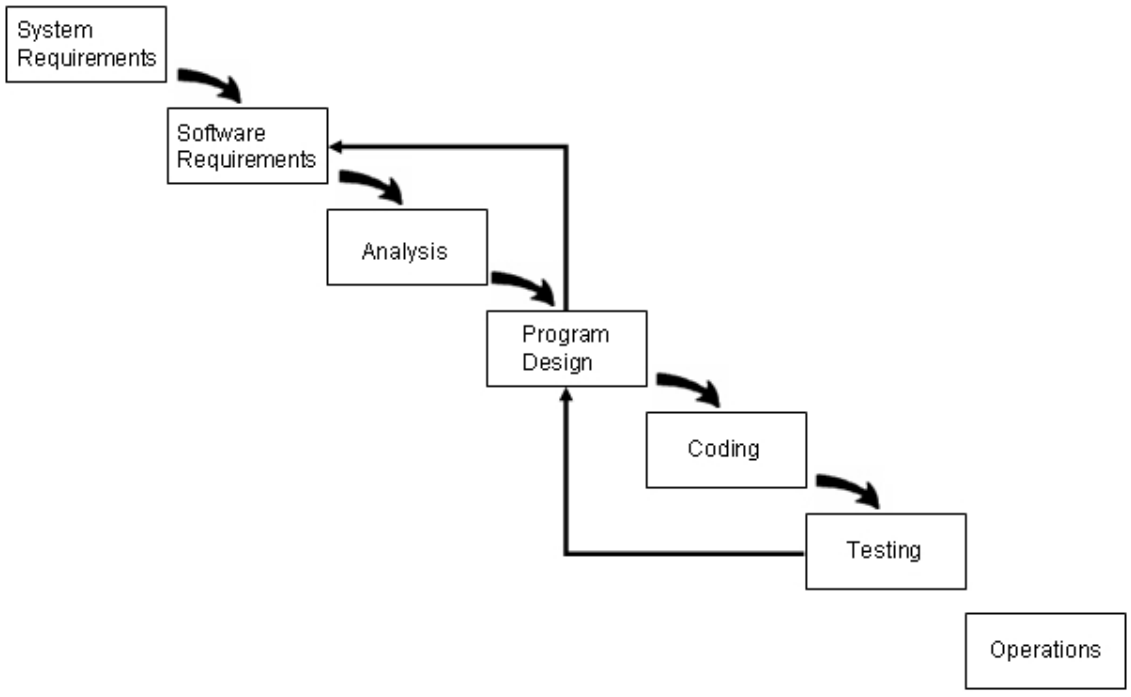


Abbildung 9: Wasserfallmodell nach [Ro87, S. 330].

Das Wasserfallmodell wird zwar teilweise immer noch eingesetzt, doch seine Verwendung ist umstritten, da es einige Probleme aufweist: Es ist sehr starr. Eine Abgrenzung der einzelnen Phasen, wie sie in diesem Modell vorgenommen wird, lässt sich in der Praxis oft nicht einhalten [RP02, Ba98].

Nachteilig an diesem Modell ist weiterhin, dass spätere Änderungen des Programms oft nur schwer möglich sind. Erschwerend kommt hinzu, dass Änderungsanforderungen oft sehr spät auffallen, da erst zu einem fortgeschrittenen Zeitpunkt im Projekt mit der Implementierung angefangen wird [La06].

3.1.2 Spiralmodell

Eine Weiterentwicklung des Wasserfallmodells stellt das Spiralmodell [Bo88] von Barry Boehm dar. Es betrachtet den Software-Entwicklungsprozess als eine Spirale, die vier Quadranten durchläuft (vergleiche Abbildung 10):

- Festlegen von Zielen, Alternativen und Bedingungen (Quadrant links oben)
- Evaluation von Alternativen, Erkennen und Beseitigen von Risiken (Quadrant rechts oben)
- Entwicklung und Verifikation des Produktes der nächsten Generation (Quadrant rechts unten)
- Planung der nächsten Phasen (Quadrant links unten)

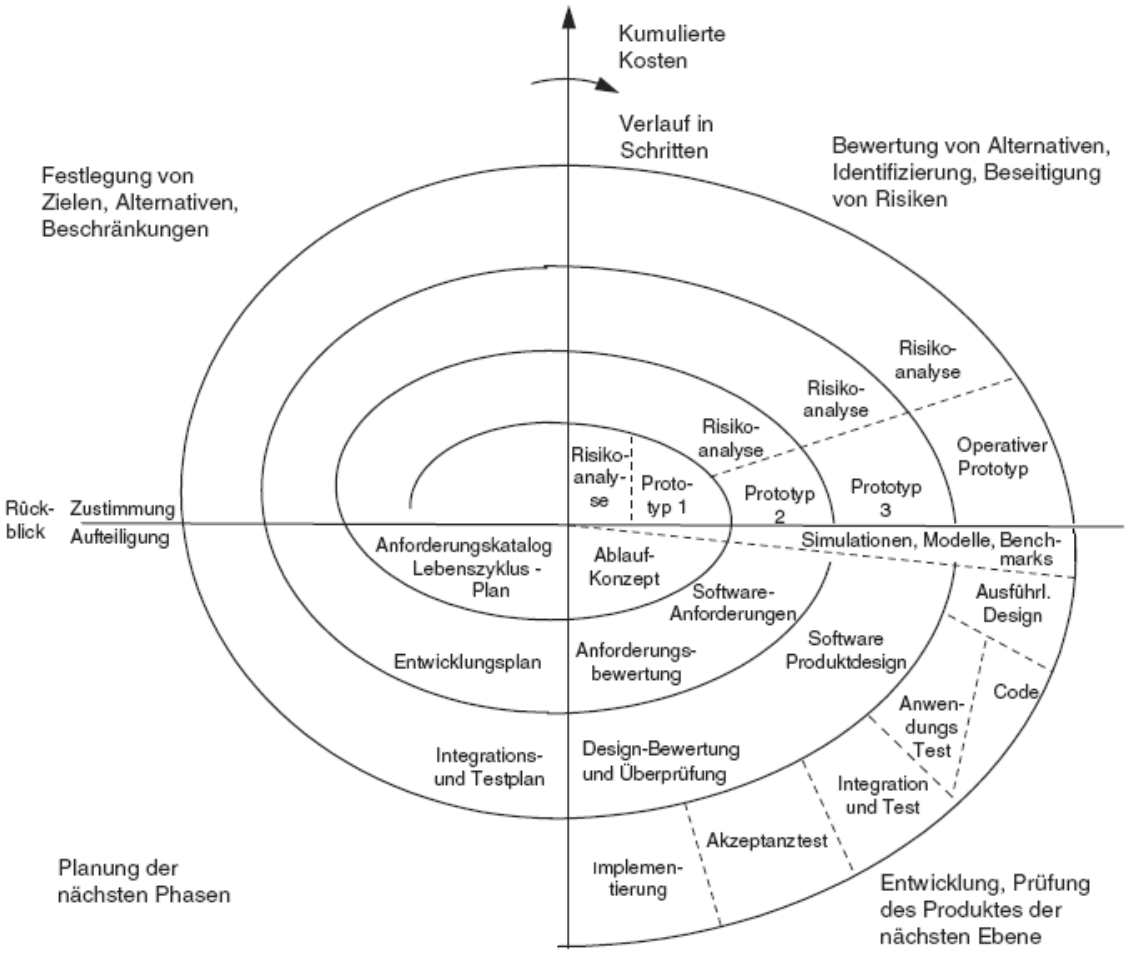


Abbildung 10: Spiralmodell in Anlehnung an [Bo88, S.64].

Die einzelnen Quadranten werden immer wieder durchlaufen, was sich in Abbildung 10 erkennen lässt. Vorteil beim Spiralmodell ist, dass es sehr flexibel ist. Außerdem zielt es nicht auf die Ablösung älterer Vorgehensmodelle ab, sondern lässt die Möglichkeit offen, diese einzubinden. Doch der Managementaufwand des Spiralmodells ist sehr hoch. Somit eignet es sich zwar aufgrund seiner Flexibilität und Offenheit gegenüber anderen Modellen gut für die Entwicklung großer Softwaresysteme, aber für kleinere und mittlere Projekte ist es weniger geeignet. Außerdem wird es oft als sehr kompliziert empfunden [Mc00].

3.1.3 V-Modell

Eine andere Weiterentwicklung des Wasserfallmodells findet sich im V-Modell [32]. Dieses ergänzt die ersten Phasen der Software-Entwicklung um Teststufen, die der Prüfung der jeweiligen Spezifikationsebene dienen. Die Idee hierzu beruht ebenfalls auf einer Arbeit von Barry Boehm [Bo79].

Das V-Modell gliedert den Entwicklungsprozess, ähnlich wie das Wasserfallmodell, in Phasen. Ein großer Vorteil ist allerdings, dass die Zusammengehörigkeit von Systemerstellung, Qualitätssicherung, Konfigurationsmanagement und Projektmanagement betont wird. Außerdem lässt es sich an projektspezifische Anforderungen anpassen [Ba98]. Doch das V-Modell weist auch Nachteile [Ba98] auf: Das Modell eignet sich zur Entwicklung großer Systeme, für kleine und mittlere Systeme bringt es zu viel bürokratischen Aufwand mit sich. Außerdem ist es recht starr, was die verwendeten Methoden angeht.

3.1.4 Prototyping

Die Beobachtung, dass Software-Entwicklung oft aufgrund mangelnder Kommunikation zwischen Auftraggebern, Anwendern und Entwicklern scheitert, führte zur Entwicklung des Prototyping. Ziel dieses Vorgehensmodells ist es, relativ früh im Projekt funktionsfähige Prototypen zur Verfügung zu stellen, an denen Teilfunktionalitäten des späteren Systems getestet werden können. Vorteil hierbei ist, dass diese Prototypen schnell und kostengünstig erstellt werden können.

In [CS89] wird der Begriff des Software-Prototyps folgendermaßen definiert:

> *"A **software prototype** is a dynamic visual model providing a communication tool for customer and developer that is far more effective than either narrative prose or static visual models for portraying functionality. It has been described as:*
>
> - *functional after a minimal amount of effort*
> - *a means for providing users of a proposed application with a physical representation of key parts of the system before system implementation*
> - *flexible modifications require minimal effort*
> - *not necessarily representative of a complete system.*"

Es lassen sich drei Arten von Prototyping unterscheiden, die sich gegenseitig nicht ausschließen [Fl84]:

- Das *explorative Prototyping* dient der Anforderungsbestimmung. Es deckt die Anwendersicht ab. Hier werden dem Benutzer Teile der Funktionalität, sowie die zugehörigen Benutzerschnittstellen in einem Prototyp zur Verfügung gestellt. Der Benutzer kann den Prototyp testen und seine Anforderungen an das System formulieren. Der Prototyp erleichtert das Verständnis des späteren Systems und unterstützt bei der Kommunikation mit dem Software Designer.
- Das *experimentelle Prototyping* hilft bei der konstruktiven Umsetzung des Systems. Hierbei können verschiedene Alternativen getestet werden. So kann die beste Lösung gefunden werden.
- Beim *evolutionären Prototyping* ist der Prototyp Ausgangspunkt für das spätere Produkt. Hier wird kontinuierlich an der Erweiterung des Prototyps, der sich üblicherweise aus mehreren Teilen zusammensetzt, gearbeitet, bis schließlich das Endprodukt erreicht ist. Die Entwicklung erfolgt kontinuierlich in Zyklen. Dadurch wird der Beobachtung Rechnung getragen, dass sich die Anforderungen an ein System meistens im Laufe der Zeit verändern.

Beim explorativen und beim experimentellen Prototyping werden „Wegwerf"-Prototypen erzeugt. Wichtig für das Endprodukt sind die Erfahrungen, die mit den Prototypen gemacht werden. Die Prototypen selbst werden üblicherweise im Endprodukt nicht weiterverwendet. Beim evolutionären Prototyping steht dagegen genau diese Weiterverwendung im Mittelpunkt.

Ein Vorteil des Prototyping ist, dass es z.B. durch die Verwendung experimenteller Prototypen ermöglicht, das Entwicklungsrisiko zu reduzieren [Ba98]. Durch Einbindung von Benutzern in Tests mit den Prototypen lässt sich außerdem eine Benutzeroberfläche erreichen, die den Anforderungen der Nutzer in hohem Maße gerecht wird [RP02]. Durch geeignete Werkzeuge, die sogenannten CASE (Computer Aided Software Engineering) Werkzeuge, lassen sich die Prototypen schnell erstellen [LD02]. Weiterhin lässt sich Prototyping in andere Vorgehensmodelle integrieren [Ba98].

Nachteilig beim Prototyping ist der höhere Entwicklungsaufwand, der entsteht, wenn Prototypen weggeworfen werden [Ba98]. Dadurch besteht auch die Gefahr, dass ein „Wegwerf"-Prototyp nicht beseitigt wird, was die Qualität des Endproduktes negativ beeinflussen kann [Ba98]. Eine weitere Gefahr liegt darin, dass die Prototypen oft als Ersatz für Dokumentation angesehen werden und somit die wichtige Arbeit der Dokumentation vernachlässigt wird [Ba98].

3.1.5 Rational Unified Process

Mit dem objektorientierten Programmierparadigma entwickelten sich auch entsprechende Vorgehensmodelle im Software Engineering. Eines dieser Modelle ist der Rational Unified Process (RUP) [Kr99]. Er setzt sich zusammen aus einem Vorgehensmodell und einem zugehörigen kommerziellen Produkt, das von der Firma Rational (mittlerweile Teil des IBM Konzerns) entwickelt wurde. Der RUP ist im Grunde genommen ein Metamodell verschiedener Vorgehensweisen. Er stellt ein Framework verschiedener Prozesselemente dar, von denen sich ein Entwicklungsteam die Elemente auswählen kann, die den jeweiligen Gegebenheiten entsprechen.

RUP wurde mit gängigen Software-Design-Techniken entwickelt. Insbesondere hat er ein zugrundeliegendes objektorientiertes Modell, das in UML notiert ist. Der RUP gliedert sich in zwei Dimensionen:

- Die horizontale Dimension stellt das Projekt auf der Zeitachse dar.
- Die vertikale Dimension entspricht den Software Engineering Aktivitäten in einem Projekt.

Die horizontale Achse unterteilt sich in vier Phasen. Diese werden in ein oder mehr Iterationen durchlaufen. Sie bilden die dynamischen Aspekte des RUP ab. Auf der vertikalen Achse finden sich die sogenannten Workflows eines Software Projektes. Sie beinhalten Aktivitäten, Disziplinen, Artefakte und Rollen des Prozesses (siehe Abbildung 11). Die Workflows erstrecken sich über alle Phasen. Sie werden in RUP zwar getrennt betrachtet, finden aber in der Realität gleichzeitig statt. Oft brauchen einzelne Workflows Ergebnisse aus anderen Workflows oder beeinflussen diese.

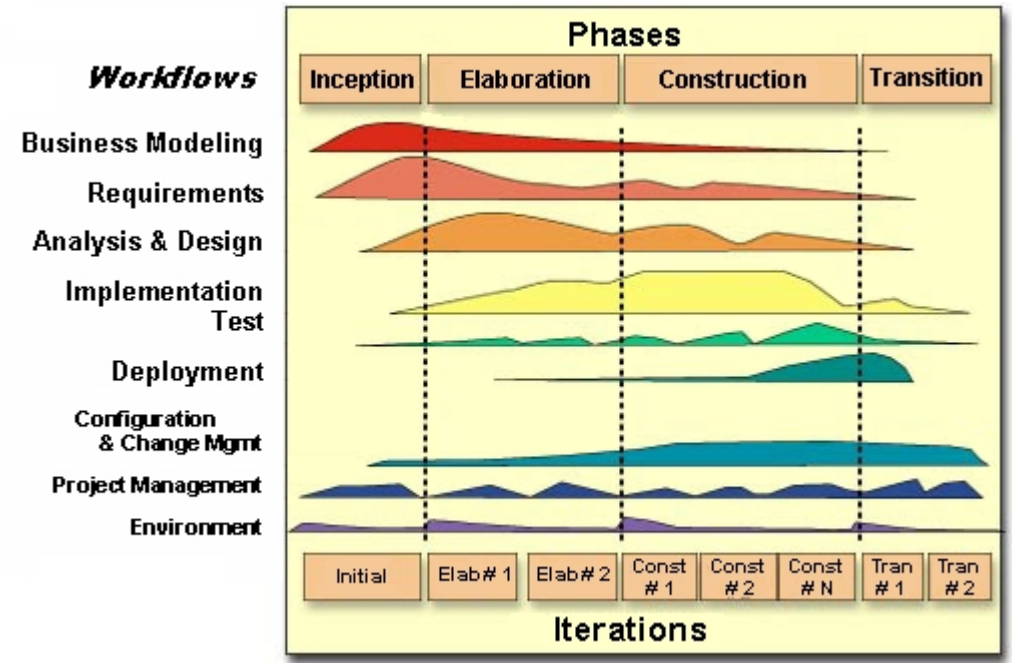


Abbildung 11: Die zwei Dimensionen von RUP [12].

Vorteilhaft beim RUP ist zum einen, dass er auf dem Wissen erfahrener Software- und Prozessentwickler beruht, wie Grady Booch, James Rumbaugh oder Ivar Jacobson [Ve00]. Außerdem verringert er Risiken und Fehler bei der Erstellung von Software [Ve00]. Allerdings ist der Prozess sehr kompliziert und somit schwer anwendbar [He03]. Des Weiteren werden die verschiedenen in einem Projekt auftretenden Rollen nicht ausreichend berücksichtigt [He03].

3.1.6 Agile Methoden

Agile Methoden wie Extreme Programming [Be99], Scrum [RJ00] oder der Agile Unified Process [1] zielen darauf ab, die Software-Entwicklung flexibler zu gestalten. Dazu

wird der bürokratische Aufwand verringert. Die Ziele und technischen Probleme der Software-Entwicklung stehen im Mittelpunkt. Ein wichtiger Bestandteil ist die enge Zusammenarbeit zwischen Entwicklern und Prozessexperten. Damit wollen sich die Vertreter der agilen Methoden bewusst von den als schwergewichtig empfundenen klassischen Modellen, wie Wasserfallmodell, V-Modell oder auch RUP, abgrenzen.

Agile Software-Entwicklung beruht auf Werten und Prinzipien, die in Methoden umgesetzt werden. Die agilen Werte sind im sogenannten „Manifesto for Agile Software Development“ festgehalten, das von 17 namhaften Software-Entwicklern wie Kent Beck, Ward Cunningham oder Martin Fowler unterzeichnet wurde. Die hier aufgeführten Werte [Be+01] stellen:

- Individuen und Interaktionen über Prozesse und Werkzeuge,
- funktionierende Software über umfangreiche Dokumentation,
- Zusammenarbeit mit dem Kunden über Vertragsverhandlungen und
- auf Änderungen antworten über einem Plan folgen.

Neben den Grundwerten finden sich im Manifest auch 12 Prinzipien, denen agile Software-Entwicklung gerecht werden soll. Hier finden sich unter anderem Kundenzufriedenheit als höchste Priorität, die Anerkennung und Akzeptanz sich verändernder Anforderungen, sowie die Betonung der Wichtigkeit von „face-to-face“ Kommunikation.

Auf den Werten und Prinzipien setzen eine Reihe von agilen Methoden wie Extreme Programming (XP) [Be99], Dynamic System Development Method (DSDM) [St97] oder Scrum [RJ00] auf. Die einzelnen agilen Methoden verfolgen zwar alle die gleichen Ideen und Prinzipien, setzen diese aber teilweise recht unterschiedlich um. Dennoch lässt sich zwischen allen Prozessen eine Verwandtschaft feststellen [Co02].

Nachteilig bei der Agilen Software Entwicklung ist zum einen, dass es schwer möglich ist, mit diesem Vorgehen in verteilten Teams zu entwickeln: Agile Prozesse bauen auf der Annahme auf, dass Kunde und Entwickler in räumlicher Nähe angesiedelt sind, so dass sich Treffen zwischen allen Beteiligten schnell anberaumen lassen. Auch die Entwicklung in großen Teams ist unzureichend berücksichtigt, da bei diesen Teams die Kommunikation zwischen allen Beteiligten nur noch schwer zu leisten ist. [TFR02]

Zum anderen ist die Wiederverwendung der Software oft nicht möglich, da agile Prozesse darauf abzielen Software zu entwickeln, die ein spezifisches Problem löst. [TFR02] Außerdem weisen agile Methoden oft das Problem auf, dass sie in Festpreisprojekten schwer einsetzbar sind, da sich der Aufwand im Vorhinein schlecht abschätzen lässt [Ab+02].

3.1.7 Zusammenfassung

Anhand eines Prototyps lässt sich eine Vorstellung der späteren Software entwickeln. Doch insbesondere die herkömmlichen Vorgehensweisen zur Software-Erstellung weisen das Problem auf, dass es eine Weile dauert, bis Prozessexperten oder potentielle Benutzer das erste Mal einen Prototyp testen können. Erschwerend kommt hinzu, dass

es nur anhand von Papier-Mock-Ups, Modellen und Spezifikationen oft schwer ist, sich vorzustellen, wie die spätere Software aussehen soll.

Das Prototyping wie auch die agilen Vorgehensweisen sind Ansätze, die dem entgegenwirken. Hier bekommen Prozessexperten oder potentielle Benutzer früh die Gelegenheit, eine Vorstellung vom späteren Produkt zu entwickeln. Doch es ist ihnen nicht möglich, selbst Prototypen zu erstellen, die als Ausgangspunkt für die Entwicklung dienen können.

Die bisher vorgestellten Ansätze binden Prozessexperten in ganz unterschiedlicher Art und Weise ein. Doch keiner der Ansätze erlaubt den Prozessexperten selbst, einen Prototyp zu gestalten, der ihr Prozessverständnis widerspiegelt. Außerdem werden die Prozessexperten nicht durchgängig in die Software-Erstellung eingebunden. Somit besteht selbst bei den Ansätzen, die Prozessexperten verhältnismäßig stark in die Entwicklung einbinden, die Gefahr, dass die Software nicht genau das Prozessverständnis der Prozessexperten wiedergibt, sondern auf dem Prozessverständnis von Software-Designern und -Entwicklern beruht.

Der in der vorliegenden Arbeit beschriebene Ansatz zielt darauf ab, Experten in der Durchführung von Anpassungsprozessen durchgängig in die Entwicklung von Software zur Unterstützung dieser Prozesse einzubinden. Dadurch lässt sich erreichen, dass die Software das Wissen der Experten beinhaltet. Um das zu ermöglichen, sollen die Prozessexperten, die hervorragendes Wissen zu den von ihnen regelmäßig durchgeführten Anpassungsprozessen haben, ihr Wissen beschreiben. Da es sich gezeigt hat, dass Prototypen dabei helfen, ein Verständnis der späteren Software zu gewinnen [CS89], sollen auch in dieser Arbeit Prototypen eingesetzt werden. Diese sollen auf den Prozessbeschreibungen beruhen und von den Prozessexperten selbst erstellt werden, damit diese direkt überprüfen können, ob der Prototyp ihrem Prozessverständnis entspricht.

Wichtig ist, dass der in dieser Arbeit beschriebene Ansatz nicht der Ablösung bereits existierender Vorgehensmodelle zur Software-Entwicklung dient. Es soll auch kein neues Vorgehensmodell vorgestellt werden. Vielmehr ist der vorliegende Ansatz als Ergänzung zu bestehenden Modellen gedacht, um gezielt Software zur besseren Unterstützung von Anpassungsprozessen zu entwickeln.

Mit dem hier vorgestellten Verfahren kann ein Prototyp für ein Prozessunterstützungswerkzeug erstellt werden. Dieser Prototyp beinhaltet das Wissen von Prozessexperten und ist somit eine wertvolle Grundlage für die weitere Entwicklung von Software zur besseren Unterstützung von Anpassungsprozessen.

3.2 Anforderungen an Prozessbeschreibungen

Um Experten in der Durchführung von Anpassungsprozessen direkter in die Erstellung von Werkzeugen zur Unterstützung dieser Prozesse einbinden zu können. müssen die Prozessexperten die von ihnen durchgeführten Anpassungsprozesse geeignet beschreiben können. In diesem Abschnitt werden Anforderungen formuliert, denen ein hierfür geeigneter Prozessbeschreibungsformalismus gerecht werden muss.

Um sowohl bei Prozessexperten als auch bei Software-Designern und -Entwicklern, Akzeptanz zu finden, muss ein Formalismus zur Prozessbeschreibung folgenden Kriterien genügen:

- Er muss ohne größeren Aufwand erlernbar sein.
- Die resultierenden Prozessbeschreibungen müssen anderen Personen Informationen über die Ausführung der Anpassungsprozesse zur Verfügung stellen, so dass diese in der Lage sind die Anpassungsprozesse durchzuführen, auch wenn sie keine Prozessexperten sind.
- Die resultierenden Prozessbeschreibungen müssen Software-Designern und -Entwicklern erlauben, ihre gewohnte Arbeitsweise beizubehalten.

Wie im vorigen Abschnitt dargestellt, besteht bei vielen Vorgehensmodellen zur Software-Erstellung das Problem, dass die Prozessexperten erst nach einiger Zeit das erste Mal einen Prototyp zu sehen bekommen. Deswegen soll in der vorliegenden Arbeit ein Verfahren entwickelt werden, das es Prozessexperten selbst erlaubt, basierend auf den von ihnen angelegten Prozessbeschreibungen einen Prototyp zu erstellen

Daraus ergeben sich weitere Anforderungen an den Formalismus zur Beschreibung der Anpassungsprozesse:

- Er muss alle zur Erzeugung eines Prototyps nötigen Informationen beinhalten.
- Er muss in einer Form vorliegen, die eine automatisierte Prototyp-Erstellung basierend auf den enthaltenen Informationen erlaubt (vergleiche Abschnitt 3.3).

Prozesse lassen sich aus verschiedenen Blickwinkeln, den sogenannten Perspektiven, betrachten und dementsprechend unterschiedlich beschreiben. Dabei werden in der Literatur vier Perspektiven unterschieden (vgl. [Aa+02] und [JB96]):

Die *Kontrollfluss-Perspektive* beschreibt Aktivitäten und deren Ausführungsreihenfolge. Sie wird auch als Prozess-Perspektive bezeichnet.

Die *Daten-Perspektive* beschäftigt sich mit den innerhalb des Kontrollflusses übergebenen Geschäfts- und Prozessdaten.

Die *Ressourcen-Perspektive* verbindet die Geschäftsstruktur mit dem Prozess, in dem sie den Aktivitäten Ressourcen in Form von Menschen und Betriebsmitteln für die Ausführung zuordnet.

Die *operationale Perspektive* ordnet den Aktivitäten Systeme und Anwendungen zu.

Die Beschreibungen von Anpassungsprozessen, die mit dem Konzept der vorliegenden Arbeit erstellt werden, geben die Kontrollfluss-Perspektive dieser Prozesse wieder. Diese Perspektive ist für den Ablauf der Tätigkeiten in einem Prozess dominant [AHD05], da sie wichtige Einsichten in den Prozess vermittelt [Aa+02]. Außerdem ist es diese Perspektive, die beschreibt, aus welchen Tätigkeiten sich ein Prozess zusammensetzt und in welcher Reihenfolge diese durchgeführt werden. Der Prozessexperte kann den Kontrollfluss detailliert beschreiben, da dieser den Anpassungsprozess so abbildet, wie der Prozessexperte ihn durchführt.

Die übrigen Perspektiven werden nicht betrachtet. Das hat folgende Gründe: Zum einen setzen alle anderen Perspektiven eine Betrachtung der Kontrollfluss-Perspektive voraus [Aa+02], zum anderen beschreiben sie Sichten, die der Prozessexperte oft nur schwer beurteilen kann. Deswegen soll es einer Person überlassen werden, diese Sichten zu betrachten, die das besser leisten kann.

3.3 Anforderungen an eine Prototyp-Erstellung

Die Prozessbeschreibungen sollen als Grundlage für die Erstellung eines Prototyps dienen, der in Form eines Wizards (vergleiche Abschnitt 2.4) erzeugt wird, da er als Grundlage für das Werkzeug zur Unterstützung von Anpassungsprozessen dienen soll. Der Prototyp gibt dem Prozessexperten die Möglichkeit, zu prüfen, ob der von ihm beschriebene Anpassungsprozess korrekt wiedergegeben wird. Um sicherzustellen, dass der Prototyp auch wirklich das Prozessverständnis der Prozessexperten abbildet, wäre es wünschenswert, wenn die Prozessexperten selbst den Prototyp basierend auf den Prozessbeschreibungen erstellen könnten.

Die meisten der Experten in der Durchführung von Anpassungsprozessen sind keine Software-Entwickler. Sie haben daher keine ausreichenden Programmierkenntnisse. Deswegen muss die Erstellung des Prototyps auch ohne Programmierkenntnisse möglich sein. Da in den Prozessbeschreibungen bereits alle benötigten Informationen zur Erzeugung eines Prototyps enthalten sind, wäre eine automatische Generierung des Prototyps aus den Prozessbeschreibungen wünschenswert.

Damit der Prototyp von Prozessexperten erzeugt werden kann, aber auch für Entwickler eine wertvolle Grundlage für die weitere Entwicklung darstellt, müssen auch der Prototyp sowie seine Generierung gewissen Anforderungen gerecht werden:

- Der erzeugte Prototyp muss alle in der Prozessbeschreibung enthaltenen Informationen berücksichtigen.
- Weiterhin muss der Prototyp die Ablauflogik des beschriebenen Anpassungsprozesses widerspiegeln (inklusive Abhängigkeiten zwischen Prozessschritten sowie atomaren Unterschritten und Informationen darüber, ob die Ausführung dieser Schritte verpflichtend ist).
- Die Generierung des Prototyps muss automatisiert erfolgen, so dass der Prozessexperte sie nur starten und dann nicht weiter eingreifen muss.
- Die Erstellung muss schnell vonstatten gehen, da der Prozessexperte wahrscheinlich mehrfach den folgenden Zyklus durchläuft: Anpassungsprozess beschreiben – Prototyp generieren – Prototyp testen – Prozessbeschreibung verändern, falls der Prototyp noch nicht den Vorstellungen entspricht, erneut Prototyp generieren etc.
- Der erzeugte Prototyp soll erweiterbar sein, um automatisierte Funktionalitäten zufügen zu können. So kann ein Entwickler aus dem Prototyp einen einsatzfähigen Wizard zur Unterstützung von Anpassungsprozessen erzeugen.

- Der Prototyp muss den im vorigen Kapitel genannten Anforderungen an ein Programm zur Prozessunterstützung gerecht werden, damit er als Basis für die Erstellung eines derartigen Programms genutzt werden kann.

Nachfolgend wird ein Konzept zur Pattern-basierten Prototyp-Erzeugung vorgestellt, das den in diesem und im vorigen Abschnitt gestellten Anforderungen gerecht wird.

3.4 Konzept einer Pattern-basierten Prototyp-Erzeugung

Wie bereits mehrfach erläutert, soll das in der vorliegenden Arbeit beschriebene Vorgehen zur Entwicklung eines prototypischen Wizards zur Unterstützung von Anpassungsprozessen auf Expertenwissen beruhen. Mit dem Verfahren erstellte Prototypen sollen dem Prozessverständnis von Prozessexperten entsprechen und deren Wissen enthalten.

Damit dies möglich ist, soll ein Prozessexperte in einem ersten Schritt selbst seinen Anpassungsprozess in Form von sogenannten Patterns beschreiben und so sein Wissen darüber bereitstellen. Das zur Verfügung gestellte Prozesswissen ist in einem zweiten Schritt die Basis für eine automatisierte Erstellung eines Prototyps in Form eines Wizards. Anhand des Prototyps kann der Prozessexperte beurteilen, ob der zu unterstützende Anpassungsprozess korrekt abgebildet ist. Durch Zufügen automatisierter Funktionen kann im dritten Schritt ein Entwickler den Prototyp zu einem voll funktionsfähigen Wizard zur Unterstützung von Anpassungsprozessen erweitern. Daraus ergibt sich ein dreistufiges Konzept (Abbildung 12), das in diesem Abschnitt erläutert wird.

Abbildung 12: Die drei Schritte der Pattern-basierten Prozessbeschreibung und Wizard-Erstellung.

Die Zielgruppe des hier vorgestellten Ansatzes sind vor allem Personen, die sich nicht mit den Methoden auskennen, die in der Software-Entwicklung verwendet werden, um Prozesse zu analysieren und zu beschreiben. Jemand der beispielsweise als Übersetzer arbeitet (ein Anpassungsprozess, der sehr oft durchgeführt wird), kennt sich sehr gut mit dem Prozess aus, der zur Durchführung einer Übersetzung benötigt wird. Üblicherweise ist ein Übersetzer aber nicht gleichzeitig ein erfahrener Software-Designer oder -Entwickler. Neben anderen Verfahren werden derzeit besonders UML [16] und ARIS [Sc01] in der Software-Entwicklung zur Prozessmodellierung verwendet [AS07]. Doch Prozessexperten haben oft keine Kenntnisse in diesen Formalismen, da sie für ihre Tätigkeit nicht vorausgesetzt werden. Schließlich brauchen sie diese Methoden nicht in ihrer täglichen Arbeit. Das gilt nicht nur für Übersetzer, sondern auch für die meisten anderen Experten in der Durchführung von Anpassungsprozessen.

Erschwerend kommt hinzu, dass UML in vielen Fällen zu komplex ist und seine Konstrukte nicht immer eindeutig sind. Dadurch ist es schwer zu lernen [SEL05]. Auch ARIS ist für die Praxis oft zu kompliziert [Sc07]. Daher ist es für Anpassungsexperten

meist nicht möglich, ihr Wissen in Form dieser Formalismen zu beschreiben. Zusätzlich sind sie oft auch nicht in der Lage zu kontrollieren, ob in den Formalismen erstellte Prozessmodelle korrekt sind. Somit werden gängige Methoden zur Prozessbeschreibung den in Abschnitt 3.2 aufgelisteten Anforderungen nicht gerecht.

Martin Fowler [Fo96] entschied sich, das in der Analysephase der von ihm durchgeführten Entwicklungsprojekte gesammelte Wissen von Prozessexperten mit Hilfe von Patterns festzuhalten. Ein Grund für die Wahl von Patterns war, dass diese in natürlicher Sprache notiert werden. Dadurch sind sie für die Personen, deren Wissen mit den Patterns beschrieben wird, auch ohne langwierige Einarbeitung gut verständlich. Eine strukturierte Form kann zusätzlich zu diesem Effekt beitragen. Dadurch können Patterns bei den Prozessexperten eine hohe Akzeptanz gewinnen [Ha01].

Patterns beschreiben Erfahrungswissen: Sie stellen das Wissen dar, wie ein Problem, das unter bestimmten Bedingungen, in einem bestimmten Kontext auftreten kann, gelöst werden kann. Patterns werden oft verwendet, um Expertenwissen zu bestimmten Prozessen oder Sachverhalten zu dokumentieren. Dadurch werden sie den Anforderungen an eine leichte Erlernbarkeit und eine gute Verständlichkeit gerecht.

Fowlers Analyse-Patterns haben ursprünglich keine feste Notations-Form, erst in späteren Arbeiten ist er zu dem Schluss gekommen, dass eine feste Notation es einfacher macht, die Patterns zu referenzieren [10]. Die meisten Patterns haben jedoch eine strukturierte Form. Diese kann außerdem das Verständnis von Patterns erleichtern, da die Notationsform als eine Art Leitfaden sowohl zur Erstellung als auch zum Verständnis der Patterns verwendet werden kann. Sie ermöglicht es außerdem, die Patterns in einem XML-Format, der sogenannten Pattern Language Markup Language (PLML) [18] abzuspeichern. PLML ist eine XML DTD, die ursprünglich als gemeinsamer Standard für HCI Patterns gedacht war. Es wird aber auch für andere Arten von Patterns verwendet.

XML ist sehr flexibel und lässt sich für unterschiedliche Zwecke einsetzen. So kann XML beispielsweise nach HTML transformiert werden. In dieser Form ist es auch für Anwender gut lesbar, die keine Kenntnisse von XML haben. Gleichzeitig ist es sehr strukturiert und dadurch maschinenlesbar. Es bietet außerdem die Möglichkeit über XML Metadata Interchange (XMI) [17] in gängige UML Modellierungswerkzeuge importiert zu werden.

Somit erfüllen Patterns, die als XML-Dateien gespeichert werden, die zuvor genannten Hauptanforderungen:

- Sie können in natürlicher Sprache notiert werden und sind somit leicht verständlich. Außerdem können sie eine feste Struktur haben, die leicht erlernbar ist.
- Sie bilden Erfahrungswissen ab und können so, beschreibt man Anpassungsprozesse mit Patterns, auch anderen Personen wichtige Informationen zur Durchführung der beschriebenen Anpassungsprozesse übermitteln.
- Sie können Prozesse so beschreiben, dass sie die zur Erstellung eines Prototyps benötigten Informationen enthalten. Dafür enthalten die Patterns Informationen, wie vorzugehen ist, wenn man die Anpassungsprozesse durchführen will. Und sie benennen die hierfür benötigten Prozessschritte.

- Sie können in XML gespeichert werden und liegen dann in einem Format vor, das maschinenlesbar ist und als Basis für die Prototyp-Erstellung verwendet werden kann.
- Sie können in XMI umgewandelt werden und ermöglichen so Software-Entwicklern und -Designern mit ihren gewohnten Methoden zu arbeiten.

Daher werden im hier vorgestellten Ansatz Anpassungsprozesse in Form von Patterns beschrieben und in XML-Dateien gespeichert.

Viele Experten in der Durchführung von Anpassungsprozessen haben keine guten XML-Kenntnisse. Daher ist es notwendig, ihnen ein Eingabewerkzeug zur Verfügung zu stellen, das sie dabei unterstützt, die Pattern-basierten Prozessbeschreibungen zu erstellen und als XML-Dateien zu speichern. Um dies zu ermöglichen, wurde das Werkzeug PIT (Abkürzung für: Process description Input Tool) entwickelt, das im folgenden Kapitel detailliert erläutert wird. Dadurch kann sichergestellt werden, dass alle benötigten Informationen zur Verfügung gestellt werden. Zusätzlich speichert das PIT die in natürlicher Sprache eingegebenen Informationen in XML und transformiert sie so in eine Form, die für die spätere Prototyp-Erstellung geeignet ist.

Somit sieht das Konzept für die Prozessbeschreibung durch Prozessexperten ein zweistufiges Vorgehen vor (siehe Abbildung 13):

1. Im ersten Schritt muss das Experten-Wissen zu den Anpassungsprozessen in einem einfach verständlichen Format gesammelt werden. Das wird über Pattern-basierte Prozessbeschreibungen gewährleistet. Der Prozessexperte erstellt die Pattern-basierte Prozessbeschreibung mit dem Werkzeug PIT.
2. Im zweiten Schritt bildet das PIT dieses Format auf eine formale Repräsentation ab, die es ermöglicht einen Prototyp basierend auf den zur Verfügung stehenden Informationen zu generieren.

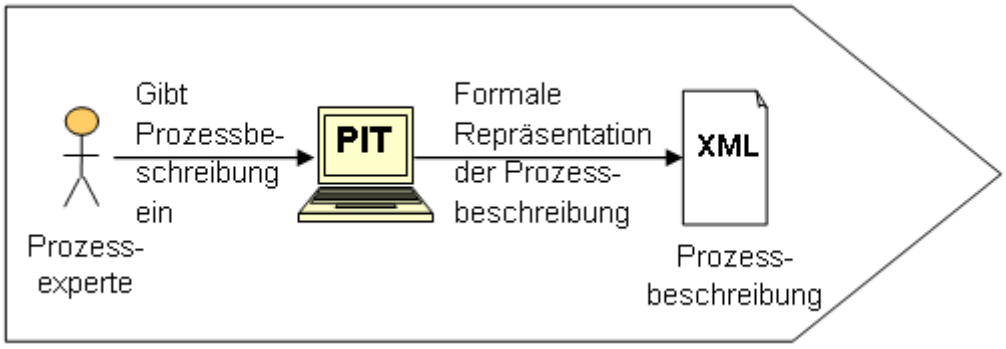


Abbildung 13: Schritt 1: Prozessbeschreibungen erstellen.

Wie bereits erläutert, kann ein basierend auf den Prozessbeschreibungen erstellter Prototyp Prozessexperten die Möglichkeit bieten, zu prüfen ob der beschriebene Anpassungsprozess korrekt abgebildet ist. Um dies zu erreichen, sollte daher Prozessexperten neben der Möglichkeit, ihr Wissen über Anpassungsprozesse zur Verfügung zu stellen, zusätzlich die Möglichkeit gegeben werden, einen Prototyp zu erstellen. Dadurch ließen

sich Prototypen erhalten, die genau die Sicht der Prozessexperten auf die Anpassungsprozesse darstellen.

Wie in Abschnitt 3.3 dargestellt, sollte die automatische Erstellung des Prototyps auch ohne Programmierkenntnisse möglich sein. Sie muss einfach handhabbar und schnell sein. Da die Prozessbeschreibungen bereits alle Informationen beinhalten, die im Prototyp benötigt werden, soll die Prototyp-Erzeugung basierend auf diesen Beschreibungen vorgenommen werden.

Damit dies möglich ist, sollen die mit dem PIT erstellten Dateien als Eingabe für ein zweites Werkzeug dienen, das die vom PIT zur Verfügung gestellten Daten interpretiert und daraus einen prototypischen Wizard erzeugt. Das zweite Werkzeug heißt WGT (Wizard Generation Tool) (siehe Abbildung 14).

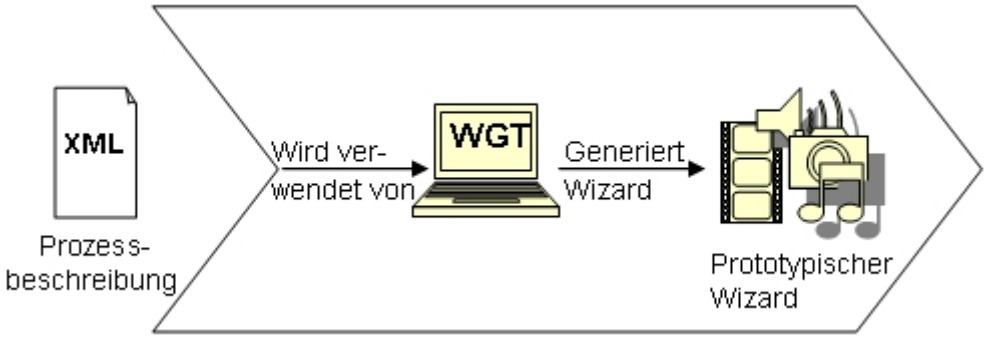


Abbildung 14: Schritt 2: Prototyp-Erstellung mit WGT.

Das WGT muss den im vorigen Abschnitt genannten Anforderungen an die automatisierte Prototyp-Erzeugung gerecht werden. Daher ist es so zu konzipieren, dass

- es alle in der Prozessbeschreibung enthaltenen Informationen interpretiert und sie in einen Prototyp überführt.
- dabei die Ablauflogik des beschriebenen Anpassungsprozesses berücksichtigt wird.
- der erzeugte Prototyp erweiterbar ist, was nachfolgend erläutert wird.
- das WGT einfach zu bedienen ist: Es muss sich aus dem PIT heraus aufrufen lassen und eigenständig, ohne dass der Prozessexperte eingreifen muss, einen prototypischen Wizard erstellen.
- das Werkzeug so gestaltet ist, dass die Erzeugung eines Prototyps nur wenige Sekunden dauert.

Der vom WGT erzeugte prototypische Wizard soll als erster Prototyp eines Unterstützungswerkzeuges für Anpassungsprozesse dienen und das Prozessverständnis der Experten abbilden. Er muss so gestaltet sein, dass er in einem dritten Schritt erweitert werden kann und so als Ausgangspunkt für die weitere Entwicklung eines Werkzeugs zur Unterstützung von Anpassungsprozessen dienen kann. Diese Erweiterung ist von erfahrenen Entwicklern vorzunehmen, da diese die hierfür benötigten Kenntnisse besitzen (Abbildung 15).

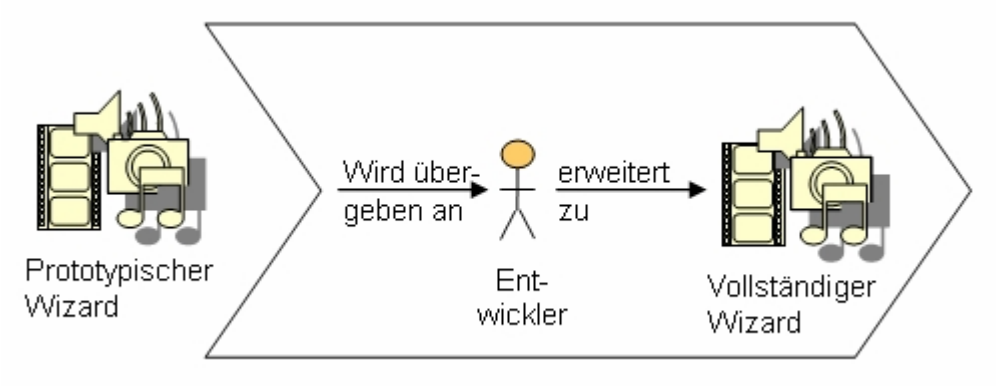


Abbildung 15: Schritt 3: Wizard-Erweiterung.

Damit ergibt sich für das in dieser Arbeit vorgestellte Verfahren zur Pattern-basierten Prototyp-Generierung das folgende dreistufige Vorgehen (Abbildung 16):

1. Erstellung Pattern-basierter Prozessbeschreibungen
2. Automatisierte Prototyp-Erzeugung
3. Erweiterung des prototypischen Wizards

Abbildung 16: Die drei Schritte der Pattern-basierten Prozessbeschreibung und Wizard-Erstellung.

Die Schritte des Ansatzes werden in den folgenden drei Kapiteln detailliert erläutert.

3.5 Zusammenfassung

In diesem Kapitel wurde das Konzept des in der vorliegenden Arbeit vorgestellten Ansatzes erläutert. Dazu wurden zuerst verschiedene Vorgehensmodelle zur Software-Entwicklung betrachtet. Dabei hat sich gezeigt, dass in gängigen Vorgehensmodellen die Prozessexperten oft erst spät und / oder nicht ausreichend in die Software-Erstellung eingebunden werden. Deswegen wurde als ein Beitrag dieser Dissertation ein Konzept entwickelt, das es erlaubt, Prozessexperten durchgängig in die Entwicklung von Software zur Unterstützung von Anpassungsprozessen einzubinden.

Die Grundlage des vorgestellten Konzeptes bilden Beschreibungen von Anpassungsprozessen, die von Prozessexperten erstellt werden. Daher wurden Anforderungen an Prozessbeschreibungen formuliert, die von Prozessexperten erstellt werden können und die auch für Software-Designer und -Entwickler in die gewohnte Arbeit integriert werden können.

Anschließend wurde dargelegt, dass eine auf den Prozessbeschreibungen basierende Erstellung von Prototypen durch die Prozessexperten wünschenswert ist, da so die

Prozessexperten in die Lage versetzt werden, ihr Verständnis von Anpassungsprozessen in Prototypen abzubilden. Die Prototypen sollen als Basis für ein Programm zur Unterstützung dieser Prozesse dienen. Daher müssen die Prototypen auch die Anforderungen an ein Programm zur Unterstützung von Anpassungsprozessen erfüllen, die in Kapitel 2 vorgestellt wurden.

Abschließend wurde das drei-stufige Konzept zur Software-Entwicklung basierend auf von Prozessexperten erstellten Prozessbeschreibungen vorgestellt. Dieses Konzept sieht eine Beschreibung der Anpassungsprozesse durch Prozessexperten in einem Pattern-basierten Format vor. Basierend auf den erstellten Prozessbeschreibungen können die Prozessexperten automatisiert einen prototypischen Wizard erzeugen. Dieser kann in einem dritten Schritt von Entwicklern als Grundlage zur weiteren Entwicklung verwendet und gegebenenfalls erweitert werden.

4 Beschreibung von Anpassungsprozessen in Pattern-Form

Dieses Kapitel präsentiert den ersten der drei Schritte des im vorigen Kapitel vorgestellten Konzeptes. Der erste Schritt dient der Erstellung von Prozessbeschreibungen in einem Pattern-basierten Format. In diesem Kapitel wird genau erläutert, wie die zur Prozessbeschreibung verwendeten Patterns aufgebaut sind und wie mit ihnen Anpassungsprozesse beschrieben werden können.

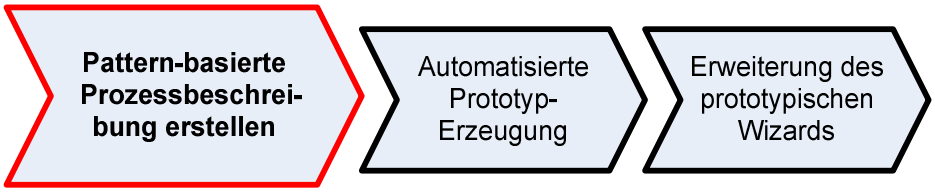


Abbildung 17: Die drei Schritte der Pattern-basierten Prozessbeschreibung und Wizard-Erstellung.

4.1 Patterns – Eine Einführung

Das in der vorliegenden Arbeit vorgestellte Konzept basiert auf der Verwendung von Prozessbeschreibungen in Pattern-Form. In diesem Abschnitt wird daher auf die Ursprünge von Patterns eingegangen. Dann werden gängige Pattern-Definitionen betrachtet. Anschließend werden Notationsformen für Patterns vorgestellt. Danach werden verschiedene Möglichkeiten zur Organisation von Patterns vorgestellt. Abschließend wird erläutert, wie Patterns geschrieben und publiziert werden.

4.1.1 Historie

Der Begriff „Pattern“, wie er in dieser Arbeit verwendet wird, basiert auf einer Idee des Architekten Christopher Alexander. Dieser stellte fest, dass in der Architektur bestimmte Probleme immer wieder auftreten [Al64]. Solche Probleme werden durch wiederkehrende Muster (englisch: Patterns) gelöst, die die Gestalt eines Ortes, einer Stadt, eines Gebäudes, also dessen spezielle Konfiguration beschreiben. Alexander definiert Patterns folgendermaßen:

> *“Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing the same twice.”* [Al^{+}77]

In „A Pattern Language“ [Al^{+}77] haben Alexander und sein Team eine Vielzahl derartiger Patterns gesammelt und veröffentlicht. Das Buch „The Timeless Way of Building“ [Al79] liefert eine philosophisch-theoretische Abhandlung und Methodenbeschreibungen zu Patterns. Doch in der Architektur fanden Alexanders Arbeiten nur geringe Beachtung.

Einige Jahre später entdeckten Ward Cunningham und Kent Beck die Arbeiten von Alexander und entschieden, diese auf die Entwicklung von Software zu übertragen. Sie

verwendeten Patterns, um ihr Wissen über die Gestaltung grafischer Benutzeroberflächen in Smalltalk aufzuschreiben. Sie fanden heraus, dass diese Patterns insbesondere für Neulinge bei der Arbeit mit Smalltalk hilfreich waren. Daher beschlossen sie, ihre Erfahrung auf der OOPSLA 1987 vorzustellen. [BC87]

Zu dieser Zeit beschäftigten sich auch James Coplien und andere mit der Möglichkeit, Erfahrungswissen in der Software-Entwicklung aufzuschreiben, teilweise in Pattern-Format [Co92], teilweise nicht in Pattern-Format [Co91]. 1991 und 1992 wurden außerdem auf der OOPSLA Workshops zu Patterns abgehalten. Hierbei kamen viele wichtige Personen der noch jungen Pattern-Community zusammen und diskutierten ihre Ideen und Erfahrungen mit Patterns.

Zur gleichen Zeit beendete Erich Gamma seine Doktorarbeit, in der er sich bereits mit Patterns beschäftigte. Er ging danach in die Vereinigten Staaten, wo er und Richard Helm, Ralph Johnson und John Vlissides (später bekannt als die „Gang of Four“) mit der Arbeit an ihren Ideen zu Software Design Patterns begannen.

Die Zahl der Personen, die sich mit Patterns beschäftigten, wurde immer größer. Und im August 1993 luden Kent Beck und Grady Booch eine Gruppe von Personen, die alle mit Patterns arbeiteten, zu einem Treffen in den Bergen von Colorado ein. Ziel war es, über die Verwendung von Patterns im Software Design zu diskutieren. Dies war das erste Treffen der sogenannten Hillside Group. Im April 1994 traf sich die Hillside Group erneut, um eine erste Konferenz zu Patternsprachen in der Programmierung zu planen. Diese Konferenzen werden seitdem regelmäßig durchgeführt und als PLoP (Pattern Languages of Programs) Konferenzen bezeichnet.

1995 wurde dann das Buch „Design Patterns - Elements of Reusable Object-Oriented Software” [Ga+95] der Gang of Four veröffentlicht. Es enthält 23 Software Design Patterns. Dieses Buch kann als der Durchbruch der Pattern-Idee in der Software Entwicklung angesehen werden, da es sehr schnell sehr populär wurde.

Kurze Zeit später wurde ein weiteres wichtiges Buch zu Patterns veröffentlicht, das sogenannte POSA-Buch: „Pattern-Oriented Software Architecture: A System of Patterns” [Bu+96b]. Es wurde von Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad und Michael Stal veröffentlicht. In diesem Buch verwenden die Autoren Patterns zur Beschreibung und Dokumentation großer Software Architekturen. Es stellte sich heraus, dass dieses Buch ein weiterer Meilenstein in der Geschichte der Patterns wurde.

Mittlerweile werden Patterns in den verschiedensten Anwendungsgebieten verwendet, wie HCI [Bo01], Security [Sc+06], Projekt Management [Co96] oder auch Pädagogik (beispielsweise die unter www.pedagogicalpatterns.org gesammelten Patterns). Dem entsprechend finden in allen Teilen der Welt neben der ursprünglichen PLoP-Konferenz weitere Konferenzen zu Patterns statt, wie die EuroPLoP in Mitteleuropa oder die KoalaPLoP in Australien.

4.1.2 Pattern-Definitionen

Die eingangs aufgeführte Pattern-Definition des Architekten Christopher Alexander ist eine der am weitesten verbreiteten Definitionen. Sie stellt somit einen guten

Ausgangspunkt für Pattern-Definitionen dar. Eine weitere Pattern-Definition von Christopher Alexander findet sich in [Al79, S.247]:

> *"Each pattern is a three part rule, which expresses a relation between a certain context, a problem, and a solution."*

Entsprechend den verschiedenen Einsatzgebieten von Patterns gibt es eine Vielzahl von Pattern-Definitionen. Viele dieser Definitionen bauen auf Alexanders Verständnis eines Patterns als Regel auf. Gamma et al. [Ga+95] sagen beispielsweise, dass sich Patterns nicht nur in der Architektur, sondern auch im Software Design finden und anwenden lassen. Für beide Arten von Patterns (Architektur-Patterns und Design-Patterns) gilt:

> *"… the core of both kinds of patterns is a solution to a problem in a context. "* [Ga+95, S.2 folgende]

Dieses Verständnis von Patterns findet sich auch in vielen anderen Arbeiten. So wird auch im bekannten POSA-Buch [Bu+96b, S.8] eine Definition von Patterns verwendet, die auf der Alexandrinischen Definition beruht:

> *"A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate."*

Richard Gabriel gehen diese Definitionen jedoch nicht weit genug. Auf der Hillside-Internetseite, einer zentralen Website [5] der Pattern-Community, schreibt er, dass Alexanders Definition über den oben aufgeführten Satz hinaus geht. Denn Alexander führt diesen folgendermaßen weiter:

> *“As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves.*
>
> *As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant.”* zitiert nach [5]

Wäre es Alexander nur auf Kontext, Problem und Lösung angekommen, so hätte er sich, laut Gabriel, die ausführlichere Erklärung gespart. Also sind auch, die sich teilweise widersprechenden Anforderungen (Forces), die auf ein Pattern einwirken, von zentraler Bedeutung für ein Pattern. Dem entsprechend hat Gabriel Patterns definiert als:

> *"Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves.”* [5]

Ein Pattern beschreibt also den Zusammenhang zwischen einem Problem, das in einem Kontext auftritt und verschiedene Forces erzeugt, die von der Lösung ausbalanciert werden. Weiterhin gibt es eine Reihe von Pattern-Definitionen, die speziell auf das jeweilige Einsatzgebiet der Patterns zugeschnitten sind. In der vorliegenden Arbeit wird die

Alexandrinische Definition eines Patterns als Regel, die eine Beziehung zwischen Problem. Kontext und Lösung beschreibt, verwendet.

4.1.3 Pattern-Elemente und Notationsformen von Patterns

So unterschiedlich die verschiedenen Pattern-Definitionen auch sind, es lassen sich doch drei Elemente ausmachen, die allen Definitionen gemeinsam sind:

Problem: Hier wird beschrieben, welches Problem durch das Pattern gelöst wird. Damit es sich um ein Pattern handelt, sollte das Problem immer wieder auftreten. Die Beschreibung des Problems kann sehr unterschiedlich aussehen. James Coplien beispielsweise formuliert das Problem als Frage. [Co94]. Bei Alexander dagegen wird das Problem sehr ausführlich beschrieben und benennt auch die Forces.

Kontext: Der Kontext beschreibt, in welchem Zusammenhang das Pattern angewandt werden kann. Oft werden hier auch Vorbedingungen für die Ausführung des Patterns benannt. In einigen Fällen enthält der Kontext die Forces (siehe unten). Diese widersprechen sich oft und es gilt herauszufinden, wie man vorgehen kann, um diese Widersprüche auszubalancieren.

Lösung: Hier wird erläutert, wie das vorher beschriebene Problem gelöst werden kann und wie die Forces ausbalanciert werden können. Wichtig ist, dass sich die Lösung bewährt haben muss. Üblicherweise wird von Patterns verlangt, dass ihre Lösung in mindestens drei Fällen erfolgreich angewandt wurde. Das wird als die sogenannte „Rule-Of-Three“ bezeichnet. [Co94]

Neben den bisher genannten Elementen gibt es noch eine Reihe weiterer Elemente, die sich oft in Patterns finden lassen:

Name: Der Name ermöglicht eine Identifizierung des Patterns. Außerdem sollte der Name bereits eine Idee vermitteln, wovon das Pattern handelt.

Forces: Hinsichtlich der Angabe, wo die Forces zu nennen sind, ist sich die Pattern-Community nicht einig. Forces können für sich allein aufgelistet werden, sie können aber auch Bestandteil von Kontext oder Lösung sein. In allen Fällen beschreiben sie aber die oft widersprüchlichen Anforderungen an ein Pattern, die von der Lösung ausbalanciert werden müssen. Das Pattern muss beschreiben, wie man mit den widersprüchlichen Anforderungen umgehen kann.

Beispiel: Das Beispiel hilft dem Leser beim Verständnis des Patterns. Meist wird eine exemplarische Anwendung des Patterns erläutert. Besonders hilfreich ist es, wenn das Beispiel durch Bilder veranschaulicht werden kann.

Resultierender Kontext: (Wird manchmal auch als *Konsequenzen* bezeichnet.) Hier wird erläutert, welche Konsequenzen sich aus der Ausführung des Patterns ergeben und wie sich der ursprüngliche Kontext verändert.

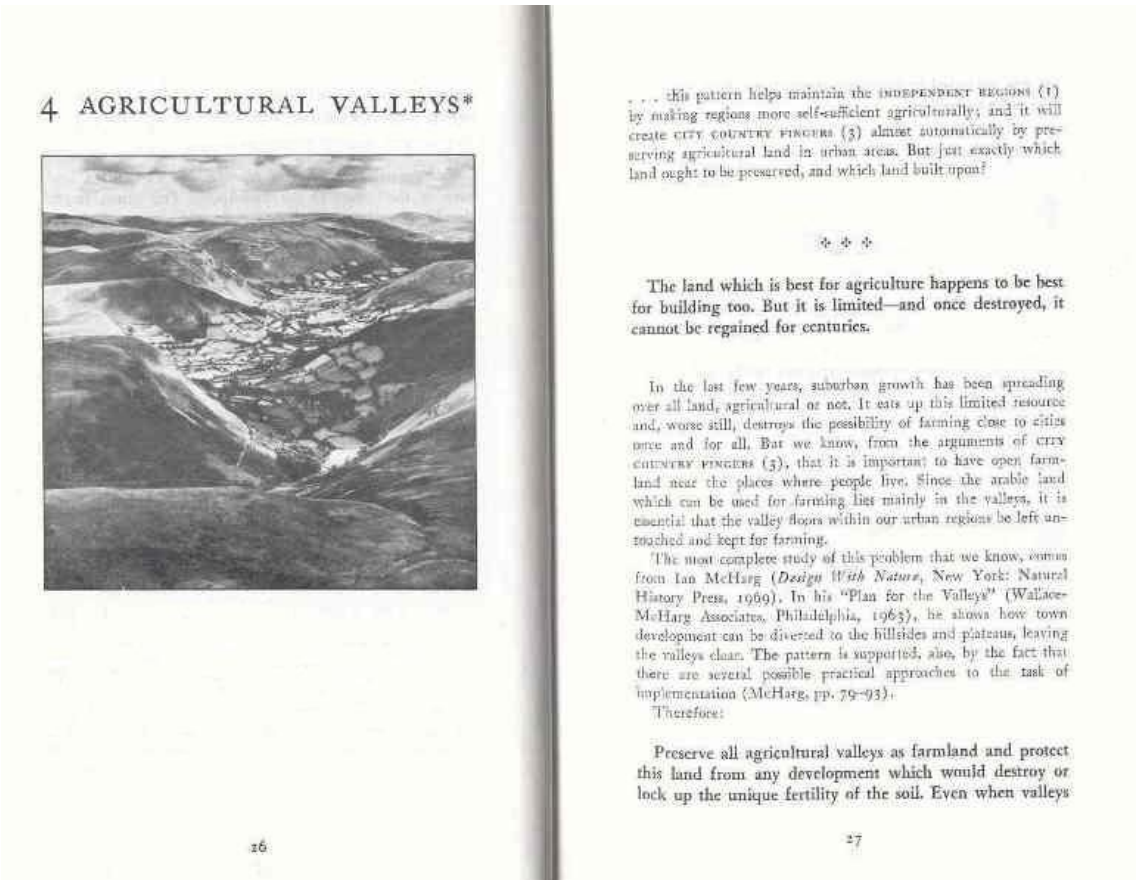
Begründung: Hier sollte erläutert werden, wie und warum das Pattern die Forces auflöst.

Verwandte Patterns: Patterns stehen nicht für sich allein, sondern in Beziehung zu einander. Die verwandten Patterns benennen, welche Patterns zum beschriebenen Pattern in welcher Beziehung stehen.

Bekannte Verwendungen: Damit ein Pattern auch wirklich als Pattern bezeichnet werden kann, muss es bereits erfolgreich eingesetzt worden sein. Das liefert den empirischen Beleg für das Pattern.

Es gibt unterschiedliche Meinungen dazu, welche der genannten Elemente verpflichtend und wie zu verwenden sind. Notationsformen von Patterns geben vor, mit welchen Elementen ein Pattern wie beschrieben werden muss. Im Folgenden wird ein Überblick über einige häufig referenzierte Pattern-Notationsformen gegeben:

Die ursprüngliche Form wurde von Alexander entwickelt und wird deswegen auch als **Alexandrinische Form** bezeichnet.



4 AGRICULTURAL VALLEYS*

. . . this pattern helps maintain the INDEPENDENT REGIONS (1) by making regions more self-sufficient agriculturally; and it will create CITY COUNTRY FINGERS (3) almost automatically by preserving agricultural land in urban areas. But just exactly which land ought to be preserved, and which land built upon?

* * *

The land which is best for agriculture happens to be best for building too. But it is limited—and once destroyed, it cannot be regained for centuries.

In the last few years, suburban growth has been spreading over all land, agricultural or not. It eats up this limited resource and, worse still, destroys the possibility of farming close to cities once and for all. But we know, from the arguments of CITY COUNTRY FINGERS (3), that it is important to have open farmland near the places where people live. Since the arable land which can be used for farming lies mainly in the valleys, it is essential that the valley floors within our urban regions be left untouched and kept for farming.

The most complete study of this problem that we know, comes from Ian McHarg (*Design With Nature*, New York: Natural History Press, 1969). In his "Plan for the Valleys" (Wallace-McHarg Associates, Philadelphia, 1963), he shows how town development can be diverted to the hillsides and plateaus, leaving the valleys clean. The pattern is supported, also, by the fact that there are several possible practical approaches to the task of implementation (McHarg, pp. 79–93).

Therefore:

Preserve all agricultural valleys as farmland and protect this land from any development which would destroy or lock up the unique fertility of the soil. Even when valleys

26

27

Abbildung 18: Ausschnitt aus einem Alexandrinischen Pattern [Al^{+}77, S. 26, 27].

Die Alexandrinischen Patterns beginnen mit einem Namen, gefolgt von Sternchen, die die sogenannte Konfidenz des Autors in das Pattern angeben: Kein Sternchen bedeutet, dass das Pattern erst ein Beispiel liefert, es sich aber vermutlich noch ändern wird. Ein Sternchen bedeutet, dass das Pattern bereits auf dem Weg ist, eine sogenannte Invariante zu beschreiben. Zwei Sternchen zeigen an, dass das Pattern bereits ausgereift ist und sich vermutlich nicht mehr ändern wird. Auf Titel und Konfidenz folgt ein Bild, meist

ein Foto, das eine typische Situation zeigt, wo das Pattern zur Anwendung kam. Danach wird der Kontext des Patterns genannt, gefolgt von einer Kurzfassung des Problems. Dieses wird anschließend ausführlich erläutert. Dabei werden auch die auftretenden Forces benannt. Durch das Wort „Therefore“ wird der Lösungsteil des Patterns eingeleitet, der mit einer Kurzfassung der Lösung beginnt. Es schließt sich eine detaillierte Lösungsbeschreibung an, die auch die Konsequenzen enthält. Auf die Lösung folgt ein Diagramm, das die Lösung verdeutlicht. Die Patterns schließen mit Beispielen, der Nennung von verwandten Patterns. Abbildung 18 zeigt exemplarisch einen Ausschnitt aus einem der Patterns aus [Al+77].

Im Gegensatz zu vielen anderen Pattern-Formen werden den Elementen bei der alexandrinischen Form nicht ihre Bezeichner voran gestellt. Das heißt, der Kontext wird beispielsweise nicht explizit durch das Wort „Context“ eingeleitet. Eine Form, wo das Voranstellen der Element-Bezeichner eingesetzt wird, ist die nachfolgend erläuterte GoF Form.

Die **GoF Form** wurde von der Gang of Four für deren Design Patterns [Ga+95] verwendet. Im Gegensatz zur sehr narrativen Form der alexandrinischen Patterns enthalten Patterns, die in der GoF Form notiert sind, wesentlich weniger narrativen Fließtext. Dafür liegt eine starke Gewichtung darauf, wie sich die Patterns implementieren lassen. Es werden in allen Patterns ausführliche Code-Beispiele vorgestellt. Die GoF-Patterns sind teilweise sehr lang und können sich über zehn und mehr Seiten erstrecken.

Patterns in der GoF Form beginnen mit ihrem *Namen* und einer *Klassifikation* also einer Einordnung in eine Gruppierung. Insgesamt gibt es sechs Gruppierungen. Diese setzen sich aus einem Gültigkeitsbereich (class und object) und einem Zweck (creational, structural und behavioral) zusammen: Ein Pattern kann eine Klasse oder ein Objekt sein und für einen der drei Zwecke verwendbar sein. Der nachfolgende *Intent* erläutert, welche Fragen und Probleme das Pattern löst. Das *Also Known As* Element nennt andere gängige Namen für das Pattern. Danach wird anhand eines Szenarios eine *Motivation* für die Anwendung des Patterns gegeben. Unter *Applicability* werden Situationen genannt, in denen das Pattern angewandt werden kann. *Structure* liefert eine grafische Darstellung der vom Pattern benötigten Klassen. *Participants* zählt die beteiligten Klassen und Objekte auf. *Collaborations* erläutert, wie die Participants zusammenarbeiten müssen. Die *Consequences* zählen die sich aus der Anwendung des Patterns ergebenden Konsequenzen auf. Unter *Implementation* finden sich Hinweise, was bei der Implementierung zu beachten ist. Ein beispielhaftes Code-Fragment (*Sample Code*) liefert eine Vorstellung davon, wie eine konkrete Umsetzung beispielsweise in Smalltalk oder C++ aussehen könnte. Die *Known uses* zählen erfolgreiche Anwendungsfälle des Patterns auf. Die Patterns enden mit einer Aufzählung verwandter Patterns (*Related Patterns*).

In GoF-Patterns werden viele UML-Diagramme zur Veranschaulichung der Motivation und / oder der Struktur genutzt. Auch in den Collaborations, den Consequences, der Implementation, sowie in den Known uses werden Diagramme verwendet.

Die im **POSA**-Buch [Bu+96b] verwendete Form ist ebenfalls sehr strukturiert und ähnelt der GoF Form, auch wenn einige Elemente anders verwendet werden oder benannt sind. Auch die Patterns im POSA-Buch erstrecken sich zum Teil über mehr als zehn

Seiten. An dieser Stelle wird auf diese Form aufgrund ihrer Ähnlichkeit zur GoF Form nicht näher eingegangen.

James Coplien vertritt die Meinung, dass Patterns unabhängig von ihrer jeweiligen Form stärker das Wesen der ursprünglichen Patterns von Alexander aufgreifen sollten [Gr04]. Das hat Eingang gefunden in die sogenannte **kanonische Form**. Diese Pattern-Form ist bewusst sehr einfach gehalten. Ihre Elemente sind ähnlich denen der GoF Form [6]. Allerdings sind Patterns in kanonischer Form wesentlich narrativer.

Welche Notationsform man wählt, hängt davon ab, was man mit den Patterns beabsichtigt. Da oft eine Form nicht genau zum geplanten Verwendungszweck von Patterns passt, finden sich viele Abarten der hier genannten Pattern-Formen, die alle bedingt sind, durch eine spezielle Verwendung der Patterns. In der vorliegenden Arbeit wird eine Notationsform verwendet, die an die POSA Form [Bu+96b] und die kanonische Form [6] angelehnt ist, da diese Notationsformen die meisten der Elemente enthalten, die für die Beschreibung der in dieser Arbeit berücksichtigten Anpassungsprozesse benötigt werden (vergleiche Abschnitt 4.2).

4.1.4 Organisation von Patterns

Ein Pattern existiert üblicherweise nicht für sich allein, sondern steht in Beziehung zu anderen Patterns, die verwandte Probleme beschreiben. Diese werden zu Pattern-Sammlungen zusammengefasst. Nach [Sc03, Ha05] lassen sich drei Typen von Pattern-Sammlungen unterscheiden: Pattern-Kataloge, Pattern-Systeme und Pattern-Sprachen. Die drei Typen stellen eine Art evolutionärer Entwicklung dar.

Die einfachste Form einer Menge zusammengehöriger Patterns stellt der *Pattern-Katalog* dar. Er enthält eine lose zusammengehörige Sammlung von Patterns. Pattern Kataloge enthalten oft Patterns verschiedener Autoren, die unterschiedliche Strukturen aufweisen. Der Pattern Katalog zeigt lose Beziehungen und eine gewisse thematische Zusammengehörigkeit unter den enthaltenen Patterns auf.

Die nächste Stufe der Pattern-Sammlung ist das *Pattern-System*. Im Gegensatz zum Pattern-Katalog weist das Pattern-System eine einheitlichere Struktur auf: Die Notation ist bei allen enthaltenen Patterns die gleiche. Außerdem sind die Beziehungen zwischen den Patterns klarer: Es wird deutlich, wie die Patterns zusammenarbeiten.

Die höchste Stufe der Pattern-Sammlungen erreichen die *Pattern-Sprachen*. Sie sollten abgeschlossen sein. Das bedeutet, sie sollten alle Patterns des behandelten Themengebietes enthalten und deren Beziehungen untereinander benennen, was sich allerdings nur schwer prüfen lässt. Alle Patterns sollten zur Lösung des gleichen Problems beitragen. Somit kann man sagen, dass die Patterns einer Pattern-Sprache zu einem großen Pattern zusammengehören.

Die Patterns dieser Arbeit formen ein Pattern-System.

4.1.5 Schreiben von Patterns

Patterns erfindet man nicht, man findet sie. Alexander hat seine Patterns nach eigenen Aussagen durch Beobachtung gefunden. In der Pattern-Community nennt man dieses

Finden von Patterns *Pattern Mining* [Bu+96b, Ri98]. Wenn man als Neuling der Meinung ist, ein Pattern gefunden zu haben und anfängt, dieses aufzuschreiben, sollte man sich vorher eine Reihe existierender Patterns anschauen. Dabei sollte man darauf achten, wie diese Patterns aufgebaut sind. Bei Patterns, die einem gut gefallen, sollte man sich überlegen, warum das so ist. Bei Patterns, die man nicht zufriedenstellend findet, sollte man sich überlegen, was dazu führt, dass man nicht zufrieden ist.

Zusätzlich zu diesem eher allgemeinen Ratschlag haben Kerth und Cunningham in [KC97] basierend auf ihrer eigenen Erfahrung drei Methoden vorgestellt, wie man neue Patterns finden kann:

Introspective Approach: Patterns finden durch Selbstbeobachtung. Patterns können dadurch gefunden werden, dass man die eigene Arbeit analysiert und ermittelt, welche Problemlösungen sich als erfolgreich und gut herausgestellt haben. Bei diesem Ansatz zum Finden von Patterns muss der Pattern-Autor sicherstellen, dass andere Experten auch der Meinung sind, dass die vorgeschlagenen Lösungen tatsächlich erprobtes Erfahrungswissen abbilden. Außerdem sollte der Autor prüfen, ob es sich tatsächlich bei den Problemlösungen um Patterns und keine Einzellösungen handelt.

Artifactual Approach: Patterns finden durch Beobachtung von Projektergebnissen. Viele der Patterns von Christopher Alexander sind durch diesen Ansatz entdeckt worden. Autoren dieser Patterns betrachten fertige Ergebnisse, bei deren Entstehung sie selbst nicht beteiligt waren. Es werden mehrere Ergebnisse, die das gleiche oder ein ähnliches Ziel erreichen sollen, verglichen. Daraus lassen sich Rückschlüsse ziehen. Allerdings ist derjenige, der die Patterns aufschreibt, nicht notwendigerweise Experte in dem Gebiet, das die Patterns behandeln. Daher kann es sein, dass die entstehenden Patterns noch nicht final sind und noch einmal überprüft werden müssen.

Sociological Approach: Patterns finden durch Beobachtung anderer. Bei diesem Verfahren beobachtet man Personen, die in der Domäne, zu der man Patterns schreibt, Experten sind. Durch Beobachten der Experten, durch Zuhören und gezieltes Interviewen, findet man Erfahrungswissen, das man dokumentieren kann. Die resultierenden Patterns sollte man den Experten zum Feedback vorlegen. Dadurch lassen sich sehr zuverlässige Patterns erstellen.

Nachdem ein Pattern gefunden und aufgeschrieben wurde, sollte es der Öffentlichkeit zugänglich gemacht werden. Eine Möglichkeit hierzu stellen Wikis dar, wie das von Ward Cunningham initiierte Portland Pattern Repository [7]. Eine weitere Möglichkeit bieten die sogenannten PLoP (Pattern Languages of Programs) Konferenzen, die Konferenzen der Pattern-Community. Auf diesen Konferenzen gibt es ein festgelegtes Verfahren, um die Qualität der veröffentlichten Patterns sicherzustellen. Bevor dieses Verfahren erläutert werden kann, muss aber zuerst betrachtet werden, was die Qualität eines Patterns ausmacht.

Doug Lea benennt in einem Paper [Le94] sechs Eigenschaften, die ein Pattern idealerweise erfüllen sollte:

1. *Encapsulation (Einkapselung)*: Patterns beinhalten ein klar definiertes Problem und dessen Lösung. Außerdem muss klar sein, wann und warum das Pattern angewandt werden kann.

2. *Generativity (Generativität)*: Patterns sollten so geschrieben sein, dass sie dabei helfen, die vorgeschlagene Lösung umzusetzen.
3. *Equilibrium (Ausgeglichenheit)*: Die Ausführung der Lösung muss dazu führen, dass die Forces aufgelöst oder zumindest minimiert werden. Es sollte klar werden, wie die Lösung das erreicht.
4. *Abstraction (Abstraktion)*: Patterns sollten eine gewisse Generalität aufweisen und von empirischen Erfahrungen und alltäglichem Wissen abstrahieren.
5. *Openness (Offenheit)*: Patterns sollten offen sein gegenüber Erweiterungen durch andere Patterns. So kann es zum Beispiel hilfreich sein, zu einem Pattern eine Reihe von Sub-Patterns anzugeben, die erläutern, wie kleinere Bestandteile des Gesamt-Problems gelöst werden können.
6. *Composibility (Zusammensetzbarkeit)*: Alle Patterns können zu einem größeren Ganzen zusammengesetzt werden. Deswegen müssen die Patterns in sinnvoller Beziehung zueinander stehen.

4.1.6 Pattern-Kultur

Um sicherzustellen, dass auf PLoP-Konferenzen Patterns veröffentlicht werden, die den genannten Qualitätskriterien gerecht werden, gibt es in der Pattern-Community ein strenges Vorgehen, das zur Veröffentlichung eines Patterns hinführt.

Hat man bei einer PLoP-Konferenz ein Paper eingereicht und das Paper ist als relevant bewertet worden, wird das Paper zum sogenannten Shepherding [Ha99] zugelassen. Beim Shepherding handelt es sich um die Betreuung des Autors durch einen erfahrenen Pattern-Schreiber, den sogenannten Shepherd (Schäfer). Der Autor übernimmt in diesem Prozess die Rolle des sogenannten Sheep (Schaf). So wie der Schäfer seine Schafe anleitet, soll der Shepherd sein Sheep anleiten. Daher der Name Shepherding.

Der Shepherd sollte sich idealerweise in der Domäne der Patterns seines Sheeps auskennen. Auf jeden Fall muss er ein erfahrener Pattern-Schreiber sein. Er gibt dem Sheep detailliertes Feedback zu den Patterns und hilft so, die Qualität der Arbeit zu verbessern und sicherzustellen, dass die Patterns den Qualitätskriterien gerecht werden. Shepherd und Sheep haben das gleiche Ziel: Das Paper soll publiziert werden. Verglichen mit einem „normalen“ Review Prozess, wie er bei vielen Konferenzen Anwendung findet, ist das Shepherding deutlich intensiver. Der Shepherd begleitet sein Sheep über eine längere Zeit und trägt zur kontinuierlichen Verbesserung des Papers bei.

An das Shepherding schließt sich ein Review-Prozess durch das Programm-Komitee und alle Shepherds an, bei dem entschieden wird, ob ein Paper zur Konferenz zugelassen wird. Auf der Konferenz selber, finden sogenannte Writer´s-Workshops statt. Diese bieten den Rahmen für einen strukturierten Feedback-Prozess durch die Workshop-Teilnehmer.

Im Workshop diskutieren die Teilnehmer ein Paper, ohne dass der Autor an der Diskussion aktiv teilnimmt. Er sitzt etwas abseits der Diskussionsrunde und beobachtet diese, ohne sich einzumischen. Er kann sich Notizen machen und hat im Anschluss an die Diskussionsrunde die Möglichkeit, Rückfragen zu stellen, falls er etwas nicht

verstanden hat. Wichtig bei den Workshops ist, dass positive Aspekte eines Papers betont werden, damit sie bei Überarbeitungen erhalten bleiben und ausgebaut werden können. Für negative Aspekte suchen die Workshop-Teilnehmer nach möglichen Verbesserungen. Diese Workshops liefern dem Autor eine Reihe wichtiger Erkenntnisse darüber, wie andere Leser seine Patterns auffassen und wo noch Raum für Verbesserungen ist. Die Autoren sollten anschließend an die Konferenz ihre Papers basierend auf den im Workshop gewonnenen Einsichten verbessern. Erst danach werden die Patterns veröffentlicht.

Verglichen mit vielen anderen Konferenzen ist das Vorgehen bei Pattern-Konferenzen sehr strukturiert und wesentlich aufwendiger. Wurde ein Paper auf einer PLoP-Konferenz publiziert, so beruht es nicht allein auf der Meinung der Autoren, sondern enthält oft auch Erfahrungen anderer Personen.

4.2 Anpassungspatterns

In dieser Arbeit soll ein Werkzeug zur besseren Unterstützung von Anpassungsprozessen erstellt werden. Wie in Kapitel 2 erläutert, wurden zu Beginn dieser Arbeit basierend auf einer Expertenumfrage Anforderungen an ein Werkzeug zur Unterstützung der hierfür benötigten Prozesse gesammelt. In dieser Befragung wurden die Experten auch gebeten, zu erläutern, wie sie bei der Durchführung der Anpassungsprozesse vorgehen.

Gemäß dem im vorigen Kapitel vorgestellten Konzept wurden die in der Befragung ermittelten Erkenntnisse über Anpassungsprozesse (vergleiche Abschnitt 2.2) in Form von Patterns notiert. Daraus entstand eine Menge initialer Patterns, die mit dem vorhin genannten Sociological Approach gefunden worden sind und somit auf Expertenwissen beruhen.

In einem zweiten Schritt wurden die Patterns Experten in der Ausführung der jeweils beschriebenen Prozesse vorgelegt. Die Patterns wurden gemäß dem Feedback der Experten verfeinert. Das führte zu einer verbesserten Version der Patterns. Da ein wichtiger Aspekt von Patterns ist, dass die in ihnen vorgestellte Lösung bereits mehrfach zum Erfolg geführt hat, wird außerdem für jedes Pattern sichergestellt, dass eine Reihe von „Known uses“ existiert. Einige davon wurden in die Patterns aufgenommen, um zu belegen, dass für jedes Pattern mindestens drei bekannte Verwendungen existieren. Bei einigen Anpassungen war es notwendig, mehrfach mit Experten zu sprechen. In diesen Fällen wurde der zweite Schritt mehrfach durchlaufen. Jeder Durchlauf hat zu einer Verbesserung des jeweiligen Patterns beigetragen.

In einem dritten Schritt wurden schließlich einige der Patterns auf Pattern Konferenzen publiziert [ZRS06b, ZRS07]. Die Patterns haben das Shepherding sowie den Writer´s Workshop durchlaufen und wurden dadurch erneut verbessert.

Das Pattern-Mining in Zusammenarbeit mit Prozessexperten stellt sicher, dass die Anpassungspatterns nicht die Meinung der Autorin widerspiegeln. Vielmehr basieren sie auf Expertenwissen, das methodisch in Zusammenarbeit mit Prozessexperten gewonnen wurde. Durch die Veröffentlichung der Patterns auf Pattern-Konferenzen wird zudem sicher gestellt, dass die Patterns hinsichtlich Form und Verständlichkeit durch eine

Reihe erfahrener Pattern-Schreiber überprüft worden sind. Abbildung 19 zeigt den besprochenen dreistufigen Prozess zu Pattern-Erstellung.

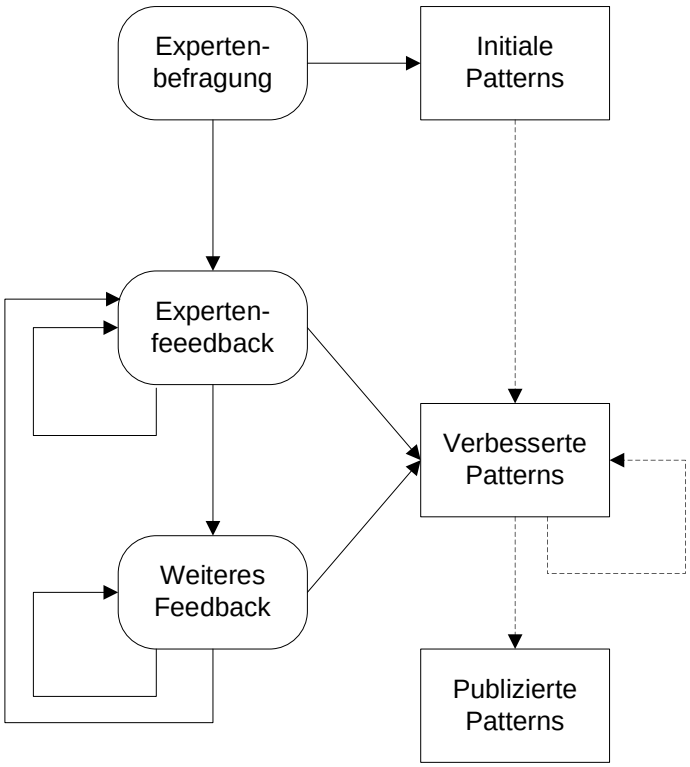


Abbildung 19: Prozess der Pattern Erstellung.

Die starke Betonung der Sammlung von Expertenwissen in den ersten zwei Schritten des eben vorgestellten Vorgehens zur Erstellung der Anpassungspatterns ist der Delphi-Methode [LT75] entnommen. Diese Methode ist ein strukturiertes Vorgehen zum Sammeln und Extrahieren von Expertenwissen. Sie basiert auf schriftlichen Expertenbefragungen [Fa04]. Im Gegensatz zur ursprünglichen Delphi-Vorgehensweise wurden in dieser Arbeit sowohl die Expertenbefragungen im ersten Schritt als auch die Feedbackrunde im zweiten Schritt der Erstellung der Anpassungspatterns nicht in schriftlicher Form durchgeführt. Es hätte die Experten unzumutbar viel Zeit gekostet, die Fragen in schriftlicher Form zu beantworten. Deswegen wurden strukturierte Telefoninterviews durchgeführt, die aufgezeichnet und anschließend verschriftlicht wurden. Basierend auf diesen Resultaten wurden die initialen Patterns, die den Ausgangspunkt für die Feedbackrunde im zweiten Schritt lieferten, sowie die durch das Feedback verbesserten Patterns erstellt.

Shepherding und Writer´s Workshops, die Elemente des dritten Schrittes der Erstellung der Anpassungspatterns, entsprechen dem traditionellen Vorgehen, um Patterns in den Proceedings einer PLoP-Konferenz zu veröffentlichen. Sie entstammen somit nicht der Delphi-Methode, tragen aber wesentlich dazu bei, dass die Qualität der Patterns verbessert wird. Die beiden ersten Schritte sichern ab, dass die Patterns inhaltlich korrekt sind, der dritte Schritt dient vor allem der Optimierung von Form und Verständlichkeit. Somit

sind die Anpassungspatterns, die diese drei Schritte durchlaufen haben, eine wertvolle Wissensquelle für andere Personen.

Anpassungspatterns beschreiben die zur Durchführung von Anpassungen benötigten Prozesse. Sie verwenden eine Notationsform, die an die POSA Form [Bu+96b] und die kanonische Form [6] angelehnt ist, da diese Notationsformen die meisten der Elemente enthalten, die für die Beschreibung der Anpassungsprozesse benötigt werden. Die Notationsform der Anpassungspatterns enthält die in Tabelle 3 aufgezählten Elemente.

Element	**Bedeutung**
Name	Eindeutige Identifizierung des Patterns.
Klassifikation	Sternchen, die den Grad des Vertrauens darstellen, das der Autor in das Pattern hat (kein Sternchen entspricht der niedrigsten Vertrauensstufe, zwei Sternchen der höchsten).
Absicht	Eine kurze Zusammenfassung des Zwecks des beschriebenen Prozesses
Kontext	Die Situation, in der der vom Pattern beschriebene Prozess auftreten kann.
Problem	Das Problem, das es durch den Prozess zu lösen gilt.
Beispiel	Hilft dem Benutzer, das Pattern zu verstehen, indem es ein Beispiel für eine erfolgreiche Anwendung des Patterns benennt.
Forces	Die Anforderungen, die beim Ausführen der Lösung beachtet werden müssen.
Lösung	Die Lösung beschreibt, wie der Prozess auszuführen ist, um das Problem zu lösen. Sie benennt auch die für die Ausführung benötigten Prozessschritte und deren Reihenfolge.
Bekannte Verwendungen	Benennt Situationen, wo die im Pattern enthaltene Prozessbeschreibung erfolgreich angewandt wurde.
Konsequenzen	Die positiven und negativen Konsequenzen, die sich aus der Ausführung der Lösung ergeben.
Verwandte Patterns	Patterns anderer Autoren, die ein ähnliches Problem behandeln.
Verbundene Patterns	Nach Ausführen der Lösung eines Patterns kann es notwendig sein, ein anderes Pattern auszuführen. Dieses wird im „connected patterns“ Element genannt.
Verwendete Patterns	Patterns, die in der Lösung genannte komplexe Prozessschritte beschreiben.

Tabelle 3: Elemente der Anpassungspatterns.

Da PLoP-Konferenzen internationale Konferenzen sind, wurden die Patterns in englischer Sprache veröffentlicht. Es gibt aber auch eine deutsche Übersetzung, da die Patterns die Grundlage für einen Wizard zur Unterstützung von Anpassungen darstellen, der auch in deutscher Sprache verfügbar sein sollte.

Abbildung 20 zeigt einen Ausschnitt aus dem Pattern zur Übersetzung. Im Anhang findet sich ein Beispiel für ein vollständiges Anpassungspattern.

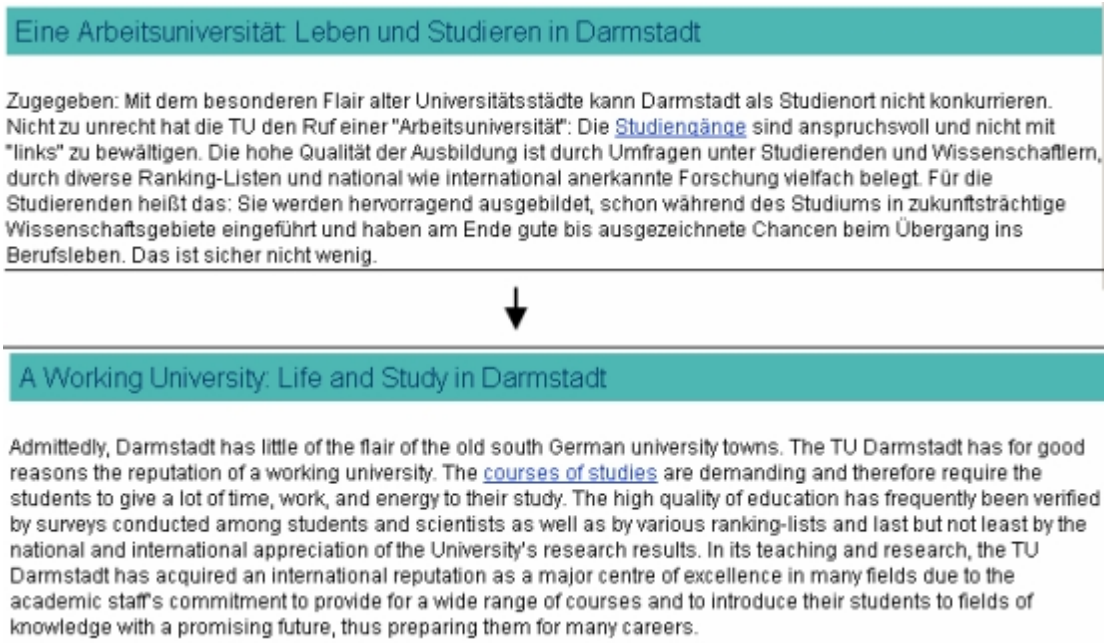
Translation *

Also known as: Multilingual Content Production

Intent: Provide content in a different language.

Context: Normally E-Learning material is first designed in one language. If versions in another language are needed the original version has to be translated.

Problem: You want to translate E-Learning content from one language to another language. Which process has to be performed to achieve a translated version?



Example for multilingual content

Abbildung 20: Ausschnitt aus dem Pattern zur Übersetzung [ZRS06b].

4.3 Ein Werkzeug zur Erstellung Pattern-basierter Prozessbeschreibungen

Wie im vorigen Kapitel erläutert, müssen die in den Patterns enthaltenen Prozessbeschreibungen in einer maschinenlesbaren Form vorliegen. Das Konzept dieser Arbeit

sieht daher vor, die Anpassungspatterns in einer an PLML (vergleiche Abschnitt 3.4) angelehnten XML-Notation zu speichern. Doch viele Experten in der Durchführung von Anpassungsprozessen beherrschen XML nicht ausreichend, um die Patterns problemlos im geforderten Format zur Verfügung zu stellen.

Um den Prozessexperten das Erstellen der Anpassungspatterns zu erleichtern, wurde daher ein prototypisches Werkzeug zur Pattern-Eingabe (PIT = Process description Input Tool) entwickelt. Der Prototyp wurde in Java 1.5 unter Eclipse 3.2 entwickelt. Er bietet ein strukturiertes Eingabeformular und unterstützt durch Hilfestellungen bei der Eingabe der Prozessbeschreibungen. Der Prozessexperte gibt seine Beschreibung in natürlicher Sprache in das Eingabeformular ein. Das PIT konvertiert die Eingabe beim Speichern in eine XML-Notation.

Eine Prozessbeschreibung muss den in Abschnitt 2.2 vorgestellten Aufbau von Anpassungsprozessen wiedergeben. Das bedeutet: Sie muss Anpassungsprozesse, Prozessschritte und atomare Unterschritte beinhalten. In den Prozessbeschreibungen werden die benötigten Prozessschritte aufgezählt, sowie deren Abhängigkeiten untereinander und Vorbedingungen für die Ausführung. Analog werden in den Prozessschrittbeschreibungen die benötigten atomaren Unterschritte aufgezählt, sowie deren Abhängigkeiten untereinander und Vorbedingungen für die Ausführung. Doch die ausführliche Beschreibung der Prozessschritte und Unterschritte erfolgt nicht innerhalb der Prozessbeschreibung, sondern separat, da sowohl Prozessschritte als auch Unterschritte in mehreren Prozessen beziehungsweise Prozessschritten verwendet werden können.

Ein weiterer Grund für die separate Beschreibung der Prozessschritte und atomaren Unterschritte ist die unterschiedliche Art der Beschreibung: Prozessbeschreibungen sind Beschreibungen eines Anpassungsprozesses auf oberster Ebene. Hier ist eine Reihe von Informationen relevant, die auf den unteren Ebenen keine Rolle mehr spielen. Hierzu zählen beispielsweise: In welcher Situation kann der Anpassungsprozess ausgeführt werden? Welche Probleme soll er lösen? Diese Informationen dienen vor allem dazu, entscheiden zu können, welcher Anpassungsprozess ausgeführt werden muss. Hat man sich für einen Anpassungsprozess entschieden, braucht man derartige Informationen für die Prozessschritte und atomaren Unterschritte nicht mehr.

Weiterhin fördert eine Trennung der Beschreibungen die Lesbarkeit. Würde man alle Beschreibungen ineinander verschachteln, so wäre eine vollständige Prozessbeschreibung sehr lang und nur noch schwer verständlich. Die Trennung in Beschreibungen der einzelnen Elemente jeder Ebene ermöglicht es dem Leser, sich den Prozess Ebene für Ebene anzuschauen und so gezielt die Teile der Beschreibung zu betrachten, die für ihn relevant sind.

Die mit dem PIT erzeugten Beschreibungen werden als XML-Dateien gespeichert: Für jeden Anpassungsprozess werden zwei Dateien erzeugt: Eine Datei, die die textuellen Beschreibungen des Anpassungsprozesses, seiner Prozessschritte und Unterschritte enthält und eine zweite Datei, die eine Darstellung der Struktur des Anpassungsprozesses enthält. Der Aufbau beider Dateien wird im Laufe dieses Abschnittes erläutert.

Abbildung 21 zeigt einen Ausschnitt der Oberfläche des Werkzeuges PIT.

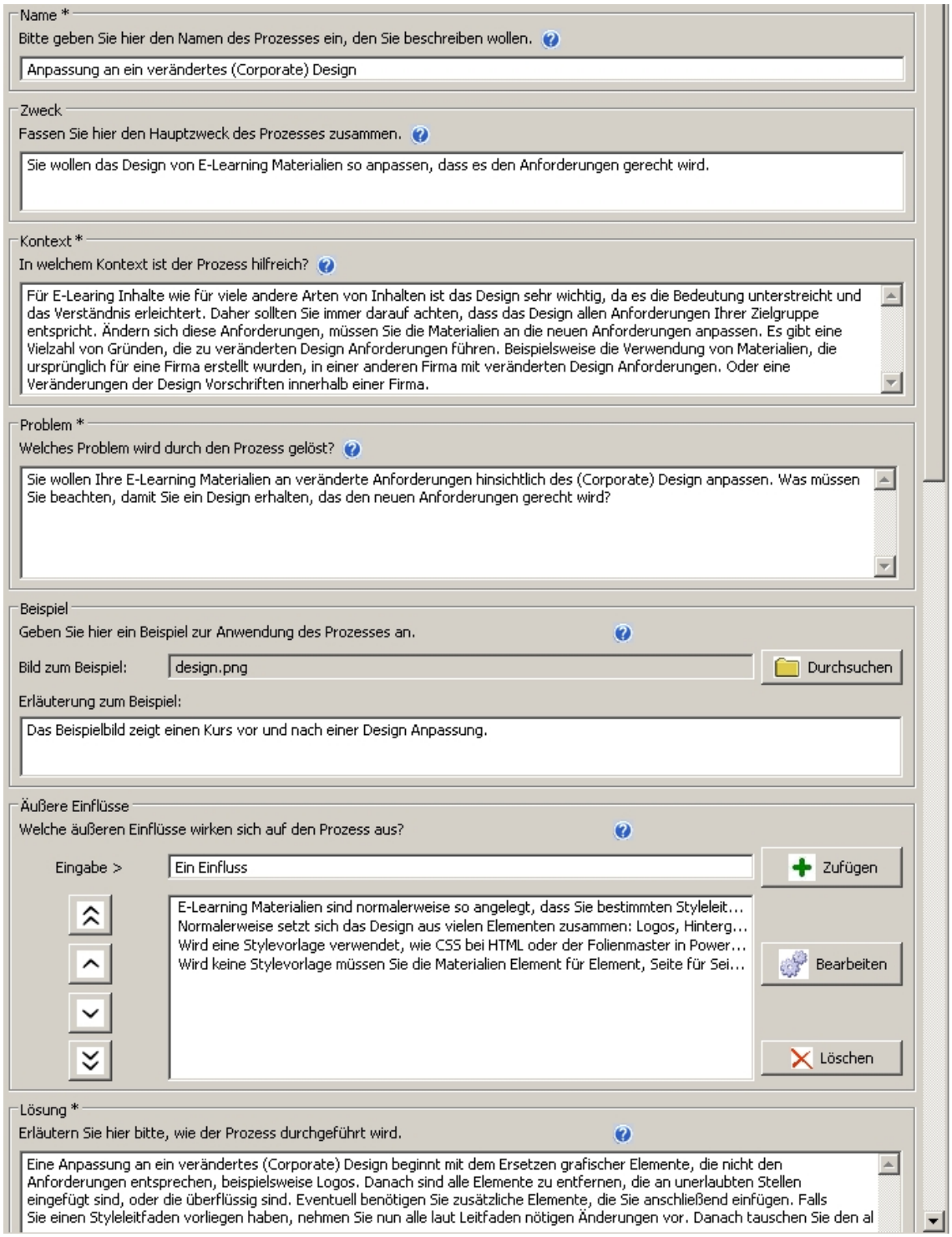


Abbildung 21: Das Prozessbeschreibungseingabewerkzeug.

Damit ein Pattern auch den Qualitätsansprüchen der Pattern-Gemeinde gerecht wird und man von einem Pattern sprechen kann, muss es einer Reihe von Anforderungen genügen. Diese wurden in Abschnitt 4.1.5 erläutert. Die in Abschnitt 4.2 vorgestellten Anpassungspatterns erfüllen diese Anforderungen. Doch die Anpassungspatterns

beschreiben derzeit nicht alle vorhandenen Anpassungsprozesse. Wenn ein Prozessexperte mit dem PIT eine Prozessbeschreibung für einen weiteren Anpassungsprozess erstellt, so wird von PIT nicht geprüft, ob ein Pattern erstellt wurde, das den Qualitätskriterien für Patterns entspricht. PIT stellt lediglich sicher, dass alle Angaben gemacht werden, die zur Erzeugung eines prototypischen Wizards notwendig sind. Die hierzu benötigten Eingabefelder werden im PIT durch Sternchen hinter dem Namen des Eingabefeldes als verpflichtende Elemente gekennzeichnet.

Folgende der in 4.2 genannten Pattern-Elemente sind bei der Erstellung einer Prozessbeschreibung verpflichtend bzw. nicht verpflichtend:

Verpflichtende Elemente	**Nicht verpflichtende Elemente**
ID	Klassifikation der Konfidenz
Name	Zweck
Kontext	Beispiel
Problem	Forces (Äußere Einflüsse)
Solution und darin enthaltene Prozessschritte	Bekannte Verwendungen
Konsequenzen	Verwandte Patterns
	Verbundene Patterns
	Verwendete Patterns

Tabelle 4: Verpflichtende und nicht verpflichtende Elemente einer Prozessbeschreibung.

Erfüllt die Beschreibung eines Anpassungsprozesses alle Qualitätskriterien und lässt sich somit als Pattern bezeichnen, so lässt sich aus ihr nicht nur ein sinnvoller Wizard generieren. Der Wizard enthält zusätzlich noch eine Reihe wertvoller Informationen, die anderen Personen helfen, nicht nur den unterstützten Anpassungsprozess ausführen zu können, sondern darüber hinaus ein tieferes Verständnis vom Prozess zu entwickeln. Eine Prozessbeschreibung muss aber nicht all diesen Anforderungen gerecht werden; und dennoch lässt sich aus ihr ein sinnvoller Wizard erzeugen. Auch Prozessbeschreibungen, die nicht den Qualitätsansprüchen eines Patterns gerecht werden, bieten wertvolles Wissen und eine gute Grundlage zur Wizard-Erzeugung.

Neben den in 4.2 genannten Pattern-Elementen gibt es noch zwei weitere Elemente, die zwar nicht zum Pattern gehören, aber notwendig sind, um einen prototypischen Wizard erzeugen zu können: Die ID des Patterns und die Prozessschritte. Die Pattern-ID ist eine eindeutige ID, die verwendet wird, um das Pattern adressieren zu können. Sie wird automatisch durch das PIT generiert. Ein Anpassungsprozess setzt sich, wie in 2.2 erläutert, aus einer Reihe von Prozessschritten zusammen. Die Lösung beschreibt die Ausführung eines Anpassungsprozesses und benennt auch die hierfür benötigten Prozessschritte. Daher sind bei den Anpassungspatterns die Prozessschritte Teil der Lösung. Da die Prozessschritte für die Ausführung des Prozesses essentiell sind, werden sie im PIT über ein separates, verpflichtendes Element angelegt.

Auch die nicht verpflichtenden Elemente sind wichtig, da sie anderen Personen wertvolle Informationen bieten, aber es ist auch möglich, ohne sie einen funktionsfähigen Prototyp zu erstellen. Bei den letzten drei Elementen (verwandte, verbundene und verwendete Patterns) kann es zudem vorkommen, dass keine derartigen Patterns bekannt sind und die Angaben daher nicht gemacht werden können.

Speichert der Prozessexperte eine Prozessbeschreibung, so werden die enthaltenen Informationen in zwei XML-Dateien abgelegt. Diese Dateien enthalten zum einen die textuelle Beschreibung des Prozesses und zum anderen, in einer separaten Datei, Informationen zum Ablauf des Anpassungsprozesses, sowie zu Abhängigkeiten und Vorbedingungen der Prozessschritte. Die Trennung in Struktur und Beschreibung ermöglicht eine leichtere Weiterverarbeitung und erhöht die Verständlichkeit. Die textuellen Informationen werden aus den Bestandteilen der Patterns generiert. Nachfolgend wird die DTD der XML-Datei gezeigt, die die textuellen Prozessinformationen enthält.

```
<!ELEMENT pattern (intent?, context, problem, example_illustration?,
example_explanation?, forces, solution, process_steps, known_uses*,
consequences, related-patterns, connected_patterns, used_patterns)>
      <!ATTLIST pattern id ID #REQUIRED
                        level-of-confidence CDATA #IMPLIED
                        name CDATA #REQUIRED >
<!ELEMENT intent (#PCDATA)>
<!ELEMENT context (#PCDATA)>
<!ELEMENT problem (#PCDATA)>
<!ELEMENT example_illustration (#PCDATA)>
<!ELEMENT example_explanation (#PCDATA)>
<!ELEMENT forces (force*)>
<!ELEMENT force EMPTY>
      <!ATTLIST force name CDATA #REQUIRED>
<!ELEMENT solution (#PCDATA)>
<!ELEMENT Process_steps (Process_step+)>
<!ELEMENT Process_step EMPTY>
      <!ATTLIST Process_step
            name Name #REQUIRED
            mandatory (true | false) “true”>
<!ELEMENT known_uses (#PCDATA)>
<!ELEMENT consequences
      (positive_consequence+, negative_consequence*)>
<!ELEMENT positive_consequence EMPTY>
      <!ATTLIST positive_consequence name CDATA #REQUIRED>
<!ELEMENT negative_consequence EMPTY>
      <!ATTLIST negative_consequence name CDATA #REQUIRED>
<!ELEMENT related_patterns (related_pattern*)>
<!ELEMENT related_pattern EMPTY>
      <!ATTLIST related_pattern
            name CDATA #REQUIRED
      description CDATA #REQUIRED
      >
<!ELEMENT connected_patterns (connected_pattern*)>
<!ELEMENT connected_pattern EMPTY>
      <!ATTLIST connected_pattern
            name CDATA #REQUIRED
            patternID ID #REQUIRED
      description CDATA #REQUIRED
```

```
      >
<!ELEMENT used_patterns (used_pattern*)>
<!ELEMENT used_pattern EMPTY>
      <!ATTLIST used_pattern
            name CDATA #REQUIRED
            patternID ID #REQUIRED
      >
```

Listing 1: DTD der textuellen Prozessbeschreibung

Viele der hier aufgeführten Elemente sind PLML [18] entnommen. Die DTD von PLML v1.1 enthält allerdings auch Elemente, die in den Patterns für die Beschreibung von Anpassungsprozessen nicht berücksichtigt wurden, da sie hier keine Rolle spielen. Diese Elemente sind: `alias`, `synopsis`, `diagram`, `rationale`, `literature`, `pattern-link` und `management`.

Außerdem finden sich in obiger DTD einige Elemente, die für die Beschreibung von Anpassungsprozessen relevant sind, aber in PLML nicht genannt werden. Diese sind:

- `Intent:` Dies ist ein wichtiges Element, da es kurz zusammenfasst, was das Ziel bzw. der Zweck des Patterns ist.
- `known_uses`: Die known_uses beschreiben Fälle, wo das Pattern bereits erfolgreich angewendet wurde. Für Patterns ist es sehr wichtig, dass dokumentiert ist, wo die Patterns eingesetzt wurden.
- `consequences`: Die Konsequenzen treten nach Anwendung des Patterns auf. Für Leser von Patterns ist es eine wichtige Information, zu wissen, welche positiven und negativen Veränderungen sich aus der Ausführung des Patterns ergeben.

Weiterhin gibt es in der oben vorgestellten DTD einige Elemente, die in PLML zwar vorkommen, dort aber eine geringfügig andere Bedeutung haben. So hat das Element `illustration` aus PLML hier den Namen `example_illustration`, was klarer verdeutlich, dass es sich bei diesem Element um eine Illustration des Beispiels handelt. PLML verwendet außerdem das Element `example`. Dieses wurde in obiger DTD mit `example_explanation` bezeichnet, zur besseren Abgrenzung vom Element `example_illustration`. Der Abschnitt `implementation` in PLML zielt klar auf eine programmiersprachliche Umsetzung ab. Im Falle der Prozessbeschreibung mit dem PIT ist diese Sicht irrelevant. Aber es ist essentiell, welche Prozessschritte umgesetzt werden müssen. Daher erhielt das Element den Bezeichner `process_steps`. Es beschreibt die zur Umsetzung der Lösung benötigten Prozessschritte. Weiterhin gibt es drei Arten von Patterns, die in Beziehung zu einem Pattern stehen können. Diese sind bei PLML alle unter `related_patterns` zusammengefasst. In obiger DTD wird zwischen `related_patterns`, `connected_patterns` und `used_patterns` unterschieden. Außerdem enthalten `connected_patterns` und `used_patterns` einen Link auf das Pattern, auf das sie sich beziehen als Attribut. Somit war ein separates Element `pattern-link`, wie in PLML verwendet, überflüssig.

Jede mit dem PIT erzeugte Pattern-basierte Beschreibung definiert einen Anpassungsprozess. Um den Ablauf eines Prozesses beschreiben zu können, enthält der Lösungsabschnitt jeder Prozessbeschreibung eine Auflistung der im Anpassungsprozess benötigten

Prozessschritte. In der XML-Datei finden sich diese im Element `process_steps`. Zusätzlich muss für jeden Prozessschritt angegeben werden, ob dessen Ausführung verpflichtend ist oder nicht, und ob er von der Ausführung oder dem Ergebnis anderer Schritte abhängt. Dadurch lässt sich ein Graph erstellen, der die Ablauflogik des Anpassungsprozesses beschreibt.

Exemplarisch sei hier ein stark vereinfachter Prozess zur Terminologieanpassung betrachtet. Dieser Prozess enthält 3 Prozessschritte:

PS1: Alle Vorkommen eines Begriffes ersetzen

PS2: Alle Vorkommen mehrerer Begriffe ersetzen

PS3: Die Anordnung der Elemente nach der Ersetzung prüfen und korrigieren

Die Durchführung des letzten Prozessschrittes ist verpflichtend, da alle Ersetzungen dazu führen können, dass sich die Anordnung der Elemente verändert hat und eventuell korrigiert werden muss. Die Reihenfolge der Durchführung der Prozessschritte ist festgelegt: Zuerst muss entschieden werden, ob PS1 oder PS2 durchgeführt werden soll. Je nachdem, wie die Entscheidung ausfällt, wird PS1 oder PS2 ausgeführt. Anschließend muss PS3 ausgeführt werden. Abbildung 22 zeigt ein Aktivitätsdiagramm, das den Kontrollfluss des Prozesses darstellt.

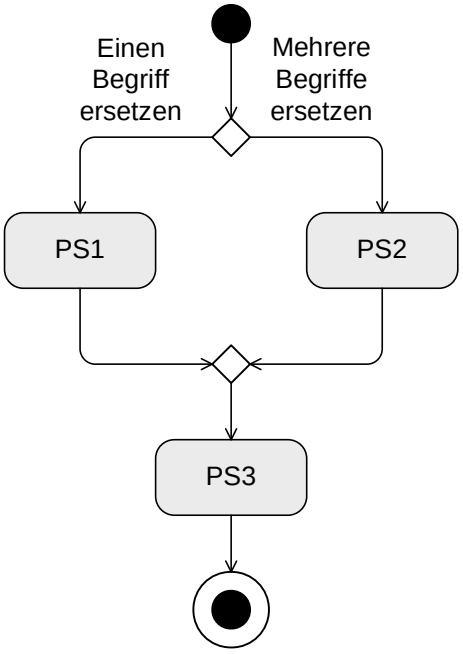


Abbildung 22: Aktivitätsdiagramm zur Terminologieanpassung.

Wie in Abschnitt 2.2 erläutert, setzt sich ein Anpassungsprozess aus Prozessschritten zusammen (vergleiche Abbildung 23). Zwischen den Prozessschritten können Abhängigkeiten bestehen. Außerdem kann es Vorbedingungen für die Ausführung der Prozessschritte geben. Alle diese Elemente sind notwendig, um den Aufbau und Ablauf eines Prozesses eindeutig beschreiben zu können. Weiterhin wichtig sind Verzweigungen und Zyklen im Ablauf des Anpassungsprozesses. Dies ist für einen Prozessexperten, der

sich nicht mit der Modellierung von Prozessen auskennt, schwer zu beschreiben. Deswegen enthält das PIT einen Dialog, der bei der Definition der Prozessschritte unterstützt (vergleiche Abbildung 24 auf Seite 68). Er wird durch einen Button gestartet, der es erlaubt, die benötigten Prozessschritte zu einer Prozessbeschreibung anzulegen. Dieser Dialog ermöglicht eine genaue Spezifikation des Prozessablaufs basierend auf den Prozessschritten, ohne dass dafür ein besonderes Wissen notwendig ist.

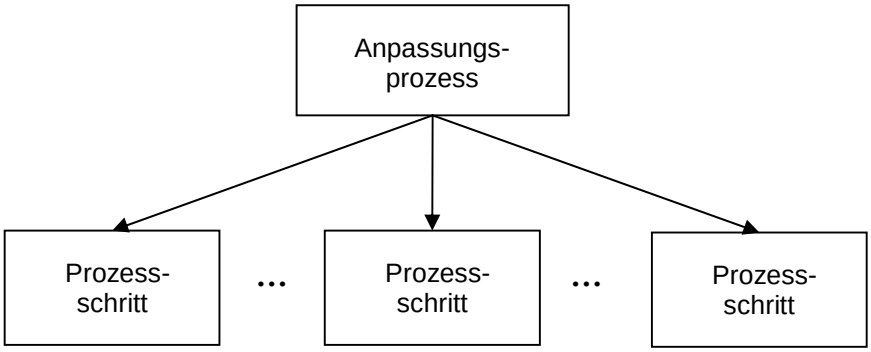


Abbildung 23: Anpassungsprozesse setzen sich aus Prozessschritten zusammen.

Es gibt folgende Elemente, die für die Beschreibung des Ablaufs der Prozessschritte eines Anpassungsprozesse im PIT zur Verfügung stehen:

- **Prozessschritte** stellen die Aktivitäten des Anpassungsprozesses dar. Im Prozessgraph entsprechen die Prozessschritte den Knoten.
- Die Prozessschritte werden in einer bestimmten **Reihenfolge** ausgeführt. Die gerichteten Kanten zwischen den Knoten im Prozessgraph repräsentieren die Reihenfolge.
- Die Ausführung von Prozessschritten kann **freiwillig** oder **verpflichtend** sein.
- **Abhängigkeiten** legen fest, dass ein Prozessschritt nur dann ausgeführt werden darf, wenn vorher eine Entscheidung getroffen wurde. Abhängig vom Resultat der Entscheidung, wird festgelegt, ob der Prozessschritt ausgeführt werden darf oder nicht. Beispielsweise muss man entscheiden, ob man einen oder mehrere Begriffe ersetzen will. Je nach Entscheidung, wird der Prozessschritt „Alle Vorkommen eines Begriffes ersetzen“ ausgeführt oder nicht.
- **Vorbedingungen**: Ein Prozessschritt B wird dann ausgeführt, wenn vorher ein Prozessschritt A ausgeführt wurde. Es kann auch sein, dass der Prozessschritt nur dann ausgeführt wird, wenn vorher mindestens ein Prozessschritt aus einer Menge von Prozessschritten ausgeführt wurde. So kann man im Beispielprozess die Anordnung der Elemente erst prüfen und korrigieren, wenn man vorher Begriffe ersetzt hat.

Vorbedingungen sind Spezialfälle von Abhängigkeiten: „Schritt A ist Vorbedingung für Schritt B“ entspricht der Entscheidung: „Wurde Schritt A ausgeführt? Dann Schritt B ausführen. Andernfalls Schritt B nicht ausführen.“ Da der Wizard die Entscheidung treffen kann, ob ein bestimmter Prozessschritt ausgeführt wurde (basierend auf der

Rückmeldung des Anwenders), wurde zur besseren Bedienbarkeit zusätzlich zur Abhängigkeit die Vorbedingung eingeführt.

Bei vielen Anpassungsprozessen ist die Reihenfolge der Ausführung einzelner Prozessschritte nicht festgelegt. Führt man den Prozess aus, kann man entscheiden, in welcher Reihenfolge man die Prozessschritte ausführen möchte. Da das für Laien verwirrend sein kann, ermöglicht das PIT nur eine strikt festgelegte, sequentielle Reihenfolge. Der Prozessexperte muss sich also für die Reihenfolge entscheiden, die ihm am sinnvollsten erscheint.

Neben der Sequenz lassen sich in der Prozessstruktur Selektion, Parallelisierung und Iteration unterscheiden [AH02]. Die Selektion wird durch die eben benannten Abhängigkeiten realisiert: Eine Abhängigkeit bedeutet, dass die Ausführung eines Prozessschrittes davon abhängt, ob eine Entscheidung mit einem bestimmten Ergebnis getroffen wurde. Abhängig vom Ergebnis der Entscheidung wird also der nächste auszuführende Prozessschritt ausgewählt.

Aktuell werden bei Abhängigkeiten nur binäre Entscheidungen berücksichtigt. Allerdings gibt es auch immer wieder Situationen, wo nicht zwischen zwei Fällen entschieden werden muss, sondern eine komplexere Fallunterscheidung notwendig ist. Aktuell ist das durch die Aneinanderreihung mehrerer binärer Entscheidungen zu beschreiben. Da dies aber nicht intuitiv ist, sollte in einer Weiterentwicklung des hier vorgestellten Prototyps die Möglichkeit vorgesehen werden, auch zwischen mehr als zwei Fällen unterscheiden zu können.

Parallelität von Prozessschritten oder atomaren Unterschritten kann mit dem PIT nicht beschrieben werden. Das hat folgenden Gründe: Der aus den PIT-Beschreibungen zu erstellende prototypische Wizard führt Benutzer mittels einer Abfolge von Dialogen durch einen Prozess, die dem Anwender nacheinander angezeigt werden und somit die Reihenfolge der durchzuführenden Aufgaben vorgeben. Diese kann der Prozessexperte beschreiben. Oft weiß der Prozessexperte aber nicht, ob sich bestimmte Tätigkeiten zur Laufzeit parallelisieren lassen. Deswegen wurde im PIT nur die sequentielle Abfolge von Prozessschritten und atomaren Unterschritten vorgesehen. Eine Parallelisierung der Ausführung kann während der Automatisierung des prototypischen Wizards durch einen Entwickler vorgenommen werden.

Derzeit im PIT-Prototyp nicht berücksichtigt sind Iterationen im Prozessablauf. Da diese in der Realität aber immer wieder der Fall sind, sollten sie in einer nächsten Version des Programms vorgesehen werden. Um sicherzustellen, dass eine Iteration beendet werden kann, muss ein Prozessschritt als Startpunkt der Iteration definiert werden und ein weiterer als Endpunkt. Nachdem der Endpunkt erreicht wurde, muss vor einem erneuten Starten der Iteration am Startpunkt entschieden werden, ob ein Abbruchkriterium erreicht wurde, oder ob die Iteration erneut durchlaufen werden soll. Dabei wäre zu berücksichtigen, dass falsch definierte Abbruchkriterien dazu führen können, dass der Wizard nicht terminieren kann.

Abbildung 24 zeigt einen Prozessschritt „Die Anordnung der Elemente nach der Ersetzung überprüfen“ aus der Terminologieanpassung. Dieser Schritt muss ausgeführt werden, falls vorher einer der ausgewählten anderen Prozessschritte ausgeführt wurde („Alle Vorkommen eines Begriffes ersetzen“, „Alle Vorkommen mehrerer Begriffe

ersetzen“). Wird keiner der anderen Prozessschritte ausgeführt, so darf der Schritt nicht ausgeführt werden. Das wird durch eine Vorbedingung ausgedrückt.

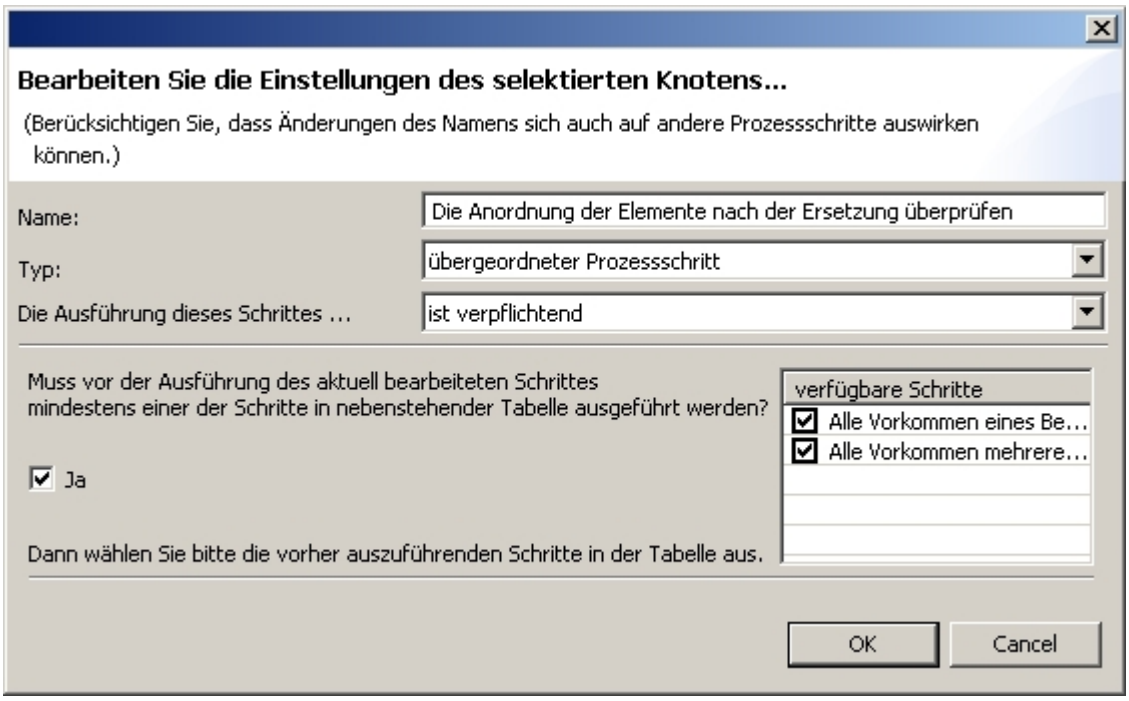


Abbildung 24: Vorbedingungen zwischen Prozessschritten definieren.

Neben der in Abbildung 24 dargestellten Erstellung und Bearbeitung von Vorbedingungen ist es für Prozessschritte auch möglich, Abhängigkeiten zu anderen Schritten festzulegen. Hierfür gibt es eine weitere Dialog-Seite, die es erlaubt, Abhängigkeiten zu beschreiben.

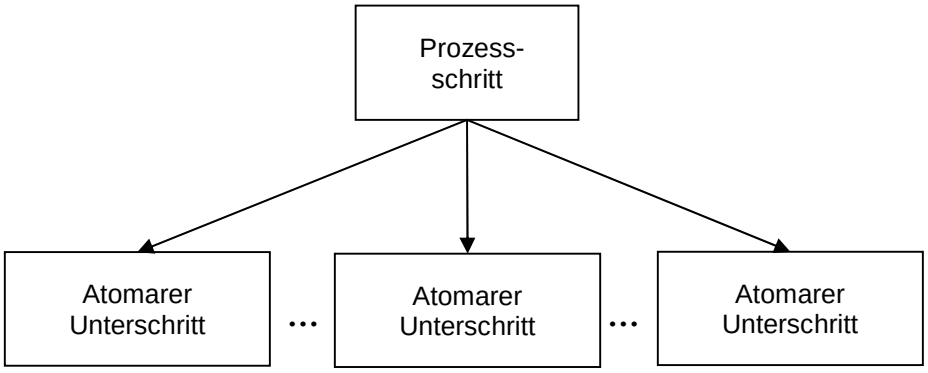


Abbildung 25: Prozessschritte enthalten atomare Unterschritte.

Neben der Erläuterung des gesamten Anpassungsprozesses durch eine Pattern-basierte Beschreibung sind die einzelnen Prozessschritte durch sogenannte *HowTos* zu erläutern, die auch mit dem PIT erstellt werden können (Abbildung 26). Die HowTos werden in einem separaten Abschnitt in der gleichen Datei, wie der zugehörige Anpassungsprozess gespeichert. Verglichen mit der ausführlichen Beschreibung des Gesamtprozesses,

sind sie deutlich kürzer: Sie enthalten den Namen des Prozessschrittes, dessen Zweck, eine Erläuterung zur Vorgehensweise und sie zählen die kleineren Prozessschritte und / oder atomaren Unterschritte auf, aus denen der Prozessschritt besteht (vergleiche Abbildung 25).

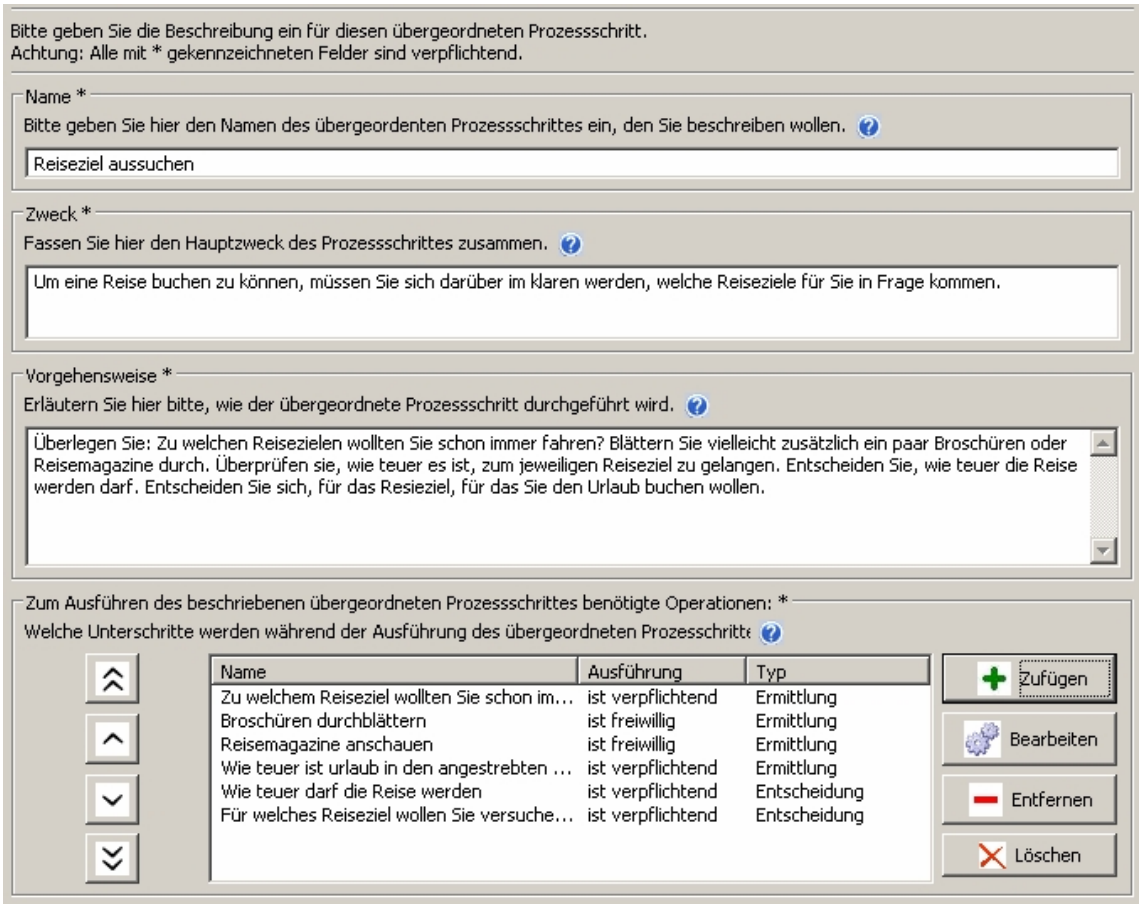


Abbildung 26: Eingabe von HowTos.

In einigen Fällen sind Prozessschritte so komplex, dass es sinnvoll ist, sie ebenfalls ausführlich mit einer Pattern-basierten Beschreibung zu erläutern. Das PIT ermöglicht es, in diesen Fällen einen Prozessschritt genauso ausführlich wie einen Anpassungsprozess zu beschreiben. Bei der Wizard-Erstellung werden diese speziellen Prozessschrittbeschreibungen gesondert interpretiert, so dass Nutzer des Wizards auf die enthaltenen Informationen zugreifen können.

Prozessschritte setzen sich aus atomaren Unterschritten zusammen. Analog dem Anlegen von Prozessschritten gibt es auch einen Dialog, der Prozessexperten beim Anlegen der atomaren Unterschritte unterstützt (vergleiche Abbildung 27). Auch hier gibt es die Möglichkeit, freiwillige oder verpflichtende Ausführung, sowie Vorbedingungen und Abhängigkeiten zu definieren. Zusätzlich muss sich der Prozessexperte entscheiden, welchem Typ ein atomarer Unterschritt angehört. Wie in Abschnitt 2.2 erläutert, gibt es drei Arten von atomaren Unterschritten:

- *Ermittlungen* werden zum Einholen von Informationen benötigt. Beispielsweise: Auffinden aller in einer Lernressource verwendeten grafischen Elemente (z.B. Bilder, Fotos, Diagramme).
- *Entscheidungen* werden benötigt, wenn die Person, die den Prozess ausführt, etwas entscheiden muss: Beispielsweise die Entscheidung, ob es grafische Elemente gibt, die nicht konform den Vorgaben sind.
- *Ausführungen* werden benötigt, wenn etwas ausgeführt wird. Beispielsweise: Löschen eines ausgewählten grafischen Elementes.

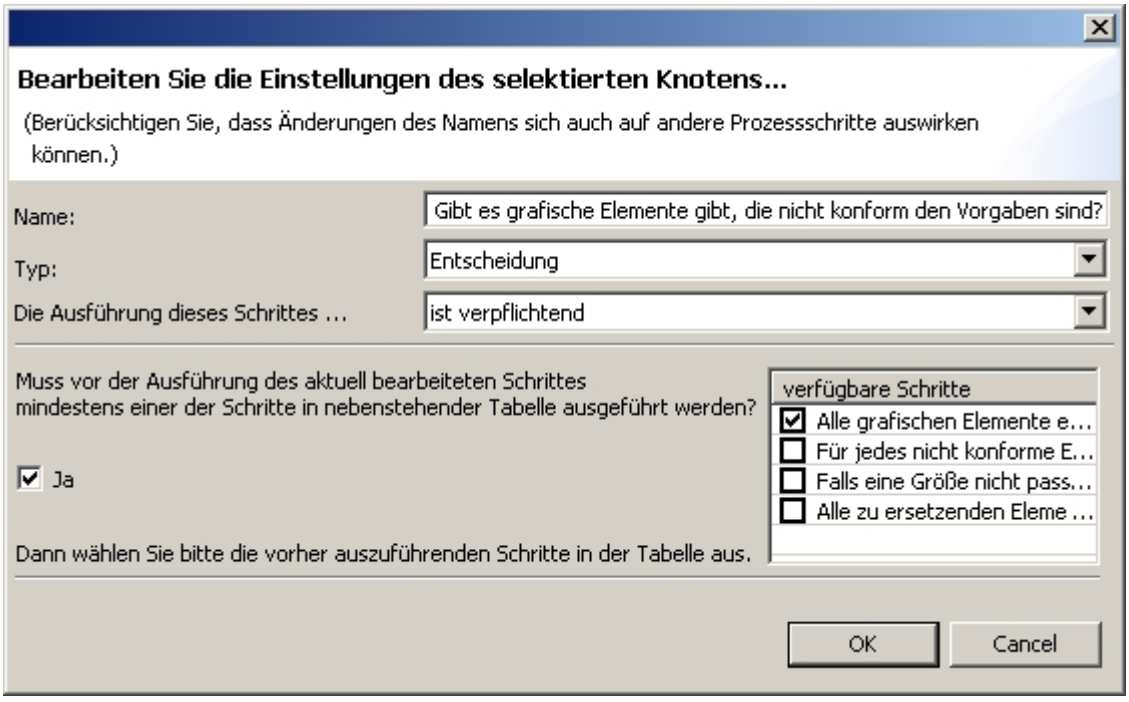


Abbildung 27: Anlegen von atomaren Unterschritten.

Für jeden atomaren Unterschritt ist zusätzlich eine kurze Beschreibung anzugeben. Diese beinhaltet eine Erläuterung, wie der Unterschritt auszuführen ist und was bei der Ausführung zu beachten ist. Die Beschreibung kann über ein weiteres Eingabeformular erstellt werden (Abbildung 28).

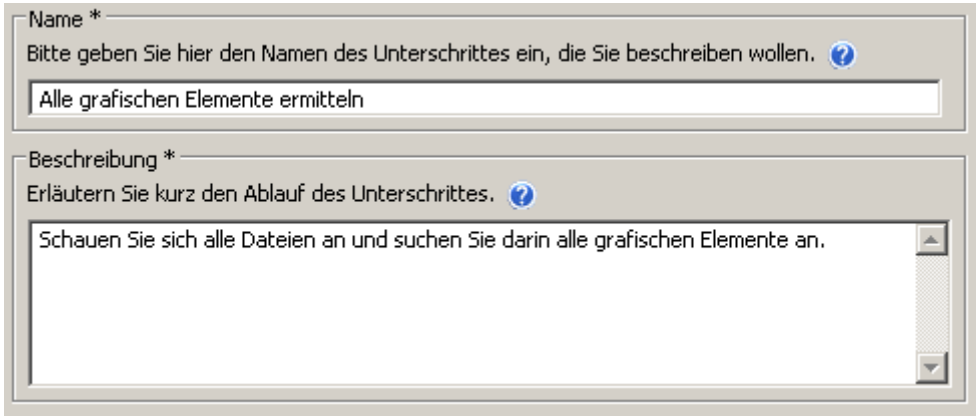


Abbildung 28: Anlegen der Beschreibung eines atomaren Unterschrittes.

Die für die Lösung benötigten Schritte (Prozessschritte und atomare Unterschritte) entsprechen Knoten in einem Graphen. Die Prozessexperten geben bei der Erstellung der Anpassunspatterns und HowTos mögliche Wege durch die Knoten vor. Diese entsprechen den Kanten zwischen den Knoten. Außerdem definieren sie, welche Knoten verpflichtend durchzuführen sind. Sie legen Abhängigkeiten zwischen Knoten und Vorbedingungen für Knoten fest. Basierend auf diesen Informationen wird eine XML-Datei erzeugt, die die Ablauflogik des Anpassungsprozesses beinhaltet. Über die eindeutige ID von Prozessen, Prozessschritten und atomaren Unterschritten ist es möglich, die enthaltenen Elemente den Elementen der textuellen Beschreibung zuzuordnen.

Nachfolgendes Beispiel zeigt einen Ausschnitt aus dem in Abbildung 22 gezeigten Prozess-Graph. Man sieht, dass jeder Anpassungsprozess eindeutig referenziert wird durch seine ID. Alle Prozessschritte, die zur Ausführung des Anpassungsprozesses benötigt werden, sind im Abschnitt `requires` aufgelistet. Für jeden Prozessschritt ist notiert, ob dessen Ausführung verpflichtend ist (`mandatory="true"`) oder nicht. Der letzte der aufgeführten Prozessschritte wird nur ausgeführt, wenn vorher mindestens einer der beiden vorigen Schritte ausgeführt wurde. Deswegen sind diese zwei Schritte im Abschnitt `precondition` beim letzten Prozessschritt aufgelistet. Für den ersten Prozessschritt sieht man außerdem die dafür benötigten atomaren Unterschritte. (Die anderen Unterschritte werden in diesem Ausschnitt nicht gezeigt.) Auch hier wird notiert, ob ein Unterschritt ausgeführt werden muss oder nicht. Außerdem kann man sehen, dass für jeden atomaren Unterschritt angegeben wird, welcher der drei Arten (Ermittlung = `query`, Entscheidung = `decision`, Ausführung = `execution`) er angehört.

```
<process id="process_pattern$56667" process-pattern="true">
  <requires fragmentRef="process_step$22357234" mandatory="false"/>
  <requires fragmentRef="process_step$28757034" mandatory="false"/>
  <requires fragmentRef="process_step$50025522" mandatory="true">
    <precondition fragmentRef="process_step$22357234"/>
    <precondition fragmentRef="process_step$28757034"/>
  </requires>
</process>
<process-step id="process_step$22357234">
  <requires functionRef="query$28693719" mandatory="true"/>
  <requires functionRef="decision$26294026" mandatory="true"/>
  <requires functionRef=" execution$24376617" mandatory="true"/>
```

```
  <requires functionRef="decision$23537464" mandatory="false"/>
  <requires functionRef=" execution$32687500" mandatory="false"/>
  <requires functionRef=" execution$22575321" mandatory="true"/>
</process-fragment>
```

Listing 2: Teil einer Prozessgraph-Datei.

Das PIT bietet verschiedene Möglichkeiten, um mit den generierten Prozessbeschreibungen zu arbeiten:

- Man kann bereits erstellte Beschreibungen editieren und so ggf. an veränderte Bedingungen anpassen.
- Außerdem ist es möglich, sich eine HTML-Darstellung der Prozessbeschreibungen in einem Webbrowser anzusehen. Dadurch kann man die Prozessbeschreibungen als fortlaufenden Text lesen und so kontrollieren, ob die Beschreibung vollständig und korrekt ist.
- Darüber hinaus ist eine Visualisierung des Prozessgraphs verfügbar, in der alle Prozessschritte und atomaren Unterschritte angezeigt werden können, sowie deren Abhängigkeiten untereinander. Anhand der Farbe kann man erkennen, welche Schritte verpflichtend (rot) und welche freiwillig (gelb) sind. (Abbildung 29 zeigt den Graphen des in Listing 2 beschriebenen Anpassungsprozesses. Man sieht die Prozessschritte der ersten Ebene. Durch Anklicken eines Prozessschrittes werden die Schritte der Ebene unter diesem Prozessschritt gezeigt. Diese sowie die Legende wurden im Screenshot aus Platzgründen abgeschnitten.)

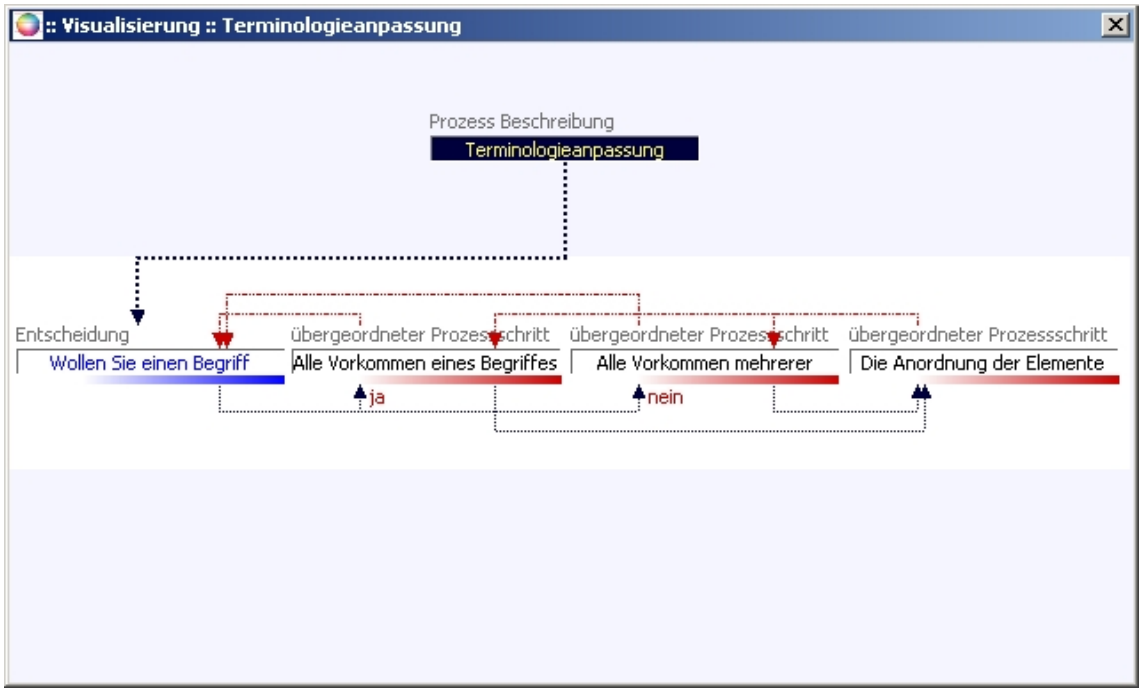


Abbildung 29: Visualisierung des Prozessgraphen.

- Weiterhin dienen die vom PIT erzeugten XML-Dateien als Import für das Wizard Generation Tool, das in Kapitel 5 detailliert erläutert wird.

- Zudem sieht das Konzept die Möglichkeit des Exports in eine XMI-Darstellung vor. XMI ermöglicht den Import in gängige UML-Werkzeuge. Allerdings ist der Export nach XMI derzeit nicht realisiert. Diese Erweiterung ist aber für eine zukünftige Version wünschenswert, da die UML-Darstellung für Software Designer und Entwickler eine wertvolle Information darstellt.

Wie bereits erläutert, sichert das PIT über Pflichtfelder ab, dass aus den erstellten Beschreibungen der Anpassungsprozesse funktionsfähige Prototypen generiert werden können. Außerdem enthält es alle Elemente, die in der Pattern-Form dieser Arbeit zur Verfügung stehen. Damit stellt das PIT aber nicht sicher, dass alle Prozessbeschreibungen auch wirklich Pattern-Anforderungen genügen. Es liegt im Ermessen des Prozessexperten, ob er sich die Mühe machen will und ein vollständiges Pattern erstellen will. Dann muss er unter anderem drei erfolgreiche Anwendungen des Patterns nennen und Kontext, Forces etc. ausführlich dokumentieren (vergleiche Abschnitt 5.1.5). Er kann sich auch entschließen, eine Prozessbeschreibung zu erstellen, aus der im Anschluss „nur“ ein Wizard generiert wird. Je nach Ziel des Prozessexperten können beide Entscheidungen richtig sein. In jedem Fall stellen die mit dem PIT erstellten Beschreibungen jedoch wertvolle Wissensquellen dar.

4.4 Verwandte Arbeiten

Die Beschreibung und Modellierung von Prozessen ist ein wichtiges Thema bei der Erstellung von Software. Dementsprechend gibt es auch eine Vielzahl von Arbeiten die sich damit beschäftigen. In vielen dieser Arbeiten werden Formalismen wie UML, ARIS, BPMN oder YAWL zur Prozessbeschreibung verwendet. Doch um mit diesen Formalismen Prozesse so beschreiben zu können, dass basierend darauf ein Wizard erstellt werden könnte, benötigt man Kenntnisse, die viele Experten in der Durchführung von Anpassungsprozessen nicht haben.

Es gibt aber auch eine Reihe von Arbeiten, die darauf abzielen, die Prozessexperten gemäß ihrer Kenntnisse einzubinden. Insbesondere im Requirements Engineering wird die Erfassung der Nutzeranforderungen betont [PH95]. Hierzu werden oft natürlichsprachliche Anforderungsbeschreibungen erstellt. Robertson [Ro01] stellt beispielsweise eine Auflistung verschiedener Techniken für das Requirements Engineering vor, die alle auf natürlicher Sprache beruhen, wie Interviews, Brainstorming oder Use Case Workshops. Auffällig hierbei ist, dass diese Techniken alle vom Prozessexperten keine oder geringe Einarbeitung erfordern. Doch die vorgestellten Techniken werden immer von Software-Designern genutzt, um die Prozessexperten zu befragen. Es ist nicht beabsichtigt, den Prozessexperten selbst in die Lage zu versetzen, einen Prototyp zu erstellen.

Eine der von Robertson vorgestellten Möglichkeiten sind natürlichsprachliche Szenarienbeschreibungen, die in diversen Arbeiten wie [Ac98], [NGS98] oder [Ro04] eingesetzt werden, um Anforderungen an ein zu erstellendes System zu beschreiben. Diese Szenarien dokumentieren anhand verschiedener Beispiele die Nutzeraktivitäten. Durch ihre Notation in natürlicher Sprache sind sie für alle an der Anforderungsanalyse beteiligten Personen gut verständlich.

Achour stellt in seiner Arbeit [Ac98] ein Modell vor, das Szenarienautoren beim Schreiben der Szenarien unterstützt. Dafür werden zum einen Richtlinien zur Verfügung gestellt, zum anderen werden die Szenarien anhand einer definierten Grammatik geprüft und korrigiert. In [NGS98] findet sich eine Anwendung des Ansatzes von Achour, bei der Szenarien verwendet werden, um Software-Anforderungen in einem Elektrizitätskonzern zu formulieren.

Doch wie bereits gesagt, zielen die von Robertson genannten Techniken nicht darauf ab, den Prozessexperten in die Lage zu versetzen, seine Prozesse eigenständig so zu beschreiben, dass daraus Software erzeugt werden kann. Sie dienen primär der Unterstützung der Software Designer. Das gilt auch für die von Achour verwendeten Szenarien.

Szenarien, wie Achour sie verwendet, werden in natürlicher Sprache notiert. Doch es gibt auch Ansätze, die die Szenarien deutlich formaler notieren. Beispielsweise verwenden Zündorf et al. [ZSW98] Story Boarding, um die Szenarien in Form von Graphen, den sogenannten Story-Diagrammen, zu sammeln. Aus diesen lässt sich, aufgrund ihrer gut definierten Notation und Semantik, (semi-)automatisch Quelltext generieren. Doch auch dieser Ansatz zielt darauf ab, Software-Designer und Entwickler zu unterstützen. Für die meisten Prozessexperten ist er nicht geeignet.

Auch Patterns haben den Eingang ins Requirements Engineering gefunden. So verwendet Robertson in [23] Patterns, um Anforderungen an Systeme zu beschreiben. Doch bei ihren Patterns ist es nicht das Ziel den Prozessexperten einen für sie gut verständlichen Formalismus an die Hand zu geben. Ihr Ziel ist viel mehr, dem Software-Designer eine Bibliothek mit wieder verwendbaren Requirement-Patterns zur Verfügung zu stellen.

Auch Sato et al. [Sa^{+}06] verwenden Patterns, um Anforderungen zu definieren. Allerdings formulieren sie nicht Nutzeranforderungen sondern Systemanforderungen. Patterns dienen hier dazu, die Entwicklung von Betriebssystemen zu vereinfachen.

Doch Patterns werden nicht nur zur Definition von Anforderungen verwendet. Oft werden sie auch eingesetzt, um Prozesse zu beschreiben. Fowler [Fo96] beispielsweise hat Pattern genutzt, um in der Analysephase von Software-Projekten das Wissen von Prozessexperten einzusammeln und in einer für die Prozessexperten verständlichen Form aufzubereiten. In [Fo96] stellt er eine Vielzahl von Patterns aus den Bereichen Handel, Messtechnik, Rechnungswesen und Organisationsstrukturen vor.

Ganz speziell auf die Beschreibung von Workflows zielen van der Aalst et al. [Aa^{+}02] ab. Sie haben eine Pattern-Sprache erstellt, die sogenannten Workflows Patterns, die dazu dienen, Anforderungen an Sprachen zur Beschreibung von Workflows zu formulieren. Doch auch hier ist nicht das Ziel, eine natürlichsprachliche Notation zu bieten, die auch für die Prozessexperten verständlich ist.

Eine andere Form der Beschreibung von Arbeitsabläufen in Prozessen wählt das APOSDLE Projekt [2]. In diesem Projekt sollen Personen bei der Durchführung von bestimmten Aufgaben (Tasks) unterstützt werden.

Um die Tasks gezielt erkennen zu können, müssen sie zuerst erfasst und modelliert werden. Dazu wurden durch Termextraktion gefundene initiale Tasks in Experteninterviews verfeinert. Die Prozessexperten wurden in strukturierten Interviews befragt, und es wurde mit der sogenannten Kartensortierung gearbeitet. Dabei erhalten die Prozess-

experten Karten mit den Tasks und müssen diese in Beziehung bringen. Außerdem wurde mit einer weiteren Form des strukturierten Interviews gearbeitet, dem sogenannten Laddering (abgeleitet vom englischen Wort für Leiter). Dies diente dazu, die Tasks auf einer kognitiven „Leiter“ hierarchisch anzuordnen. [Gh+08]

Die so gesammelten und strukturierten Tasks wurden mit Hilfe von Templates beschrieben. Die informellen Templates wurden anschließend semi-automatisch in eine formale Darstellung mit YAWL transferiert [Go+08].

Die bei APOSDLE verwendeten Techniken zur Task-Modellierung zielen auf die automatische Erkennung von Tasks ab. Es war nicht das Ziel, die Tasks so von Prozessexperten beschreiben zu lassen, dass daraus Unterstützungswerkzeuge generiert werden können. Die Task-Beschreibungen von APOSDLE sind deswegen nicht für den Zweck dieser Arbeit geeignet.

Neben der Erstellung der Beschreibungen von Anpassungsprozessen durch Prozessexperten ist ein weiterer wichtiger Aspekt in diesem Kapitel die Formalisierung von Anpassungspatterns sowie deren werkzeuggestützte Eingabe. Auch in diesen Bereichen gibt es bereits Arbeiten. Exemplarisch sollen hier drei Arbeiten betrachtet werden.

In ihrer Dissertation [Ha05] hat Mariele Hagen einen Ansatz zur Formalisierung von Software-Entwicklungsprozessen Patterns mit der UML-basierten Patternsprache PROPEL (Process Pattern Description Language) vorgestellt. Diese Sprache kann sowohl einzelne Prozesse als auch deren Beziehungen untereinander beschreiben [HG04]. Mit PROPEL können Software Entwicklungsmodelle definiert und diese Modelle an spezifische Situationen in Projekten angepasst werden.

PROPEL basiert auf UML. Das ermöglicht die Verwendung gängiger Prinzipien aus Software-Design und -Entwicklung, da laut Hagen UML der im Software-Design am häufigsten verwendete Modellierungsformalismus ist [Ha05]. PROPEL formalisiert Prozess-Patterns [BR02] [Gna+01]. Aber PROPEL ist sehr formal, und verwendet viele Komponenten aus UML. Man muss also UML bereits beherrschen, um PROPEL erlernen zu können. PROPEL ist für Software-Designer gedacht, die üblicherweise über UML-Kenntnisse verfügen. Aber für Personen ohne UML-Kenntnisse ist es nur schwer erlernbar.

Borchers hat in seiner Dissertation ebenfalls ein Werkzeug zum Editieren von Patterns vorgestellt [Bo01]. Dieses Werkzeug zielt auf Software-Ingenieure und HCI-Experten ab. Sie können mit PET Patterns schreiben, lesen und bearbeiten. Die Patterns müssen im XML Format eingegeben werden und können dann über einen Browser betrachtet werden. Die Visualisierung von Zusammenhängen zwischen verschiedenen Patterns ist leicht verständlich. Aber die Eingabe der Patterns im XML-Format ist, wie bereits erläutert, für den Ansatz der vorliegenden Arbeit ungeeignet.

Schobert und Schümmer [SS06] stellten auf der EuroPLoP 2006 ein Werkzeug zur Eingabe und Visualisierung von Patterns vor: CoPE (Collaborative Pattern Editor). Es stellt einen Texteditor zum Anlegen der Patterns und eine interaktive Diagrammsicht zum Erstellen einer sogenannten Pattern-Landkarte zur Verfügung. So ermöglicht es CoPE Pattern-Autoren, die Texte eines Patterns und eine grafische Visualisierung der Zusammenhänge mehrerer Patterns in Patternlandkarten zu erstellen. Ein Webinterface ermög-

licht es, dass mehrere Nutzer gemeinsam an einem Pattern arbeiten. Die erzeugten Patterns können auf einer Internetseite veröffentlicht oder in XML gespeichert werden.

Die Arbeit von Schobert und Schümmer trifft recht gut die Bedürfnisse der Prozessexperten bei der Patterneingabe. Allerdings ist CoPE nicht in der Lage, Prozessschrittbeschreibungen und Unterschrittbeschreibungen anzulegen und die strukturellen Informationen eines Anpassungsprozesses in der für die Wizard Generierung benötigten Form zu speichern. Daher konnte auch CoPE nicht als Werkzeug zur Eingabe der Prozessbeschreibungen verwendet werden.

4.5 Zusammenfassung

In diesem Kapitel wurde die Beschreibung von Anpassungsprozessen mit einer Notationsform vorgestellt, die die Beschreibungselemente eines Patterns verwendet und dabei insbesondere die Beschreibung von Prozessschritten und atomaren Unterschritten erlaubt. Dabei handelt es sich um den ersten Schritt des von der vorliegenden Arbeit vorgeschlagenen Konzeptes zur Software-Entwicklung basierend auf von Prozessexperten erstellten Pattern-basierten Beschreibungen von Anpassungsprozessen.

Zuerst wurde erläutert, welchen historischen Ursprung Patterns haben, welche Definitionen vorkommen und welche Notationsformen es gibt. Es wurde erläutert, dass Patterns nicht für sich allein existieren, sondern in Pattern-Kataloge, -Systeme und -Sprachen zusammengefasst werden. Anschließend wurde erklärt, wie man Patterns finden, schreiben und publizieren kann.

Basierend auf den gewonnenen Erkenntnissen über Patterns wurde eine spezielle Art von Patterns vorgestellt, die Anpassungsprozesse beschreiben. Zum Erfassen dieser Patterns wurde als ein Beitrag dieser Arbeit eine für die Beschreibung von Prozessabläufen in Anpassungsprozessen geeignete Pattern-Notation entwickelt.

Danach wurde gezeigt, wie die Anpassungspatterns zur Prozessbeschreibung mit Hilfe des Werkzeuges PIT erstellt werden können. Das PIT stellt zu diesem Zweck Eingabemasken und verschiedene Dialoge zu Verfügung. Diese unterstützen Anwender dabei Anpassungsprozesse und die darin benötigten Prozessschritte und atomaren Unterschritte anzulegen und in XML-Dateien zu speichern. Die vom PIT erzeugten XML-Dateien dienen als Eingabe für ein Werkzeug zur Erzeugung eines prototypischen Wizards, das im nächsten Kapitel erläutert wird. Die Entwicklung und Umsetzung des PIT stellt einen weiteren Beitrag dieser Arbeit dar.

Abschließend wurden verwandte Arbeiten betrachtet und in Bezug zum Vorgehen dieser Arbeit gestellt.

5 Automatisierte Prototyp-Erzeugung

Die automatisierte Prototyp-Erzeugung, basierend auf den mit Hilfe des PIT erstellten Beschreibungen von Anpassungsprozessen, bildet den zweiten Schritt des in der vorliegenden Arbeit vorgestellten Konzeptes. In diesem Kapitel wird das Vorgehen bei der automatisierten Prototyp-Erzeugung detailliert betrachtet.

Abbildung 30: Die drei Schritte der Pattern-basierten Prozessbeschreibung und Wizard-Erstellung.

5.1 Ansatz zur automatisierten Prototyp-Erzeugung

Der Prozessexperte erzeugt mit Hilfe des Eingabewerkzeuges PIT Beschreibungen von Anpassungsprozessen in einer Pattern-Notation. Daraus soll automatisiert ein prototypischer Wizard erstellt werden, um überprüfen zu können, ob die beschriebenen Anpassungsprozesse korrekt abgebildet sind.

Die vom PIT erzeugten XML-Dateien dienen daher als Eingabe für das Wizard-Generierungswerkzeug (WGT), das im zweiten Schritt des vorgestellten Ansatzes Verwendung findet (vergleiche Abbildung 30). Beim WGT handelt es sich um einen Prototyp, der in Java 1.5 unter Eclipse 3.2 entwickelt wurde. Der Prototyp basiert auf einem gemeinsam mit Horneff erarbeiteten Konzept und dessen Implementierung, die in [Ho07] vorgestellt wird. Basierend auf den Informationen, die vom Prozessexperten mit Hilfe des PIT in den Prozessbeschreibungen zur Verfügung gestellt wurden, generiert das WGT automatisch einen prototypischen Wizard. Dieser stellt als Prototyp eines Werkzeuges zur Unterstützung von Anpassungsprozessen den Ausgangspunkt für die weitere Entwicklung dar.

Eine wichtige Anforderung an die Generierung des Prototyps ist, dass sie einfach zu bedienen sein muss. Anwender, die mit dem WGT arbeiten, müssen keine Programmierkenntnisse haben. Sie stellen ihr Wissen über die zu unterstützenden Anpassungsprozesse mit Hilfe vom PIT im benötigten Format zur Verfügung. Das WGT übernimmt dann basierend auf den Prozessbeschreibungen die Erstellung eines Prototyps. Ein wichtiger Punkt hierbei ist, dass das WGT schnell und einfach gestartet werden kann. Deswegen gibt es zwei Möglichkeiten, das WGT zu starten: Zum einen kann es über einen Menü-Eintrag im PIT gestartet werden. Der Anwender muss dann lediglich angeben, wo der fertige Prototyp gespeichert werden soll (siehe Abbildung 31). Das WGT liest die aktuell im PIT geöffneten Prozessbeschreibungen und erzeugt einen Prototyp für diese Prozessbeschreibungen. Dabei können sowohl ein als auch mehrere Prozesse berücksichtigt werden. Zum anderen kann das WGT als eigenständiges Werkzeug gestartet werden. Dann ist zusätzlich anzugeben, welche vom PIT erzeugten Dateien für die

Prototyp-Generierung berücksichtigt werden sollen. Auch hierbei können ein oder mehrere Prozesse berücksichtigt werden. In beiden Fällen wird die eigentliche Prototyp-Erzeugung gestartet durch Drücken des Buttons „Wizard-Generierung starten“.

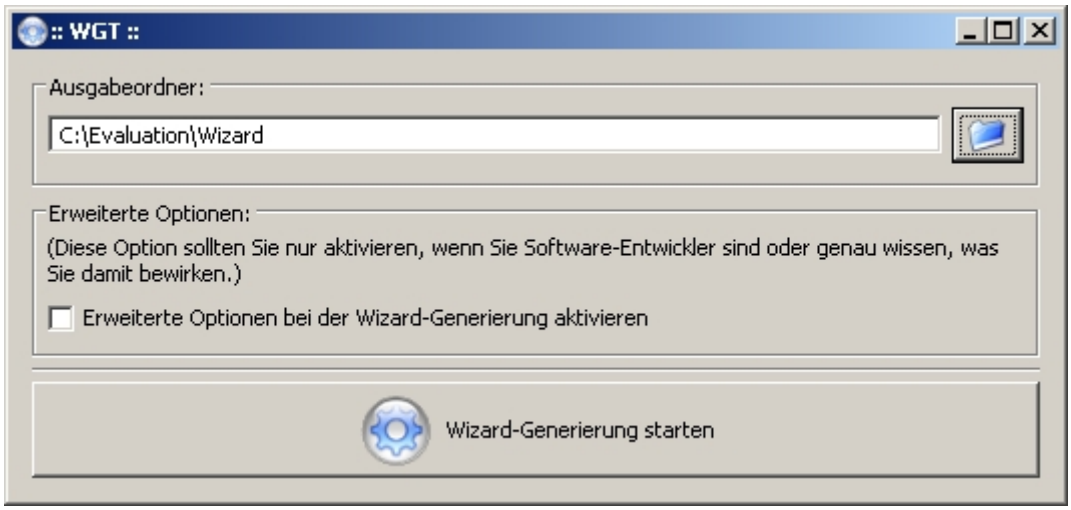


Abbildung 31: Benutzeroberfläche von WGT.

Anwender erstellen mit dem PIT Prozessbeschreibungen, die als Modelle der Anpassungsprozesse aufgefasst werden können. Das WGT generiert hieraus einen Prototyp. Dabei kann ein Prototyp einen oder mehrere Anpassungsprozesse beinhalten. Da es wahrscheinlich ist, dass mehrere Prozessexperten auch mehrere Prototypen erzeugen und da große Teile des Quelltextes für alle erstellten Prototypen gleich sind, wurde in der vorliegenden Arbeit eine Template-basierte Quelltext-Generierung [SV05] gewählt: Von den Prozessexperten werden die benötigten Informationen zu den Anpassungsprozessen zur Verfügung gestellt. Mit diesen Informationen können anhand vordefinierter Quelltext-Templates die Prototypen erstellt werden.

Der Ablauf der Prototyp-Erstellung setzt sich aus drei Schritten zusammen, die nacheinander durchlaufen werden:

1. Der erste Schritt besteht darin, die vom PIT erzeugten XML-Dateien einzulesen und zu parsen. In dieser Phase werden die Informationen über den Prozessfluss aller bei der Wizard-Generierung zu berücksichtigenden Prozesse in ein internes Modell transferiert, das die Struktur der Anpassungsprozesse darstellt. So lässt sich sicherstellen, dass der Prototyp die vom Prozessexperten beschriebene Ablauflogik widerspiegelt. Außerdem werden die in den Prozessbeschreibungen enthaltenen sprachlichen Informationen extrahiert.
2. Der zweite Schritt dient dem Befüllen der Quelltext-Templates: Es existieren verschiedene Arten von Quelltext-Templates. Diese werden gemäß der Struktur der Prozessabläufe instanziiert, gruppiert und mit den Textinformationen der Prozessbeschreibungen gefüllt. In diesem Schritt werden alle für den Prototyp benötigten Java-Klassen erzeugt.
3. Im dritten Schritt werden die erzeugten Klassen gespeichert. Die Klassen werden kompiliert und es wird ein lauffähiger Prototyp erstellt.

Abbildung 32 zeigt die 3 Phasen der Prototyp-Erzeugung:

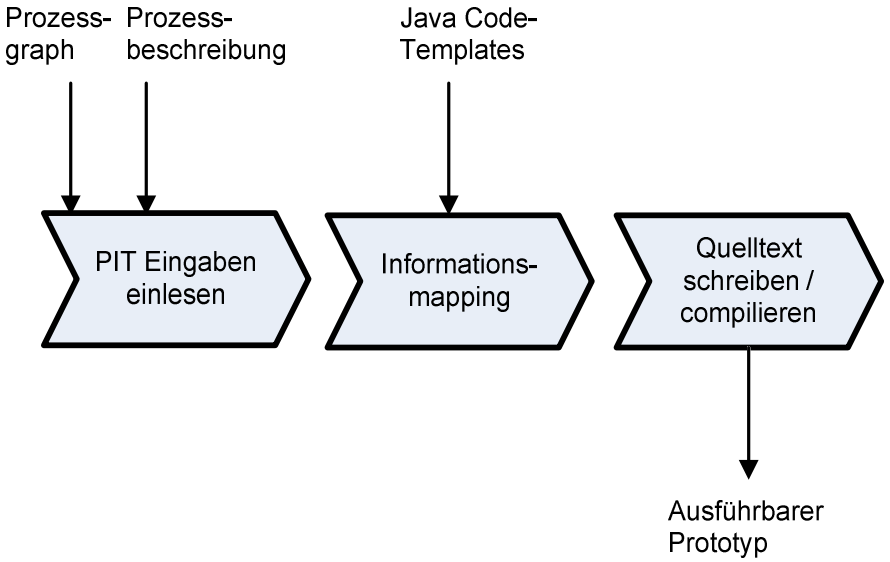


Abbildung 32: Die drei Phasen von WGT.

Der durch das WGT erzeugte Prototyp basiert auf dem Model-View-Controller Prinzip in der Interpretation von Eckstein [8]:

- *Model* = Die Regel-, Aktionen- und Datenschicht. Der Prozessgraph enthält die Regeln zum Ablauf der Anpassungsprozesse. Diese werden aus den Informationen zum Prozessfluss erzeugt, die im Prozessgraphen der PIT-Dateien enthalten sind. Darin werden auch Abhängigkeiten und Vorbedingungen berücksichtigt.
- *View* = Die Präsentationsschicht. Die Seiten des Prototyps sind die grafische Benutzerschnittstelle. Sie enthalten die textuellen Informationen der Pattern-ähnlichen Prozessbeschreibungen und basieren auf den Templates, die vom WGT mit den extrahierten Informationen gefüllt wurden. Die Seiten enthalten eine detaillierte Beschreibung, wie die Anpassungsprozesse, Prozessschritte und atomaren Unterschritte durchzuführen sind.
- *Controller* = Die Kontrollschicht. Der Controller reicht Ereignisse, die vom Anwender ausgelöst werden, an das Modell weiter und Reaktionen, die durch das Modell hervorgerufen werden, zurück zum Anwender. Der Controller kontrolliert, dass der Prozessfluss wie vom Modell vorgegeben abläuft. Abhängig von Benutzereingaben entscheidet der Controller, welcher Schritt wann ausgeführt wird. Dadurch wird die korrekte Einhaltung des Prozessablaufs sichergestellt.

  Weiterhin enthält der Controller Informationen über den aktuellen Zustand des Prototyps, indem er die bereits ausgeführten Schritte und deren Resultate speichert.

Der Controller ist zwischen Modell und Präsentationsschicht angesiedelt. Er verarbeitet die Benutzereingaben und gibt sie an das Modell weiter. Außerdem übermittelt er

Informationen, die aus Änderungen des Modells resultieren, an die Präsentationsschicht. Modell und Präsentationsschicht sind dadurch klar von einander getrennt. Diese konsequente Trennung erleichtert spätere Änderungen am auf dem Prototyp basierenden Wizard, da nicht bei jeder Änderung alle Teile des Programms angepasst werden müssen.

Der Hauptzweck des WGT ist es, einen Prototyp basierend auf den Informationen zu erstellen, die der Prozessexperte mit dem PIT bereitgestellt hat. Dieser Prototyp sollte so gestaltet sein, dass der Prozessexperte direkt nach der Prototyp-Generierung evaluieren kann, ob der Prototyp alle benötigten Informationen beinhaltet und die dargestellten Anpassungsprozesse korrekt abbildet. Damit der Prototyp zu einem einsatzfähigen Werkzeug zur Unterstützung von Anpassungsprozessen erweitert werden kann, muss er den in Abschnitt 2.3 definierten Anforderungen an ein Werkzeug zur Prozessunterstützung gerecht werden. Die Generierung des Prototyps berücksichtigt diese Anforderung folgendermaßen:

- *Aufgabenangemessenheit*: Um diesen Grundsatz zu erreichen, sollen Prozessexperten die Anpassungsprozesse beschreiben und anhand der automatisch generierten Prototypen prüfen, ob die Prototypen die beschriebenen Prozesse korrekt abbilden. Sie können Prozessbeschreibungen so lange verändern, bis die daraus erzeugten Prototypen die Prozesse korrekt abbilden.
- *Selbstbeschreibungsfähigkeit*: Die Beschreibungen der Prozessexperten sollen extrahiert und im Prototyp zur Verfügung gestellt werden, so dass Benutzer des Prototyps jederzeit wissen, was wann zu tun ist. Außerdem wird eine Hilfe für die Benutzung des Gesamtsystems angelegt.
- *Steuerbarkeit*: Mit „Vor“ und „Zurück“ Buttons können Benutzer im Prototyp navigieren. Außerdem ist eine Anzeige vorhanden, die anzeigt, wie viele Unterschritte eines Prozessschrittes bereits durchgeführt wurden und wie viele noch durchzuführen sind
- *Erwartungskonformität*: Der Ablauf des Prototyps orientiert sich an der Beschreibung des Prozessexperten. Dieser bekommt die Möglichkeit, den Ablauf durch die Prozessbeschreibung zu beeinflussen und kann so den Prototyp derart gestalten, dass er erwartungskonform ist.
- *Fehlertoleranz*: Bei der späteren Automatisierung des Prototyps ist zu berücksichtigen, dass, wo immer möglich, Eingaben der Benutzer direkt bei der Eingabe auf Fehler überprüft werden. Außerdem ermöglicht eine Vorschaufunktion, die Auswirkungen von Änderungen zu überprüfen, bevor sie endgültig ausgeführt werden. Falls dabei festgestellt wird, dass Fehler gemacht wurden, ist es möglich, Änderungen zu verwerfen.
- *Individualisierbarkeit*: Über Benutzereinstellungen können Benutzer den Prototyp an ihren aktuellen Kenntnisstand anpassen. Dazu werden zwei Modi vorgesehen: für Personen, die noch nicht oft mit dem Werkzeug gearbeitet haben und für Personen, die sich bereits gut mit der Anwendung auskennen. Außerdem wird das Werkzeug in mehreren Sprachen angeboten.

- *Lernförderlichkeit*: Zum einen enthält das Programm die Beschreibung der Prozessexperten und die schrittweise Führung durch die Anpassungsprozesse, die ein Erlernen des Programms erleichtern sollen. Zum anderen wird ein Hilfedokument erzeugt, das anhand einfach verständlicher Beispiele ermöglicht, die Bedienung des Programms zu üben.

5.1.1 Zur Quelltextgenerierung verwendete Templates

Ein Wizard besteht aus einer Abfolge von Dialogen, die in Form von Bildschirmseiten realisiert sind. Die einzelnen Seiten enthalten üblicherweise sogenannte Composites, die für die Anzeige der Benutzeroberfläche benötigte Elemente gruppieren. Die für die Prototyp-Erzeugung benötigten Seiten und Composites basieren auf Templates.

Es gibt insgesamt 25 verschiedene Templates für die verschiedenen Aufgaben, die bei der Ausführung des Prototyps berücksichtigt werden müssen. Grundsätzlich lassen sich zwei Arten von Templates unterscheiden:

1. Templates, in die während der Prototyp-Generierung aus den PIT-Dateien gewonnene Informationen eingetragen werden müssen. Viele dieser Templates stellen nur einen Teil einer Klasse dar, beispielsweise ein einzelnes Composite. Sie müssen mit anderen Templates kombiniert werden, um eine vollständige Klasse zu erhalten.
2. Templates, die bei der Erzeugung des Prototyps ohne Veränderung in den Prototyp übernommen werden. Bei diesen handelt es sich meist um vollständige Klassen, beispielsweise die Klasse, die den bereits angesprochenen Controller realisiert.

Nachfolgend werden ausgewählte Templates beider Gruppen erläutert. Dabei werden die Templates in der Reihenfolge erklärt, in denen sie Nutzern bei der Ausführung des Prototyps angezeigt werden. (Im Anhang findet sich eine Auflistung aller bei der Wizard Generierung verwendeten Templates.)

APStartPage

Mit diesem Template wird die Startseite des Prototyps generiert (Abbildung 33). Diese listet die vom Prototyp unterstützten Anpassungsprozesse auf. Der Nutzer kann den durchzuführenden Prozess wählen. Bei der Auswahl hilft zum einen eine HTML-Hilfeseite des Anpassungsprozesses, die den Prozess beschreibt. Sie ist über das blaue Fragezeichen hinter jedem Prozess aufrufbar. Zum anderen gibt es über den entsprechenden Button die Möglichkeit, eine zusätzliche Hilfe bei der Prozessauswahl zu erhalten. Das Drücken dieses Buttons ruft die Seite *ChoosePatternDialog* auf. Außerdem zeigt die Startseite, zu welchem Grad ein Anpassungsprozess automatisiert ist (vgl. Abschnitt 6.2.4). Direkt nach der Prototyp-Generierung sind alle Anpassungsprozesse durch Anleitung unterstützt, aber nicht automatisiert, da die Automatisierung erst nach der Prototyp-Generierung durch einen Entwickler vorgenommen wird.

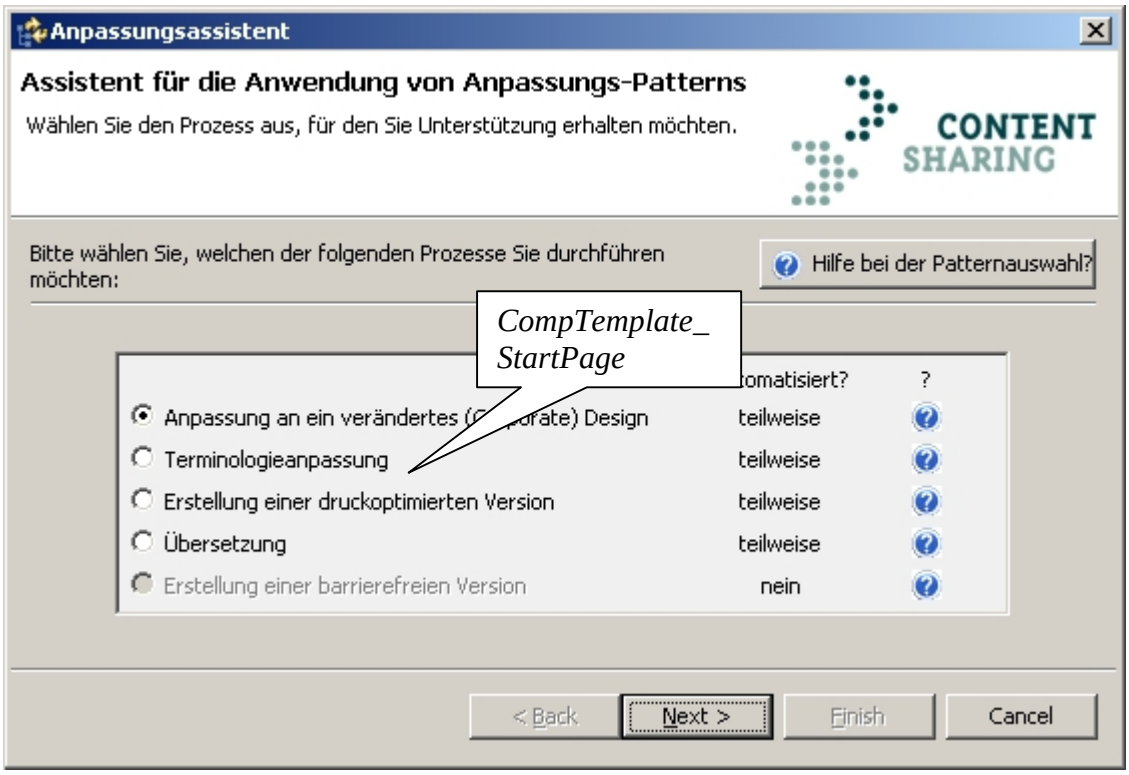


Abbildung 33: Auswahl aus unterstützten Anpassungsprozessen.

CompTemplate_StartPage:

Diese Composites zeigen auf der Startseite (*APStartPage*) alle vom Prototyp beschriebenen Anpassungsprozesse, deren Unterstützungsgrad sowie das blaue Fragezeigen für den Hilfelink an, so dass ein Anwender wählen kann, welchen Prozess er durchführen möchte (Abbildung 33). Das Composite wird für jeden unterstützten Anpassungsprozess einmal erzeugt.

ChoosePatternDialog

Wenn ein Nutzer den Prototyp startet, weiß er gegebenenfalls nicht, welchen Anpassungsprozess er benötigt. Deswegen gibt es auf der Startseite die Möglichkeit, zusätzliche Informationen zu den im Prototyp abgebildeten Anpassungsprozessen zu erhalten. Das Template *ChoosePatternDialog* erzeugt die dazu benötigte Seite, die alle Prozesse auflistet und zu jedem Prozess Informationen anzeigt, die aus Platzgründen auf der Startseite nicht gezeigt werden, die aber bei der Auswahl des richtigen Prozesses helfen können (Abbildung 34).

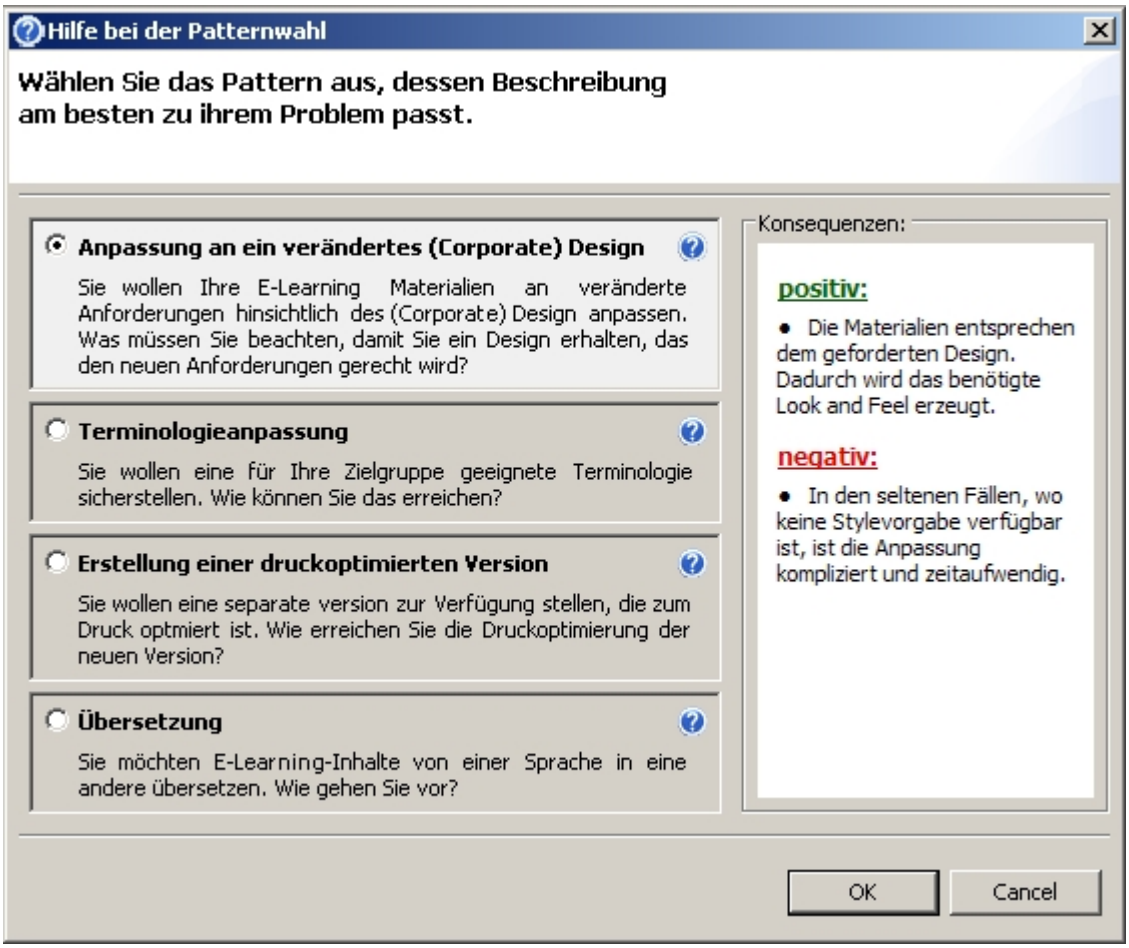


Abbildung 34: Unterstützung bei der Prozessauswahl.

APProcessFragmentsPage

Seiten, die mit diesem Template erzeugt werden, listen alle in einem Anpassungsprozess enthaltenen Prozessschritte auf (Abbildung 35). Außerdem wird es ermöglicht, zu wählen, welche der freiwilligen Prozessschritte man ausführen möchte. Die Seite wird für jeden im Wizard enthaltenen Anpassungsprozess einmal erzeugt.

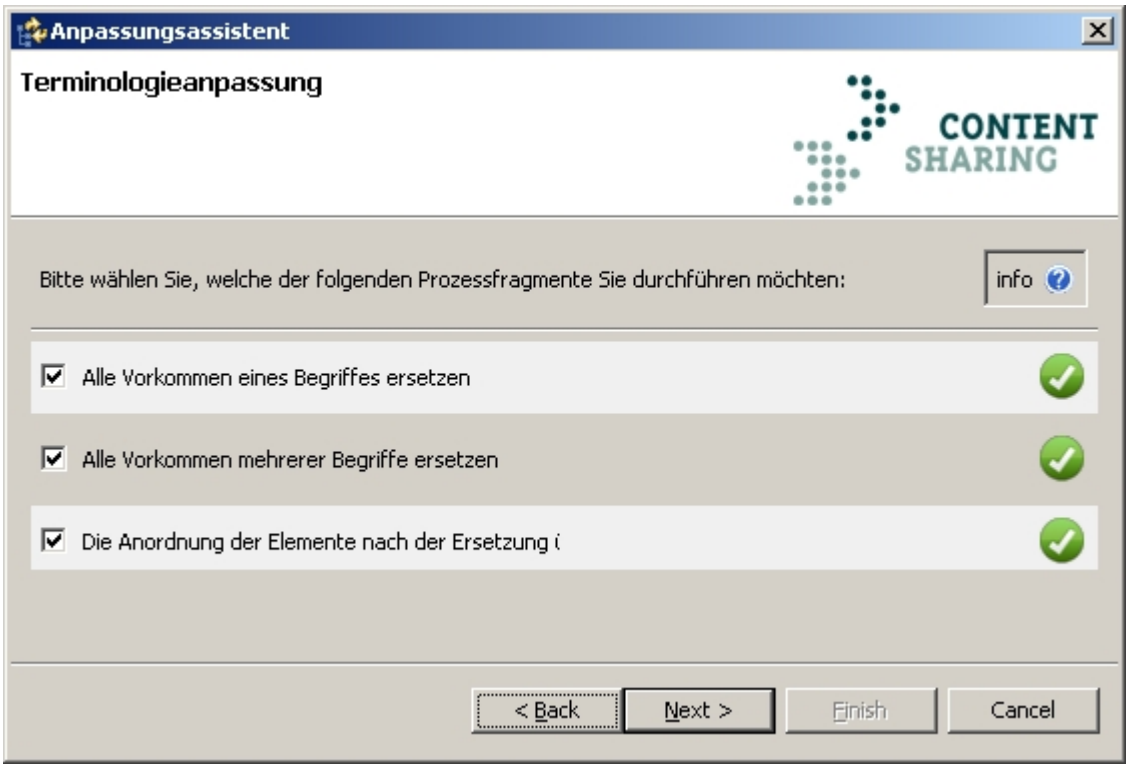


Abbildung 35: In einem Prozess enthaltene Prozessschritte .

APFragmentWizardPage

Die mit diesem Template angelegte Seite gibt einen Überblick über einen Prozessschritt und benennt die darin enthaltenen Unterschritte (Abbildung 36). So kann sich ein Anwender einen Überblick verschaffen, was im Prozessschritt geschieht. Diese Seite wird für alle Prozessschritte, die in den vom Wizard unterstützten Anpassungsprozessen enthaltenen sind, einmal angelegt.

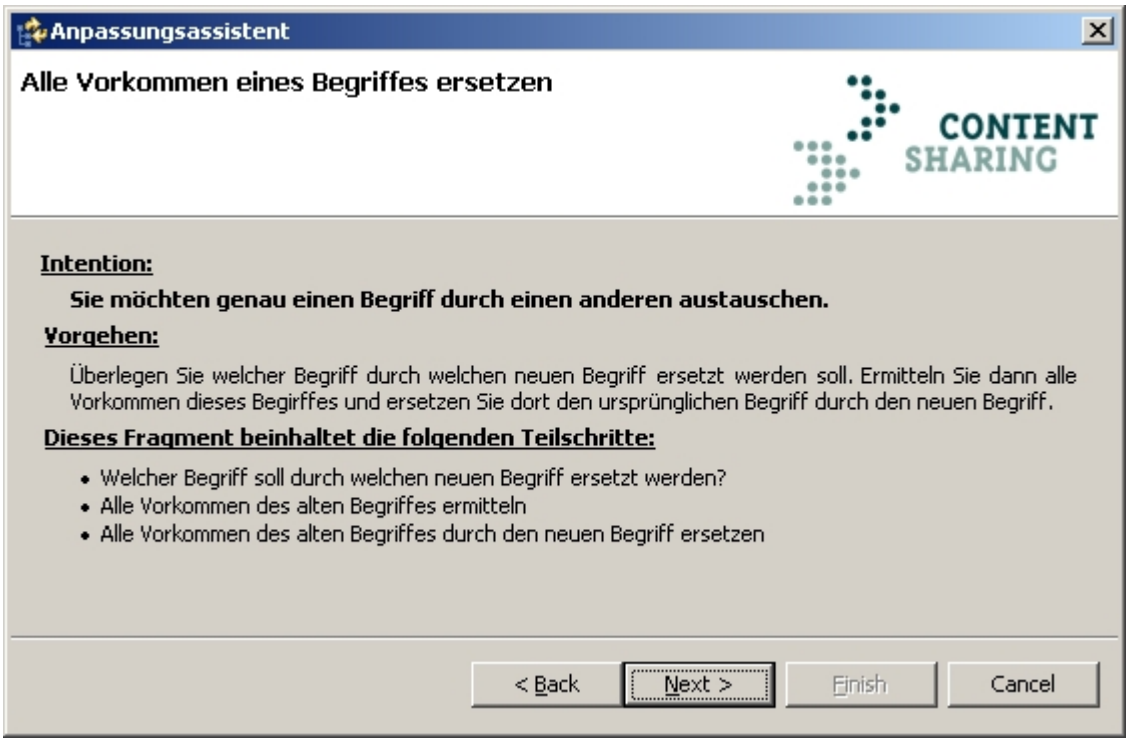


Abbildung 36: Überblicksseite für einen Prozessschritt.

APFragmentFunctionsPage

Dieses Template wird genutzt, um eine Seite anzulegen, die eine Gruppe von atomaren Unterschritten anzeigt (Abbildung 37). Die einzelnen Unterschritte werden durch Composites realisiert, die mit dem Template *CompTemplate_Function* angelegt wurden. Basierend auf dem Template werden alle Seiten erzeugt, die Unterschritte anzeigen.

Abbildung 37 zeigt eine Prozessschrittseite vor der Ergänzung automatisierter Funktionen. Die Informationen zu den atomaren Unterschritten werden aus der Prozessbeschreibung gewonnen. Um die Ablauflogik zu steuern, muss der Nutzer des Prototyps außerdem die Ergebnisse der Durchführung jedes Unterschrittes rückmelden. Dazu wählt er beispielsweise aus, dass er Elemente gefunden hat (vergleiche Abbildung 37). Erst dann wird der nächste Schritt aktiviert, im Beispiel ist das die Entscheidung darüber, ob es zu löschende Elemente gibt. Welche Rückmeldung möglich ist, hängt vom Typ des Unterschrittes ab. Der Prototyp reagiert also entsprechend des Unterschritttyps und des zugrunde liegenden Modells auf die Rückmeldungen des Anwenders.

CompTemplate_Function

Hier wird ein Quelltextfragment für ein Composite zur Verfügung gestellt. Für jeden auf einer Seite enthaltenen Unterschritt wird der zugehörigen Seite, die auf *APFragmentFunctions_Page* basiert, ein neues Composite zugefügt, das die Angaben zum Unterschritt enthält (siehe Abbildung 37). Das Composite wird für jeden im Prozess enthaltenen Unterschritt einmal erzeugt. Abhängig davon, ob der Unterschritt vom

Typ Ermittlung, Entscheidung oder Ausführung ist, wird das Composite unterschiedlich gestaltet. Abbildung 37 zeigt drei dieser Composites.

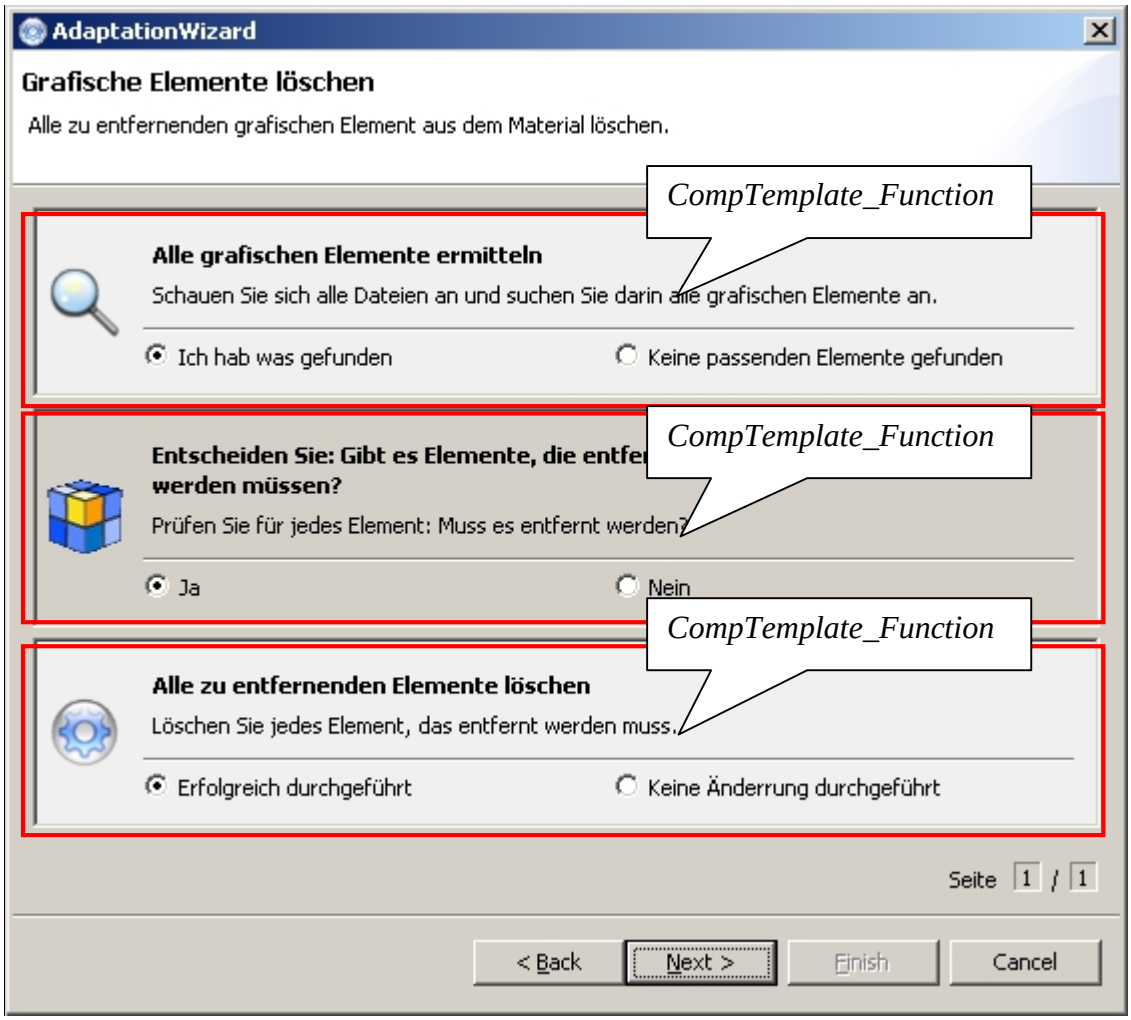


Abbildung 37: Anzeige einer Gruppe von Unterschritten.

APDecisionWizardPage

Es gibt Fälle, wo ein Prozessschritt nur ausgeführt wird, wenn der Anwender vorher eine bestimmte Entscheidung mit „ja“ oder „nein“ beantwortet hat (Abbildung 38). Die mit diesem Template erzeugte Seite präsentiert die Entscheidungsfrage und zeigt abhängig von der Entscheidung des Anwenders die nächste Seite. Basierend auf dem Template werden für alle entsprechenden Entscheidungen die Seiten erzeugt.

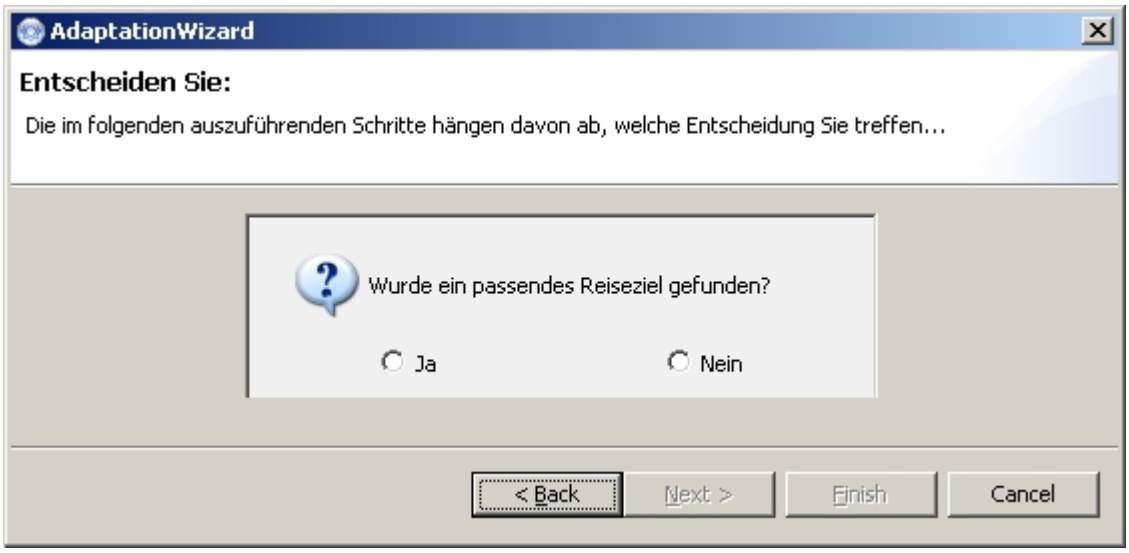


Abbildung 38: Anzeige von Entscheidungen.

APFinalWizardPage

Diese Klasse stellt die letzte Seite des Prototyps dar. Sie bietet dem Anwender die Möglichkeit, den Prototyp zu beenden, oder bei Bedarf einen weiteren Anpassungsprozess zu starten (Abbildung 39). Sie gibt auch Hinweise, falls im Anschluss an den aktuellen Anpassungsprozess andere Prozesse ausgeführt werden sollten.

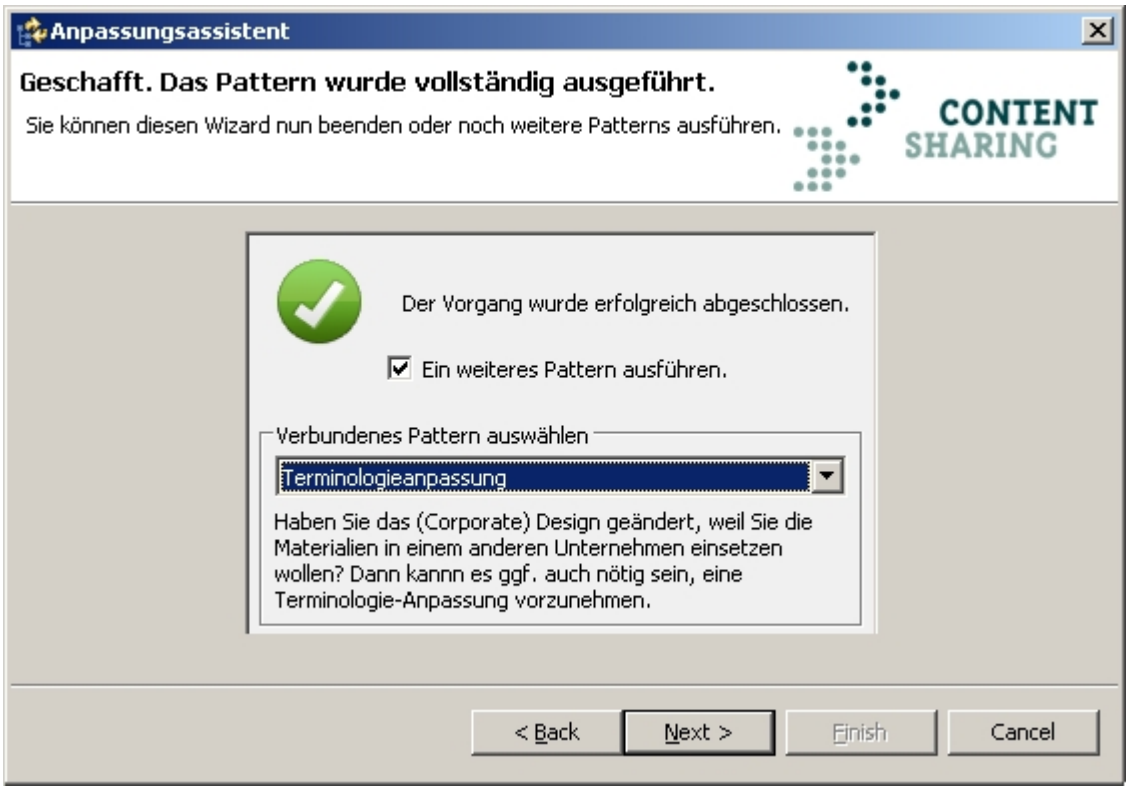


Abbildung 39: Letzte Seite des Prototyps.

Direkt nach seiner Erzeugung führt der Prototyp Anwender durch die Anpassungsprozesse und erklärt Schritt für Schritt, was jeweils zu tun ist. Die atomaren Unterschritte sind jedoch noch nicht automatisiert. Wie bereits erläutert wurde, ist es aber möglich, automatisierte Funktionen zuzufügen: Im dritten Schritt des in dieser Arbeit vorgeschlagenen Konzeptes wird Entwicklern die Möglichkeit gegeben, den durch das WGT erzeugten Prototyp um automatische Funktionen zu erweitern (vergleiche Kapitel 6).

Damit dies möglich ist, werden bereits bei der Prototyp-Generierung durch das WGT Klassen und Methoden zur Verfügung gestellt, die in einem automatisierten Wizard benötigt werden. Der automatisch vom WGT generierte Quelltext enthält außerdem eine Vielzahl von Kommentaren, die es Entwicklern erleichtern, den Quelltext zu verstehen und die zur Automatisierung geeigneten Stellen zu identifizieren.

5.1.2 Optionen bei der Seitenaufteilung

Ein Problem, das bei der Erstellung des Prototyps auftaucht, ist das der Verteilung der atomaren Unterschritte auf den Bildschirmseiten des Prototyps. Es gibt verschiedene Möglichkeiten, wie man es lösen kann:

Man könnte für jeden Prozessschritt eine Seite anlegen, auf der der gesamte Prozessschritt erläutert wird. Weiterhin könnte man für jeden atomaren Unterschritt eine eigene Seite anlegen, die den Unterschritt erläutert. Da viele Prozessschritte eine ganze Reihe von Unterschritten enthalten, würde diese Lösung zu einer Unmenge von Seiten führen, die teilweise nur ein oder zwei Sätze enthalten. Da das für Anwender unbefriedigend ist, wurde diese Lösung verworfen.

Eine weitere Möglichkeit wäre, für jeden Prozessschritt eine Seite anzulegen und darauf sowohl die Erläuterung des Prozessschrittes, als auch die darin enthaltenen atomaren Unterschritte anzuzeigen. Doch viele Prozessschritte enthalten zehn und mehr Unterschritte. In diesen Fällen würden die Seiten sehr groß und überfüllt mit Informationen. Erschwerend kommt hinzu, dass die Informationen zu Prozessschritten und Unterschritten unterschiedlich sind, so dass Anwender mit Informationen unterschiedlicher Art überflutet würden. Aus diesen Gründen wurde auch die zweite Möglichkeit verworfen.

Stattdessen wurde eine Mischform beider Möglichkeiten gewählt: Für jeden Prozessschritt wird eine Übersichtsseite angelegt, die erläutert, wie der Prozessschritt auszuführen ist und welche atomaren Unterschritte er beinhaltet. Der erzeugte Prototyp sieht einen Experten- und einen Laien-Modus vor, was es ermöglicht, die Übersichtsseite nur Laien zu zeigen, für die diese Information hilfreich ist.

Die atomaren Unterschritte wurden zu Gruppen zusammengefasst. Jeweils eine Gruppe wird auf einer Seite angezeigt. Standardmäßig werden fünf atomare Unterschritte auf einer Seite gezeigt. Die Anzahl fünf wurde gewählt, da nach Miller das Kurzzeitgedächtnis eine Aufnahmefähigkeit von 7 +/- 2 Informationseinheiten hat [Mi56]. Bei einer Gruppierung von maximal fünf Unterschritten kann man also sicher gehen, dass Anwender in der Lage sind, die gezeigten Informationen aufzunehmen.

Um zu erreichen, dass logisch zusammengehörige atomare Unterschritte auch auf einer Seite angezeigt werden, ist es möglich, die Anzahl der auf einer Seite gezeigten Unterschritte zu ändern. Dadurch ist es außerdem möglich, bei der Aufteilung den Platzbedarf

der automatisierten Unterschritte zu berücksichtigen, falls dieser bereits bei der Erzeugung des Prototyps bekannt ist. Zu diesem Zweck existiert im WGT die Option, die Seitenaufteilung bei der Prototyp-Erzeugung zu beeinflussen. Dazu ist auf der WGT-Oberfläche die Option „Erweiterte Optionen bei der Wizard-Generierung aktivieren“ zu selektieren (vergleiche Abbildung 40).

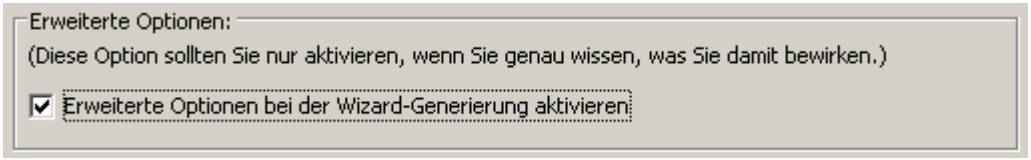


Abbildung 40: Erweiterte Optionen für die Prototyp-Generierung wählen.

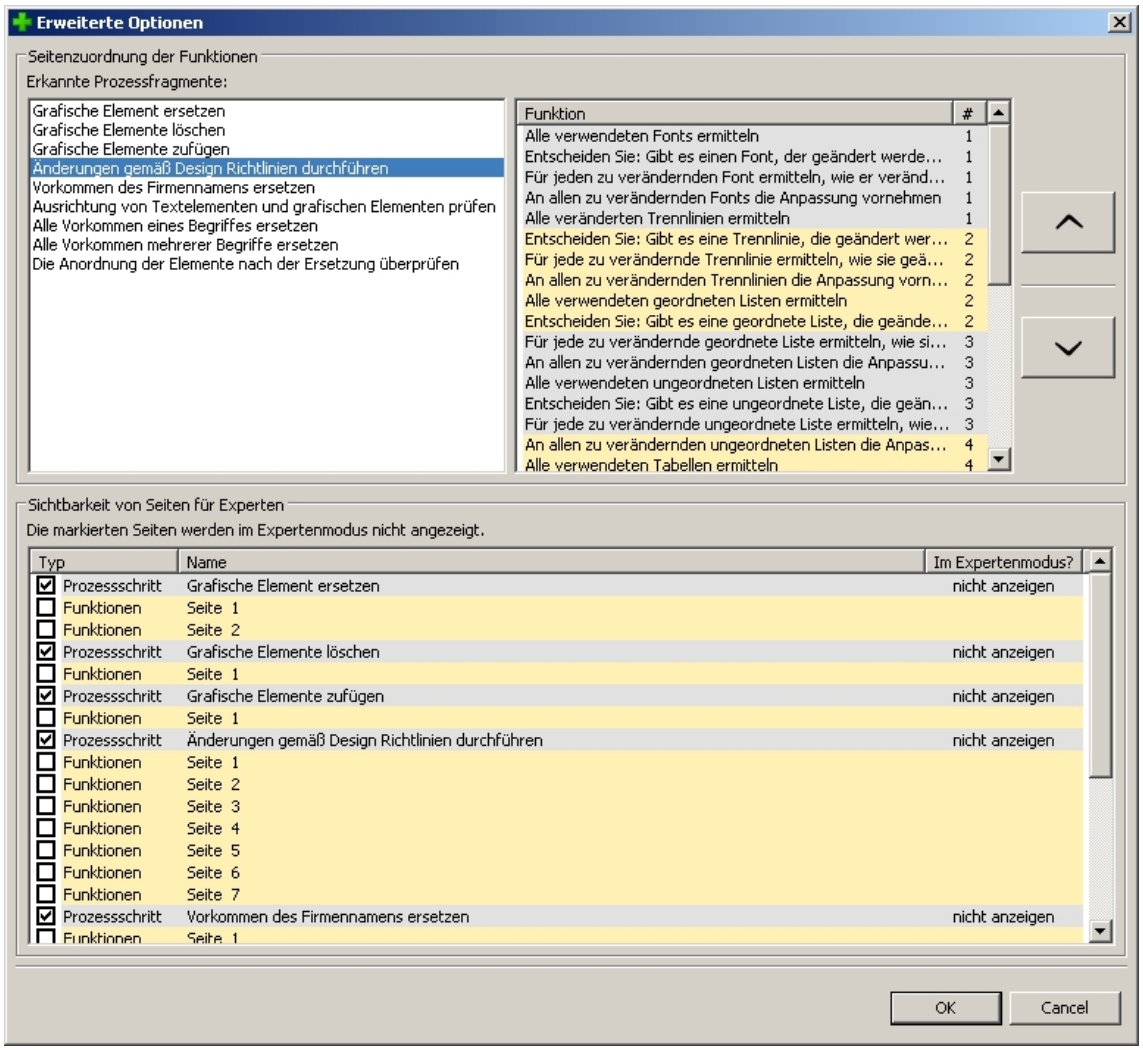


Abbildung 41: Erweiterte WGT Optionen.

Wählt man diese Option, öffnet sich ein Fenster (siehe Abbildung 41), das die Zuordnung der atomaren Unterschritte zu Seiten zeigt. Nutzer können die Zuordnung den jeweiligen Erfordernissen anpassen. Durch Auswahl eines Schrittes in der Liste der vorhandenen Prozessschritte werden die darin enthaltenen atomaren Unterschritte

angezeigt. Farbige Unterlegungen verdeutlichen, welche Unterschritte auf derselben Seite angezeigt werden. Wählt man einen Unterschritt aus und drückt auf die Pfeile auf der rechten Seite, so wird der ausgewählte Schritt auf die Seite vor oder hinter der aktuell zugeordneten Seite platziert. Wählt man in Abbildung 41 beispielsweise „Änderungen gemäß Design Richtlinien durchführen“ und drückt dann den Pfeil nach unten, so würde dieser atomare Unterschritt auf der zweiten Seite angezeigt werden.

5.1.3 Weitere Optionen bei der Prototyp-Erstellung

Im Gespräch mit den Prozessexperten wurde immer wieder der Wunsch nach einem Laien- und Expertenmodus im Prozessunterstützungswerkzeug geäußert (vergleiche Abschnitt 2.3). Der Laienmodus ist für Anwender gedacht, die das Werkzeug noch nicht gut kennen. In diesem Modus sind sehr ausführliche Informationen zu jedem Anpassungsprozess und dessen Durchführung enthalten. Anwender, die regelmäßig mit dem Werkzeug arbeiten, können in den Expertenmodus wechseln. Der Expertenmodus enthält weniger Anleitungen und kann dadurch schneller ausgeführt werden. Jeder Benutzer kann den Modus so ändern, dass er seinen Bedürfnissen entspricht.

Im Expertenmodus werden die Hilfetexte deutlich knapper gezeigt. Die Seiten, die ausschließlich eine Erläuterung der Prozessschritte enthalten, werden standardmäßig übersprungen. In der Abbildung der erweiterten WGT-Optionen (Abbildung 41) sieht man im oberen Bereich des Dialoges, die Möglichkeit der Zuordnung von Unterschritten zu Seiten. Im unteren Bereich des Dialoges gibt es außerdem die Möglichkeit, festzulegen, welche Seiten im Expertenmodus angezeigt werden sollen und welche nicht. Möchte man dem Experten beispielsweise nur die Seiten zeigen, die auch automatisierte Funktionen enthalten, kann man das dadurch erreichen, dass man bei der Prototyp-Erzeugung in den erweiterten Optionen alle anderen Seiten als auszublenden wählt.

Bei der Prototyp-Generierung werden die atomaren Unterschritte gruppiert und Seiten zugeordnet. Es wird entschieden, welche Seiten im Expertenmodus ausgeblendet werden sollen. Basierend auf den Templates werden die benötigten Java-Klassen angelegt und mit den zugehörigen Texten befüllt. Wenn diese Schritte abgeschlossen sind, startet das WGT die Kompilierung der Klassen und erzeugt einen ausführbaren Prototyp.

Der so erzeugte, ausführbare Wizard ist ein erster Prototyp des endgültigen Werkzeuges zur Unterstützung von Anpassungsprozessen. Er bietet Prozessexperten die Möglichkeit zu prüfen, ob der Prototyp die beschriebenen Anpassungsprozesse korrekt abbildet und sinnvoll unterstützt. Ist dies nicht der Fall, kann der Prozessexperte die Prozessbeschreibung an den Stellen, wo es nötig ist, anpassen und dann eine verbesserte Version des Prototyps generieren. Das lässt sich so lang wiederholen, bis der erzeugte Prototyp den Wünschen des Prozessexperten entspricht.

Zusammen mit den im ersten Schritt des vorgestellten Konzeptes erzeugten Prozessbeschreibungen dient der Prototyp als eine Kommunikationsgrundlage zwischen Prozessexperten und Entwicklern. Basierend auf dem Prototyp kann erreicht werden, dass alle Beteiligten ein gemeinsames Verständnis der abgebildeten Anpassungsprozesse erlangen. Außerdem lassen sich anhand des Prototyps Ideen für die weitere Entwicklung anschaulich diskutieren. Weiterhin bietet der Prototyp ein Quelltext-Grundgerüst, das, wie bereits erläutert, erweitert werden kann. Diese Erweiterung wird in Kapitel 6 vorgestellt.

5.2 Verwandte Arbeiten

5.2.1 Modellgetriebene Software-Entwicklung, Modellgetriebene Architektur und Generative Programmierung

Die mit dem PIT erzeugten Prozessbeschreibungen können als Modelle der beschriebenen Anpassungsprozesse aufgefasst werden. Die automatische Erstellung von Software basierend auf Modellen wird unter anderem in der sogenannten modellgetriebenen Software-Entwicklung (MDSD) behandelt. Dabei handelt es sich um ein Vorgehen zur Erstellung von Software, bei dem Software-Systeme durchgängig durch Modelle beschrieben werden, aus denen sich Quelltext erzeugen lässt [SV05]. Die Modelle sind das zentrale Konzept. Aus ihnen heraus werden die Software-Systeme erzeugt. Treten Änderungen auf, wird zuerst das Modell geändert. Durch Transformationen des Modells wird dann auch entsprechend der geänderte Quelltext erzeugt.

MDSD kann von einer existierenden Anwendung oder einer Referenzimplementierung ausgehen. Deren Quelltext gilt es dann zu analysieren. Dabei ist herauszufinden, was individueller Quelltext ist, der nur in der speziellen Anwendung Einsatz findet, was schematischer Quelltext ist, der zwar nicht für alle Applikationen gleich ist, aber doch immer einem ähnlichen Schema folgt, und was generischer Quelltext ist, der für alle derartigen Applikationen gleich ist. Ziel ist es, den schematischen Teil zu identifizieren und dafür ein für die Anwendungsdomäne spezifisches Modell zu erstellen. Dieses wird in einer Domänen-spezifischen Sprache (DSL) erstellt, die z.B. UML-basiert sein kann. Diese beschreibt die Bedeutung des Modells. Ein Metamodell definiert die im Modell verwendeten Elemente. Das Modell enthält somit formalisiertes Wissen über die Anwendungsdomäne. Es lässt sich durch Transformatoren in Applikationen überführen, die auf der Zielplattform ausführbar sind. Abbildung 42 verdeutlicht dieses Vorgehen.

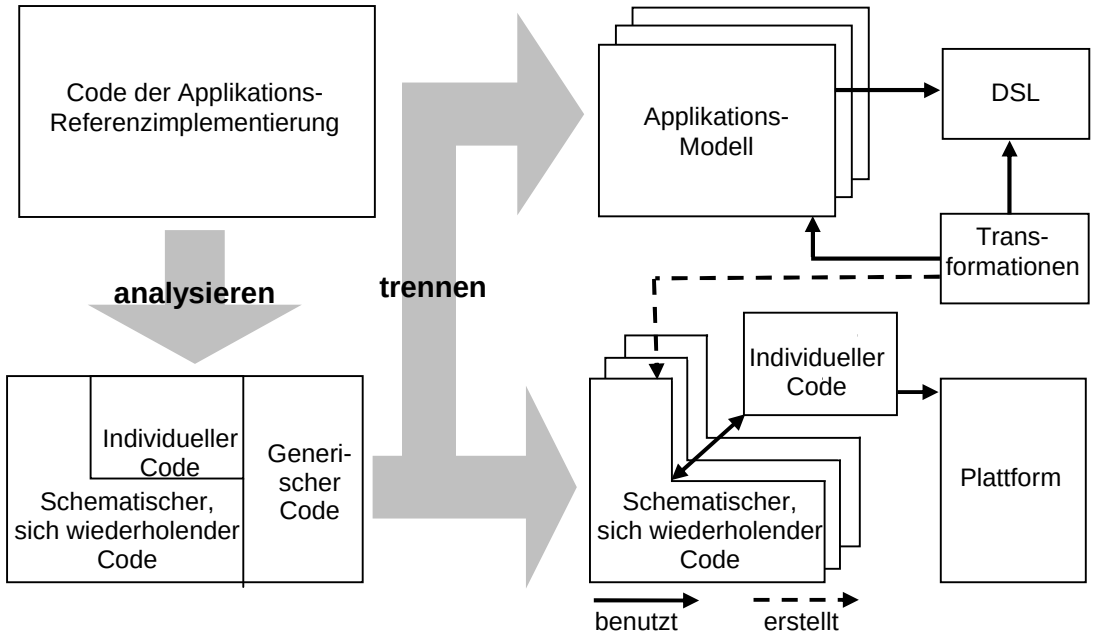


Abbildung 42: Vorgehen beim MDSD [SV05, S.17].

MDSD ist als Überbau über verschiedene Techniken und Bereiche der Modell-getriebenen Software-Entwicklung gedacht [SV05]. So umfasst es z.B. die Modell-getriebene Architektur (MDA), ebenso wie die generative Programmierung.

Bei der MDA [14] handelt es sich um einen Standard der OMG. Wie bei allen Ansätzen des MDSD wird auch hier aus formalen Modellen Quelltext generiert. Zentral ist das plattformunabhängige Modell (PIM). Daraus wird mittels Modell-Transformationen ein plattformspezifisches Modell (PSM) erstellt, beispielsweise für J2EE oder .NET. Das PSM wird mittels Modell-zu-Code-Transformationen in Quelltext für die entsprechende Plattform transformiert. Es ist auch möglich mit Modell-zu-Modell-Transformationen ein PSM in ein anderes PSM zu transferieren. (Vergleiche Abbildung 43).

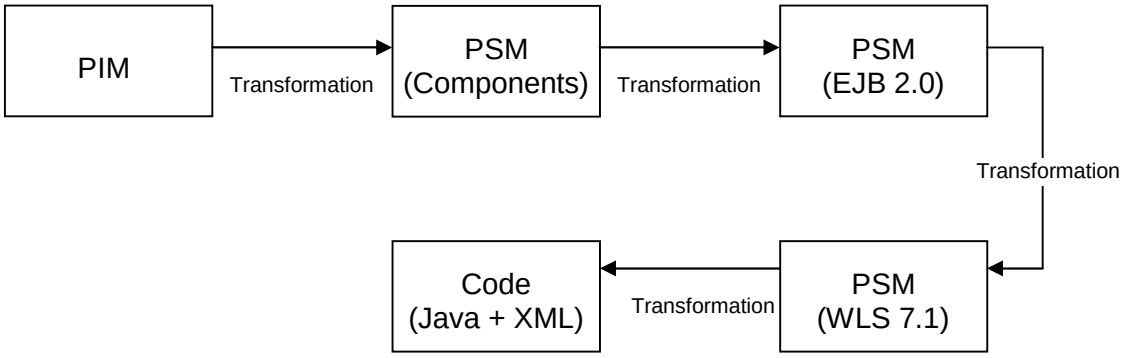


Abbildung 43: MDA-Transformationen [SV05, S.18].

Nach [Vö05] ist MDSD vor allem in folgenden drei Punkten generischer als MDA:

- In der MDA sind nur DSLs erlaubt, die auf der sogenannten MOF (Meta Object Facility) basieren. Bei der MOF handelt es sich um ein von der OMG definiertes Metametamodell. Beim MDSD sind auch andere DSLs erlaubt.
- Das Metametamodell MOF dient dazu MOF-konforme Modellierungssprachen wie die UML oder das Common Warehouse Modell (CWM) zu beschreiben. MDSD sieht das MOF als ein mögliches Metametamodell. Es ist aber nicht darauf festgelegt, sondern erlaubt auch andere Metamodelle.
- Bei der MDA sollten Transformationen auf Modell-Transformationssprachen aufbauen, die ebenfalls standardisiert sind. Auch hierbei lässt MDSD auch andere Möglichkeiten zur Transformation als die von der OMG standardisierten zu.

Eine weitere spezielle Form des MDSD ist die generative Programmierung [EC00]. Da das MDSD jünger ist als die generative Programmierung stellt das MDSD historisch gesehen eine Generalisierung der generativen Programmierung dar.

Bei der generativen Programmierung werden generative Domänenmodelle erstellt, aus denen durch die Spezifikation bestimmter Anforderungen die Anwendungen erzeugt werden können. Ziel ist es, aus einem Modell, das eine gesamte Systemfamilie abbildet, automatisiert ein fertiges Produkt zu erstellen. Der Hauptunterschied zum MDSD liegt in der Betonung der Systemfamilie: In der generativen Programmierung werden nicht einzelne „Mitglieder“ der Familie beschrieben, sondern die Modelle sind so gestaltet,

dass sich alle Mitglieder einer Software-System-Familie aus dem gleichen Modell generieren lassen.

Dazu wird folgendes Vorgehen durchgeführt (vergleiche Abbildung 44):

Der Problemraum (*Problem Space*) legt die domänenspezifischen Konzepte und Bestandteile fest. Dafür werden verschiedene Modelle erstellt, die oft in textueller Form vorliegen. Im Lösungsraum (*Solution Space*) wird die fertige Anwendung durch ihre elementaren Komponenten realisiert. Diese sollen maximal kombinierbar sein und möglichst wenig Redundanz aufweisen. Das Konfigurationswissen (*Configuration Knowledge*) beschreibt, wie man vom Problemraum zum Lösungsraum gelangt. Dafür legt es illegale Kombinationen der Bestandteile fest, es definiert Standardeinstellungen und Standardabhängigkeiten. Außerdem beinhaltet es Konstruktionsregeln und mögliche Optimierungen. Der Generator ist also auch Bestandteil des Konfigurationswissens.

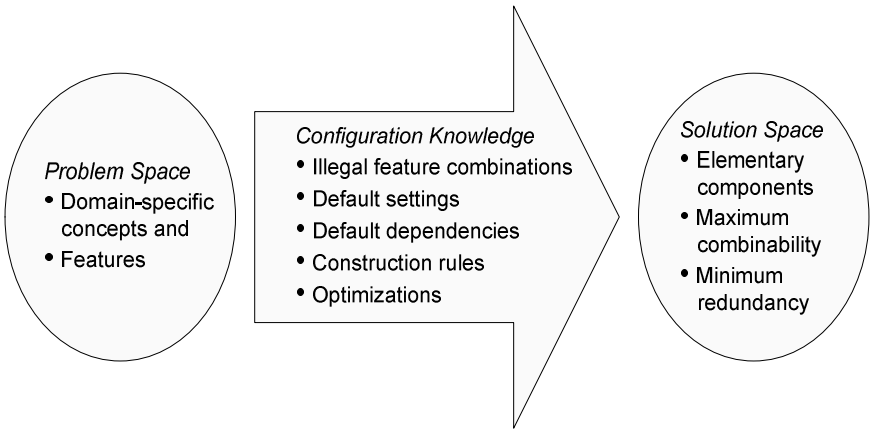


Abbildung 44: Elemente des generativen Domänenmodells [EC00, S.6].

Bei der generativen Programmierung wird das Modell durch einen Generator, der eine Modell-zu-Code-Transformation vornimmt, direkt in eine Anwendung überführt. Oft werden Templates zur Transformation der Modelle in Anwendungen verwendet. Generative Programmierung wird vor allem zur Erzeugung kleiner, hoch effizienter Produkte verwendet. Der in dieser Arbeit vorgestellte Ansatz erzeugt Quelltext basierend auf einem Modell (den Pattern-basierten Prozessbeschreibungen) unter Zuhilfenahme von Quelltext-Templates. Er lässt sich somit der generativen Programmierung zuordnen.

Zusammenfassend lässt sich sagen, dass alle drei eben vorgestellten Ansätze die automatische Generierung von Quelltexten basierend auf Modellen vorschlagen. In Rahmen dieser Dissertation wird ein Ansatz verwendet, der auf der generativen Programmierung beruht. Diese lässt sich als Spezialform des MDSD auffassen. Die MDA ist zu stark an die Verwendung von UML gebunden, was wie bereits Abschnitt 3.4 erläutert von Prozessexperten oft nicht beherrscht wird. Die Modelle in der generativen Programmierung dagegen sind freier, so dass auch die in dieser Arbeit verwendeten Pattern-basierten Prozessbeschreibungen als Modelle dienen können, auf deren Basis der Quelltext der zu generierenden Anwendung erzeugt wird.

5.2.2 Automatische Erzeugung von Quelltext auf der Basis von UML und XML

Eine Arbeit zur automatischen Erzeugung von Quelltext auf der Basis von UML ist die von Sturm et al. [SVB02]. Sie stellen ein Verfahren zur Generierung von Quelltext aus UML mit Velocity Templates vor. Ziel ihrer Arbeit war es, ein Framework zu erzeugen, das UML Modelle in Quelltext übersetzen kann. Ihre Arbeit wird im Quelltext Generierungs-Framework von Poseidon für UML [16] verwendet.

Das Framework enthält diverse Klassen für die verschiedenen UML Elemente und eine Reihe von Templates zur Erzeugung von Java Quelltext und HTML. Die Templates sind in der Velocity Template Language geschrieben und werden von der Velocity Template Engine interpretiert. Velocity [31] ist ein OpenSource Projekt, das eine Template-Sprache und deren Einbindung in Java anbietet.

Beim Ansatz von Sturm et al. werden bei jedem Durchgang der Quelltext-Generierung vorbereitete Elemente, die die benötigten Modell-Informationen bereitstellen, initialisiert und die Velocity Templates werden damit befüllt. Der Ansatz weist für den Zweck dieser Arbeit den Nachteil auf, dass er auf UML-Modellen beruht.

Eine Arbeit, die sich mit der automatisierten Erstellung eines prototypischen Wizards basierend auf einer XML-Spezifikation beschäftigt, ist die Arbeit von Turau [Tu02]. Er stellt ein Framework vor, das es ermöglicht, automatisch Wizards zur Dateneingabe im Internet zu erzeugen.

Um dies zu erreichen, setzt Turau ein Framework basierend auf dem Model-View-Controller Prinzip [21] ein. Das Modell wird durch Anwendungsdaten und Geschäftslogik dargestellt. Es bestimmt Zugriff, Veränderung, Validierung und Speicherung von Daten. Durch Anwendung der Geschäftslogik definiert der Controller das Verhalten der Applikation. Dadurch ändert sich auch der Zustand des Modells. Der Controller wählt auch die Seiten des Views aus, die verwendet werden, um den Zustand des Modells anzuzeigen. Die Views sind die Präsentationsschicht des Frameworks.

Das Framework basiert auf einer deklarativen Spezifikation, die in einer XML-Datei realisiert ist. Diese Datei legt alle möglichen Dateneingaben und deren Wertebereich fest und wie diese Daten auszuwerten sind, wie sie auf den Webseiten angezeigt werden sollen, in welcher Reihenfolge sie gesammelt werden müssen und welche Aktionen darauf ausgeführt werden müssen. Basierend auf diesen Angaben werden die Seiten des Views sowie das interne Modell von einem Quelltext-Generator erzeugt.

Das Ergebnis ist eine prototypische Webanwendung, die Daten in Form eines Wizards einsammelt, validiert und entsprechend auf die Dateneingabe reagiert. Dieser Prototyp kann zum Testen genutzt werden. Durch Änderung der XML-Spezifikation können Änderungen am Prototyp erreicht werden. Dieser wird so lange angepasst, bis das Ergebnis den Wünschen entspricht. Dann kann er durch Erweiterung der rudimentären Eingabe-Formulare zu einer endgültigen Anwendung erweitert werden.

Der in der vorliegenden Arbeit vorgestellte Ansatz verwendet ebenfalls ein Modell in Form einer XML-Spezifikation als Grundlage zur automatischen Generierung einer Anwendung, die mit Nutzern interagiert. Daher wird Turaus Idee der automatischen Generierung einer Anwendung basierend auf dem Model-View-Controller-Prinzip

übernommen. Mit Turaus Ansatz werden Formulare zur Dateneingabe im Internet generiert. Mit dem in der vorliegenden Dissertation vorgestellten Ansatz werden Wizards zur Prozessunterstützung erzeugt. Dabei treten Probleme auf, die bei Turau nicht relevant sind, beispielsweise die Aufteilung atomarer Unterschritte auf Seiten. Daher wurde in der vorliegenden Arbeit die Idee, aus einer XML-Spezifikation eine Anwendung basierend auf dem Model-View-Controller-Prinzip zu erzeugen, erweitert. So entstand ein Konzept, das den Anforderungen an eine automatische Erzeugung eines Werkzeuges zur Unterstützung von Anpassungsprozessen (vergleiche Abschnitte 2.3 und 2.4) gerecht wird.

5.2.3 Automatische Erzeugung von Quelltext basierend auf Patterns

Auch die Verwendung von Patterns als Grundlage zur automatischen Quelltext-Generierung wird in einigen Arbeiten behandelt. So stellen beispielsweise Budinsky et al. [Bu+96a] einen Ansatz vor, der aus Design Patterns [Ga+95] automatisiert Quelltext erzeugt. Hintergrund ihrer Arbeit ist die Beobachtung, dass Design Patterns Expertise in der Entwicklung objektorientierter Software abbilden. Sie erläutern systematisch, wie sich oft auftretende Design-Probleme lösen lassen. Nachteilig ist aber, dass in jedem neuen Entwicklungsprojekt die Lösungen der Patterns neu implementiert werden müssen. Deswegen haben Budinsky et al. ein Werkzeug entwickelt, dass es ermöglicht, automatisch den zu den Design Patterns passenden Quelltext zu erzeugen. Dadurch soll der Entwickler entlastet werden und sich so auf die Optimierung der übrigen Entwicklung konzentrieren können.

Um das zu ermöglichen, bietet das Werkzeug die Option, durch die Patterns zu blättern. Für jedes Pattern existiert eine Seite, mit der sich der zum Pattern passende Quelltext erzeugen lässt. Da der erzeugte Quelltext von einigen Faktoren abhängt, kann der Nutzer diese spezifizieren. Hierunter fallen abhängig vom jeweiligen Pattern einige generelle Angaben, wie z.B. der Name der zu erzeugenden Komponente, sowie Pattern-spezifische Angaben. Außerdem wird berücksichtigt, dass die Patterns verschiedene Forces benennen, also bei der Entwicklung je nach Gegebenheiten unterschiedliche Kompromisse eingegangen werden müssen. Nutzer können wählen, welche Forces sie bevorzugen. Dann wird die hierzu passende Quelltext-Variante erzeugt.

Der vom Code-Generator erzeugte Quelltext muss vom Nutzer des Werkzeuges per Cut & Paste an die richtigen Stellen im eigenen Quelltext eingefügt werden. Budinsky et al. bewerten dieses Vorgehen selbst als nachteilig, da spätere Änderungen am Modell ein erneutes Kopieren und Einfügen des generierten Quelltextes bedeuten. Sie schlagen zur Lösung vor, eine Klasse mit dem generierten Quelltext zu erzeugen und eine Subklasse mit den vom Benutzer gemachten Änderungen. Dann würde die Subklasse bei erneutem Generieren des Quelltextes nicht überschrieben, könnte aber die Änderungen von der Superklasse erben.

Eine weitere Arbeit aus dem Bereich der automatisierten Quelltext-Generierung basierend auf Patterns wird in [EYG97] vorgeschlagen. Diese Arbeit konzentriert sich auf die Formalisierung und Implementierung der Lösungs-Teile von Design Patterns. Dafür schlagen die Autoren ein prototypisches Werkzeug vor, das die Spezifikation von Design Patterns und deren Realisierung in einem bereits existierenden Quelltext erlaubt.

Annahme ist, dass Patterns meist nicht dadurch realisiert werden, dass man neue Klassen implementiert, sondern dass man sie in existierende Programme einfügt. Daher ist das Ziel bei Eden et al., die Lösung eines Patterns in den existierenden Programmcode einzufügen.

Dafür wird folgendes Konzept vorgeschlagen: Das existierende Programm wird geparst. Ein menschlicher Programmierer wählt dann ein Pattern aus, das er anwenden möchte. Lässt sich das Pattern im existierenden Programm anwenden, wird es umgesetzt. Ist der Anwender nicht zufrieden mit dem Resultat, kann er die Anwendung des Patterns rückgängig machen, oder Teile des Resultates verändern. Außerdem ist es möglich, die Spezifikation der Patterns zu verändern, so dass das Resultat sich ändert. Der Nutzer kann die Schritte so lang ausführen, bis das Ergebnis seinen Anforderungen entspricht. Dann wird der Quelltext geschrieben. Dabei werden Kommentare eingefügt, die dem Anwender sagen, wo noch manuelle Änderungen vorzunehmen sind.

Bei Edens Ansatz wird nur die Lösung von Patterns betrachtet. Design Patterns enthalten aber auch eine Reihe weiterer Bestandteile, die wichtig sind und in einem Werkzeug zur automatisierten Quelltext-Generierung ebenfalls berücksichtigt werden sollten. Zu diesen Bestandteilen zählen beispielsweise die Forces, die bei Budinsky et al. eingebunden sind, bei Eden aber außer acht gelassen werden. Eine Einbindung dieser Bestandteile scheint aber gerade im Hinblick auf die Anpassungsprozesse wünschenswert.

Neben den bereits vorgestellten Arbeiten zu Design Patterns gibt es auch Ansätze, die sich mit dem Einsatz anderer Patterns für die automatisierte Erzeugung von Programmen beschäftigen. Hier wären beispielsweise die Arbeiten von Braga et al. zu nennen. In [BGM04] stellen sie einen Ansatz vor, wie sich mit Hilfe von domänenspezifischen Patterns und einem darauf basierenden Framework automatisiert Applikationen erstellen lassen. Die Patterns bilden eine Domänen-spezifische Pattern-Sprache, bestehend aus Analyse Patterns. Jedes Pattern stellt in Form von Klassendiagrammen die Lösung eines spezifischen Problems dar, das in der modellierten Domäne auftaucht.

Voraussetzung dafür, dass der Ansatz angewandt werden kann, ist dass der Software-Entwickler die Pattern-Sprache der zu betrachtenden Domäne sowie ein Dokument mit Anforderungen verfügbar hat. Zuerst muss der Software-Entwickler die Anforderungen analysieren. Basierend darauf wählt er eine passende Pattern-Sprache. Mit der Pattern-Sprache modelliert er dann das zu erstellende System. Braga et al. empfehlen, dies in Form eines UML-Klassendiagramms zu tun. Das Diagramm wird dann mit den Anforderungen verglichen, um Klassen zu finden, die durch die Pattern-Sprache nicht oder nur unzureichend abgedeckt werden.

Im zweiten Schritt des Vorgehens werden basierend auf dem Analysemodell die Klassen der zu erstellenden Applikation erzeugt. Damit dies möglich ist, muss ein Framework existieren, das für jedes Pattern der Pattern-Sprache eine Klasse enthält. Zu diesem Zweck haben Braga et al. das GREN Framework [BM02] entwickelt. Es basiert auf der Pattern-Sprache GRN [BGM99], die das Ressourcen-Management in Unternehmen modelliert. Um das Framework instanziieren zu können, wurde der GREN-Wizard [BM03] entwickelt, der die Manipulation des Frameworks mittels einer grafischen Benutzeroberfläche erlaubt.

Im dritten Schritt wird der erzeugte Quelltext erweitert. Hierbei werden die im ersten

Schritt ermittelten Anforderungen berücksichtigt, die von der Pattern-Sprache nicht abgedeckt worden sind. Diese Erweiterung kann sich als schwierig gestalten. Deswegen schlagen Braga et al. vor, die Ergebnisse der Erweiterungen zu dokumentieren und die Erfahrungen in das Framework einzubauen.

Ziel aller Arbeiten zur Quelltext-Generierung aus Patterns ist, Entwickler bei der Erstellung komplexer Anwendungen zu unterstützten. Somit werden bei allen Arbeiten Kenntnisse im Bereich der Software-Entwicklung vorausgesetzt, welche bei der Zielgruppe der vorliegenden Arbeit, den Prozessexperten, meist nicht vorhanden sind. Außerdem erstellen die Personen, die den Quelltext generieren nicht auch die Patterns, die als Grundlage der Quelltext-Generierung verwendet werden, wie das in der vorliegenden Arbeit der Fall ist.

5.3 Zusammenfassung

In diesem Kapitel wurde der zweite Schritt des Konzeptes zur Software-Entwicklung basierend auf von Prozessexperten erstellten Pattern-basierten Beschreibungen von Anpassungsprozessen vorgestellt: Die automatisierte Generierung eines prototypischen Wizards aus den mit Hilfe vom PIT erzeugten Prozessbeschreibungen.

Da die Prozessexperten, auf die diese Arbeit abzielt, meist nicht programmieren können, wurde das Werkzeug WGT zur Prototyp-Generierung entwickelt. In Abschnitt 5.1 wurde dieses Werkzeug vorgestellt. Dabei wurden der Aufbau des Werkzeuges und dessen Funktionsweise detailliert erläutert. Das WGT realisiert die in Abschnitt 3.3 formulierten Anforderungen an eine automatisierte Prototyp-Generierung. Ein wichtiger Aspekt bei der automatischen Erzeugung eines prototypischen Wizards ist die Aufteilung der Seiten. Dieses Problem und die im Rahmen dieser Dissertation erarbeitete Lösung wurden ebenfalls in Abschnitt 5.1 vorgestellt. Die Konzeption und Entwicklung des Werkzeuges WGT zur automatisierten Prototyp-Generierung stellt einen der Beiträge dieser Dissertation dar.

Anschließend wurden in Abschnitt 5.2 verwandte Arbeiten betrachtet und in Bezug zum Vorgehen dieser Arbeit gestellt. Es wurde erläutert, dass es bereits einige Arbeiten zur automatisierten Quelltexterzeugung gibt. Doch diese werden nicht allen in Abschnitt 3.3 vorgestellten Anforderungen gleichzeitig gerecht. Deswegen wurde das in Abschnitt 5.1 behandelte Werkzeug zur Prototyp-Generierung entwickelt. Im folgenden Kapitel wird erläutert, wie sich durch Erweiterung um automatisierte Funktionen aus einem mit dem WGT erzeugten Prototyp ein einsatzfähiges Werkzeug zur Unterstützung von Anpassungsprozessen erzeugen lässt.

6 Wizard-Erweiterung zur (semi-)automatischen Unterstützung von Anpassungsprozessen

Zur Zeitersparnis sowie zur Fehlerverringerung bei der Durchführung der Anpassungsprozesse ist es wünschenswert, den Wizard, wo sinnvoll möglich, durch automatisierte Funktionen zu ergänzen. Dazu wird der vom Prozessexperten erzeugte prototypische Wizard im dritten Schritt des in der vorliegenden Dissertation vorgestellten Konzeptes einem Entwickler gegeben. Dieser kann basierend auf dem im folgenden Abschnitt vorgestellten Vorgehen den Wizard um automatisierte Funktionen erweitern.

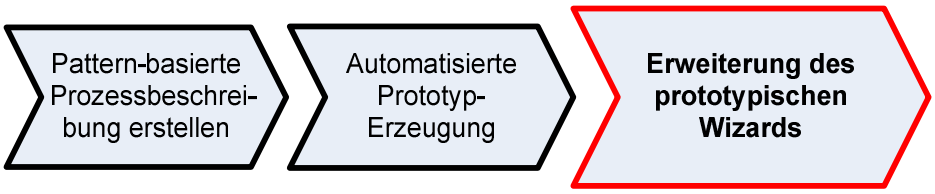


Abbildung 45: Die drei Schritte der Pattern-basierten Prozessbeschreibung und Wizard-Erstellung.

6.1 Vorgehen bei der Wizard-Erweiterung

Die verwandten Arbeiten zur automatischen Quelltext Generierung aus Patterns (vergleiche Abschnitt 5.2.3) haben gezeigt, dass die manuelle Erweiterung von automatisch erzeugtem Quelltext gut dokumentiert werden muss und dass spätere Änderungen schwierig sind. Um dieses Problem bereits bei der Erzeugung des prototypischen Wizards durch das WGT einzuschränken, werden bei der Wizard-Generierung durch das WGT Kommentare in den Quelltext eingefügt. Sie helfen einem Entwickler, die Stellen zu identifizieren, die bei einer Erweiterung geändert werden müssen. Außerdem werden Klassen und Methoden zur Verfügung gestellt, die für eine Automatisierung der atomaren Unterschritte des prototypischen Wizards benötigt werden.

Anpassungsprozesse verändern die Dateien, aus denen Lernressourcen bestehen. Um die Ausführung eines atomaren Unterschrittes zu automatisieren, muss ein Entwickler formatspezifische Funktionen implementieren, die den Unterschritt für ein Dateiformat realisieren. Dabei kann es vorkommen, dass ein Unterschritt durch mehrere Funktionen realisiert werden muss. Beispielsweise gibt es eine Reihe von Formaten, wo die Ersetzung eines Bildes durch Löschen des ursprünglichen und Zufügen eines neuen Bildes geschieht. In anderen Formaten (z.B. HTML) ist das Ersetzen eines Bildes durch Ändern einer Referenz realisiert.

Die Klasse `Controller.java` enthält eine Liste aller Unterschritte, für die eine automatisierte Implementierung zur Verfügung steht. Diese Liste ist leer, solange der Wizard keine automatisierten Unterschritte enthält. Erweitert ein Entwickler den Wizard durch Zufügen automatisierter Funktionen, so enthält die Liste eine Menge von Objekten, die der Identifikation der zur Automatisierung verwendeten Funktionen dienen. Dabei wird auch angegeben, für welche Formate eine Funktion automatisiert ist.

In Anpassungsprozessen gibt es eine Reihe von Funktionen, die in mehreren Prozessschritten verwendet werden. Daher werden die Funktionen in einer Art „Pool“ gesammelt. Das ermöglicht es, aus verschiedenen Prozessschritten auf dieselben Funktionen zuzugreifen.

Alle Klassen, die Funktionen realisieren, müssen das Interface `APAutomater` implementieren. Dadurch verfügen sie über folgende Methoden:

- Die Methode `getComposite()` liefert das Composite, das gezeichnet werden muss, um die Benutzerschnittstelle einer Funktion im prototypischen Wizard darzustellen (siehe Abschnitt 5.1.1). Dieses Composite ist innerhalb der Funktionsklasse zu implementieren.
- Mit den Methoden `getData()` und `setData()` können die benötigten Daten von und für die Funktion bereitgestellt werden. Allerdings müssen diese Methoden bei der Implementierung einer neuen Funktion vom Entwickler an den jeweiligen Bedarf angepasst werden, da je nach Anpassungsprozess und unterstütztem Dateiformat unterschiedliche Arten von Daten in Funktionen benötigt werden. Deswegen ist es sehr wichtig, dass der Entwickler genau spezifiziert und dokumentiert, welche Daten die von ihm implementierten Funktionen benötigen beziehungsweise zur Verfügung stellen.
- Die Methode `isComplete()` liefert `true` zurück, falls die Funktion vollständig ausgeführt wurde. Das wird benötigt, um zu prüfen, ob eine nachfolgende Funktion ausgeführt werden kann.
- Die Methode `updateControls()` kann aufgerufen werden, falls die Darstellung der Funktion in der Benutzeroberfläche aktualisiert werden soll, zum Beispiel um neue Daten anzuzeigen.

Das Interface `APAutomater` stellt eine Menge von Kernmethoden zur Verfügung. Weitere Methoden müssen vom Entwickler Format- und Funktions-spezifisch implementiert werden. Das ist darin begründet, dass die Automatisierung der Funktionalitäten je nach unterstützten Anpassungsprozessen und in einer Lernressource verwendeten Formaten sehr unterschiedlich erfolgen kann. Deswegen muss der Entwickler vor einer Erweiterung des Wizards prüfen, welche Funktionalitäten er zur Verfügung stellen möchte, und dementsprechend die Klassen, die die automatisierten Funktionen implementieren, für den konkreten Fall passend erweitern.

Während der Laufzeit des Wizards wird beim Anzeigen einer Seite für jede auf der Seite enthaltene Funktion mit der Controller-Methode `getAutomater()` geprüft, ob die Funktion automatisiert wurde. In diesem Fall liefert die Methode die Klasse zurück, die die Funktion implementiert. Diese Klasse enthält die Methode `getComposite()`, die das für die Anzeige der Funktion benötigte Composite realisiert. Dieses Composite wird dann auf der passenden Wizard-Seite angezeigt. Liefert `getAutomater()` `null` zurück, wird das vom WGT erzeugte Standard-Composite angezeigt. Bei komplexen Funktionen kann es notwendig sein, von diesem Vorgehen abzuweichen und die Composites direkt in der Klasse einer Seite zu implementieren, da es dann gegebenenfalls notwendig ist, die Anzeige abhängig vom übergeordneten Prozessschritt zu gestalten.

Abbildung 46 zeigt die Klassen und Interfaces, die während der Wizard-Automatisierung eine Rolle spielen sowie deren Attribute und Methoden. Außerdem wird hier exemplarisch die Klasse einer implementierten Funktion zum Löschen von Bildern gezeigt. Diese enthält die angepassten Methoden `getData()` und `setData()`.

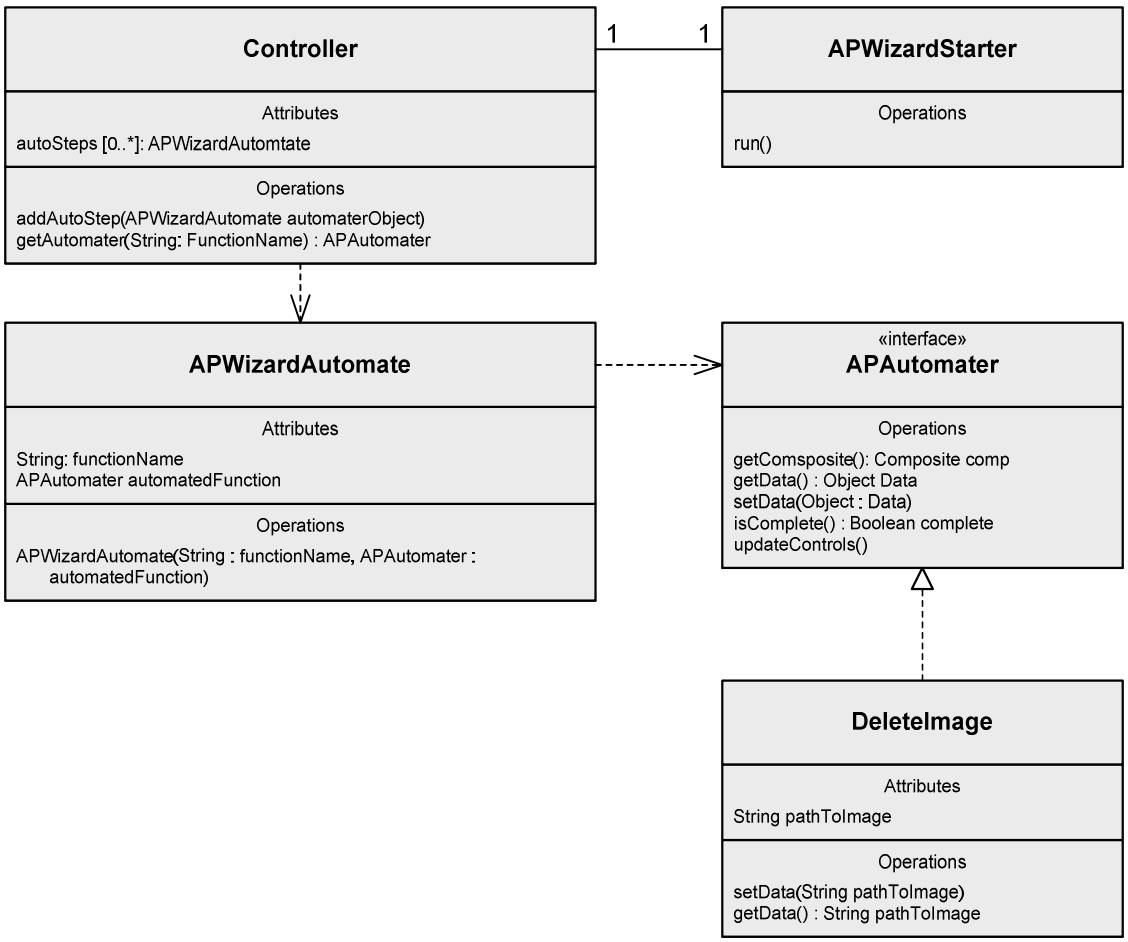


Abbildung 46: Klassendiagramm der bei der Automatisierung benötigten Klassen und Interfaces, sowie einer Beispiel-Funktion.

Alle Quelltext-Stellen, die für die Automatisierung relevant sind, werden durch entsprechende Kommentare gekennzeichnet. Durch weitere Kommentare lassen sich auch die durch Erweiterungen hinzu gekommenen Quelltextstellen und der ursprüngliche Quelltext gut trennen. Für einen Entwickler ist es so möglich, dem Wizard eigene Funktionen hinzuzufügen und auch später noch zu erkennen, wo der Wizard erweitert wurde.

6.2 Ein Werkzeug zur Unterstützung von Anpassungsprozessen

Das in dieser Arbeit vorgestellte Konzept zur automatischen Erzeugung eines Wizard basierend auf Beschreibungen von Anpassungsprozessen in Pattern-Form wurde entwickelt, um ein Werkzeug zur Unterstützung der in Kapitel 2 vorgestellten Anpassungsprozesse zu entwickeln. Wie in Kapitel 2 erläutert, ist die Anpassung eine der Aktionen des Repurposing. Oftmals kommt sie nicht allein vor, sondern in Zusammenhang

mit den beiden anderen Repurposing-Aktionen: Modularisierung und Aggregation. Deswegen sollte ein Anpassungswerkzeug nicht für sich allein stehen, sondern in ein größeres Framework eingebettet werden, das die Durchführung aller im Repurposing notwendigen Aktionen erlaubt. Im Rahmen des Content Sharing Projektes [4], das im folgenden Abschnitt vorgestellt wird, wurde ein derartiges Framework entwickelt.

6.2.1 Das Content Sharing Projekt

Das Content Sharing Projekt war ein vom deutschen Bundesministerium für Wirtschaft und Technologie gefördertes Forschungsprojekt. Ziel des Projektes war es, einen Marktplatz zum Austausch von Lernmodulen zwischen Herstellern von Lernressourcen und deren Konsumenten aufzubauen. Hintergrund hierfür war die Beobachtung, dass die Erstellung qualitativ hochwertiger Lernressourcen sehr kostenintensiv ist. Deswegen sollte eine Möglichkeit geschaffen werden, Lernressourcen in modularer Form für eine Wiederverwendung zur Verfügung zu stellen. Ein geeignetes Modularisierungskonzept erleichtert die Wiederverwendung [SN04], da man so genau diejenigen Module beziehen kann, die man zur Wiederverwendung benötigt. Beispielsweise kann ein Konsument einen kompletten Kurs wiederverwenden wollen, während ein anderer Konsument nur ein darin enthaltenes Flash-Modul benötigt.

Um dies zu ermöglichen, unterstützt das Content Sharing Projekt das folgende Szenario: Die Hersteller von Lernressourcen, die sogenannten Contentproduzenten, stellen die von ihnen erzeugten Lernressourcen auf einem Marktplatz in modularer Form für andere Nutzer zur Verfügung. Contentnutzer entnehmen die Module, die sie benötigen und stellen sie zu einer neuen Lernressource zusammen, die ihren Anforderungen entspricht. Die sogenannten Contentveredler erzeugen neue oder veränderte Lernressourcen durch Kombination vorhandener Lernressourcen mit neuen Lernressourcen oder durch Anpassung bereits existierender Lernressourcen. Diese neuen Lernressourcen können ebenfalls auf dem Marktplatz zur Verfügung gestellt werden (vergleiche Abbildung 47).

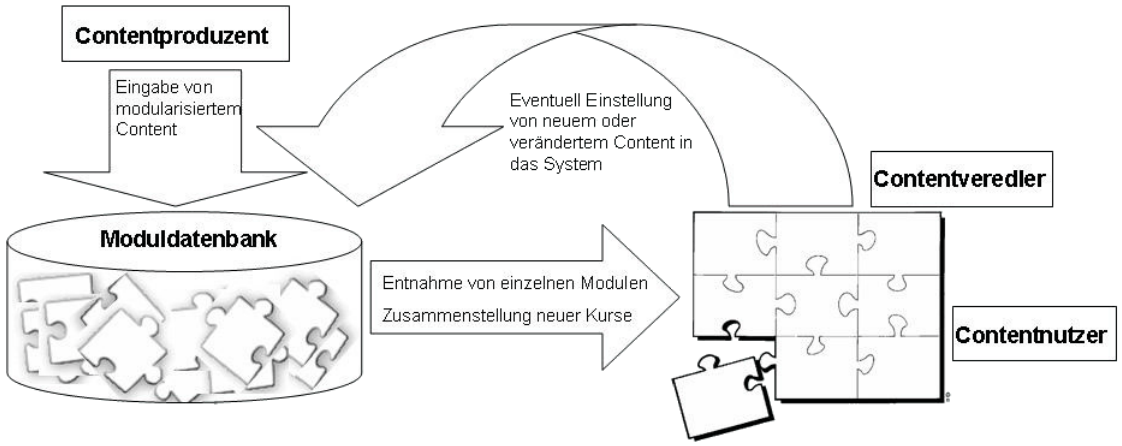


Abbildung 47: Das Content Sharing Szenario [4].

An diversen Stellen in diesem Szenario ist Repurposing notwendig, um Lernressourcen in der gewünschten Form zu erhalten: Der Contentproduzent sollte seine Lernressourcen bereits in modularer Form zur Verfügung stellen. Das erleichtert anderen Nutzern die

Wiederverwendung [Se02] und ermöglicht es, gezielt die Module beziehen zu können, die benötigt werden, und nicht einen kompletten Kurs kaufen zu müssen. Die Module können, den jeweiligen Anforderungen entsprechend, zu neuen Lernressourcen aggregiert werden. Dabei ist allerdings zu beachten, dass der bereits erwähnte Mosaik-Effekt entstehen kann. Durch Anpassung der wiederverwendeten Module an den neuen Einsatzkontext kann dem entgegen gewirkt werden.

Doch alle diese Aktionen sind kompliziert. Es wären Werkzeuge wünschenswert, die Anwender bei der Durchführung des gesamten Repurposing unterstützen. Da alle Repurposing-Aktionen zusammenhängen, wurde im Rahmen des Content Sharing Projektes eine Repurposing Suite entwickelt, die das gesamte Repurposing unterstützt. Sie wird im folgenden Abschnitt vorgestellt.

6.2.2 Die Repurposing Suite

Die Repurposing Suite [Me08] unterstützt Anwender dabei, Lernressourcen so zu verändern, dass sie für einen neuen Einsatzkontext geeignet sind. Dabei ist zu berücksichtigen, dass Lernressourcen üblicherweise aus einer Vielzahl von Dateien und Formaten bestehen. Außerdem muss das Werkzeug alle Aufgaben des Repurposing unterstützen: Modularisierung, Aggregation und Anpassung.

Um Format-unabhängige und Dateigrenzen-übergreifende Bearbeitung zu ermöglichen, arbeitet das Repurposing-Werkzeug mit einem Modell der zu bearbeitenden Lernressource. Dieses Modell basiert auf der sogenannten Content Ontology [Be$^+$06], einer Ontologie, die mögliche Elemente einer Lernressource, deren Eigenschaften sowie deren Beziehungen untereinander abbildet. Die Elemente der Lernressource können dabei ganz unterschiedlicher Granularität sein: Es werden sowohl gesamte Dokumente als auch einzelne Textfragmente oder Bilder abgebildet.

Das Modell der Lernressource ist eine aus drei Schichten bestehende Repräsentation der zu bearbeitenden Ressource [Me$^+$06] (vergleiche Abbildung 48):

1. Auf der untersten Ebene befindet sich die physikalische Repräsentation PCR (Physical Content Representation) der Ressource. Diese enthält die Dateien der zu bearbeitenden Ressource in der Form, in der sie auf einem Speichermedium abgelegt sind.
2. Darüber liegt eine objektorientierte Darstellung der Ressource, das sogenannte OOCR (Object Orientet Content Representation). Das OOCR ist eine Baumdarstellung der kompletten Lernressource. Die Elemente des Baumes werden dabei als von Java manipulierbare Objekte angelegt.
3. Die oberste Schicht wird durch eine semantisch angereicherte Darstellung der Lernressource gebildet. Diese wird als SCR (Sematic Content Representation) bezeichnet. Das SCR wird durch einen RDF Graphen realisiert, der neben der aus dem OOCR übernommenen Struktur, Relationen und Eigenschaften auch semantische Angaben zu den Elementen der Lernressource enthält.

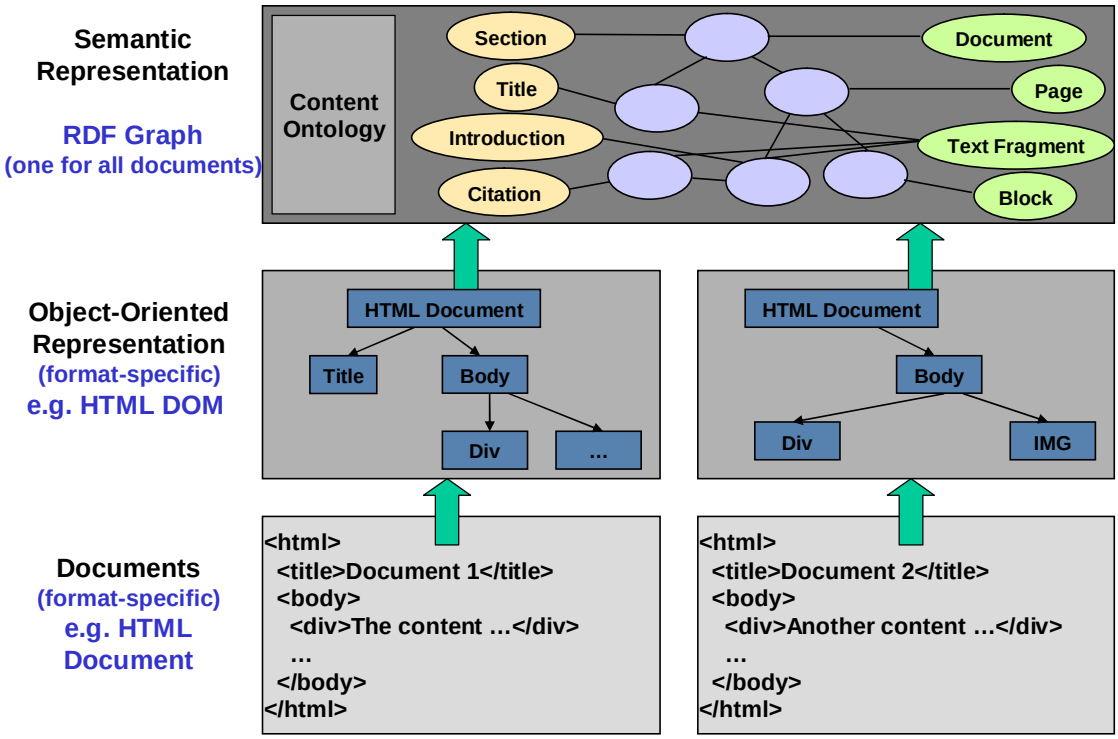


Abbildung 48: Schichten der Ressourcen-Repräsentation [Me08]

Die dreistufige Repräsentation der Lernressourcen ist der Kern des sogenannten Repurposing-Frameworks [Me+06]. Das Framework wiederum ist zentraler Bestandteil der Repurposing Suite, die sich aus dem Repurposing-Framework und den Repurposing-Anwendungen zusammensetzt. Abbildung 49 zeigt den Aufbau der Repurposing Suite:

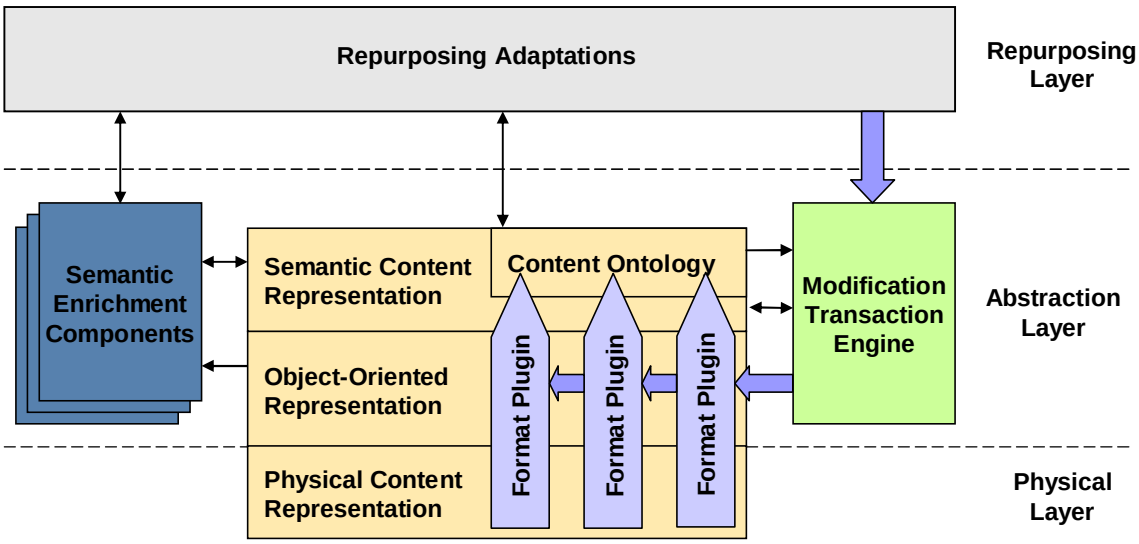


Abbildung 49: Die Komponenten der Repurposing Suite [Me+06].

Auf der untersten Ebene des Repurposing-Frameworks befindet sich die physikalische Repräsentation einer Lernressource. Mit Hilfe von Eclipse PlugIns werden die Elemente

der physikalischen Schicht abhängig von ihrem Format auf das OOCR abgebildet. Diese PlugIns werden als Format-PlugIns bezeichnet. Für jedes Format existiert ein eigenes PlugIn. Ein Application Programming Interface (API) erlaubt es, bei Bedarf weitere PlugIns für neue Formate zuzufügen. Dadurch wird die Forderung nach einer Erweiterbarkeit hinsichtlich neuer Formate erfüllt.

Das OOCR ist eine Java-Baumstruktur, die die gesamte Lernressource in Form von Java-Objekten darstellt. Für jedes Dokument der Lernressource existiert ein eigener Unterbaum. Dieser enthält die im Dokument enthaltenen Elemente, wie Links, Bilder, Absätze etc. Für jeden dieser Elementtypen existiert ein Format-übergreifendes Interface, das die Methoden und Eigenschaften des jeweiligen Elementtyps festlegt. Die Format PlugIns enthalten eine formatspezifische Implementierung der Interfaces. Dadurch wird es ermöglicht, das OOCR aus den Dateien zu generieren und nach der Bearbeitung verlustfrei in die physikalischen Objekte zurück zu transferieren.

Neben Methoden zum Aufbau und Speichern des OOCR enthalten die Interfaces auch Methoden, um das OOCR ins SCR zu überführen. Das SCR ist eine semantische Repräsentation der zu bearbeitenden Lernressource. Direkt nach seiner Erzeugung bildet es die Struktur des OOCR und die Eigenschaften der OOCR-Objekte ab. Bei Bedarf wird es um semantische Angaben erweitert. Das SCR wird in Form eines RDF-Graphen [20] erstellt, dessen Elemente durch die bereits vorgestellte Content-Ontologie definiert sind.

Welche Eigenschaften und Beziehungen für ein Element im SCR berücksichtigt werden, ist formatspezifisch festgelegt. Somit sind die Format PlugIns dafür zuständig, dass diese Informationen im SCR zur Verfügung gestellt werden. Für die semantische Anreicherung des SCR existieren semantische Anreicherungskomponenten, die sogenannten SECs (Semantic Enrichement Component). Diese werden bei Bedarf von den Repurposing-Anwendungen aufgerufen. So kann z.B. die Anpassungsanwendung bei der Übersetzung die passende SEC dazu veranlassen, für alle Textstellen zu prüfen, in welcher Sprache diese vorliegen. Dieses Vorgehen stellt sicher, dass immer nur die Informationen im SCR enthalten sind, die auch tatsächlich benötigt werden.

Nutzer interagieren mit den Repurposing-Anwendungen. Diese haben Zugriff auf das SCR. Da dieses von Formaten und Dateigrenzen abstrahiert, wird so die gewünschte Format-unabhängige und Dateigrenzen-übergreifende Bearbeitung ermöglicht. Über den Zugriff der Repurposing-Anwendungen auf das SCR wird Anwendern Zugang auf die zur Durchführung einer speziellen Aufgabe benötigten, im SCR enthaltenen Informationen ermöglicht. Der Nutzer kann in der Benutzeroberfläche Änderungen an der Lernressource auslösen, die in die SCR übernommen werden.

Um die Änderungen speichern zu können, muss erst das OOCR verändert werden, das dann in die physikalischen Dateien überführt werden kann. Doch eine Rücktransformation des SCR auf das OOCR kann zu Informationsverlusten führen, da das SCR meist nicht die vollständigen Informationen aller Objekte enthält. Um dies zu vermeiden, wurde ein weiterer Bestandteil des Repurposing-Frameworks entwickelt: Die Modification Transaction Engine (MTE).

Die MTE wird von den Repurposing-Anwendungen über die im OOCR benötigten Änderungen, die atomaren Unterschritten vom Typ „Ausführung“ entsprechen, informiert. Sie ermittelt daraufhin für alle betroffenen Elemente die zuständigen Format PlugIns

und sorgt dafür, dass die Änderungen von den Format PlugIns im OOCR vorgenommen werden, das verlustfrei in die jeweiligen Elemente der physikalischen Ebene überführt werden kann. Weiterhin werden die im OOCR vorgenommen Änderungen durch die Format PlugIns auch im SCR durchgeführt. Die MTE bekommt dazu den Namen des gewünschten Unterschrittes und die Elemente, für die dieser Unterschritt ausgeführt werden soll, übergeben. Anhand der Elemente kann die MTE die zuständigen Format PlugIns ermitteln. Diese wiederum bekommen die Änderung und die Elemente, deren Format sie behandeln, übergeben. Die Format PlugIns realisieren dann die Unterschritte, indem sie die zur Ausführung benötigten automatisierten Funktionen aufrufen, um das OOCR gemäß der gewünschten Änderung zu verändern.

Die Repurposing-Anwendungen sind als Eclipse PlugIns realisiert und über eine API ins Framework eingebunden. So ist es jederzeit möglich, bei Bedarf neue Anwendungen anzubinden. Im Rahmen des Content Sharing Projektes wurden drei Anwendungen zur Unterstützung bei der Durchführung aller Aktionen des Repurposing entwickelt:

Die **Modularisierungsanwendung** ist für die Zerlegung einer Lernressource in Module zuständig. Dabei ist es möglich, die gewünschte Granularität der Module vorzugeben. Die Modularisierungsanwendung wird detailliert vorgestellt in [Me08].

Die **Aggregationsanwendung** erlaubt es, einzelne Module zu einem neuen Kurs zusammenzufügen. Auch diese Anwendung wird in [Me08] detailliert erläutert.

Die **Anpassungsanwendung** unterstützt Anwender bei der Durchführung der bereits erläuterten Prozesse zur Anpassung existierender Lernressourcen an neue Einsatzszenarien.

Zusätzlich wurde ein Strukturediteor entwickelt, der es erlaubt, die Anordnung der Module innerhalb einer Lernressource zu verändern. Außerdem gibt es ein Werkzeug zur Verwaltung aller Module, die zur Bearbeitung zur Verfügung stehen.

Zentral für die Bearbeitung von Lernressourcen ist der sogenannte Moduleditor (vergleiche Abbildung 50). Der Moduleditor realisiert das Framework der Repurposing Suite [Me08]. Über ihn sind die Repurposing-Anwendungen an das Framework angebunden und stehen Benutzern zur Verfügung. Er stellt außerdem eine Benutzeroberfläche zur Verfügung, über die auf die zu bearbeitenden Module und die in einem Modul enthaltenen Submodule zugegriffen werden kann.

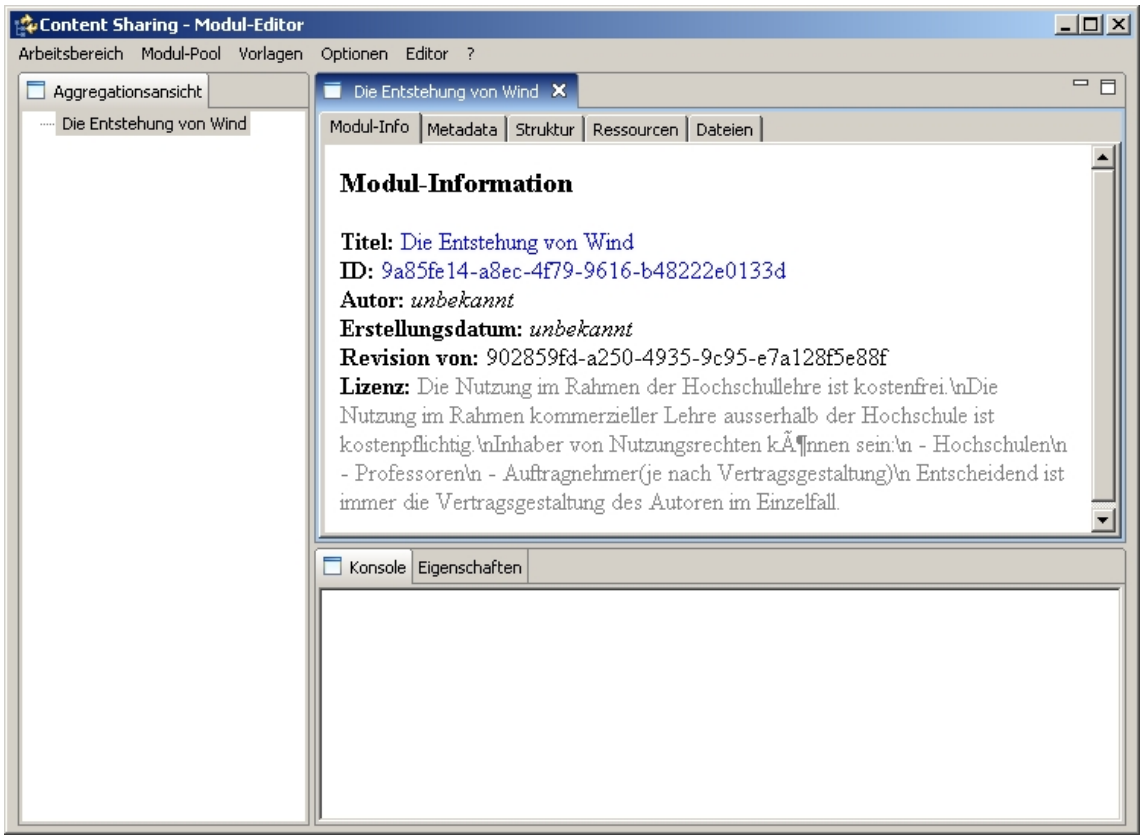


Abbildung 50: Die Benutzeroberfläche des Moduleditors.

6.2.3 Funktionsweise des Anpassungswerkzeuges

Im vorigen Abschnitt wurde die Repurposing Suite vorgestellt und es wurde ein Überblick über deren Bestandteile gegeben. In diesem Abschnitt wird eine der Repurposing-Anwendungen genauer vorgestellt: das Anpassungswerkzeug. Zuerst wird die Funktionsweise des Werkzeugs betrachtet. Dann wird erläutert, wie dieses Werkzeug durch Integration und Erweiterung des automatisch generierten prototypischen Wizards zur Unterstützung von Anpassungsprozessen (vergleiche Abschnitt 5.1) entstanden ist.

Das Anpassungswerkzeug unterstützt Anwender bei der Durchführung von Anpassungsprozessen. Ausgangspunkt für die Entwicklung des Anpassungswerkzeuges war die in Abschnitt 2.3 präsentierte Anforderungsanalyse. Entwickelt wurde das Werkzeug mit dem in dieser Arbeit vorgestellten Ansatz: Basierend auf einer Benutzerumfrage wurden in Zusammenarbeit mit Prozessexperten Anpassungspatterns erstellt. Diese wurden in das PIT eingegeben. Basierend auf den resultierenden Prozessbeschreibungen wurde mit dem WGT ein prototypischer Wizard erstellt. Dieser wurde in die Repurposing Suite integriert und um automatisierte Funktionen erweitert.

Das Anpassungswerkzeug ist Bestandteil der Repurposing Suite und über den Moduleditor angebunden. Nutzer wählen zuerst innerhalb des Moduleditors das Modul aus, das sie anpassen möchten. Dabei kann es sich um einen kompletten Kurs handeln; es kann aber auch ein kleinerer Bestandteil eines Kurses angepasst werden, etwa weil er

nachträglich in einen anderen, bereits existierenden Kurs eingefügt wurde. Um das Anpassungswerkzeug zu starten, ist aus der Reihe der Operationen, die für ein Modul möglich sind, der Punkt „Anpassungswerkzeug starten“ zu wählen (Abbildung 51).

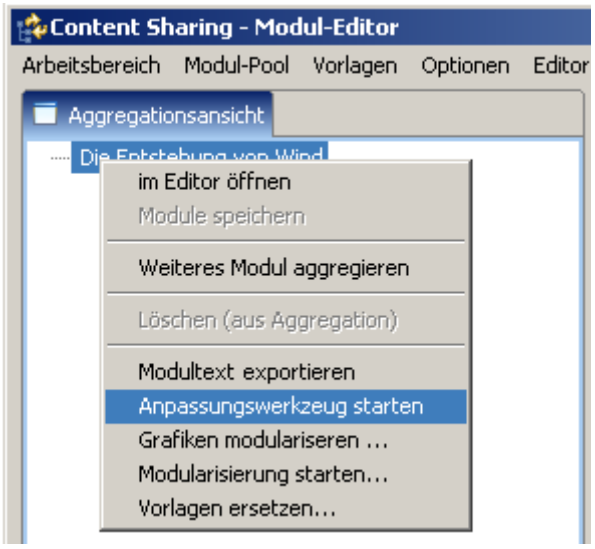


Abbildung 51: Starten des Anpassungswerkzeuges.

Nach Starten des Anpassungswerkzeuges öffnet sich der Wizard, der die Benutzer durch die Anpassungsprozesse führt (siehe Abbildung 52).

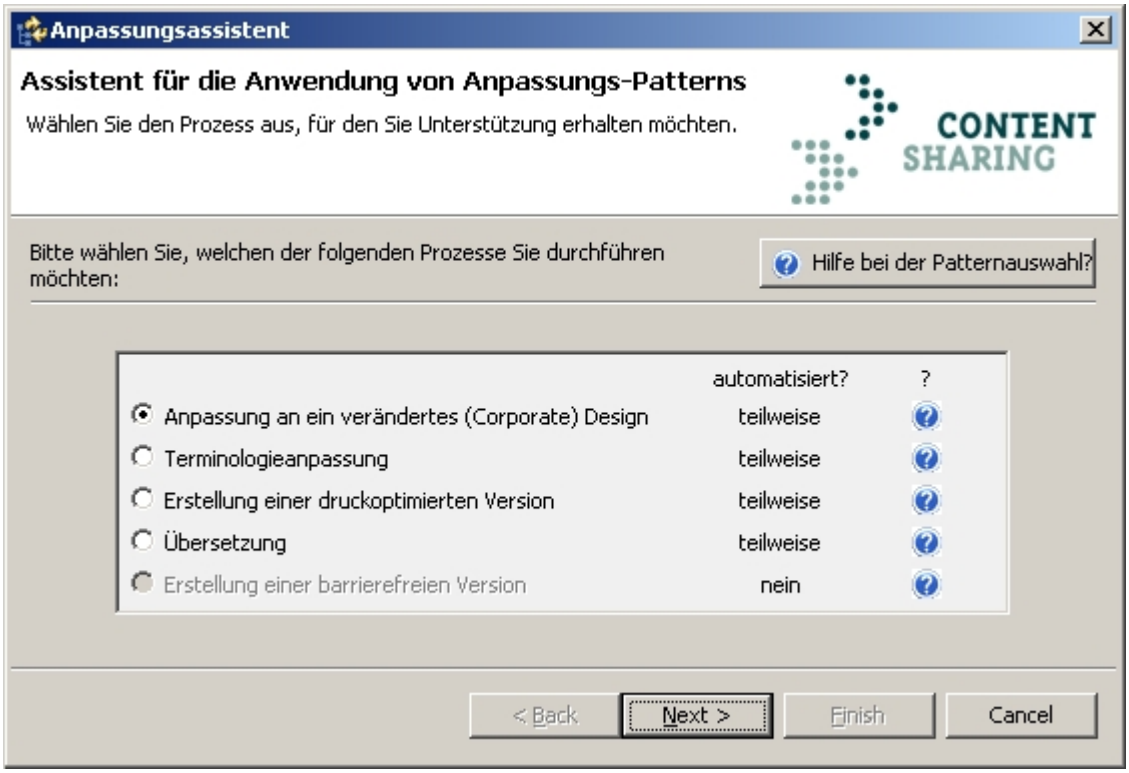


Abbildung 52: Die Startseite des Wizard des Anpassungswerkzeuges.

Der Wizard, durch den das Anpassungswerkzeug realisiert ist, basiert auf dem vom WGT aus den Anpassungspatterns erzeugten Prototyp. Ein Anwender sieht auf der Startseite eine Auflistung der vom Anpassungswerkzeug unterstützten Anpassungsprozesse. Außerdem bekommt er angezeigt, zu welchem Grad jeder Prozess unterstützt wird. In Abschnitt 6.2.4 wird erläutert, wie diese Informationen gewonnen werden. Über den Button „Hilfe bei der Prozessauswahl“ ist es zudem möglich, weitere Informationen aus den Prozessbeschreibungen angezeigt zu bekommen, die bei der Auswahl des durchzuführenden Prozesses helfen.

Das Anpassungswerkzeug soll nicht nur für Neulinge und Laien eine wertvolle Unterstützung sein, sondern auch Experten die tägliche Arbeit erleichtern. Um dies zu ermöglichen, wurden im Anpassungswerkzeug zwei Modi vorgesehen: Ein Modus für Neulinge und Laien sowie ein weiterer Modus für Experten. Anwender haben die Möglichkeit, den für sie geeigneten Modus auszuwählen. Während Neulingen und Laien ausführliche Erläuterungen zu den einzelnen Schritten des Wizards angeboten werden, werden Experten direkt durch den Anpassungsprozess geführt, ohne dass ihnen eine Vielzahl von Erklärungen gegeben wird, die sie nicht benötigen. Sowohl für Neulinge als auch für Experten ist auf der Startseite zu jedem Anpassungsprozess eine Erläuterung verfügbar. Auch diese ist für Neulinge deutlich detaillierter, während sie für Experten nur einen groben Überblick bietet. Bei Bedarf können sich aber auch Experten die ausführliche Erklärung anzeigen lassen.

Die Erläuterung zur Prozessdurchführung lässt sich durch Anklicken des Fragenzeichens hinter dem jeweiligen Prozess aufrufen (siehe Abbildung 52). Diese Anleitung wird ebenfalls aus der Pattern-basierten Prozessbeschreibung generiert (vergleiche Abschnitt 5.3). Dazu wird die XML-Darstellung mit Hilfe von Stylesheets nach HTML umgewandelt und als Hilfedokument (vergleiche Abbildung 53) zur Verfügung gestellt.

Abhängig davon, ob der Benutzer im Experten- oder Laienmodus arbeitet, werden unterschiedliche Stylesheets verwendet. Dadurch wird erreicht, dass die Sicht den jeweiligen Anforderungen gerecht wird. Abbildung 53 zeigt einen Ausschnitt aus einem derartigen Hilfedokument in der Sicht des Laien.

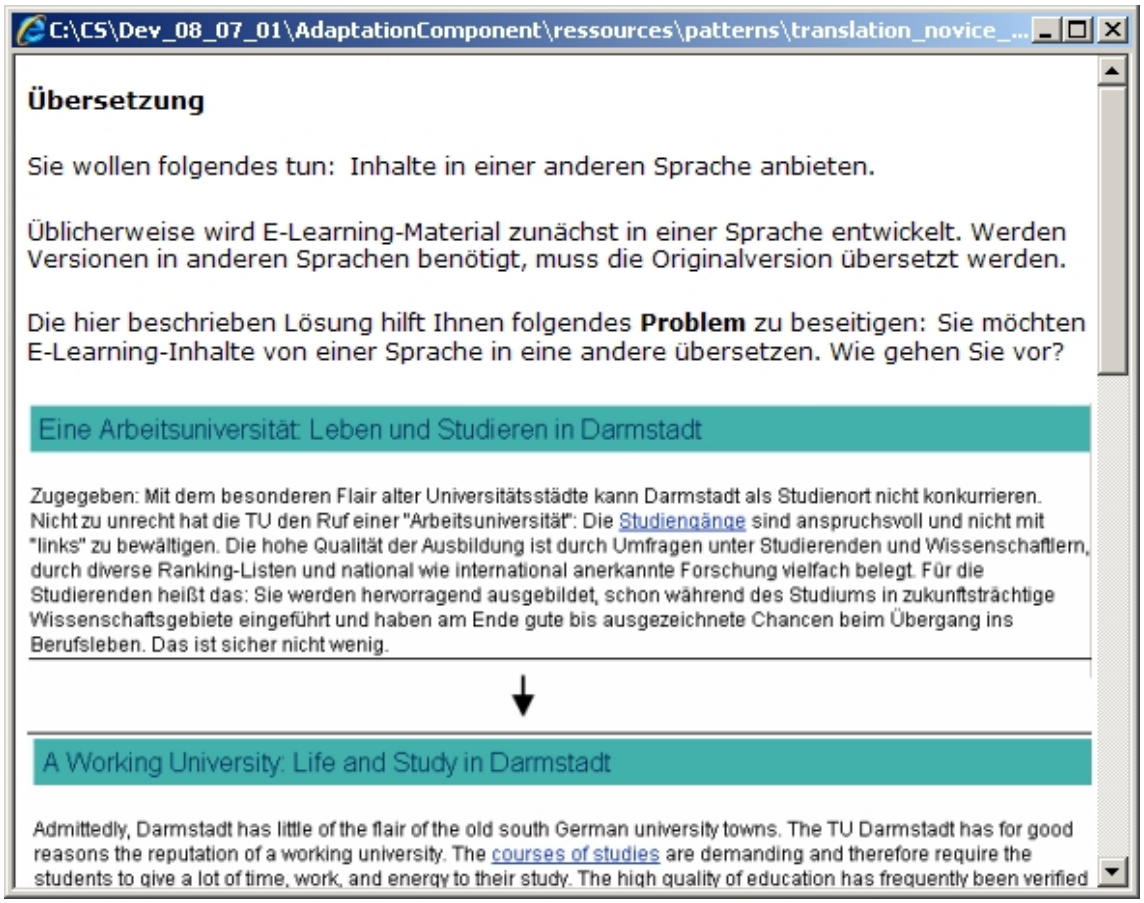


Abbildung 53: Das Hilfedokument zur Anpassung „Übersetzung".

Das Anpassungswerkzeug unterstützt derzeit 5 Anpassungen. Für alle unterstützten Anpassungen wurden Prozessexperten befragt, wie sie bei der jeweiligen Anpassung vorgehen. Dieses Wissen ist sowohl in die Anleitung für die Prozesse, als auch in den zur Unterstützung der Prozesse erstellten Prototypen eingeflossen. Vier der Anpassungsprozesse werden teilautomatisiert unterstützt. Für den fünften Prozess, die Anpassung zum Erhalten eine barrierefreie Version, ist eine Anleitung zur manuellen Durchführung des Prozesses erhältlich.

Folgende Anpassungen, die nachfolgend vorgestellt werden, sind unterstützt:

1. Übersetzung
2. Anpassung an ein verändertes (Corporate) Design
3. Anpassung, um eine druckoptimierte Version zu erhalten
4. Terminologieanpassung
5. Anpassung, um eine barrierefreie Version zu erhalten

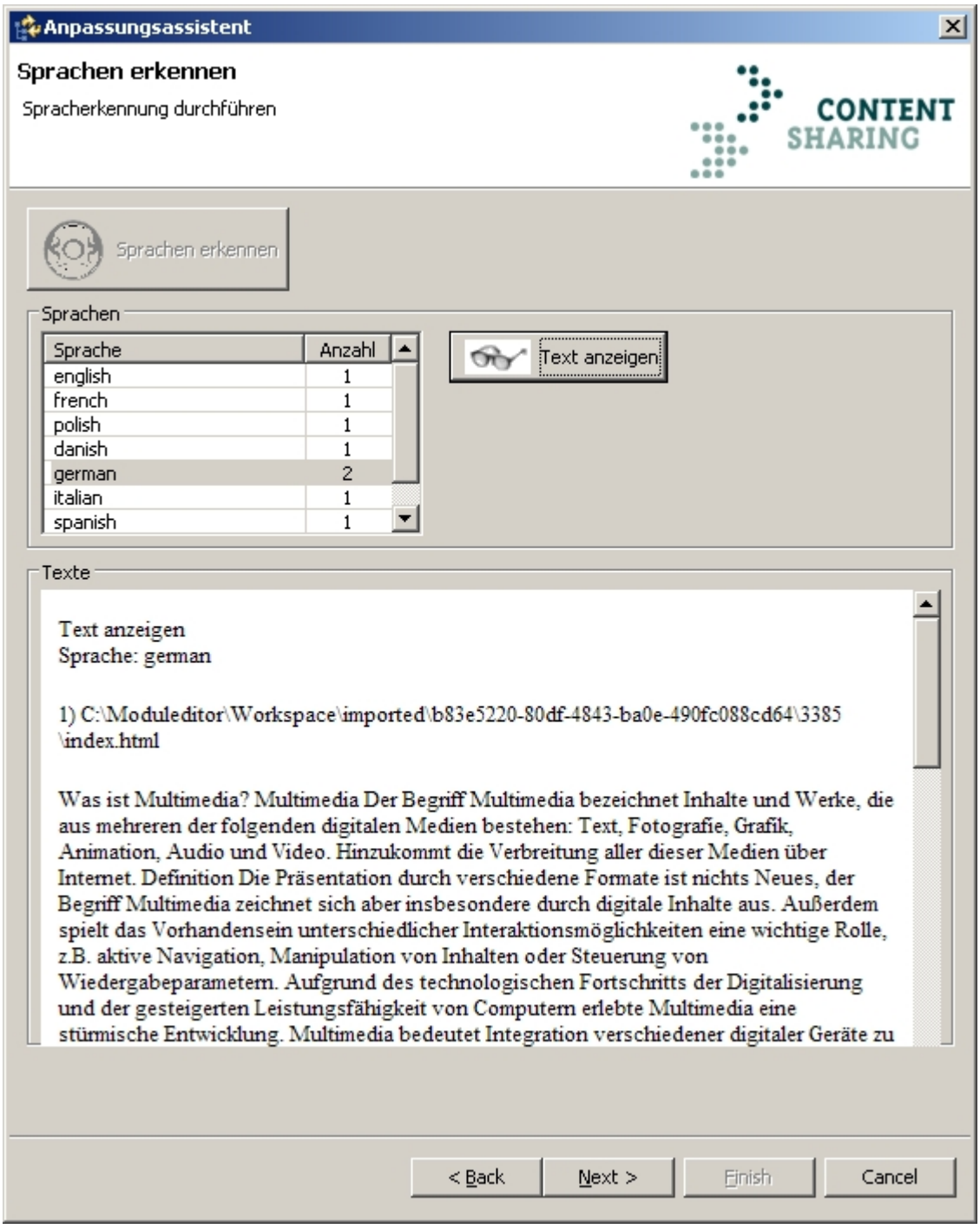


Abbildung 54: Bei der Übersetzung Sprachen erkennen.

Da es derzeit keine zufriedenstellenden vollautomatischen Übersetzungswerkzeuge gibt, unterstützt das Anpassungswerkzeug bei der **Übersetzung**, überlässt aber die eigentliche Übersetzung einem Menschen. Dazu bietet es die Möglichkeit, die im ausgewählten Modul vorhandenen Sprachen zu ermitteln (Abbildung 54). Dafür wurde eine SEC entwickelt, die basierend auf Heuristiken pro Textstelle die Sprache der jeweiligen Textstelle feststellt. Dadurch lässt sich schnell ermitteln, ob ein Text Sprachen enthält, die

die Zielgruppe nicht versteht und die deswegen übersetzt werden müssen. Dies kann beispielsweise bei nicht übersetzten Zitaten in der Originalsprache der Fall sein.

Nutzer können die Texte, die übersetzt werden sollen, in eine csv-Datei exportieren. Die exportierten Texte können so in eine Reihe anderer Programme importiert und dort übersetzt werden. Die eigentliche Übersetzung sollte manuell geschehen, da derzeit noch keine Programme existieren, die bei der Übersetzung die Qualität eines menschlichen Übersetzers erreichen. Das ist insbesondere bei Lerninhalten wichtig, da hier fehlerhafte Ausdrücke den Lernenden das Verständnis erschweren.

Man kann eine exportierte Textdatei auch mit dem Anpassungswerkzeug öffnen und dann Textstelle für Textstelle durch die Datei navigieren. Dabei sieht man, ob es für bestimmte Textstellen bereits eine Übersetzung gibt. Ist das nicht der Fall, kann man eine Übersetzung anlegen. Dadurch erspart man sich, dass man nach kleinen Änderungen in einer Datei den kompletten Kurs noch einmal übersetzen muss. Es reicht, den geänderten Teil zu übersetzen. Außerdem kann man häufig verwendete Formulierungen in dieser Form bereithalten und bei neuen Übersetzungen wiederverwenden.

Für den Übersetzungs-Wizard wurde mit Hilfe vom PIT eine Prozessbeschreibung erstellt, die über die Anleitung (vergleiche Abbildung 53) in das Anpassungswerkzeug eingebunden ist. Über diese Anleitung erhalten Anwender eine ausführliche Beschreibung, wie sie bei Übersetzungen vorgehen können. Da es hier aber, wie erwähnt, noch keine sinnvolle, automatisierte Unterstützung gibt, wurde nicht mit dem WGT ein Wizard erzeugt. Stattdessen wurden nur die Funktionalitäten, die oft im Rahmen der Übersetzung benötigt werden, zur Verfügung gestellt. Diese waren bereits aufgrund einer vorherigen prototypischen Implementierung vorhanden. Sollten zu einem späteren Zeitpunkt geeignete Funktionen für eine automatisierte Übersetzung zur Verfügung stehen, so kann mit dem WGT ein Wizard für die Übersetzung erzeugt werden und die Funktionen können, wie in Abschnitt 6.1 erläutert, zugefügt werden.

Die **Anpassung an ein verändertes (Corporate) Design** wird teilweise automatisch unterstützt. Folgende Funktionen werden automatisiert zur Verfügung gestellt: Man kann Bilder löschen (Abbildung 55), ersetzen oder ihre Größe ändern. Zusätzlich lassen sich Hintergrundfarben verändern. Außerdem kann man Schriftart, -farbe und -größe anpassen. In allen Fällen ermittelt das Anpassungswerkzeug jeweils alle Vorkommen der zu verändernden Objektart, also Bilder (Abbildung 56), Hintergründe und Schriftstile, und listet diese den Benutzern auf. Nutzer können dann auswählen, welche Elemente gelöscht oder verändert werden sollen. Zusätzlich gibt es eine Anleitung, die die gesamte Anpassung (auch die nicht automatisierten Teile) erläutert.

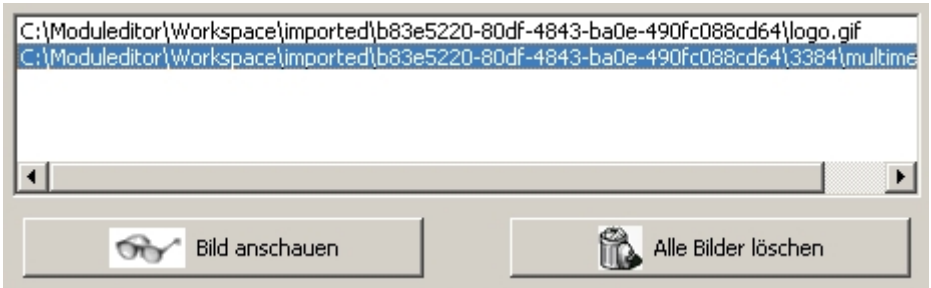


Abbildung 55: Bilder löschen.

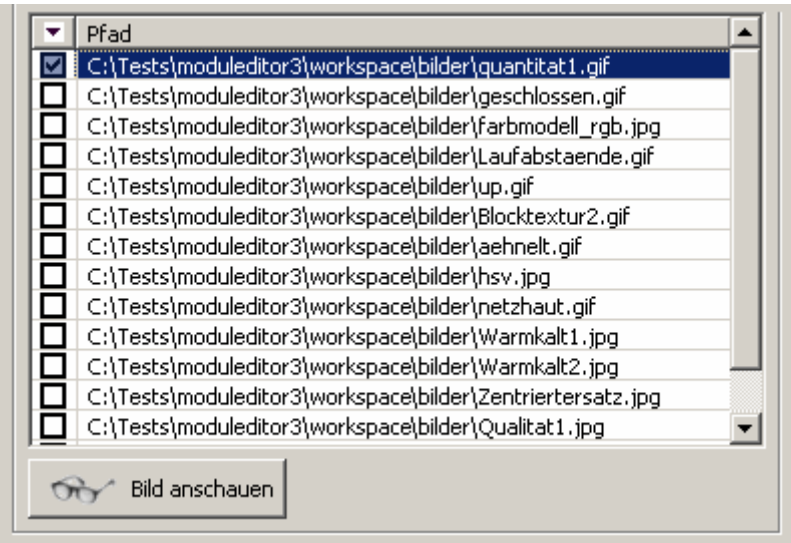


Abbildung 56: Bilder auswählen.

Durch Zufügen geeigneter SECs wäre es außerdem möglich, Elemente auch inhaltlich zu unterscheiden. So könnte man beispielsweise erkennen, welche Bilder Logos darstellen, welche Fotografien etc. Das würde es ermöglichen, beispielsweise gezielt alle Logos zu identifizieren und zu ersetzen. Im Rahmen dieser Arbeit wurde die Möglichkeit vorgesehen, derartige SECs zu integrieren. Sie wurden aber nicht realisiert. Die Umsetzung müsste in einer Nachfolgearbeit geschehen.

Der für die Unterstützung der Anpassungen an ein verändertes (Corporate) Design benötigte Wizard wurde mit dem PIT und dem WGT erzeugt. Die automatisierten Funktionen beruhen auf einer gemeinsamen Arbeit zur Anpassung des Layouts von Lernressourcen mit Metzger [Me+07]. In dieser Arbeit wurden die im Anpassungswerkzeug eingebundenen Möglichkeiten zur Anpassung an ein verändertes (Corporate) Design untersucht und teilweise implementiert.

Bei der **Anpassung zur Erzeugung einer druckoptimierten Version** wird für das ausgewählte Modul eine separate Druckversion angelegt. Für diese Version wird für alle Elemente überprüft, ob sie sich sinnvoll auf einer Seite in einem von den Anwendern festzulegenden Seitenformat (z.B. DIN A4 oder Letter) drucken lassen. Elemente die zu groß sind, werden Benutzern gemeldet (Abbildung 57). Diese haben dann die Möglichkeit, die Elemente mit Hilfe des Anpassungswerkzeuges zu verkleinern. Dabei wird berücksichtigt, dass es Elemente gibt, bei denen es möglich ist, sie in der Höhe zu unterteilen und auf zwei Seiten auszudrucken. Dies ist beispielsweise bei textuellen Elementen möglich. Es gibt aber auch Elemente, bei denen eine Aufteilung auf zwei Seiten nicht gewünscht ist, beispielsweise bei Bildern. Bei textuellen Objekten ist zu berücksichtigen, dass der Seitenumbruch an einer sinnvollen Stelle stattfindet. Ein Umbruch innerhalb eines Wortes ist beispielsweise nicht sinnvoll.

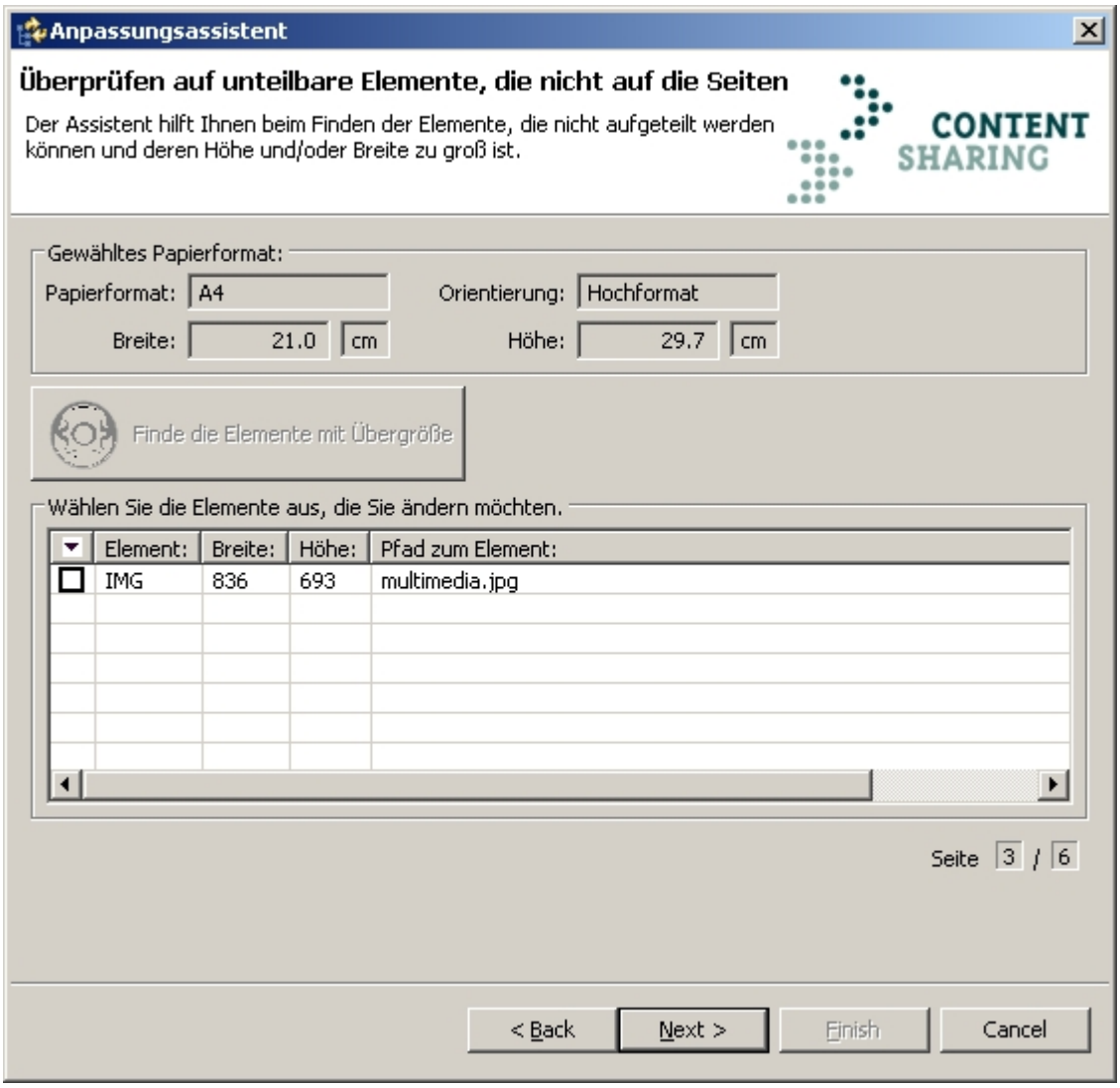


Abbildung 57: Elemente, die zu groß sind für die Seitengröße.

Die Druckbarkeitsanpassung beruht auf einer gemeinsamen Arbeit mit Horneff [Ho06]. Resultat dieser Arbeit war ein Prototyp zur Druckbarkeitsanpassung. Dieser wurde in das Anpassungswerkzeug integriert. Es wurde daher kein zusätzlicher Wizard mit dem WGT erzeugt. Aber mit PIT wurde eine Prozessbeschreibung basierend auf den bei der Prototyp-Entwicklung gesammelten Erfahrungen erstellt. Diese wurde als Anleitung zur Druckbarkeitsanpassung zur Verfügung gestellt.

Hintergrund der **Terminologieanpassung** ist, dass unterschiedliche Firmen oder auch unterschiedliche Branchen häufig eine spezielle Terminologie verwenden. Wird ein Kurs in einem anderen Einsatzgebiet wiederverwendet, so ist daher zu überlegen, ob die Terminologie anzupassen ist. Das Anpassungswerkzeug unterstützt hierbei, in dem es zum einen die Möglichkeit bietet, einzelne Begriffe zu ersetzen. Beispielsweise könnte in der Firma, für die ein Kurs ursprünglich erstellt wurde, der Begriff Peer-to-Peer verwendet werden. In einer anderen Firma, die den Kurs wiederverwenden möchte, wird

dagegen die Abkürzung P2P verwendet. Mit Hilfe des Wizards kann man alle Vorkommen von „Peer-to-Peer“ durch „P2P“ ersetzen (Abbildung 58).

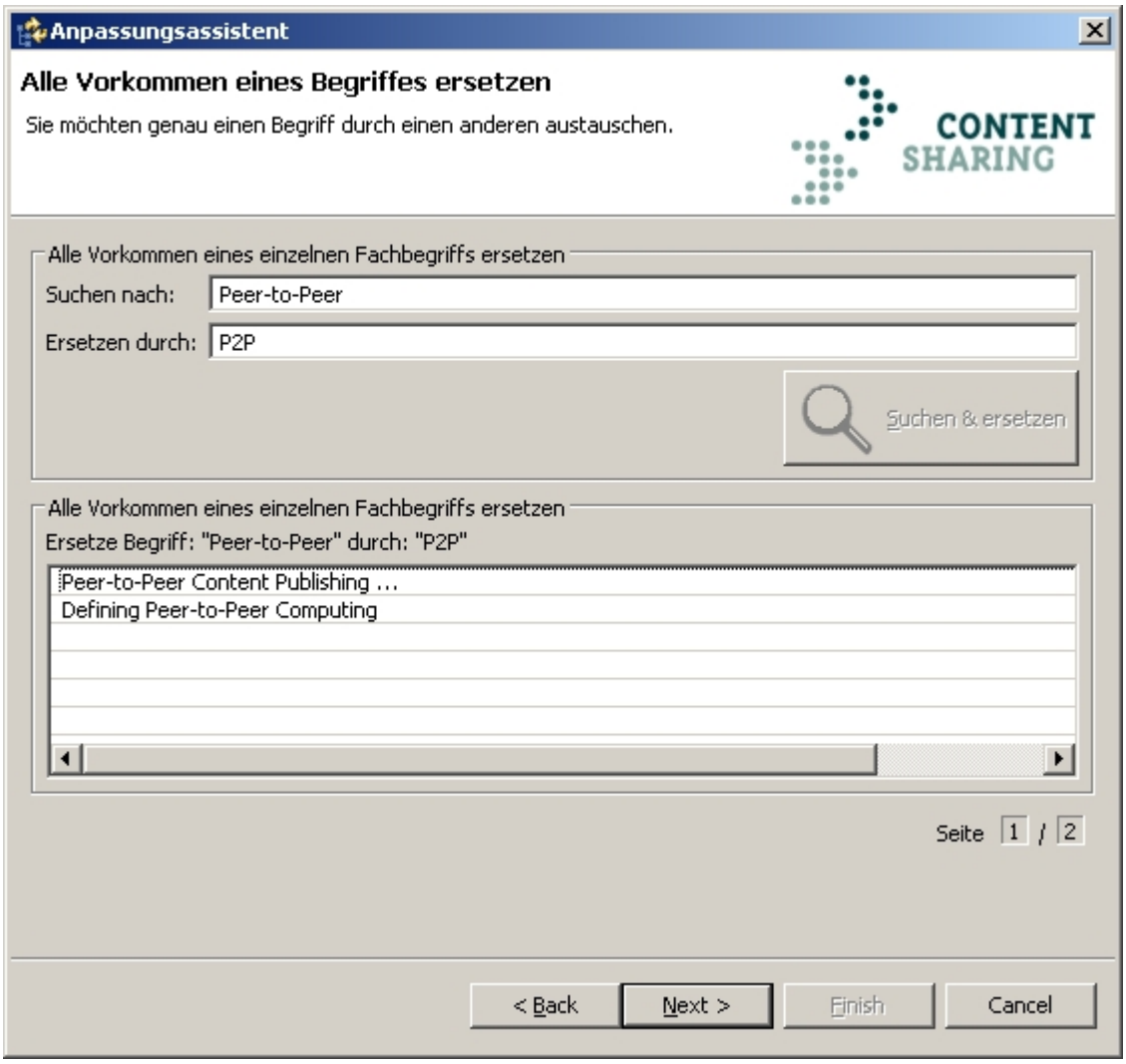


Abbildung 58: Einen Begriff ersetzen.

Ist nicht nur ein Begriff zu ersetzen, sondern eine Menge von Begriffen, so ermöglicht das Anpassungswerkzeug, eine Liste mit Termpaaren zu pflegen, die Terme und deren Ersetzungen enthält (Abbildung 59). Das Anpassungswerkzeug stellt Funktionen zur Verfügung, um diese Liste anzulegen und zu pflegen. Außerdem ist eine Funktion vorhanden, die alle Terme der Liste durch deren Ersetzungen austauscht. Die Ersetzungsliste kann immer wieder genutzt werden und so sicherstellen, dass bei verschiedenen Lernressourcen für eine Zielgruppe immer die gleiche Terminologie verwendet wird.

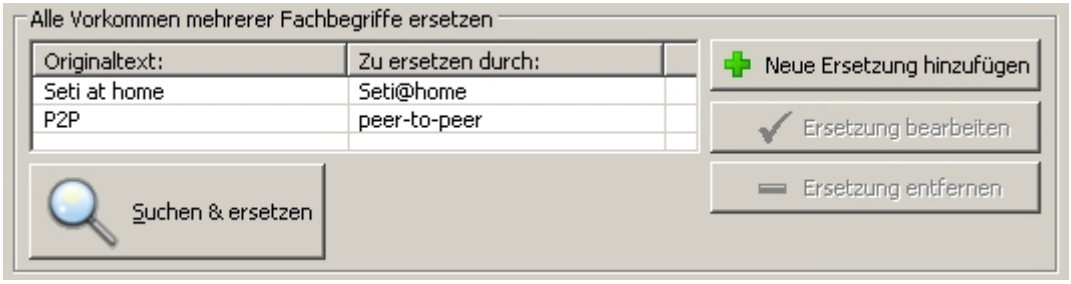


Abbildung 59: Mehrere Begriffe ersetzen.

Für die Terminologieanpassung wurde mit dem PIT eine Prozessbeschreibung erstellt, die mit dem WGT in einen prototypischen Wizard überführt wurde, der die Basis für die Terminologieanpassung des Anpassungswerkzeuges bildet.

Für die **Anpassung zum Erzeugen barrierefreier Versionen** ist derzeit eine Unterstützung mittels eines Leitfadens zur Durchführung (siehe Abbildung 60) realisiert. Eine Integration des WGT-Prototyps und eine teilweise Automatisierung sind in zukünftigen Arbeiten zu realisieren. Der Leitfaden basiert erneut auf einer mit dem PIT erstellten Prozessbeschreibung und erläutert Anwendern, was sie berücksichtigen müssen, wenn sie vorhandene Lernressourcen barrierefrei gestalten wollen.

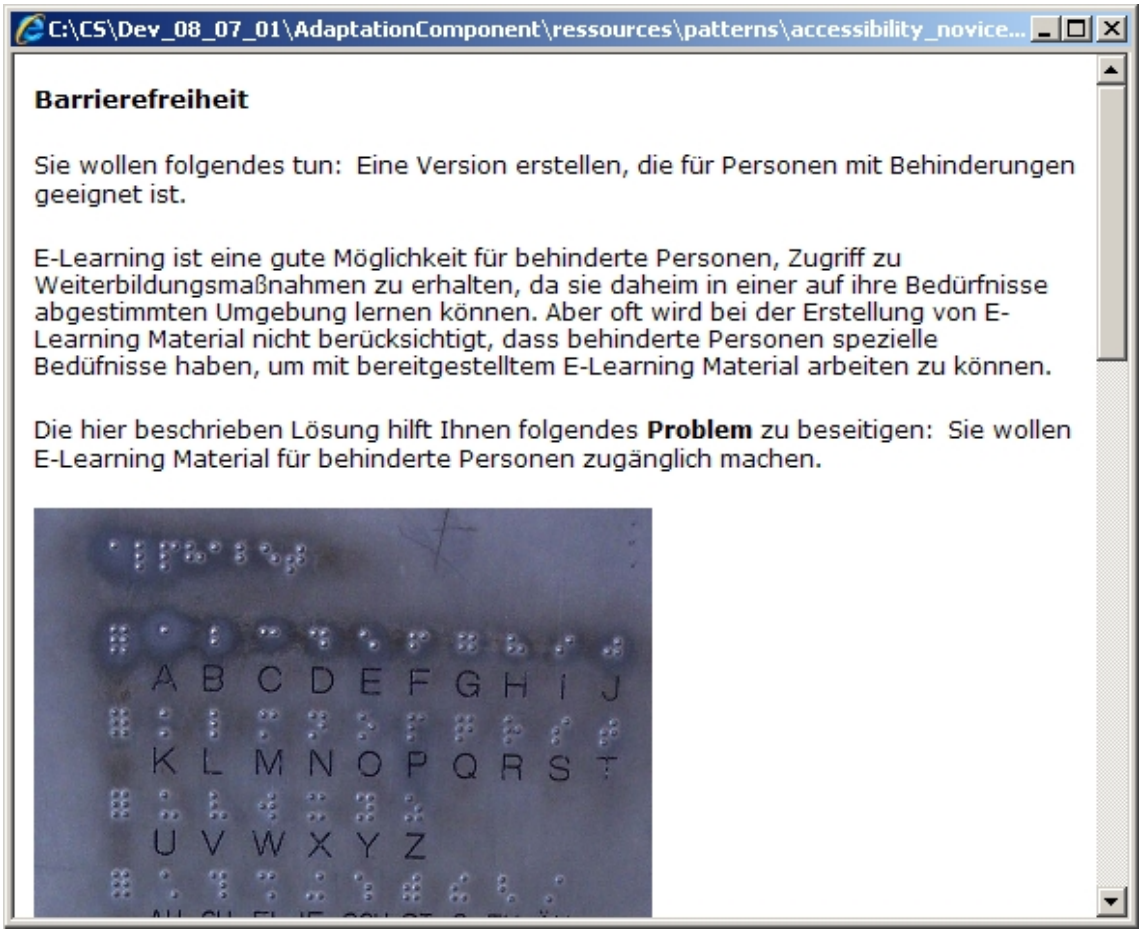


Abbildung 60: Hilfe zu Barrierefreiheit.

Das Anpassungswerkzeug führt Anwender schrittweise durch die genannten Anpassungsprozesse. Es unterstützt Anwender dabei, die benötigten Anpassungsprozesse, Prozessschritte und atomaren Unterschritte zu identifizieren und deren Durchführung zu steuern. Dazu tritt es mit dem Anwender in Interaktion, um die zur Durchführung nötigen Angaben vom Anwender zu erfragen, dem Anwender relevante Systeminformationen zu zeigen und Rückmeldung über den aktuellen Stand der Durchführung zu geben.

Das Anpassungswerkzeug ist aufgebaut wie in Abschnitt 5.1 erläutert: Es enthält für jede zur Verfügung stehende Anpassungsart einen Eintrag auf der Startseite, so dass Benutzer die von ihnen benötigte Anpassungsart auswählen können. Für jeden Prozess existiert eine Seite, auf der Anwender alle enthaltenen Prozessschritte sehen und wählen können, welche der freiwilligen Schritte sie ausführen wollen. Weiterhin existiert eine Seite pro Prozessschritt, die einen Überblick über den Prozessschritt gibt und die enthaltenen atomaren Unterschritte benennt. Die Unterschritte sind in logischen Einheiten auf Seiten gruppiert, die die Durchführung der Unterschritte erläutern.

Zwischen den einzelnen Anpassungen bestehen Beziehungen. So führt zum Beispiel eine Terminologieanpassung dazu, dass eine eventuell existierende Übersetzung auch angepasst werden muss. Ein weiteres Beispiel ist, dass nach Erstellen einer Druckversion, diese noch einmal hinsichtlich des Corporate Design überprüft werden sollte, da für Druckinhalte eventuell andere Regeln gelten als für Bildschirmausgaben. Das Anpassungswerkzeug weist Anwender nach Beendigung eines Anpassungsprozesses auf Abhängigkeiten zu anderen Anpassungsprozessen hin. Ist ein Anpassungsprozess beendet, so prüft das Anpassungswerkzeug, ob in der zugehörigen Prozessbeschreibung Abhängigkeiten eingetragen sind. Ist dies der Fall, so gibt das Anpassungswerkzeug einen Hinweis darauf, welche Prozesse aus welchem Grund als nächstes ausgeführt werden sollten. Dadurch wird vermieden, dass Anwender versehentlich vergessen, die Folgen eines durchgeführten Anpassungsprozesses zu überdenken.

Im folgenden Abschnitt wird erläutert, wie der vom WGT erzeugte prototypische Wizard in die Repurposing Suite integriert wurde, um das hier beschriebene Anpassungswerkzeug zu realisieren, und welche Erweiterungen vorgenommen wurden, um die vorgestellten Funktionalitäten zu ermöglichen.

6.2.4 Realisierung des Anpassungswerkzeuges

Für die Erstellung des Anpassungswerkzeuges wurden mit dem PIT Prozessbeschreibungen erstellt, die mit dem WGT in einen prototypischen Wizard überführt wurden. Dieser Wizard stellt die Basis des Anpassungswerkzeuges dar. Da das Anpassungswerkzeug Bestandteil der Repurposing Suite ist, musste der Wizard sowohl in die Repurposing Suite integriert werden, als auch um automatisierte Funktionen angereichert werden.

Die Repurposing-Anwendungen bieten Anwendern die Möglichkeit, existierende Lernressourcen so zu verändern, dass sie für neue Anforderungen geeignet sind. Gestartet werden sie durch Auswahl einer Option im Kontextmenü eines im Moduleditor geöffneten Moduls. Dabei wird der Repurposing-Anwendung eine Referenz auf das zu bearbeitende Modul übergeben. Um das Anpassungswerkzeug als Bestandteil der Repurposing

Suite verfügbar zu machen, musste es an das im Moduleditor zur Verfügung gestellte Repurpsoing-Framework angebunden werden.

In den folgenden Abschnitten werden zuerst die Schritte aufgeführt, die notwendig waren, um das Anpassungswerkzeug durch Anbindung an den Moduleditor in die Repurposing Suite zu integrieren. Anschließend wird die Erweiterung des Wizards um automatisierte Funktionalitäten erläutert.

Übersichtlichkeit gewinnen:

Der erste Schritt der Integration des generierten Wizards in die Repurposing Suite war die Aufteilung der erzeugten Klassen in vier Pakete, die die Klassen zu logischen Einheiten zusammenfassen und die Übersichtlichkeit erhöhen (siehe Abbildung 61). Dies war notwendig, da während der Automatisierung noch eine Reihe weiterer Klassen zugefügt wurden.

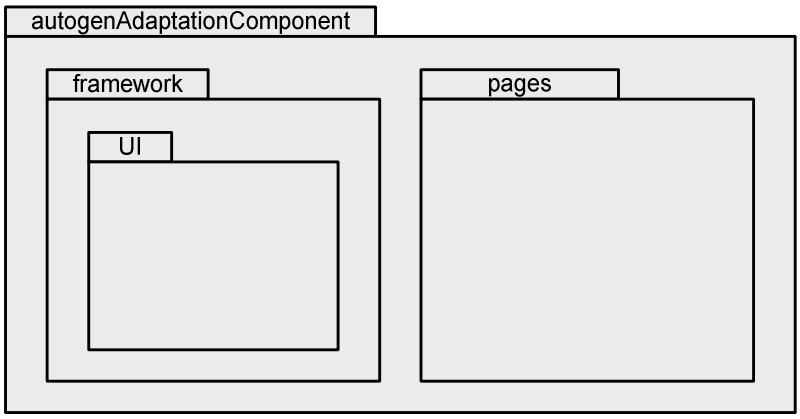


Abbildung 61: Package-Struktur für den erweiterten Anpassungs-Wizard

Anbindung des Anpassungswizards an das Repurposing-Framework

Um den automatisch erzeugten Wizard in das Repurposing-Framework einzubinden, wurde der vom WGT erzeugte Wizard-Controller um Methoden erweitert, die die MTE aufrufen, um vom Anwender gewünschte Modifikationen an die zuständigen PlugIns weiterzugeben.

Weiterhin enthielt der automatisch erzeugte Wizard nur eine recht einfache Steuerung, wann zur Folgeseite und zurück gewechselt werden darf. Um hier eine Ablaufsteuerung zu ermöglichen, die auch die Ergebnisse der automatisierten Funktionen berücksichtigt, wurden die Methoden zum Vor- und Zurückwechseln im Wizard-Controller erweitert.

Erweiterung des prototypischen Wizards

Um automatisiert Modifikationen an den Lernressourcen ausführen zu können, musste der erzeugte Prototyp erweitert werden. Dazu wurden die in Kapitel 6.1 vorgestellten Möglichkeiten genutzt. Abbildung 62 verdeutlicht das Vorgehen.

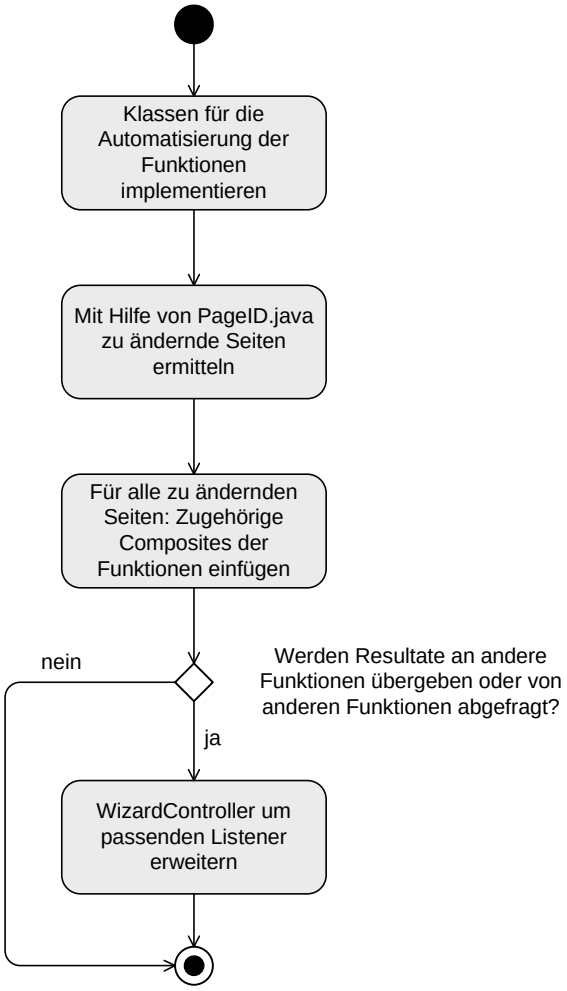


Abbildung 62: Vorgehen bei der Erweiterung des Wizards.

Zuerst wurden im Paket `framework` die Klassen erstellt, die für die Durchführung der automatisierten Funktionen zuständig sind. Das entspricht der Realisierung des bereits erwähnten Funktionspools. Hierbei wurden beispielsweise Funktionen zum Finden, Löschen und Ersetzen von Bildern oder zum Ändern von Fonts implementiert. Die Klassen greifen auf das SCR zu und lösen Aktionen der SECs aus. Beispielsweise gibt es im Funktionspool Klassen zum Sammeln von Informationen zu Hintergrundfarben und -bildern oder zu Fonts, sowie Klassen zum Auswählen bestimmter Elemente, beispielsweise Bilder.

Um die Funktionen Anwendern zur Verfügung zu stellen, müssen sie an der richtigen Stelle im Wizard angezeigt werden. Dazu muss herausgefunden werden, welche Unterschritte die Funktion enthalten und in welchem Prozessschritt sie verwendet werden.

Eine vorher erzeugte Datei (`PageID.java`) enthält Zuordnungen der Namen von Prozessschritten zu den zugehörigen IDs. Zusätzlich findet man in der Datei auch die IDs

der Unterschritte. Mit ihrer Hilfe kann vom Entwickler ermittelt werden, welche Prozessschritte die gesuchten atomaren Unterschritte enthalten.

Die Unterschritte werden durch Composites auf der Seite des zugehörigen übergeordneten Prozessschrittes angezeigt. Die Composites sind durch die ID des jeweiligen Unterschrittes gekennzeichnet. Standardmäßig sind abhängig vom Typ eines Unterschrittes vom WGT erzeugte Composites eingefügt. Zur Laufzeit wird geprüft, ob für den Unterschritt eine Automatisierung vorhanden ist. Ist dies nicht der Fall, wird das Standard-Composite angezeigt, andernfalls wird das Composite der automatisierten Funktion abgefragt und angezeigt.

Für die Anpassungsanwendung wurden die Composites für die automatisierten Funktionen nicht in den jeweiligen Funktionsklassen abgelegt. Hintergrund ist, dass je nach Verwendung der Funktion die Anzeige differiert. So ist beispielsweise die Auswahl von Bildern aus der Menge der gefundenen Bilder beim Löschen durch die gleiche Funktion realisiert, wie beim Ersetzen. Die Anzeige unterscheidet sich aber in den beiden Fällen. Daher wurde bei der Automatisierung der Anpassungsanwendung davon abgesehen, die Composites in den Klassen der entsprechenden Funktionen abzulegen. Stattdessen wurden sie direkt in die jeweilige Klasse der Seite integriert, die die Anzeige des atomaren Unterschrittes enthält. Dazu wurde die Abfrage auf Automatisierung, sowie das darin enthaltene Standard-Composite entfernt. An deren Stelle wurde das passende Composite für die Anzeige der automatisierten Funktion eingefügt.

Da Funktionen oft Resultate aus anderen Funktionen benötigen bzw. Resultate für andere Funktionen liefern, wurde der Wizard-Controller zusätzlich um Methoden erweitert, die es erlauben, verschiedene Listener zu registrieren. Dadurch kann für jede Funktion zur Laufzeit der passende Listener im Wizard-Controller registriert werden. So lässt sich sicherstellen, dass der Wizard-Controller zur Laufzeit alle Resultate der automatisierten Funktionen sammelt und diese bei Bedarf weitergibt und so die Ablaufsteuerung kontrolliert.

Im Anhang der Arbeit finden sich eine ausführliche Erläuterung der Integration und Erweiterung des Anpassungswizards sowie Listings, die an einem konkreten Beispiel einen Quelltextausschnitt einer Wizard-Seite vor und nach der Automatisierung zeigen. Außerdem enthält der Anhang eine Liste der Funktionen, die automatisiert zur Verfügung stehen.

Abbildung 63 und Abbildung 64 zeigen eine Wizard-Seite vor und nach dem Zufügen automatisierter Funktionen.

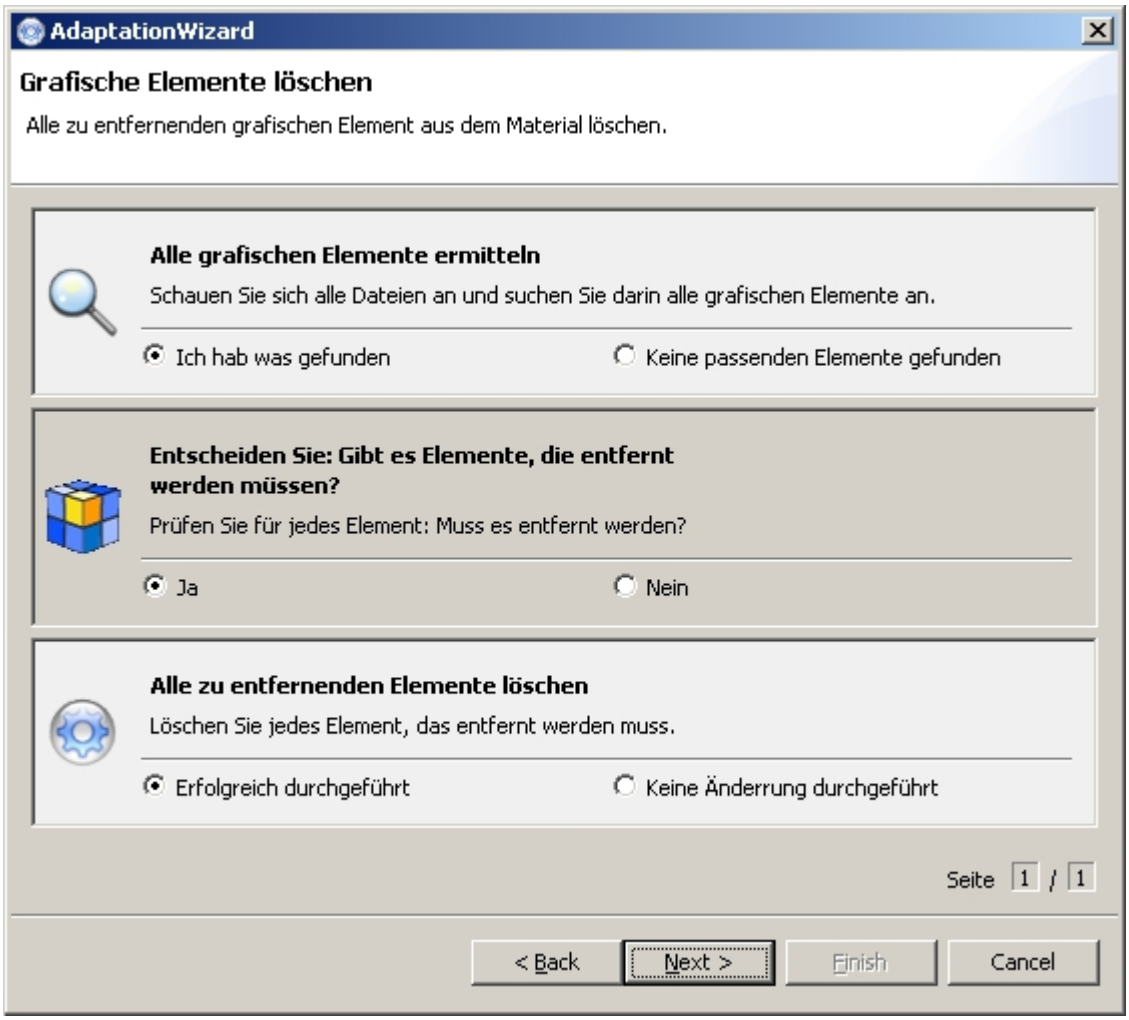


Abbildung 63: Eine Wizard-Seite vor der Automatisierung.

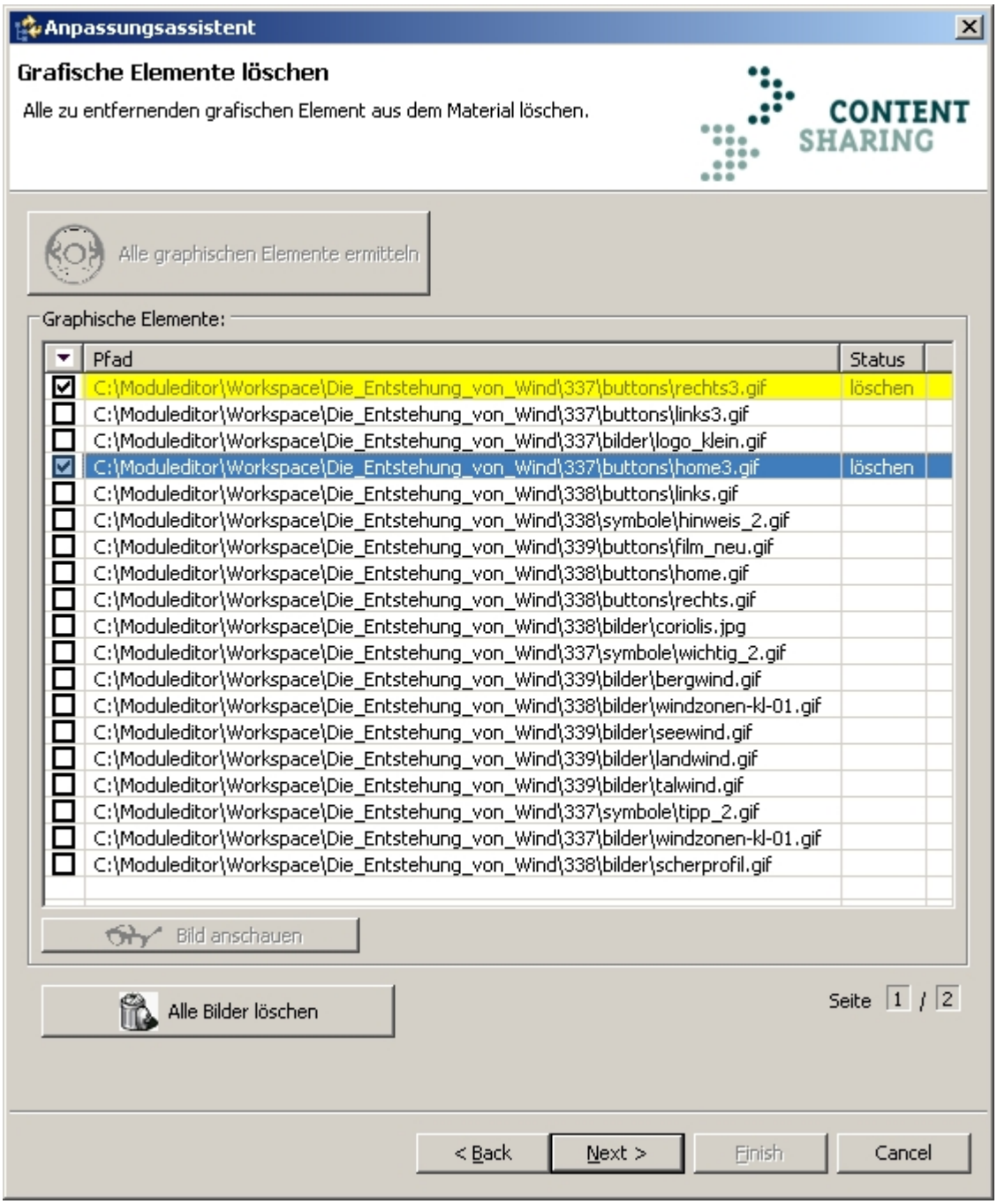


Abbildung 64: Die gleiche Wizard-Seite nach der Automatisierung.

Die Startseite eines mit WGT erzeugten Wizards enthält Angaben darüber, zu welchem Grad eine automatisierte Unterstützung der Prozesse erfolgt. Der Grad der automatischen Unterstützung eines Anpassungsprozesses ergibt sich daraus, für wie viele der im Prozess benötigten atomaren Unterschritte automatisierte Funktionen zur Verfügung stehen. Dabei ist zu berücksichtigen, dass es für einige Formate eine weiter reichende Unterstützung gibt, als für andere Formate.

Um diese zu ermöglichen, wurde eine XML-Datei erstellt, die pro Anpassungsprozess die darin enthaltenen atomaren Unterschritte enthält. Diese Datei wird als Format-Funktions-Tabelle (FFT) bezeichnet. Für jeden Unterschritt ist defaultmäßig angegeben, dass keine automatisierte Unterstützung vorliegt (`supported="no"`). Jedes Format-PlugIn enthält eine Kopie dieser Datei. Fügt ein Entwickler nun für ein Format die automatisierte Unterstützung eines atomaren Unterschrittes zu, so ändert er die Angabe in der XML-Datei des PlugIns auf `supported="yes"`. Wird die Unterstützung erweitert, z.B. aufgrund neuer technischer Möglichkeiten, so wird dies nach einer Änderung in der entsprechenden FFT bei der Anzeige des Grades der Automatisierung berücksichtigt. Listing 3 enthält einen Ausschnitt aus der FFT für HTML.

```
<functionTable>
  <function id="determine_all_used_images" supported="yes" />
  <function id="select_images_for_deletion" supported="yes" />
  <function id="delete_images_selected_for_deletion" supported="yes"
   />
   ...

  <function id="Detect_all_page_breaks" supported="no" />
  <function id="Check_page_breaks_if_reasonable" supported="no" />
  <function id="Determine_page_size_for_prinitng supported="yes" />
  <function id="Detect_iamges_too_large_for_page_size supported="yes"
   />
  <function id="Determine_new_image_size" supported="yes" />
  <function id="Resize_image" supported="yes" />
</functionTable>
```

Listing 3: Ausschnitt der HTML-FFT-Datei

Das Anpassungswerkzeug bekommt beim Start das zu bearbeitende Modul übergeben. Aus dem SCR ermittelt es die im Modul enthaltenen Formate. Für jedes Format liest es die zugehörige FFT ein. Da bekannt ist, welche atomaren Unterschritte in welchen Prozessschritten enthalten sind, kann ermittelt werden, zu welchem Grad jeder Prozessschritt automatisiert ist. Daraus lässt sich dann auch ermitteln, zu welchem Grad der gesamte Prozess unterstützt wird. Derzeit sind drei Grade berücksichtigt: Nicht automatisiert unterstützt, teilautomatisiert unterstützt und vollautomatisiert unterstützt.

6.3 Wizard-Erweiterung durch Einbindung von Webservices

Im vorigen Abschnitt wurde erläutert, wie der mit dem WGT generierte Anpassungs-Wizard in die Repurposing Suite integriert und mit automatisierten Funktionen erweitert wurde. Dabei wurden die Funktionen von einem Entwickler speziell für diesen Zweck implementiert und im Funktionspool zur Verfügung gestellt. Das Konzept zur Wizard-Erweiterung gibt aber nicht vor, wie die Funktionen zu realisieren sind. Dadurch ist es möglich, verschiedene Arten von Funktionen einzubinden. So wäre es auch denkbar, die Funktionen über Webservices zu realisieren. Daher wird an dieser Stelle ein kurzer Überblick über Webservices gegeben.

Webservices sind wiederverwendbare, abgeschlossene und plattformunabhängige Software-Komponenten, die mittels Standardprotokollen über das Internet miteinander kommunizieren und aufgerufen werden können [Ch07, Be08, 33]. Webservices führen

Funktionen durch, die von einfachen Anfragen bis hin zu komplexen Prozessen reichen können [Ch07].

Die Zusammenfassung mehrerer derartiger Webservices zu einer komplexen Anwendung wird als Webservice-Komposition bezeichnet. Hauptbeweggründe hierfür sind Wiederverwendung und Kostenreduktion. Die Komposition von Webservices umfasst die Orchestrierung und die Choreographie [Ch+07], welche zwei sich überschneidende Sichten auf die Webservice-Komposition darstellen. Bei der Orchestrierung gibt es eine zentrale Instanz, die die Interaktion zwischen der Instanz, die einen Prozess durchführt, und den an der Ausführung beteiligten Webservices kontrolliert [CM07]. Choreographie dagegen organisiert die Komposition von Webservices in einer eher kollaborativen, interaktiven Art. Bei der Choreographie gibt es keine zentrale Kontrollinstanz. Damit dies möglich ist, muss der Datenaustausch zwischen allen beteiligten Webservices gut definiert sein [CM07].

Erweitert man den vom WGT erstellten Wizard mit Webservice-Schnittstellen, so lässt sich diese Erweiterung in Form einer Webservice-Komposition realisieren. Allerdings ist zu bedenken, dass für eine Prozessunterstützung mit Webservices eventuell eine Internet- bzw. Intranetanbindung bestehen muss, je nachdem wo die Webservices realisiert sind. Weiterhin ist eine Webservice-Einbindung nur dann möglich, wenn überhaupt entsprechende Webservices verfügbar sind. Es ist unwahrscheinlich, dass für das Löschen eines einzelnen Bildes in einem bestimmten Format Webservices bereitgestellt werden, da es sich hierbei um eine sehr spezielle Aufgabe handelt. Aber es wäre denkbar, das für die Übersetzung von Textstellen irgendwann geeignete Webservices zur Verfügung stehen werden.

Möchte man nun einzelne Funktionen eines vom WGT generierten Wizards durch Webservices realisieren, so sind Klassen zu implementieren, die mit den Webservices kommunizieren und die Interaktion mit dem Anwender ermöglichen. Diese Klassen entsprechen den bereits vorgestellten Funktionsklassen und sollten ebenfalls das Interface `APAutomater` implementieren. Die Funktionsklassen können die benötigten Services aufrufen und über bereitgestellte Composites die Darstellung auf der Benutzeroberfläche realisieren. Die Einbindung der Webservices kann statisch bereits während der Entwicklung vorgenommen werden, oder dynamisch zur Laufzeit. Allerdings setzt die dynamische Einbindung voraus, dass geeignete Webservices durch eine Suche aufgefunden werden können, was derzeit aufgrund einer fehlenden einheitlichen Auszeichnung der Webservices noch Probleme bereitet [Sc+08].

Da die Einbindung von Webservices zur Wizard Erweiterung nicht Fokus dieser Arbeit ist, wird an dieser Stelle nicht weiter darauf eingegangen. Diese stellt aber einen interessanten Ausgangspunkt für zukünftige Arbeiten dar.

6.4 Zusammenfassung

In diesem Kapitel wurde der dritte Schritt des Konzeptes zur Software-Entwicklung basierend auf von Prozessexperten erstellten Pattern-basierten Beschreibungen von Anpassungsprozessen vorgestellt: Die Erweiterung des mit dem WGT erzeugten prototypischen Wizards um automatisierte Funktionen.

In Abschnitt 6.1 wurde das Vorgehen zur Wizard-Erweiterung erläutert, das einen der Beiträge dieser Dissertation darstellt. Dabei wurden Klassen und Methoden vorgestellt, die es Entwicklern erleichtern, automatisierte Funktionen zuzufügen. Anschließend wurde verdeutlicht, wie die Erweiterung des mit WGT erzeugten Wizards in der Praxis aussieht. Dazu wurde das Anpassungswerkzeug betrachtet, das zur besseren Unterstützung der in Kapitel 2 vorgestellten Anpassungsprozesse dienen soll und das in Abschnitt 2.3 vorgestellte Konzept für ein Anpassungswerkzeug realisiert. Die Konzeption und Umsetzung des Anpassungswerkzeuges sind ein weiterer Beitrag der vorliegenden Arbeit.

Das Anpassungswerkzeug wurde im Rahmen des Content Sharing Projektes entwickelt, das in Abschnitt 6.2.1 dargestellt wurde. Das Anpassungswerkzeug ist Bestandteil einer Repurposing Suite, die bei der Durchführung aller Repurposing-Aktionen unterstützt. Diese wurde in 6.2.2 vorgestellt.

Anschließend wurde das Anpassungswerkzeug selbst und dessen Funktionsweise erläutert. Die vom Anpassungswerkzeug unterstützten Prozesse wurden mit Hilfe des PIT beschrieben. Dann wurde mit dem WGT ein prototypischer Wizard zur Unterstützung der Prozesse erzeugt. Das Anpassungswerkzeug wurde dadurch erzeugt, dass dieser Wizard in die Repurposing Suite integriert und um automatische Funktionen erweitert wurde. Für die Erweiterung wurden Funktionen entwickelt, die eine teilweise automatisierte Unterstützung der Anpassungsprozesse erlauben. Die Konzeption und die Entwicklung dieser Funktionen bilden ebenfalls einen der Beiträge dieser Arbeit. Ergebnis der Integration und Erweiterung war ein einsatzfähiges Werkzeug zur Unterstützung der Anpassungsprozesse.

7 Übertragbarkeit auf andere Prozesse

7.1 Motivation

In Kapitel 2 wurde die Gruppe der Prozesse betrachtet, die zur Anpassung bereits existierender Lernressourcen für das E-Learning an veränderte Einsatzbedingungen notwendig sind. Es hat sich gezeigt, dass diese Prozesse sehr komplex sind und zu ihrer Durchführung Expertenwissen notwendig ist. Doch in der Realität werden die Anpassungsprozesse oft von Personen durchgeführt, die keine Prozessexperten sind. Damit auch diese Personen ein zufriedenstellendes Resultat bei der Durchführung der Anpassungsprozesse erzielen können, ist eine geeignete Unterstützung wünschenswert, die auf dem notwendigen Expertenwissen beruht und dieses den Laien zugänglich macht.

Die Analyse verwandter Arbeiten (vergleiche Abschnitt 2.5) sowie eine Befragung von Experten in der Durchführung von Anpassungsprozessen (Abschnitt 2.2. und Abschnitt 2.3) haben gezeigt, dass es ein derartiges Werkzeug derzeit nicht gibt. In dieser Arbeit wurde daher ein Konzept vorgestellt, das es erlaubt, ein Werkzeug zur Unterstützung von Anpassungsprozessen zu erstellen, das auf dem Wissen von Prozessexperten beruht.

Doch neben den Anpassungsprozessen gibt es noch viele weitere Prozesse, bei denen sich die gleiche Problematik beobachten lässt. Somit stellt sich die Frage, ob das in dieser Arbeit vorgestellte Konzept auch für die Erstellung von Werkzeugen zur Unterstützung anderer Prozesse verwendet werden kann.

Das Konzept zur Erstellung eines Werkzeuges zur Unterstützung von Anpassungsprozessen berücksichtigt den Aufbau und die speziellen Merkmale, die Anpassungsprozesse aufweisen. Daher wird in diesem Kapitel untersucht, ob es auch andere Prozesse gibt, die den gleichen Aufbau und die gleichen Merkmale wie Anpassungsprozesse aufweisen und ob sich das vorgestellte Konzept auch auf diese Prozesse anwenden lässt.

7.2 Merkmale betrachteter Prozesse

7.2.1 Aufbau der Prozesse

Das Werkzeug zur Unterstützung von Anpassungsprozessen spiegelt den Aufbau der Anpassungsprozesse wieder. Damit dies möglich ist, berücksichtigen das PIT und das WGT diesen Aufbau. Sollen diese Werkzeuge für andere Prozesse eingesetzt werden und soll auch der erzeugte Wizard den Aufbau anderer Prozesse geeignet abbilden, so müssen diese Prozesse den gleichen Aufbau wie Anpassungsprozesse haben.

Wie in Abschnitt 2.2.4 erläutert, setzen sich Anpassungsprozesse aus einer Reihe von Prozessschritten und atomaren Unterschritten zusammen. Dieser Aufbau gilt aber nicht nur für Anpassungsprozesse, sondern findet sich auch in der Literatur zu anderen Prozessen: Allen in Abschnitt 2.1 aufgeführten Definitionen ist gemeinsam, dass Prozesse sich aus einer Folge von Schritten zusammensetzen. Diese Schritte werden in den genannten Definitionen als Aktionen, Operationen oder Tätigkeiten bezeichnet. Die Zerlegung eines Prozesses in seine Schritte wird als Dekomposition bezeichnet und ist ein

wichtiger Bestandteil bei der Modellbildung [HKM00]. Somit lässt sich sagen, dass es auch andere Prozesse gibt, die sich aus einer Reihe von Schritten zusammensetzen.

Bei Anpassungsprozessen lassen sich zwei Arten von Schritten feststellen: Prozessschritte, die der logischen Gruppierung kleinerer Schritte dienen, und atomare Unterschritte, die sich nicht weiter unterteilen lassen. Auch diese Aufteilung findet sich in anderen Prozessen (siehe beispielsweise [Ac98]). Folgendes Beispiel verdeutlicht das:

Betrachtet man den Prozess einer Reisebuchung auf einer abstrakten Ebene, so muss man sich für ein Reiseziel entscheiden, um eine Reise zu buchen. Man kann dann eine Pauschalreise oder eine Individualreise buchen, je nach persönlichen Vorlieben. Wenn man will, kann man außerdem einen Mietwagen buchen. Somit setzt sich der Prozess „Reisebuchung“ zusammen aus „Reiseziel festlegen“, „Individualreise buchen“, „Pauschalreise buchen“ und „Mietwagen buchen“.

Der Prozess lässt sich aber auch deutlich detaillierter beschreiben. So kann man, um eine Reise zu buchen, im Internet ein Portal öffnen, dort die gewünschte Reisezeit eingeben und das gewünschte Reiseziel. Dann lässt man sich eine Auswahl möglicher Transportmittel, Unterkünfte etc. anzeigen. Alternativ könnte man sich entscheiden, im Reisebüro buchen zu wollen und sich überlegen, bei welchem Reisebüro man buchen will, sich dann zu diesem Reisebüro begeben etc.

Die erste, grobe Beschreibung nennt die Prozessschritte des Prozesses zur Reisebuchung, die zweite, detaillierte Beschreibung nennt atomare Unterschritte. Es gibt also eine Gruppe von Prozessen, die sich in die bereits definierten zwei Arten von Schritten zerlegen lassen [Ac98]:

1. Die Schritte, die der logischen Gruppierung mehrerer kleinerer Schritte dienen, und die in dieser Arbeit als *Prozessschritte* bezeichnet wurden.
2. Die Schritte, die sich nicht weiter unterteilen lassen, und die in dieser Arbeit als *atomare Unterschritte* bezeichnet wurden.

Damit ergibt sich für diese Prozesse ebenfalls ein hierarchischer Aufbau (vergleiche Abbildung 65):

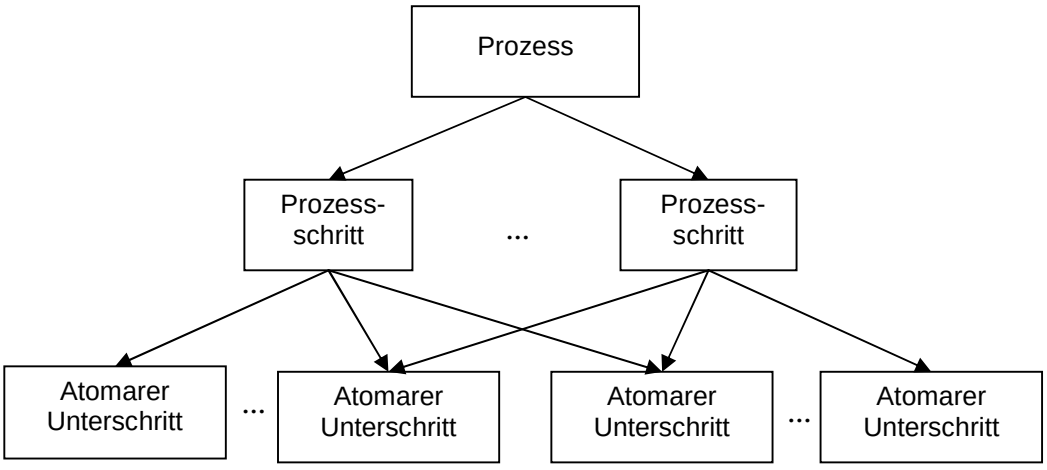


Abbildung 65: Hierarchischer Aufbau von Prozessen.

In Abschnitt 2.2.4 wurde erläutert, dass sich bei Anpassungsprozessen drei Arten von atomaren Unterschritten unterscheiden lassen: *Ermittlungen*, *Entscheidungen* und *Ausführungen*. Die Festlegung der drei Arten von atomaren Unterschritten beruht auf einer Aufteilung von Tätigkeiten in zwei Gruppen, die in der Betriebswirtschaftslehre allgemein für Tätigkeiten vorgenommen wird: Entscheidungen und Ausführungen [He92]. Zusätzlich zu den zwei Gruppen von Tätigkeiten wurde noch eine weitere Gruppe eingeführt: Ermittlungen, die für Entscheidungen und Ausführungen notwendige Informationen bereitstellen. Da diese Aufteilung für verschiedenste Tätigkeiten gilt, lassen sich auch bei anderen Prozessen diese drei Arten von atomaren Unterschritten unterscheiden. Das oben genannte Beispiel zur Reisebuchung verdeutlicht das. Hier finden sich unter anderem folgende Unterschritte:

- Ermittlungen: Welchen Kriterien muss ein potentielles Reiseziel genügen? Welche Ziele entsprechen diesen Kriterien?
- Entscheidungen: Welches der infrage kommenden Ziele soll gewählt werden?
- Ausführungen: Buchen einer Pauschalreise.

Somit lässt sich sagen, dass der in Kapitel 2 vorgestellte Aufbau von Anpassungsprozessen sich auch bei einer Reihe anderer Prozesse wiederfinden lässt.

7.2.2 Weitere Merkmale betrachteter Prozesse

In Abschnitt 2.2.5 wurden neben dem hierarchischen Aufbau weitere Merkmale der Anpassungsprozesse vorgestellt. Das in dieser Arbeit vorgestellte Konzept baut auf diesen Merkmalen auf. Sollen andere Prozesse von diesem Konzept unterstützt werden, müssen sie daher ebenfalls diese Merkmale aufweisen:

1. Oft ist nur eine Person für die Durchführung der Prozesse zuständig. Das bedeutet, dass eine Person die Prozesse durchführen kann. Es kann aber auch vorkommen, dass mehrere Personen an der Ausführung beteiligt sind. Aber es gibt immer eine Person, die die Hauptausführende ist und / oder die Aufgaben der anderen koordiniert.
2. Um bei der Durchführung der Prozesse ein optimales Resultat zu erzielen, ist Spezialwissen nötig. In der Realität kommt es aber immer wieder vor, dass auch Laien die Prozesse durchführen müssen. Diese können ebenfalls zufrieden stellende Resultate erzielen, wenn sie geeignet unterstützt werden.
3. Die Struktur der Prozesse ist strikt definiert und lässt sich bereits vor der Ausführung des Prozesses vollständig beschreiben. Es ist also möglich, anzugeben, aus welchen Prozessschritten sich ein Prozess zusammensetzt, in welcher Reihenfolge diese durchlaufen werden und welchen Bedingungen die Ausführung unterliegt.
4. Experten können beschreiben, wie sie die Prozesse durchführen, so dass auch Laien anhand der Beschreibung die Prozesse durchführen können.
5. Die Prozesse werden regelmäßig durchgeführt. (Andernfalls wäre der Aufwand für die Entwicklung eines Unterstützungswerkzeugs zu hoch.)

Das dritte Merkmal stellt sicher, dass ein Prozess sich aus Prozessschritten und atomaren Unterschritten zusammensetzt und somit der geforderte Aufbau gegeben ist.

Insbesondere in kleineren Unternehmen, die sich nicht für jede Aufgabe einen Spezialisten leisten können [Br07], findet man immer wieder Prozesse, die die hier vorgestellten Merkmale aufweisen. Dazu zählen beispielsweise die Sicherheit von IT-Systemen oder die Pflege von Webseiten sowie die bereits erwähnten Prozesse zur Anpassung existierender Lernressourcen an veränderte Einsatzszenarien. In folgendem Abschnitt werden beispielhaft einige Prozesse vorgestellt, die die eben genannten Merkmale aufweisen.

7.3 Beispielprozesse

Neben dem bereits ausführlich erläuterten Beispiel der Anpassungsprozesse gibt es eine Reihe weiterer Prozesse, die die eben genannten Merkmale aufweisen. So gehören die Prozesse, die nötig sind, um eine sichere IT-Landschaft zu erreichen, in die Gruppe der vorher charakterisierten Prozesse:

1. Gerade in kleineren Unternehmen kommt es oft vor, dass nur eine Person alle notwendigen Sicherheitsprozesse durchführen muss.
2. Um die Prozesse wirklich gut durchzuführen, sind Expertenkenntnisse notwendig. Doch in der Realität führen oft Laien die Prozesse durch [Sc03].
3. Es lassen sich alle Prozessschritte und atomaren Unterschritte, die notwendig sind, um einen Sicherheitsprozess umzusetzen, deren Ausführungsreihenfolge, sowie deren Abhängigkeiten untereinander benennen. [Sc^{+}06]
4. In [Sc^{+}06] finden sich detaillierte Beschreibungen der Prozesse anhand derer auch Laien die Prozesse durchführen können.
5. Diejenigen, die Angriffe auf IT-Systeme planen und durchführen, entwickeln stetig neue Möglichkeiten, um die Sicherheitsmechanismen zu umgehen [Ja02]. Deswegen ist es ständig notwendig, zu prüfen, ob die Systemlandschaft auch weiterhin sicher ist. Außerdem müssen bei jedem neuen System erneut die Prozesse durchgeführt werden, die das System sicher werden lassen. Somit müssen Prozesse zum Erreichen der IT-Sicherheit regelmäßig ausgeführt werden.

Eine weitere Gruppe von Prozessen, die die genannten Merkmale aufweist, sind die Prozesse zur Erstellung und Wartung von HTML-Seiten. Alle oben aufgeführten Merkmale werden von diesen Prozessen erfüllt:

1. Wie im Falle der IT-Sicherheit ist es insbesondere in kleineren Unternehmen oft so, dass nur eine Person die Erstellung und Wartung der Webseiten des gesamten Internetauftrittes durchführen muss.
2. Diese Person bräuchte Expertenkenntnisse für ein perfektes Resultat. Aber oft hat die Person die notwendigen Kenntnisse nicht.
3. Man kann für die Prozesse zur Erstellung und Pflege von Webseiten die zur Durchführung notwendigen Prozessschritte und atomaren Unterschritte sowie deren Reihenfolge benennen. Weiterhin lassen sich Abhängigkeiten zwischen Prozessschritten und atomaren Unterschritten festlegen [Gr03].

4. Wie in [Gr03] deutlich wird, lässt sich beschreiben, wie Experten die Prozesse durchführen. Mit diesen Beschreibungen können auch Laien die Prozesse durchführen.
5. Die meisten Unternehmen pflegen regelmäßig Änderungen in ihren Internetauftritt ein. Außerdem kommt es auch immer wieder vor, dass neue Seiten zu einem bestehenden Auftritt zugefügt werden. Somit werden die Prozesse regelmäßig durchgeführt.

Auch der bereits genannte Prozess der Reisebuchung weist die in Abschnitt 7.2 vorgestellten Merkmale auf:

1. Üblicherweise bucht eine Person eine Urlaubsreise. Dabei kann es sein, dass derjenige, der die Reise antreten will, einen Angestellten eines Reisebüros die Buchung vornehmen lässt. Es kann aber auch vorkommen, dass er die Reise selbst z.B. über das Internet bucht. In seltenen Fällen lässt die Person, die verreisen möchte, die Buchung durch mehrere Personen ausführen, z.B., um Mietwagen, Hotel und Beförderungsmittel für Hin- und Rückreise separat zu buchen. Doch in diesem Fall ist dennoch eine Person Hauptausführende: Die Person, die verreisen will, trifft die Entscheidungen, was gebucht werden soll und veranlasst die anderen Personen, die Buchungen vorzunehmen.
2. Es ist nicht unbedingt nötig, Spezialwissen zu haben, um eine Reise zu buchen. Doch es ist hilfreich, wenn man bei der Buchung unterstützt und so verhindert wird, dass man z.B. zu geringe Umsteigezeiten bei Zugfahrten einplant, oder den Flughafentransfer bei einer Flugreise nicht berücksichtigt.
3. Es lassen sich alle Prozessschritte und atomaren Unterschritte angeben, aus denen sich der Prozess der Reisebuchung zusammensetzt. Außerdem kann man die Reihenfolge aller Schritte und deren Abhängigkeiten untereinander beschreiben.
4. Der Prozess lässt sich detailliert beschreiben, so dass auch Laien anhand der Beschreibung in der Lage sind, den Prozess durchzuführen.
5. Eine Vielzahl von Personen bucht im Laufe ihres Lebens mehrfach eine Reise, sei es um in Urlaub zu fahren, oder aus geschäftlichen Gründen. Somit wird der Prozess immer wieder ausgeführt.

Als abschließendes Prozessbeispiel sei die Einstellung neuer Mitarbeiter in Firmen oder in einem Institut an einer Hochschule genannt. Diese Prozesse laufen in verschiedenen Firmen oder Instituten geringfügig unterschiedlich ab. Doch alle diese Prozesse weisen die oben vorgestellten Merkmale auf:

1. Bei der Einstellung neuer Mitarbeiter ist oft eine Reihe von Personen beteiligt, aber üblicherweise gibt es eine Person, die veranlasst, dass die anderen Personen ihre Aufgabenteile ausführen.
2. Oft gibt es Details, die nur jemand weiß, der den Prozess oft durchführt. Ist derjenige in Urlaub oder aus anderen Gründen verhindert, kommt es aber immer wieder vor, dass auch jemand, der kein Experte ist, den Prozess durchführen muss.

3. Man kann für die Einstellung neuer Mitarbeiter die Prozessschritte und atomaren Unterschritte benennen, die zur Durchführung notwendig sind. Außerdem lassen sich die Reihenfolge der Schritte und Abhängigkeiten der Schritte untereinander feststellen.
4. Es gibt eine Reihe von Tätigkeiten, die bei jeder Einstellung eines neuen Mitarbeiters durchgeführt werden müssen. Diese können so beschrieben werden, dass auch ein Laie sie veranlassen und falls nötig durchführen kann.
5. Üblicherweise werden regelmäßig neue Mitarbeiter eingestellt. Somit werden die Prozesse mehrfach durchgeführt.

Diese Aufzählung beispielhafter Prozesse macht deutlich, dass es eine Reihe von Prozessen gibt, die - so unterschiedlich sie auch sind - alle die gleichen Merkmale wie die in dieser Arbeit betrachteten Anpassungsprozesse zeigen. Nachfolgend wird daher exemplarisch untersucht, ob sich der in dieser Arbeit vorgestellte Ansatz auch auf andere Prozesse anwenden lässt, die die genannten Merkmale aufweisen.

7.4 Untersuchung der Übertragbarkeit auf andere Prozesse

In den Kapiteln 4 und 6 wurde gezeigt, wie der Ansatz dieser Dissertation für die Erstellung eines Werkzeuges zur Unterstützung von Anpassungsprozessen verwendet werden kann. In diesem Kapitel wurde eine Reihe weiterer Prozesse vorgestellt, die die eben benannten Merkmale aufweisen. An ihnen wurde gezeigt, dass sich für diese Prozesse mit dem in der vorliegenden Arbeit vorgestellten Konzept Prozessbeschreibungen und Wizards zur Prozessunterstützung erstellen lassen.

Insgesamt werden drei der eben vorgestellten Arten von Prozessen betrachtet: Prozesse aus dem Bereich der IT-Sicherheit, ein Prozess, der bei der Einstellung neuer Mitarbeiter durchgeführt wird, und ein Prozess zur Buchung einer Pauschalreise.

7.4.1 Prozesse aus der IT Sicherheit

In [Sc$^+$06] stellen Schumacher et al. Patterns für Sicherheitsaspekte in den verschiedenen Ebenen eines Unternehmens vor. Dabei betrachten sie beispielsweise das gesamte Unternehmen, die eingesetzten IT-Systeme, aber auch den einzelnen Mitarbeiter.

Wie bereits in Abschnitt 7.3 dargelegt, zeigen die Prozesse aus der IT Sicherheit die gleichen Merkmale wie Anpassungsprozesse. Im Rahmen der Untersuchung der Übertragbarkeit auf andere Prozesse wurde daher an einigen Prozessen aus diesem Bereich getestet, ob sie sich in Pattern-basierter Form beschreiben lassen und ob basierend auf den Prozessbeschreibungen ein prototypischer Wizard zur Prozessunterstützung generiert werden kann.

In [Sc$^+$06] werden die Probleme, die im Bereich der IT Sicherheit auftreten, und deren Lösungen bereits in Pattern-Form beschrieben. Viele dieser Patterns beschreiben Prozesse, die sich durch Wizards unterstützen lassen. Doch das in [Sc$^+$06] vorgestellte Pattern-Format stimmt nicht vollständig mit dem in dieser Arbeit verwendeten Format überein. Daher wurde im Rahmen dieser Dissertation für das Pattern-Format aus [Sc$^+$06] eine Abbildung auf das in dieser Arbeit verwendete Format entwickelt.

Außerdem wurden exemplarisch einige der Patterns aus dem Bereich der Zugangskontrolle [Sc+06] in das Format dieser Arbeit überführt.

Die Patterns von Schumacher et al. enthalten alle Elemente, die im Rahmen dieser Arbeit benötigt werden, um Prozesse zu beschreiben. Insbesondere enthalten sie einen Implementierungsabschnitt, der erläutert, wie die jeweilige Lösung umgesetzt werden kann. In diesem Abschnitt werden detailliert die Schritte beschrieben, die zur Ausführung der Lösung notwendig sind. In den Erläuterungen zu den einzelnen Schritten der Implementierung finden sich auch die benötigten atomaren Unterschritte. Es ergibt sich folgende Abbildung der Elemente der Security-Patterns auf die Elemente, der in dieser Arbeit verwendeten Patterns zur Prozessbeschreibung (vergleiche Tabelle 5):

Element in den Prozessbeschreibungs-Patterns	**Zugehöriges Element in den Security Patterns**
Name	Name
Klassifikation	Diese kommt in [Sc+06] nicht vor. Da die Klassifikation kein Pflichtelement ist, wurde keine nachträgliche Klassifikation der Patterns vorgenommen.
Absicht	Zu Beginn jedes Patterns gibt es eine kurze Zusammenfassung, die der Absicht entspricht.
Kontext	Context
Problem	Problem
Beispiel	Example
Forces	Sind bei [Sc+06] Bestandteil des Problems und wurden für die Transformation in die Prozessbeschreibungs-Patterns aus dem Problem herausgelöst
Lösung	Solution
Bekannte Verwendungen	Known Uses
Konsequenzen	Consequences
Verwandte Patterns	Sind in der „See Also“ Sektion enthalten und müssen dort herausgelöst werden
Verbundene Patterns	Sind in der „See Also“ Sektion enthalten und müssen dort herausgelöst werden
Verwendete Patterns	Sind in der „See Also“ Sektion enthalten und müssen dort herausgelöst werden
Prozessschritte und atomare Unterschritte	Sind im Implementation-Abschnitt enthalten und müssen dort herausgelöst werden

Tabelle 5: Abbildung der Pattern Elemente.

Somit lassen sich die Security-Patterns aus [Sc+06] in die Prozessbeschreibungspatterns dieser Arbeit transformieren. Aus den so entstandenen Prozessbeschreibungen zur Zugangskontrolle wurde mit dem WGT ein prototypischer Wizard erzeugt, der Nutzer durch Prozesse der Zugangskontrolle führt.

7.4.2 Prozess zur Einstellung neuer Mitarbeiter

Neben Prozessen aus der IT-Sicherheit wurden in Abschnitt 7.3 auch die Prozesse zur Einstellung neuer Mitarbeiter in Firmen oder Hochschulinstituten beschrieben. In Abschnitt 7.3 wurde bereits erläutert, dass diese Prozesse die in 7.2 genannten Merkmale aufweisen.

Doch die Einstellung neuer Mitarbeiter läuft in allen Firmen oder Instituten etwas anders ab. Deswegen wurde im Rahmen der Evaluation exemplarisch der Prozess der Einstellung neuer Mitarbeiter an einem Institut beschrieben. Dazu wurde der Prozess mit Hilfe des PIT in Pattern-basierter Form beschrieben. Aus dieser Beschreibung wurde dann mit Hilfe des WGT ein prototypischer Wizard erzeugt. Dabei hat sich gezeigt, dass das vorgeschlagene Vorgehen gut für den Prozess geeignet ist. Der entstandene Wizard bildet alle für die Einstellung eines neuen Mitarbeiters benötigten Schritte ab und führt somit auch Personen, die keine Experten in der Prozessausführung sind, wie beispielsweise eine Urlaubsvertretung, durch den Prozess und unterstützt bei der Durchführung der notwendigen Schritte.

7.4.3 Prozess zur Reisebuchung

In Abschnitt 7.3 wurde bereits gezeigt, dass der Prozess der Reisebuchung die geforderten Merkmale aufweist. Zur Evaluation der Werkzeuge PIT und WGT wurde von Testpersonen im Rahmen eines Benutzertests der Prozess einer Reisebuchung beschrieben. Dieser Test wird in Abschnitt 8.1 detailliert erläutert.

Im Benutzertest wurde von den Testpersonen aus Zeitgründen nur eine stark verkürzte Prozessbeschreibung erstellt. Aber im Vorfeld des Tests wurde auch eine ausführliche Prozessbeschreibung angelegt. Im Anhang findet sich die für die Reisebuchung mit Hilfe vom PIT erzeugte Prozessbeschreibung. Außerdem werden dort einige Screenshots des mit dem WGT generierten prototypischen Wizards gezeigt. Dieser Wizard erläutert Schritt für Schritt, wie man bei der Buchung einer Reise vorzugehen hat. Es ist möglich, einzelne Funktionen zu automatisieren, oder auf existierende Werkzeuge zur Reisebuchung z.B. über Links hinzuweisen.

7.5 Zusammenfassung

Es gibt eine Reihe von Prozessen, für deren Ausführung Expertenwissen notwendig ist. Aber in der Realität kommt es oft vor, dass Laien die Prozesse durchführen müssen. Doch mit Hilfe eines geeigneten Werkzeuges, das Laien Expertenwissen zur Verfügung stellt, können auch die Laien diese Prozesse zufriedenstellend durchführen. Im Rahmen dieser Arbeit wurde ein Konzept zur Entwicklung eines derartigen Werkzeuges für den Spezialfall der Anpassungsprozesse vorgestellt.

In diesem Kapitel wurde gezeigt, dass die in Anpassungsprozessen ermittelten Merkmale auch bei anderen Prozessen zu finden sind. Dazu wurde zuerst gezeigt, dass es auch andere Prozesse gibt, die den gleichen Aufbau wie Anpassungsprozesse haben. Dann wurde erläutert, dass diese Prozesse ebenfalls die weiteren Merkmale der Anpassungsprozesse aufweisen. Anschließend wurden einige Beispielprozesse vorgestellt, die die vorher benannten Merkmale und den erläuterten Aufbau aufweisen.

Weiterhin wurde für einen Teil der Beispielprozesse untersucht, ob sich für sie mit dem in dieser Arbeit vorgestellten Ansatz Werkzeuge zur Prozessunterstützung erzeugen lassen. Dazu wurden für die Prozesse mit dem PIT Prozessbeschreibungen erstellt, die die Durchführung der jeweiligen Prozesse erläutern. Basierend auf den Prozessbeschreibungen wurden mit dem WGT prototypische Wizards generiert. Diese Wizards bilden die Prozessstruktur ab und erläutern anderen Personen schrittweise die unterstützten Prozesse.

Somit lässt sich sagen, dass der in dieser Dissertation vorgestellte Ansatz zur Patternbasierten Prozessbeschreibung und automatischen Wizard-Generierung, sowie die dafür entwickelten Werkzeuge PIT und WGT auch für die untersuchten Prozesse anwendbar sind.

Für eine generelle Erweiterung dieser Arbeit auf andere Prozesse wäre zu untersuchen, ob die hier vorgestellten Merkmale vollständig sind, oder ob es Prozesse gibt, die zwar alle Merkmale aufweisen, aber dennoch nicht für den vorgestellten Ansatz geeignet sind. Da der Schwerpunkt dieser Arbeit aber in der Entwicklung eines Werkzeuges zur Unterstützung von Anpassungsprozessen lag, wird die Übertragbarkeit auf andere Prozesse an dieser Stelle nicht detaillierter betrachtet.

8 Evaluation

In der vorliegenden Arbeit wurde zur Umsetzung des ersten Hauptziels der Arbeit ein Konzept zur Pattern-basierten Beschreibung für Anpassungsprozesse vorgestellt. Dieses Konzept und die zu dessen Umsetzung entwickelten Werkzeuge erlauben es Prozessexperten unabhängig von ihrem Vorwissen im Hinblick auf Prozessmodellierung und Softwareerstellung, Anpassungsprozesse so zu beschreiben, dass sich aus den Beschreibungen prototypische Wizards erzeugen lassen. Die erzeugten Wizards dienen als Basis für die Entwicklung eines Werkzeuges zur Unterstützung von Laien und Experten bei der Durchführung der Anpassungsprozesse. Dieses Werkzeug realisiert das zweite Hauptziel der Arbeit.

Um nachzuweisen, dass das Konzept Prozessexperten unabhängig von ihren Vorkenntnissen ermöglicht, Prozessbeschreibungen und darauf basierende prototypische Wizards zu erstellen, wurde im Rahmen der vorliegenden Arbeit eine Evaluation durchgeführt. Diese untersuchte weiterhin, ob das mit dem vorgestellten Konzept erstellte Werkzeug (vergleiche Abschnitt 6.2) zur Unterstützung von Anpassungsprozessen auch tatsächlich geeignet bei der Prozessdurchführung unterstützt.

Daher untersuchte die Evaluation folgende zwei Annahmen:

1. Das Konzept und die dafür entwickelten Werkzeuge PIT (siehe Kapitel 4) und WGT (siehe Kapitel 5) eignen sich zur Prozessbeschreibung durch Prozessexperten, unabhängig von deren Vorkenntnissen in Software-Design und -Entwicklung.
2. Der mit dem vorgestellten Konzept entwickelte Wizard zur Unterstützung von Anpassungsprozessen verbessert die Durchführung der unterstützten Prozesse.

Dementsprechend teilte sich die Evaluation in zwei Tests auf, die in den beiden folgenden Abschnitten vorgestellt werden.

8.1 Evaluation der Werkzeuge PIT und WGT

Hauptgrund für das in dieser Arbeit vorgestellte Konzept war die Beobachtung, dass Software zur Unterstützung von Anpassungsprozessen oft nicht das Prozessverständnis von Prozessexperten widerspiegelt und deswegen keine geeignete Führung durch die Prozesse anbietet. Grund hierfür ist oft, dass die Prozessexperten nur unzureichend in den Prozess der Software-Entwicklung eingebunden werden. Das führt dazu, dass ihr Wissen oftmals nicht vollständig in die Entwicklung und in spätere Werkzeuge zur Prozessunterstützung einfließt. Daher sollten die Prozessexperten, auch wenn sie keine Kenntnisse in Prozessmodellierung haben, in die Erstellung der Software eingebunden werden. In diesem Abschnitt wird untersucht, ob das zu diesem Zweck entworfene Konzept und die dafür entwickelten Werkzeuge PIT und WGT geeignet sind, dieser Anforderung nachzukommen.

Die Werkzeuge PIT und WGT erlauben sowohl Personen mit als auch Personen ohne speziellem Vorwissen in Software-Design und -Entwicklung, ihre Kenntnisse über Anpassungsprozesse so zur Verfügung zu stellen, dass daraus ein Prototyp erzeugt werden

kann. Dieser Prototyp ermöglicht es dem Prozessexperten, zu prüfen, ob der von ihm beschriebene Prozessablauf geeignet abgebildet wird. Er dient als Basis für die weitere Entwicklung und stellt eine wertvolle Kommunikationsgrundlage für alle an einem Entwicklungsprojekt beteiligten Personen dar. Außerdem enthält der Prototyp das Wissen der Prozessexperten und kann somit bereits direkt nach seiner Erstellung genutzt werden, um andere Personen durch die abgebildeten Anpassungsprozesse zu führen.

Besonders wichtig beim vorgestellten Konzept ist, dass ein Prozessexperte die Anpassungsprozesse beschreibt, für die ein Unterstützungswerkzeug entwickelt werden soll. Die im Rahmen dieser Arbeit verwendeten Anpassungspatterns wurden in Zusammenarbeit mit Prozessexperten erstellt und dann in das PIT eingegeben. Aus den entstandenen Prozessbeschreibungen wurde mit dem WGT der prototypische Wizard für das Anpassungswerkzeug erstellt. Im Rahmen der Evaluation soll gezeigt werden, dass Prozessexperten auch ohne Unterstützung in der Lage sind, das PIT und das WGT zu verwenden, um Prozesse zu beschreiben und prototypische Wizards zur Prozessunterstützung zu erzeugen. Für dieses Vorgehen soll es keinen Unterschied machen, welche Vorkenntnisse im Software Engineering der Prozessexperte besitzt.

Um dies nachzuweisen, wurde ein Benutzertest durchgeführt, der im Rahmen einer Studienarbeit [Bu08] am Institut für Psychologie der Technischen Universität Darmstadt mitbetreut wurde. Annahme des Tests war, dass es eine Gruppe von Prozessexperten gibt, die Grundkenntnisse im Bereich der Informatik haben, die bei der Erstellung der Prozessbeschreibungen und der Generierung des Wizards hilfreich sind. Diese Gruppe wird als Gruppe der ITler bezeichnet, da ihre Mitglieder meist aus der Informatik oder verwandten Bereichen stammen. Weiterhin gibt es eine zweite Gruppe von Prozessexperten, die keine entsprechenden Vorkenntnisse, insbesondere keine Prozessmodellierungsvorkenntnisse besitzt. Sie wird als die Gruppe der Nicht-ITler bezeichnet.

Der durchgeführte Benutzertest sollte die folgenden Hypothesen beweisen:

H1: Mit dem PIT und dem WGT sind **nur geringfügig weniger** Nicht-ITler als ITler in der Lage, aus einer Prozessbeschreibung einen Wizard zu generieren.

H2: Nicht-ITler machen **nur geringfügig mehr** Fehler bei der Anwendung des Werkzeuges PIT als ITler.

Die genaue Zusammensetzung der untersuchten Testgruppen sowie der Aufbau des Tests werden im folgenden Abschnitt erläutert.

8.1.1 Testaufbau

Die Bestimmung der optimalen Stichprobengröße ergab eine Anzahl von 35 Versuchspersonen [Bo05, Seite 258 und Seite 303]. Der Test baut auf einem mehrfaktoriellen 2x2x2 Versuchsplan auf. Dieser enthält drei **unabhängige Variablen (UV)** in Form von Organismusvariablen:

- Vorkenntnisse in Software-Design und -Entwicklung, aufgeteilt in die Gruppen „ITler“ (z.B. Informatiker und Elektro- und Informationstechniker) und „Nicht-ITler“ (z.B. Pädagogen, Soziologen, Philosophen, Psychologen)
- Alter, aufgeteilt in die Gruppen „jung“ (20-26 Jahre) und „alt“ (27-61 Jahre)

- Geschlecht, aufgeteilt in die Gruppen „männlich“ und „weiblich“

Daraus ergaben sich 6 Bedingungen: ITler versus Nicht-ITler, männlich versus weiblich, alt versus jung. Dies führte zu insgesamt acht Testgruppen zu je vier Personen (siehe Tabelle 6).

	IT-Kenntnisse				Keine IT-Kenntnisse			
	Männlich		Weiblich		Männlich		Weiblich	
	Alt	Jung	Alt	Jung	Alt	Jung	Alt	Jung
Anzahl Personen	4	4	4	4	4	4	4	4

Tabelle 6: Versuchsplan.

Das zentrale Grenzwerttheorem [Bo05, Seite 93 f.] besagt, dass die Mittelwerte einer Verteilung bei einer Stichprobengröße von mindestens 30 annähernd normalverteilt sind. Daher wurde eine tatsächliche Stichprobengröße von 32 gewählt, die sich aus acht Testgruppen zu je vier Teilnehmern zusammensetzte.

Für die **abhängigen Variablen (AV)** des Tests wurden Funktionalität und Benutzerfreundlichkeit der Werkzeuge PIT und WGT betrachtet. Gemessen wurden die AVs zum einen anhand der bei den Tests entstandenen Prozessbeschreibungen. Hier wurden die gemachten Fehler gezählt und die für die Erstellung benötigte Zeit gemessen. Zum anderen wurde in einem Online-Fragebogen die Einschätzung der Testpersonen zu Benutzerfreundlichkeit und Funktionalität abgefragt.

Der Test wurde an einem Notebook durchgeführt, auf dem die Programme PIT und WGT installiert waren. Alle Testteilnehmer erhielten die gleiche Aufgabe: Sie sollten alle den gleichen Prozess mit dem PIT beschreiben und mit dem WGT für diesen Prozess einen Wizard erzeugen. Als Prozess wurde der Prozess der Reisebuchung, der in Kapitel 7 vorgestellt wurde, gewählt. Grund hierfür war, dass keine 32 Testpersonen gefunden werden konnten, die alle Prozessexperten im gleichen Anpassungsprozess sind und den oben genannten Kriterien (Vorkenntnisse, Geschlecht und Alter) entsprachen. Deswegen musste ein Prozess gewählt werden, bei dem sich eine geeignete Gruppe von Testpersonen finden ließ.

Ein Prozess, den viele Personen bereits durchgeführt haben, ist die Buchung einer Reise. Daher wurde untersucht, ob der Prozess der Reisebuchung den gleichen Aufbau wie Anpassungsprozesse hat und die gleichen Merkmale aufweist (vergleiche Kapitel 7). Da dies zutraf, wurde zudem mit dem PIT eine ausführliche Prozessbeschreibung für den Prozess der Reisbuchung erstellt. Basierend auf dieser Beschreibung wurde außerdem mit dem WGT ein prototypischer Wizard erstellt. (Die Prozessbeschreibung und einige Screenshots des Wizards finden sich im Anhang.) Der Prozess der Reisebuchung ist also ein Prozess, der den untersuchten Testpersonen bekannt ist und der von dem in dieser Arbeit vorgestellten Konzept unterstützt wird.

Den Teilnehmern wurde eine Anleitung zur Verfügung gestellt, die die Verwendung von PIT erläutert. Die Anleitung war wie folgt aufgebaut:

Zuerst wurde der hierarchische Aufbau von Prozessen, wie er bei Anpassungsprozessen und auch beim Reisebuchungsprozess zu finden ist, erläutert. Danach wurde erklärt, wie sich mit dem PIT die Beschreibung für einen Prozess anlegen lässt. Anschließend wurde das Vorgehen für Prozessschritte erläutert, dann das für atomare Unterschritte. Abschließend wurde erklärt, wie ein Anwender eine erstellte Prozessbeschreibung speichern kann. Die Anleitung verwendet Beschreibungen und Screenshots zur Verdeutlichung der Bedienung vom PIT. Zusätzlich wird die Bedienung anhand eines stark verkürzten Beispielprozesses erläutert, der beschreibt, wie man beim Prozess des Kuchenbackens vorzugehen hat.

Sowohl für den Prozess, den die Testpersonen mit dem PIT beschreiben sollten, als auch für den Beispielprozess wurden Prozesse gewählt, die die Testpersonen im täglichen Leben bereits mehrfach durchgeführt haben, so dass man davon ausgehen kann, dass sie die nötigen Kenntnisse haben, um die Prozesse zu beschreiben.

Zusätzlich zur Bedienungsanleitung für das PIT erhielten die Testpersonen eine Erläuterung des Testablaufs, in der auch die Bedienung vom WGT erläutert war. Im WGT musste die Testperson lediglich den Button zum Start der Wizard-Generierung drücken. Die Eingabe des Zielordners war bereits vorgegeben, um sicherstellen zu können, dass die Testpersonen die Wizards im korrekten Ordner erstellen. Die Pfade zu den PIT-Dateien wurden durch das PIT vorgegeben.

Außerdem erhielten die Testpersonen folgende, verkürzte Beschreibung einer Reisebuchung, die sie als Prozess mit dem PIT beschreiben sollten:

„Sie wollen eine Pauschalreise buchen. Dabei gehen Sie folgendermaßen vor: Als erstes müssen Sie Ihr Reiseziel aussuchen. Dazu müssen Sie mögliche Reiseziele ermitteln, beispielsweise über Kataloge aus dem Reisebüro oder per Internet. Haben Sie ein Reiseziel gefunden? Falls ja, fahren Sie als zweites damit fort, eine Pauschalreise zu buchen. Dazu müssen Sie mögliche Pauschalreisen ermitteln. Danach müssen Sie entscheiden: Gibt es eine Pauschalreise, die Sie buchen wollen? Falls ja, müssen Sie diese Pauschalreise buchen. Danach können Sie als drittes, wenn sie das wollen, noch einen Mietwagen buchen. Dazu müssen Sie zuerst einmal alle verfügbaren Mietwagen ermitteln. Danach müssen Sie entscheiden: Gibt es einen Wagen, den Sie mieten wollen? Falls ja, müssen Sie diesen Mietwagen buchen.“

Das Beispiel „Reisebuchen“ wurde stark verkürzt beschrieben. Es wurden nur die für den Beispielprozess unbedingt benötigten Prozessschritte aufgezählt. Bereits für das Anlegen der verkürzten Prozessbeschreibung brauchten die Testteilnehmer durchschnittlich eine Stunde. Eine längere Beschreibung hätte unzumutbar viel Zeit benötigt. Ein weiterer Grund für dieses Vorgehen war, dass alle Testpersonen zwecks Vergleichbarkeit genau den gleichen Prozess beschreiben sollten.

Zielgruppe des PIT sind Prozessexperten, die regelmäßig den von ihnen beschriebenen Prozess durchführen. Beim Test wurde der Prozess so gewählt, dass die Testteilnehmer ihn kennen. Da sie aber nicht unbedingt Experten in der Durchführung des Prozesses sind, wurden Hilfestellungen gegeben, um die Strukturierung des Prozesses und die Definition der Prozessschritte zu erleichtern. Der Text wurde anhand der übergeordneten Prozessschritte in Abschnitte geteilt, die einzelnen übergeordneten Prozessschritte und Unterschritte wurden im Text fett gedruckt und einige Schlüsselworte, die bei der

Definition hilfreich sind, kursiv gedruckt. So konnte sichergestellt werden, dass die Testpersonen die Kenntnisse zur Prozessbeschreibung hatten, die auch ein Prozessexperte gehabt hätte.

Im Vorfeld der Evaluation wurde ein Prätest mit acht Teilnehmern durchgeführt. Dabei wurden vier ITler und vier Nicht-ITler getestet. Beide Gruppen setzten sich aus je zwei Frauen und zwei Männern zusammen. Die Ergebnisse des Prätests wurden in der Anleitung und in der Testdurchführung berücksichtigt. Außerdem wurde ein Fragebogen von den Teilnehmern des Prätests hinsichtlich Vollständigkeit und Verständlichkeit bewertet. Basierend auf dem Feedback der Teilnehmer des Prätests wurden einige Formulierungen im Fragebogen geändert.

Der Benutzertest ermittelt, ob alle Testpersonen in der Lage sind, mit dem PIT und dem WGT einen vorgegebenen Prozess korrekt zu beschreiben und einen Wizard zu erstellen. Um auch die Einschätzung der Testteilnehmer zur Bedienbarkeit der Werkzeuge ermitteln zu können, wurde außerdem ein Fragebogen entwickelt, der auf Standardfragebögen [9, 19, SP05] aufbaut.

Der Fragebogen wurde in Form eines **Online-Fragebogens** erstellt. Er umfasste insgesamt 25 Fragen, die sich aus folgenden Frageformaten zusammensetzten:

- 7 Fragen als semantisches Differential,
- 3 Ja- / Nein-Fragen,
- 3 Multiple-Choice-Fragen (mit 4, 5 und 8 Antwortmöglichkeiten),
- 7 Antwortfelder für freie Antworten,
- 5 Fragen zu demographischen Daten.

Die Testpersonen beantworteten fünf Fragen zu ihren demographischen Daten (Alter, Geschlecht, Bildung ...), vier Fragen zu ihren PC-Vorkenntnissen (Dauer der Zeit am PC, Nutzungsart des PCs, Kenntnisse in Prozessmodellierung und Programmierung), elf Fragen zur Bewertung des PIT (Anordnung der Informationen am Bildschirm, erwartetes Programmverhalten, verständliche Bedienung, Vorhandensein nötiger Funktionen, Möglichkeit zur Änderung von Fehlereingaben und Anleitung) und vier allgemeine Fragen zum Test (Wunsch, Eingaben im Nachhinein zu ändern, Anmerkungen zum Test und zum Wizard). Außerdem musste eine ID angegeben werden, die für die Zuordnung der Fragebögen zu den Testergebnissen verwendet wurde.

8.1.2 Testablauf

Der Test teilte sich in 32 Einzeltests auf, an denen je eine Testperson teilnahm. Die Rahmenbedingungen waren bei allen Tests gleich.

Zu Beginn eines Tests, wurde die teilnehmende Person begrüßt. Es wurde erläutert, was der Zweck des Tests ist (im Rahmen einer Dissertation zwei Werkzeuge hinsichtlich Bedienbarkeit und Benutzerfreundlichkeit testen) und wie lang der Test ungefähr dauern wird. Außerdem wurde auf Anonymität und Freiwilligkeit des Tests hingewiesen.

Anschließend wurde der Testperson detailliert erläutert, was ihre Aufgabe ist. Außerdem wurde der Aufbau der PIT-Anleitung erklärt. Die Durchführung des Tests dauerte durchschnittlich eine Stunde. Dabei legten die Testteilnehmer eine Prozessbeschreibung

des Prozesses „Reise buchen“ an. Wurde der Prozess korrekt angelegt, so mussten innerhalb des Prozesses drei Prozessschritte und sieben atomare Unterschritte angelegt und spezifiziert werden. Prozess, Prozessschritte und atomare Unterschritte waren zu beschreiben. Danach war mit dem WGT ein prototypischer Wizard basierend auf der Prozessbeschreibung anzulegen. Diesen Wizard sollte die Testperson ausführen und prüfen, ob er mit ihren Erwartungen übereinstimmt. Nachdem die Testperson den Test durchgeführt hatte, wurde sie gebeten, einen Fragebogen auszufüllen. Das Ausfüllen des Fragebogens dauerte durchschnittlich 5 Minuten.

8.1.3 Ergebnisse

Die Auswertung des Benutzertests betrachtete zwei Aspekte:

1. Die erstellten Prozessbeschreibungen - Für die Auswertung dieses Aspektes wurden pro Test die Fehler gezählt, die beim Anlegen der Prozessbeschreibungen gemacht wurden.
2. Die Antworten auf den Fragebogen – Bei diesem Aspekt wurden die Antworten auf die Fragen des Fragebogens ausgewertet.

Auswertung der Prozessbeschreibungen

Zur Bewertung der Prozessbeschreibungen wurde im Vorfeld des Tests eine Fehleranalyse durchgeführt. Dabei haben drei verschiedene Personen mit dem PIT und dem WGT gearbeitet und anschließend überlegt, welche Fehler bei der Arbeit mit diesen Werkzeugen auftreten können. Alle drei Personen stellten unabhängig von einander die gleichen möglichen Fehlerquellen fest:

1. Beim Anlegen der Prozessschritte und Unterschritte sowie bei deren Spezifikation können Fehler auftreten.
2. Beim Anlegen der Beziehungen zwischen Prozessschritten bzw. Unterschritten können Fehler auftreten.

Bei der Bedienung vom WGT können vom Anwender keine Fehler gemacht werden, da dieser nur auf einen Button drücken muss, der in der Testanleitung mit Erklärung und Screenshot benannt ist. Wird kein Wizard erzeugt, so liegt das an einer fehlerhaften und / oder unvollständigen Prozessbeschreibung und ist somit bedingt durch Fehler, die während der Arbeit mit dem PIT gemacht wurden. Deswegen wurde das WGT nicht separat bewertet.

Alle Testteilnehmer haben im Rahmen der Evaluation mit dem PIT eine Prozessbeschreibung zum Prozess „Reise buchen“ angelegt. Daraus wurde dann mit dem WGT ein Wizard erstellt. Nach Abschluss aller Tests wurden die Ergebnisse ausgewertet. Für die Auswertung wurden pro Test folgende Fragen beantwortet:

1. Wurde überhaupt ein Wizard generiert?
2. Enthält die Prozessbeschreibung alle im Prozess benötigten Prozessschritte und atomaren Unterschritte?

3. Wurden die Prozessschritte korrekt spezifiziert hinsichtlich freiwilliger / verpflichtender Ausführung, Abhängigkeiten und Vorbedingungen?
4. Wurden die atomaren Unterschritte korrekt spezifiziert hinsichtlich ihres Typs (Ermittlung, Entscheidung, Ausführung), freiwilliger / verpflichtender Ausführung, Abhängigkeiten und Vorbedingungen?

Die erste Frage war mit „Ja“ oder „Nein“ zu beantworten. Da alle Testteilnehmer einen lauffähigen Wizard erstellt haben, wurde die Frage in allen 32 Fällen mit „Ja“ beantwortet.

Bei der zweiten Frage wurde betrachtet, ob die angelegten Prozessschritte und atomaren Unterschritte korrekt angelegt und benannt wurden. Wurden die Prozessschritte und die atomaren Unterschritte nicht korrekt angelegt, so ergab das jeweils einen Fehler.

Für die Beantwortung der dritten Frage wurde für alle Prozessschritte betrachtet, ob deren Ausführung korrekt den Werten „freiwillig“ oder „abhängig“ zugeordnet worden war. Pro falsche Zuordnung wurde ein Fehler gezählt. Außerdem wurde betrachtet, ob alle Abhängigkeiten und Vorbedingungen korrekt definiert worden waren. Auch hierbei wurde pro falsche Zuordnung ein Fehler gerechnet.

Bei der Auswertung der atomaren Unterschritte (vierte Frage) wurde betrachtet, ob der Typ korrekt vergeben war. Für jeden falschen Typ wurde ein Fehler gerechnet. Weiterhin wurde ermittelt, ob die Ausführung korrekt den Werten „freiwillig“ oder „abhängig“ zugeordnet worden war. Pro falsche Zuordnung wurde ein Fehler gezählt. Außerdem wurde betrachtet, ob alle Abhängigkeiten und Vorbedingungen korrekt definiert worden waren. Auch hierbei wurde pro falsche Zuordnung ein Fehler gerechnet.

Bei der gesamten Auswertung wurden Folgefehler und Wiederholungsfehler nicht gezählt. Dadurch sollte berücksichtigt werden, dass bestimmte Fehler dazu führen, dass Folgefehler gemacht werden, die andernfalls nicht gemacht worden wären. So führt beispielsweise das Vergessen einer Entscheidung dazu, dass von dieser Entscheidung auch keine Schritte abhängen können. Die fehlende Abhängigkeit wurde in einem derartigen Fall nicht als zusätzlicher Fehler gewertet.

Insgesamt wären 56 Fehler möglich gewesen.

Die Auswertung der Testergebnisse hinsichtlich der gemachten Fehler ergab folgende Werte:

	Minimum	**Maximum**	**Mittelwert Fehler**	**Standardabweichung Fehler**
ITler	0	8	1,44	2,476
Nicht-ITler	0	9	2,75	2,352
Gesamt	0	9	2,09	2,467

Tabelle 7: Auswertung der Testergebnisse hinsichtlich Fehlern.

Daraus lässt sich erkennen, dass beide Gruppen sehr gut in der Lage waren, die Prozesse zu beschreiben. Die ITler haben allerdings im Durchschnitt etwas besser abgeschnitten

als die Nicht-ITler. (Der Unterschied zwischen beiden Gruppen betrug im Durchschnitt etwa 1,3 Fehler.) Somit lässt sich folgern: Sowohl Personen mit, als auch Personen ohne Vorkenntnisse in Software-Design und -Entwicklung sind in der Lage, mit dem PIT Prozesse so zu beschreiben, dass daraus mit dem WGT ein Wizard erstellt werden kann. Allerdings schneiden Personen mit Vorkenntnissen geringfügig besser ab. Der Unterschied ist allerdings im Verhältnis zur Anzahl maximal möglicher Fehler gering.

Bei den Mittelwerten ist zu beachten, dass diese von den Extrema abstrahieren und die eigentliche Verteilung der Fehler nicht erkennen lassen. Die hohe Standardabweichung lässt auf eine breite Streuung der Fehleranzahlen schließen. Das ist insbesondere in der Gruppe der ITler relevant, da hier die Verteilung der Resultate nur schwach dem Kriterium der Normalverteilung genügt, was mit dem Kolmogorov-Smirnov-Anpassungstests überprüft worden war. Hierauf wird im Laufe dieses Abschnittes noch einmal näher eingegangen. An dieser Stelle soll zusätzlich die Fehlerverteilung in beiden Gruppen aufgezeigt werden, um die genaue Zusammensetzung dieser Verteilungen darstellen zu können.

Abbildung 66 zeigt die Verteilung der Fehleranzahl in der Gruppe der ITler (dunkel) und in der Gruppe der Nicht-ITler (hell). Man kann erkennen, dass in der Gruppe der ITler fast alle Personen keine, ein oder zwei Fehler gemacht haben. Außerdem gab es in dieser Gruppe zwei Personen, die überdurchschnittlich viele Fehler gemacht haben. Bei den Nicht-ITlern waren die Fehler normalverteilt: Es wurden null bis neun Fehler gemacht.

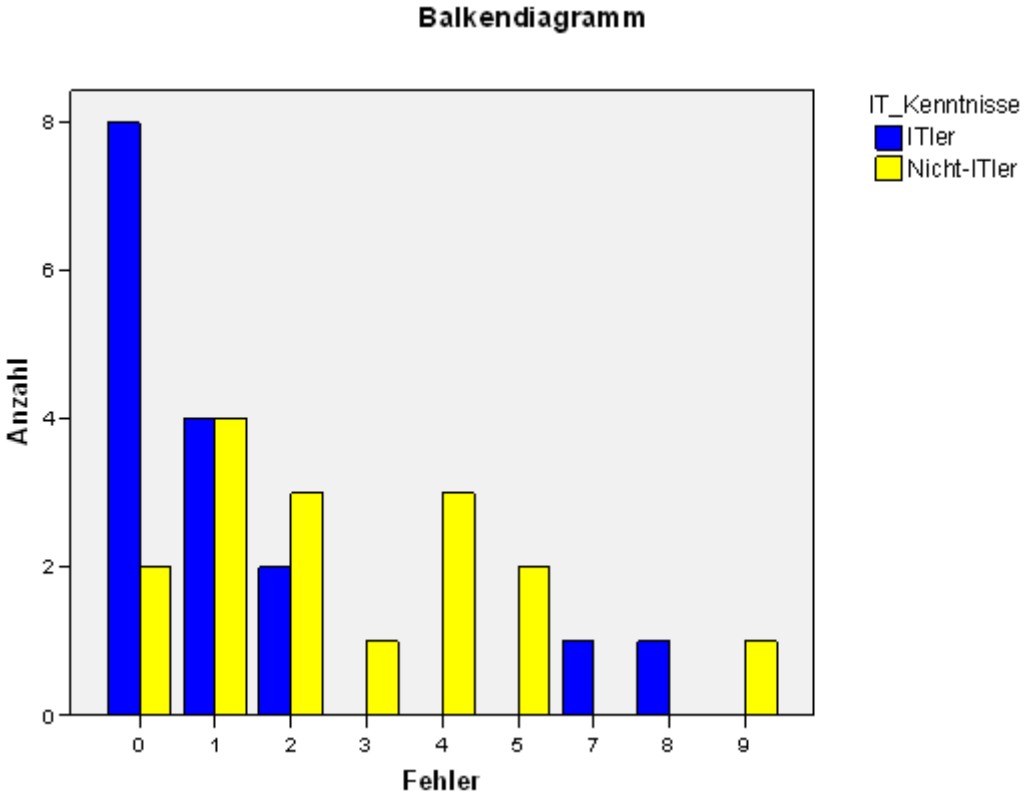


Abbildung 66: Das Diagramm zur Anzahl der Fehler.

Doch auch unter Berücksichtigung der einzelnen Fehlerergebnisse lässt sich im Hinblick auf die gesamt mögliche Fehlerzahl sagen, dass sowohl ITler als auch Nicht-ITler gut in der Lage sind, mit dem PIT und dem WGT Prozesse zu beschreiben und Wizards zu generieren, da in beiden Gruppen selbst die Personen, die viele Fehler gemacht haben, noch weit von der maximal möglichen Fehlerzahl entfernt lagen.

Im Test wurde den Testpersonen keine Zeit zur Korrektur von Fehleingaben gegeben, nachdem sie den mit dem WGT erstellten Wizard angesehen hatten. Viele der Personen gaben aber an, nachdem sie den Wizard gesehen hatten, gern noch Korrekturen vornehmen zu wollen. Man kann also davon ausgehen, dass bei einem zweiten Test mit den gleichen Testgruppen vermutlich nur noch sehr wenige Fehler beim Anlegen der Prozessbeschreibungen gemacht worden wären und der Unterschied zwischen beiden Gruppen geringer ausgefallen wäre.

Neben den gemachten Fehlern wurde auch berücksichtigt, wie viel Zeit eine Testperson bei der Prozessbeschreibung benötigt hatte. Hierbei zeigte sich, dass beide Gruppen im Mittel etwa eine Stunde benötigt hatten. Tabelle 8 listet die Mittelwerte und Standardabweichungen auf:

	Minimum	**Maximum**	**Mittelwert Zeit (Anzahl Minuten geteilt durch 60)**	**Standardabweichung Zeit (Anzahl Minuten geteilt durch 60)**
ITler	0,58	1,83	1,0573	0,32306
Nicht-ITler	0,67	1,92	1,0135	0,26834
Gesamt	0,58	1,92	1,0354	0,29298

Tabelle 8: Benötigte Zeit.

Die durchschnittliche Differenz zwischen beiden Gruppen macht ca. 3 Minuten aus, was bezogen auf die durchschnittliche Gesamtzeit von rund 60 Minuten sehr gering ist. Somit gab es in Bezug auf diesen Faktor keinen nennenswerten Unterschied zwischen beiden Gruppen.

Zu Beginn von Abschnitt 8.1 wurden Hypothesen vorgestellt, die in diesem Test untersucht wurden. Nachfolgend wird gezeigt, ob sie sich bestätigt haben.

H1: Mit dem PIT und dem WGT sind **nur geringfügig** weniger Nicht-ITler als ITler in der Lage, aus einer Prozessbeschreibung einen Wizard zu generieren.

Um **Hypothese H1** zu bestätigen, mussten keine weiteren Analysen vorgenommen werden, da alle Personen mit dem WGT in der Lage waren, einen lauffähigen Wizard basierend auf ihren Prozessbeschreibungen zu erzeugen. Somit war diese Hypothese erfüllt.

Ergebnis: Es haben genauso viele Nicht-ITler wie ITler einen Wizard generiert.

H2: Nicht-ITler machen **nur geringfügig** mehr Fehler bei der Anwendung des PIT (Erstellen der Prozessbeschreibung) als die ITler.

Um **Hypothese H2** zu bestätigen, wurde eine multivariate Varianzanalyse mit 3 Faktoren durchgeführt: Alter, Geschlecht, IT-Kenntnisse. Im Folgenden werden die für die Evaluation von H2 relevanten Ergebnisse dargestellt. Dabei werden die Faktoren Alter und Geschlecht außer Acht gelassen, da sie für diese Arbeit keine zusätzlichen Erkenntnisse bringen.

Im Rahmen der Varianzanalyse wurde die Fehlerverteilung in den Gruppen berechnet. Dabei wurde berechnet, ob es einen signifikanten Unterschied zwischen den Fehlerzahlen beider Gruppen gibt. Ergebnis der Varianzanalyse war, dass es keinen signifikanten Unterschied gab.

Annahme bei der Varianzanalyse war, dass die Fehlerergebnisse beider Gruppen normalverteilt sind. Ob diese Vorraussetzung erfüllt war, wurde mit Hilfe des Kolmogorov-Smirnov-Anpassungstests überprüft. Dabei ergab sich für beide Gruppen eine Normalverteilung der Ergebnisse. Allerdings waren die Bedingungen für eine Normalverteilung in der Gruppe der ITler nur schwach erfüllt.

Daher wurden noch der Mann-Whitney-U-Test [Bo05] durchgeführt, der keine Normalverteilung der Daten voraussetzt. Dieser Test ergab einen signifikanten Zusammenhang zwischen Fehleranzahl und IT-Kenntnissen. Ein Grund für dieses Verhalten könnte sein, dass die Testdaten nur knapp den Anforderungen an eine Normalverteilung genügen, was sich in der Fehlerverteilung (siehe Tabelle 7) erkennen lässt. Betrachtet man allerdings den Unterschied im Verhältnis zur möglichen Maximalzahl von Fehlern, so fällt der Unterschied gering aus.

Ergebnis: Nicht-ITler machen zwar mehr Fehler als ITler, aber die Differenz der Mittelwerte ist gering.

Als **Endergebnis** lässt sich somit sagen, dass es zwischen ITlern und Nicht-ITlern zwar einen Unterschied bei der Menge der in Prozessbeschreibungen gemachten Fehler gibt, der vermutlich in den Vorkenntnissen der ITler begründet ist. Im Mittel war dieser Unterschied aber gering. Außerdem hat sich gezeigt, dass beide Gruppen bei der Arbeit mit dem PIT gute Resultate erzielen. Würde man mit beiden Gruppen einen weiteren Test durchführen, so würde der Unterschied vermutlich aufgrund des Übungseffektes geringer ausfallen. Alle Personen waren in der Lage mit dem WGT aus den Prozessbeschreibungen einen prototypischen Wizard zu erzeugen. Somit konnten beide Hypothesen bestätigt werden.

Auswertung der Fragebögen

Zur Auswertung der Fragebogenergebnisse wurde eine Faktorenanalyse durchgeführt, um zu ermitteln, welche Fragen miteinander zusammenhängen. Dabei ergab sich, dass sich die sieben Fragen des Fragebogens zum PIT zu zwei Faktoren zusammenfassen lassen: Nutzerzufriedenheit und Bewertung der Funktionalität.

Folgende Fragen tragen zum Faktor der Nutzerzufriedenheit (Faktor F1) bei:

- Wie hilfreich war die Anleitung für die Arbeit mit dem PIT?
- War die Anordnung der Informationen auf dem Bildschirm übersichtlich?
- War die Bedienung vom PIT verständlich?
- Hat sich das Programm immer so verhalten, wie Sie das erwartet haben?
- Wie hoch war der Aufwand zum Beseitigen von Fehlern?

Folgende Fragen lassen sich dem Faktor der Bewertung der Funktionalität (Faktor F2) zuordnen:

- Enthält das Programm PIT alle für die Aufgabe benötigten Funktionen?
- Hatten Sie das Gefühl bei der Arbeit mit dem PIT, Eingaben gemacht zu haben, welche Ihnen eigentlich überflüssig erschienen?

Die Tabelle 9 zeigt die Ergebnisse der Faktorenanalyse für alle Fragen. Fragen, die für einen Faktor einen hohen Korrelationswert ergeben, werden diesem Faktor zugeordnet, da es für diesen Faktor einen hohen Zusammenhang gibt. Alle Fragen, die demselben Faktor zugeordnet werden, stehen in hohem Zusammenhang miteinander. Das bedeutet, dass eine Person, die eine der Fragen in einer bestimmten Form beantwortet, die anderen Fragen des gleichen Faktors ähnlich beantwortet. In Tabelle 9 lässt sich anhand der Korrelationswerte erkennen, welche Fragen dem gleichen Faktor angehören. Die Werte, die zur Zuordnung einer Frage zu einem bestimmten Faktor führen, sind fett gedruckt.

	Faktor	
Frage zu:	**F1**	**F2**
Hilfreiche Anleitung	**0,738**	-0,046
Übersichtliche Informationsanordnung	**0,782**	0,222
Verständliche Bedienung	**0,845**	0,081
Vorhersehbares Programmverhalten	**0,869**	0,147
Aufwand Fehlerbeseitigung	**0,602**	0,409
Enthält benötigte Funktionen	0,157	**0,692**
Überflüssige Eingaben	0,015	**0,836**

Tabelle 9: Ergebnisse der Faktorenanalyse für den Fragebogen.

Bei allen Fragen gab es die Möglichkeit, mittels einer Skala von 6 bis 1 zu bewerten, wobei 6 der beste Wert war und 1 der schlechteste. Sowohl bei der Nutzerzufriedenheit als auch bei der Bewertung der Funktionalität waren die Bewertungen beider Gruppen positiv. Es ergab sich kein signifikanter Unterschied zwischen den Gruppen. Tabelle 10

zeigt die Auswertung der Fragebögen in Bezug auf die beiden Faktoren. Die getrennte Auswertung aller Fragen findet sich im Anhang.

	Minimum	**Maximum**	**Mittelwert**	**Standard-abweichung**
Benutzerzufriedenheit (ITler)	4	5,8	5,013	0,5390
Benutzerzufriedenheit (Nicht-ITler)	3	6	4,875	0,8941
Bewertung der Funktionen (ITler)	3	6	4,906	1,0363
Bewertung der Funktionen (Nicht-ITler)	3,5	6	5,25	0,7303

Tabelle 10: Auswertung Fragebogen.

Insgesamt ergab sich ein Mittelwert von 4,944 für die Zufriedenheit und 5,078 für die Bewertung der Funktionen. Maximal möglicher Wert in beiden Fällen war 6.

Als Endergebnis des Fragebogens lässt sich somit sagen, dass die Benutzerzufriedenheit hinsichtlich des PIT sowie die Bewertung der im PIT enthaltenen Funktionen bei allen Anwendern sehr positiv waren.

8.1.4 Zusammenfassung der Ergebnisse

Somit lässt sich als Endergebnis für die gesamte Evaluation zum PIT und zum WGT sagen, dass sowohl Personen mit als auch ohne IT-Vorkenntnisse gut in der Lage sind, den in dieser Arbeit vorgestellten Ansatz zur Pattern-basierten Prozessbeschreibung zu nutzen. Dabei ist es nicht erforderlich, dass sie bestimmte Vorkenntnisse mitbringen. Sie müssen lediglich in der Lage sein, den von ihnen beschriebenen Prozess detailliert zu kennen und ihn beschreiben zu können. Da auch die Zufriedenheit der Nutzer mit dem Werkzeug PIT und dessen Funktionalitäten hoch war, ist auch nicht die Hürde zu überwinden, dass die Anwender mit Werkzeugen arbeiten müssen, die nicht ihren Wünschen entsprechen.

Das in dieser Arbeit vorgestellte Konzept und die dafür entwickelten Werkzeuge erlauben es also, dass Prozessexperten die von ihnen durchgeführten Prozesse so beschreiben, dass daraus automatisiert ein prototypischer Wizard erzeugt werden kann. Prozessbeschreibungen und Wizards sind die Basis für die weitere Entwicklung und stellen eine wichtige Wissensquelle und Kommunikationsgrundlage dar. So lassen sich die Prozessexperten unter Verwendung des in dieser Arbeit vorgestellten Konzeptes maßgeblich in den Prozess der Software-Entwicklung einbinden.

Das erste Hauptziel dieser Arbeit war die Erstellung eines Konzeptes und eines Frameworks für einen einfach handhabbaren Prozessbeschreibungsformalismus als Basis für

die Erstellung eines Werkzeugs zur Unterstützung von Nicht-Experten bei der Durchführung von Anpassungsprozessen. Im Rahmen des in diesem Abschnitt betrachteten Benutzertests wurde gezeigt, dass das Ziel eines einfach handhabbaren Prozessbeschreibungsformalismus erreicht wurde. Anhand des basierend auf den Prozessbeschreibungen erstellten Anpassungswerkzeuges wird im folgenden Abschnitt untersucht, ob sich dieses Werkzeug zur Prozessunterstützung auch für Nicht-Experten eignet.

8.2 Evaluation des Anpassungswerkzeuges

Das in dieser Dissertation vorgestellte Konzept ermöglicht es Prozessexperten, die von ihnen durchgeführten Anpassungsprozesse so zu beschreiben, dass basierend auf den Prozessbeschreibungen ein prototypischer Wizard erstellt werden kann. Dieser Wizard dient als Basis für die weitere Entwicklung. Durch Erweiterung um automatisierte Funktionalitäten lässt er sich zu einem voll einsatzfähigen Programm zur Prozessunterstützung erweitern. Dieses Programm erlaubt es auch Personen, die keine Prozessexperten sind, die Anpassungen durchzuführen.

In Kapitel 6 wurde bereits erläutert, wie der basierend auf Anpassungspatterns erstellte Wizard um automatisierte Funktionen erweitert wurde und so den Anforderungen an ein Anpassungswerkzeug (vgl. Abschnitte 2.3 und 2.4) gerecht wird. Ziel des so erstellten Wizards ist es, Nutzer, auch wenn sie keine Prozessexperten sind, bei der Durchführung von Anpassungsprozessen geeignet zu unterstützen. Um zu zeigen, dass diese Anwender mit dem Anpassungswerkzeug in der Lage sind, Anpassungsprozesse mit einem guten Resultat und in einer angemessenen Zeit durchzuführen, wurde ein weiterer Benutzertest durchgeführt. Dieser wurde ebenfalls im Rahmen einer Studienarbeit [Bu08] am Institut für Psychologie an der Technischen Universität Darmstadt mitbetreut.

Annahme bei dem Test war, dass Anwender in der Lage sind, mit dem Anpassungswerkzeug existierende Lernressourcen schneller und mit weniger Fehlern anpassen zu können, als mit auf dem Markt existierenden Werkzeugen zur Bearbeitung von HTML-Dateien. Um dies nachzuweisen, wurde ein Test mit zwei Gruppen durchgeführt. Beide Gruppen hatten die gleiche Aufgabe: Sie sollten eine Reihe von oft vorkommenden Anpassungsarten in E-Learning Kursen, bestehend aus HTML-Dateien, durchführen. Die eine Gruppe führte die Aufgaben mit dem Anpassungswerkzeug durch, die anderen mit einem WYSIWYG HTML-Editor. Der durchgeführte Benutzertest sollte die folgenden Hypothesen nachweisen:

H1: Mit dem Anpassungswerkzeug lassen sich die Anpassungen **schneller** als mit einem herkömmlichen Werkzeug durchführen.

H2: Mit dem Anpassungswerkzeug werden **weniger** Fehler bei Anpassungen gemacht als mit einem herkömmlichen Werkzeug.

Die genaue Zusammensetzung der untersuchten Nutzergruppen sowie der Aufbau des Tests werden im folgenden Abschnitt erläutert.

8.2.1 Testaufbau

Auch der zweite Test baut auf einem mehrfaktoriellen 2x2x2 Versuchsplan auf. Dieser enthält drei **unabhängige Variablen (UV)**:

- Verwendetes Bearbeitungswerkzeug, aufgeteilt in die Gruppe der Personen, die mit dem Anpassungswerkzeug gearbeitet haben und die Gruppe der Personen, die mit einem anderen Werkzeug (Netscape Composer) gearbeitet haben
- Alter, aufgeteilt in die Gruppen „jung“ (20-28 Jahre) und „alt“ (29-61 Jahre)
- Geschlecht, aufgeteilt in die Gruppen „männlich“ und „weiblich“

Alter und Geschlecht sind Organismusvariablen.

Daraus ergaben sich 6 Bedingungen: Anpassungswerkzeug versus Vergleichswerkzeug, männlich versus weiblich, alt versus jung. Dies führte zu insgesamt 8 Gruppen (siehe Tabelle 11).

	Anpassungswerkzeug				Vergleichswerkzeug			
	Männlich		Weiblich		Männlich		Weiblich	
	Alt	Jung	Alt	Jung	Alt	Jung	Alt	Jung
Anzahl Personen	4	4	4	4	4	4	4	4

Tabelle 11: Versuchsplan des zweiten Versuches.

Auch in diesem Test wurde aus den gleichen Gründen wie beim in Abschnitt 8.1 beschriebenen Test eine Stichprobengröße von 32 gewählt, die sich aus acht Testgruppen zu je vier Teilnehmern zusammensetzte.

Die **abhängigen Variablen (AV)** bestehen aus der Zeit, die für die Durchführung des Tests benötigt wurde, der Anzahl der gemachten Fehler sowie der Einschätzung der Testpersonen hinsichtlich Aufgabenangemessenheit, Transparenz des Programms, sowie der Informationsanordnung auf dem Bildschirm. Gemessen wurden die AV zum einen durch Zählen der gemachten Fehler und Messen der benötigten Zeit. Zum anderen wurde in einem Online-Fragebogen die Einschätzung der Testpersonen hinsichtlich der übrigen AVs abgefragt.

Alle Testteilnehmer erhielten die gleiche Aufgabe: Sie sollten in drei E-Learning Kursen, die aus einer Reihe von HTML-Dateien bestanden, typische Anpassungsprozesse durchführen. Den Teilnehmern wurde eine Anleitung zur Verfügung gestellt, die die Anpassungsprozesse und die Bedienung des zu verwendenden Werkzeuges erklärte. Damit für alle Teilnehmer die Vorraussetzungen gleich waren, wurden nur Teilnehmer berücksichtigt, die bereits HTML-Kenntnisse hatten. Allen Teilnehmern wurde eine Anleitung zur Verfügung gestellt, die für jedes Werkzeug detailliert dessen Funktionsweise beschreiben und die in den einzelnen Kursen durchzuführenden Anpassungsprozesse benennen. Die Hälfte der Teilnehmer arbeitete mit dem in dieser Arbeit vorgestellten Anpassungswerkzeug. Die andere Gruppe arbeitete mit dem Vergleichswerkzeug.

Als Vergleichswerkzeug sollte ein Werkzeug gewählt werden, dass möglichst einfach zu bedienen ist und erlaubt, die vorgegebenen Anpassungen mit geringem Zeitaufwand durchzuführen. Die Wahl fiel auf den Netscape Composer, Version 7.1. Dieses Werkzeug ist frei erhältlich. Es bietet die Möglichkeit, HTML-Dateien in einem WYSIWYG-Editor zu ändern. Das hat den Vorteil, dass man nicht erst im Quelltext suchen muss, sondern direkt die angegebenen Elemente ändern kann und auch direkt sieht, ob man eine Änderung korrekt ausgeführt hat. Außerdem ist der Composer auch für Nutzer ohne Kenntnisse des Werkzeuges mit einer geeigneten Anleitung einfach zu bedienen.

Folgende Aufgaben waren im Rahmen des Tests durchzuführen:

- In einem Kurs war ein Bild auszutauschen und die Größe des neuen Bildes war anzupassen, was eine gestalterische Anpassungsart darstellt.
- In einem anderen Kurs mussten alle Vorkommen eines Logos und alle Vorkommen einer Hintergrundfarbe geändert werden, was ebenfalls gestalterischen Anpassungsarten entspricht. Außerdem war eine inhaltliche Anpassungsart durchzuführen: Es waren alle Vorkommen eines Firmennamens zu ändern.
- In einem weiteren Kurs waren alle Vorkommen eines Begriffes durch einen anderen Begriff zu ersetzen, was ebenfalls eine inhaltliche Anpassungsart ist.

Im Vorfeld der Evaluation wurde ein Prätest mit vier Teilnehmern durchgeführt. Dabei testete jede Testperson mit beiden Werkzeugen. Um Auswirkungen der Reihenfolge der Werkzeuge zu vermeiden, testeten zwei Teilnehmer erst mit dem Anpassungswerkzeug und danach mit dem Vergleichswerkzeug. Die anderen beiden Teilnehmer arbeiteten in umgekehrter Reihenfolge. Die Testgruppe setzte sich aus je zwei Frauen und zwei Männern zusammen. Die Ergebnisse des Prätests wurden in den Anleitungen und im Fragebogen berücksichtigt.

Der Test ermittelt, ob die Benutzer, die das Anpassungswerkzeug einsetzen, schneller und mit weniger Fehlern die Anpassungen durchführen können, als die Personen, die den Netscape Composer verwenden. Um auch die Einschätzung der Testteilnehmer zur Bedienbarkeit der Werkzeuge ermitteln zu können, wurde außerdem ein Fragebogen entwickelt, der auf Standardfragebögen [9, 19, SP05] aufbaut.

Der Fragebogen wurde in Form eines **Online-Fragebogens** erstellt. Er umfasste insgesamt 24 Fragen, die sich aus folgenden Frageformaten zusammensetzten:

- 9 Fragen als semantisches Differential,
- 2 Ja- / Nein-Fragen,
- 3 Multiple-Choice-Fragen (mit 4, 5 und 8 Antwortmöglichkeiten),
- 5 Antwortfelder für freie Antworten,
- 5 Fragen zu demographischen Daten.

Die Testpersonen beantworteten fünf Fragen zu ihren demographischen Daten (Alter, Geschlecht, Bildung ...), drei Fragen zu ihren PC-Vorkenntnissen (Dauer der Zeit am PC, Nutzungsart des PCs, Kenntnisse in Programmierung), 14 Fragen zur Bewertung des jeweils zur Durchführung der Anpassungen verwendeten Werkzeuges (Anordnung der Informationen am Bildschirm, erwartetes Programmverhalten, verständliche Bedienung, Vorhandensein nötiger Funktionen, Möglichkeit zur Änderung von

Fehlereingaben und Anleitung) und eine allgemeine Frage zum Test (Anmerkungen zum Test). Außerdem musste eine ID angegeben werden, die für die Zuordnung der Fragebögen zu den Testergebnissen verwendet wurde.

Der Fragebogen war so konzipiert, dass mit ihm die in Abschnitt 2.4 gestellten Anforderungen an ein Werkzeug zur Prozessunterstützung überprüft werden konnten. Lediglich die Individualisierbarkeit wurde nicht abgefragt, da diese im Test keine Rolle spielte.

8.2.2 Testablauf

Der Test teilte sich in 32 Einzeltests auf, an denen je eine Testperson teilnahm. Die Rahmenbedingungen waren bei allen Tests gleich. In allen Tests wurden die im vorigen Abschnitt genannten Aufgaben in den drei vorgegebenen Kursen durchgeführt.

Um einen Einfluss der Reihenfolge der Kurse und der Aufgaben auszuschließen, wurde die Reihenfolge der drei zu bearbeitenden Kurse rotiert. Dazu wurde folgendes Schema verwendet: Der erste Teilnehmer bearbeitete die Kurse in der Reihenfolge ABC, der zweite in der Reihenfolge BCA, der dritte in der Reihenfolge CAB. Danach wurde wieder von vorn begonnen. Mit welchem Werkzeug ein Testteilnehmer zu arbeiten hatte, wurde per Los festgelegt. Dem entsprechend wurde das für die Testdurchführung verwendete Notebook vorbereitet und die passende Anleitung wurde ausgelegt.

Zu Beginn eines Tests, wurde die teilnehmende Person begrüßt. Es wurde erläutert, was der Zweck des Tests ist (im Rahmen einer Dissertation Werkzeuge hinsichtlich diverser Faktoren vergleichen) und wie lang der Test ungefähr dauern wird. Außerdem wurde auf Anonymität und Freiwilligkeit des Tests hingewiesen. Es wurde bewusst nicht erwähnt, dass nachgewiesen werden sollte, dass das Anpassungswerkzeug besser als das Vergleichswerkzeug ist. Auch die Anzahl der getesteten Werkzeuge wurde nicht genannt. Ziel war es, eine neutrale Bewertung durch die Testpersonen zu erhalten. Anschließend wurde jeder Testperson detailliert erläutert, was ihre Aufgabe war. Dazu wurde folgender Text vorgetragen:

„Stellen Sie sich vor, Sie sind in Ihrem Unternehmen für die Erstellung von E-Learning Kursen zuständig. Zurzeit haben Sie die Aufgabe, drei Kurse, die Sie von anderen Unternehmen zur Verfügung gestellt bekommen haben, so zu verändern, dass diese in Ihrem Unternehmen eingesetzt werden können. Die neuen Kurse haben ein anderes Design, als das in Ihrem Unternehmen verlangt wird. Außerdem haben Sie beim Durchlesen der Texte festgestellt, dass einige Begriffe verwendet werden, die in Ihrem Unternehmen anders verwendet werden. Sie müssen also das Aussehen der Kurse so ändern, dass es den von Ihrem Unternehmen vorgegebenen Design-Richtlinien entspricht. Außerdem verändern Sie die Terminologie in der Art, dass alle Begriffe den Vorgaben Ihres Unternehmens entsprechen.“

Der Testperson wurde ein ausgedrucktes Dokument zur Verfügung gestellt, das zum einen die Funktionsweise des zu verwendenden Werkzeuges erläuterte, zum anderen die durchzuführenden Anpassungsaufgaben benannte. Die Durchführung des Tests dauerte durchschnittlich circa 20 Minuten. Nachdem die Testperson den Test durchgeführt hatte, wurde sie gebeten, einen Fragebogen auszufüllen. Das Ausfüllen des Fragebogens dauerte durchschnittlich 5 Minuten.

8.2.3 Ergebnisse

Die Auswertung des Benutzertests betrachtete drei Aspekte:

1. Die Auswertung der Testergebnisse hinsichtlich der für die Durchführung der Anpassungen benötigten Zeit – zu diesem Zweck wurde die Zeit gestoppt.
2. Die Auswertung der Testergebnisse hinsichtlich der gemachten Fehler – hierfür wurden die Fehler gezählt.
3. Die Auswertung der Antworten auf den Fragebogen - für diesen Aspekt wurden die Antworten auf die Fragen des Fragebogens ausgewertet.

Auswertung der Testergebnisse in Bezug auf Zeit und Fehler

Die zur Durchführung der Anpassungen benötigte Zeit wurde gestoppt. Die Auswertung der Ergebnisse führte zu den in Tabelle 12 gezeigten Werten für Minimum, Maximum, Mittelwert und Standardabweichung:

	Minimum	**Maximum**	**Mittelwert Zeit (in Minuten)**	**Standardabweichung Zeit (in Minuten)**
Composer	12,46	30,39	20,4969	2,35241
Anpassungswerkzeug	10,28	17,15	13,6662	5,91599
Gesamt	10,28	30,39	17,0816	5,62612

Tabelle 12: Testergebnisse für die Zeit.

Die durchschnittliche Differenz zwischen beiden Gruppen macht ca. 7 Minuten aus, was bezogen auf die durchschnittliche Gesamtzeit von rund 17 Minuten ein hoher Wert ist. Um zu ermitteln, ob es einen signifikanten Zusammenhang zwischen Zeit und verwendetem Werkzeug gibt, wurde die Korrelation nach Pearson gemessen. Diese ergab für die Korrelation zwischen Zeit und verwendetem Werkzeug einen Wert von -0,617 und eine sehr hohe Signifikanz von 0,0. Daraus ergibt sich folgender Zusammenhang: Die Personen, die das Anpassungswerkzeug verwenden, benötigen weniger Zeit als die Personen, die den Composer verwenden.

Weiterhin wurde eine multivariate 3-faktorielle Varianzanalyse durchgeführt, mit den Faktoren Alter, Geschlecht und verwendetes Werkzeug. Diese führte ebenfalls zu einem signifikanten Einfluss des verwendeten Werkzeuges auf die Zeit.

Neben der benötigten Zeit wurde auch berücksichtigt, wie viele Fehler eine Testperson bei der Anpassung gemacht hat. Die Anzahl der Fehler wurde ermittelt, indem die angepassten Kurse betrachtet wurden. Jede Abweichung von der Vorgabe in der Anleitung wurde als Fehler gezählt. Tabelle 13 listet die Werte für Minimum, Maximum, Mittelwert und Standardabweichung auf:

	Mini-mum	Maxi-mum	Mittelwert Fehler	Standardabwei-chung Fehler
Composer	0	4	0,75	1,18322
Anpassungswerkzeug	0	2	0,3125	0,60208
Gesamt	0	4	0,5313	0,94985

Tabelle 13: Testergebnisse für den Fehler.

Daraus lässt sich erkennen, dass Personen, die mit dem Anpassungswerkzeug gearbeitet haben, geringfügig weniger Fehler gemacht haben, als Personen, die mit dem Composer gearbeitet haben. Daher gibt es keinen eindeutigen Zusammenhang zwischen verwendetem Werkzeug und Fehleranzahl. Allerdings werden Fehler oft aufgrund von nachlassender Konzentration bei sich wiederholenden Tätigkeiten gemacht. Dieser Effekt tritt aber erst nach einiger Zeit ein. Es ist damit zu rechnen, dass bei einem längeren Versuch mit dem Anpassungswerkzeug weniger Fehler gemacht worden wären, als mit dem Vergleichswerkzeug, da sich wiederholende Aufgaben, wie das Austauschen eines Logos in mehreren Dateien vom Anpassungswerkzeug automatisiert ausgeführt werden. Der Nutzer muss diese Aufgabe nur einmal durchführen. Beim Vergleichswerkzeug dagegen, muss der Nutzer das Logo in allen Dateien austauschen.

Zusammenfassend lässt sich als Ergebnis der Bewertung der Testresultate sagen, dass die Hypothesen weitgehend erfüllt sind:

H1: Mit dem Anpassungswerkzeug lassen sich die Anpassungen **schneller** als mit dem herkömmlichen Werkzeug (Netscape Composer) durchführen.

Ergebnis: Die Durchführung ist mit dem Anpassungswerkzeug signifikant schneller als mit dem Composer.

H2: Mit dem Anpassungswerkzeug werden **weniger** Fehler bei Anpassungen gemacht als mit dem herkömmlichen Werkzeug.

Ergebnis: Mit dem Anpassungswerkzeug werden geringfügig weniger Fehler gemacht als mit dem Composer. Es lässt sich aber kein signifikantes Ergebnis feststellen.

Auswertung der Fragebögen

Zur Auswertung der Fragebogenergebnisse wurde eine Faktorenanalyse durchgeführt, um zu ermitteln, welche Fragen miteinander zusammenhängen. Dabei ergab sich, dass sich sieben der acht Fragen des Fragebogens zu zwei Faktoren zusammenfassen lassen: Aufgabenangemessenheit der Werkzeuge sowie Bewertung der Vorhersehbarkeit und Verständlichkeit. Die Frage „War die Anordnung der Informationen auf dem Bildschirm übersichtlich?“ konnte keinem Faktor eindeutig zugeordnet werden und muss für sich allein betrachtet werden. Die Frage „Falls Sie Fehler korrigieren mussten: Wie groß war der Aufwand, um diese umzuändern?“ war optional, da nicht alle Personen Fehler

gemacht haben. Sie wurde nur von sechs Personen beantwortet und konnte somit nicht ausgewertet werden.

Folgende Fragen tragen zum Faktor der Aufgabenangemessenheit (Faktor F1) bei:

- Wie hilfreich war die Anleitung für die Arbeit mit dem Composer / Anpassungswerkzeug?
- Hat der Composer / das Anpassungswerkzeug Sie informiert, welche Eingaben zulässig oder nötig sind?
- War der erforderliche Aufwand für Ihr Arbeitsergebnis angemessen?
- Wie war der Composer / das Anpassungswerkzeug auf die Anforderungen der Arbeit zugeschnitten?
- Haben Sie sich im Composer / Anpassungswerkzeug zusätzliche Funktionen zur Bearbeitung der Aufgabe gewünscht?

Folgende Fragen lassen sich dem Faktor der Bewertung der Transparenz des Programms (Faktor F2) zuordnen:

- Hat sich der Composer / das Anpassungswerkzeug so verhalten, wie Sie es erwartet haben?
- War die Bedienung des Composers / Anpassungswerkzeuges gut verständlich?

Die nachfolgende Tabelle zeigt die Ergebnisse der Faktorenanalyse für alle Fragen. Fragen, die für einen Faktor einen hohen Korrelationswert ergeben, werden diesem Faktor zugeordnet. Alle Fragen, die demselben Faktor zugeordnet werden, stehen in hohem Zusammenhang miteinander. Das bedeutet, dass eine Person, die eine der Fragen in einer bestimmten Form beantwortet, die anderen Fragen des gleichen Faktors ähnlich beantwortet. In Tabelle 14 lässt sich anhand der Korrelationswerte erkennen, welche Fragen dem gleichen Faktor angehören. Die Werte, die zur Zuordnung einer Frage zu einem bestimmten Faktor führen, sind fett gedruckt.

	Faktor	
Frage zu	**F1**	**F2**
Hilfreiche Anleitung	**0,756**	-0,064
Information über zulässige Eingaben	**0,622**	0,322
Aufwand angemessen	**0,942**	0,027
Auf Anforderung zugeschnitten	**0,937**	0,053
Zusätzliche Funktionen gewünscht	**0,828**	0,185
Vorhersehbares Programmverhalten	-0,066	**0,634**
Verständliche Bedienung	0,081	**0,833**
Übersichtliche Informationsanordnung	0,371	0,492

Tabelle 14: Ergebnisse der Faktorenanalyse für den Fragebogen.

Bei allen Fragen gab es die Möglichkeit, mittels einer Skala von 6 bis 1 zu bewerten, wobei 6 der beste Wert war und 1 der schlechteste. Tabelle 15 zeigt die Auswertung der Fragebögen in Bezug auf beide Faktoren, sowie in Bezug auf die nicht zuordenbare Frage zur Informationsanordnung. Die getrennte Auswertung aller Antworten findet sich im Anhang.

In Tabelle 15 lässt sich erkennen, dass die Ergebnisse des zweiten Faktors, sowie die der separat betrachteten Frage nach der Informationsanordnung auf dem Bildschirm für beide Werkzeuge fast gleich ausgefallen sind. Die Bewertung der Fragen, die zum Faktor der Aufgabenangemessenheit (F1) gehören, fällt für das Anpassungswerkzeug deutlich besser aus. Um herauszufinden, ob ein signifikanter Zusammenhang zwischen der Bewertung des Fragebogens und dem verwendeten Werkzeug besteht, wurde die Korrelation nach Pearson zwischen diesen Variablen berechnet. Dabei hat sich gezeigt, dass kein Zusammenhang zwischen dem verwendeten Werkzeug und der Bewertung des zweiten Faktors bzw. der separat betrachteten Frage besteht.

	Minimum	**Maximum**	**Mittelwert**	**Standardabweichung**
F1: Aufgabenangemessenheit (Composer)	1,2	6,0	4,138	1,4908
F1: Aufgabenangemessenheit (Anpassungswerkzeug)	4,4	5,8	5,4	0,3933
F2: Bewertung der Transparenz des Programms (Composer)	4	6	5,5	0,6831
F2: Bewertung der Transparenz des Programms (Anpassungswerkzeug)	4	6	5,250	0,6583
Übersichtliche Informationsanordnung (Composer)	4	6	5,31	0,704
Übersichtliche Informationsanordnung (Anpassungswerkzeug)	4	6	5,69	0,602

Tabelle 15: Auswertung der Fragebögen.

Die Korrelation nach Pearson für den Zusammenhang zwischen der Bewertung der Aufgabenangemessenheit und dem verwendeten Werkzeug ergab einen Wert von 0,513 und eine hohe Signifikanz von 0,03. Daraus ergibt sich folgender Zusammenhang: Die Personen, die das Anpassungswerkzeug verwenden, sind deutlich zufriedener in Bezug auf die Aufgabenangemessenheit als die, die den Composer verwenden.

Als Endergebnis des Fragebogens lässt sich somit sagen, dass die Bewertung des verwendeten Werkzeuges in Bezug auf die Aufgabenangemessenheit bei den Testpersonen, die mit dem Anpassungswerkzeug gearbeitet haben, deutlich besser war, als bei der Vergleichsgruppe, die mit dem Composer gearbeitet hat. Die Transparenz des Programms sowie die Informationsanordnung auf dem Bildschirm wurden von beiden Testgruppen positiv bewertet, wobei sich hier kein Unterschied für die beiden Werkzeuge feststellen lässt.

8.2.4 Bewertung hinsichtlich der Anforderungen zur Wizard Gestaltung

In Abschnitt 2.4 wurden Anforderungen an ein Werkzeug zur Unterstützung von Prozessen gestellt. Anhand des Anpassungswerkzeuges wird in diesem Abschnitt betrachtet, in wie weit diese Anforderungen erfüllt wurden, dazu wurden zum Überprüfen einiger Anforderungen Fragen im Fragebogen gestellt. Da im Test nicht alle Anforderungen eine Rolle spielten, beziehungsweise einige Anforderungen von den Testpersonen schwer zu beurteilen gewesen wären, wurden nicht alle Anforderungen im Fragebogen abgefragt. Tabelle 16 zeigt, welche Anforderungen durch welche Fragen im Fragebogen überprüft wurden:

Anforderung	**Frage im Fragebogen**
Aufgabenangemessenheit	Auf Anforderung zugeschnitten Aufwand angemessen Zusätzliche Funktionen gewünscht
Selbstbeschreibungsfähigkeit	Übersichtliche Informationsanordnung Verständliche Bedienung
Steuerbarkeit	Diese Anforderung wurde nicht abgefragt, da zu wenige Aufgaben durchgeführt wurden, als dass die Testpersonen dieses Kriterium hätten beurteilen können. Teilweise fließt die Steuerbarkeit aber auch in die verständliche Bedienung mit ein.
Erwartungskonformität	Vorhersehbares Programmverhalten
Fehlertoleranz	Information über zulässige Eingaben (Wurde nur eingeschränkt abgefragt, da den Testpersonen keine Zeit zum Korrigieren von Fehlern gegeben wurde.)
Individualisierbarkeit	Wurde nicht abgefragt, da im Rahmen des Tests keine Individualisierung vorgenommen wurde.
Lernförderlichkeit	Hilfreiche Anleitung

Tabelle 16: Zuordnung Fragen zu Anforderungen.

Nachfolgend wird für jede Anforderung betrachtet, wie diese von den Testpersonen bewertet wurde. Zusätzlich wird erläutert, welche Maßnahmen bei der Realisierung des Werkzeuges ergriffen wurden, um der Anforderung gerecht zu werden.

- Um die **Aufgabenangemessenheit** zu erreichen, wurden die Prozesse des Anpassungswerkzeuges von Prozessexperten beschrieben. Auf diesen Beschreibungen basiert das Werkzeug. Dadurch wurde sichergestellt, dass das Vorgehen von Prozessexperten abgebildet wird. Die Testpersonen haben die hierzu gestellten Fragen folgendermaßen beantwortet (Es war möglich von 1 bis 6 zu bewerten, wobei 6 die beste Bewertung war):
  - Auf Anforderung zugeschnitten: Mittelwert: 5,56, Standardabweichung: 0,512
  - Aufwand angemessen: Mittelwert: 5,69, Standardabweichung: 0,479.
  - Zusätzliche Funktionen gewünscht: Mittelwert: 5,25, Standardabweichung:1,065.

  Somit lässt sich sagen, dass die Testpersonen die Aufgabenangemessenheit positiv bewertet haben und die Anforderung erfüllt ist.
- **Selbstbeschreibungsfähigkeit**: Das Anpassungswerkzeug erläutert dem Benutzer, was zum Durchführen der Aufgabe zu tun ist. Die Erläuterungen werden aus den Beschreibungen der Prozessexperten gewonnen. Die Benutzer wurden hinsichtlich der verständlichen Bedienung des Programms befragt und haben diese positiv bewertet. Die Testpersonen haben die hierzu gestellten Fragen folgendermaßen bewertet:
  - Übersichtliche Informationsanordnung: Mittelwert: 5,69, Standardabweichung: 0,602.
  - Verständliche Bedienung: Mittelwert: 5,38 Standardabweichung: 0,719.

  Somit lässt sich sagen, dass die Testpersonen die Selbstbeschreibungsfähigkeit ebenfalls positiv bewertet haben. Die entsprechende Anforderung ist erfüllt.
- **Steuerbarkeit**: Mit „Vor“ und „Zurück“ Buttons kann ein Benutzer im Wizard navigieren. Außerdem wird angezeigt, wie viele Unterschritte bereits durchgeführt wurden und wie viele noch durchzuführen sind. Dadurch wurde die Anforderung realisiert. (Im Fragebogen gab es keine Frage zu dieser Anforderung.)
- **Erwartungskonformität**: Der Ablauf des Wizards orientiert sich an der Beschreibung der Prozessexperten und bildet deren Erwartungen ab. Die Testpersonen haben die hierzu gestellte Frage folgendermaßen bewertet:
  - Vorhersehbares Programmverhalten: Mittelwert: 5,13, Standardabweichung: 1,015.

  Auch die Erwartungskonformität wurde positiv beurteilt und ist somit erfüllt.
- **Fehlertoleranz**: Wo immer möglich, werden Eingaben der Benutzer direkt bei der Eingabe auf Fehler überprüft. Außerdem ermöglicht es eine

Vorschaufunktion, die Auswirkungen von Änderungen zu überprüfen, bevor sie endgültig ausgeführt werden. Falls dabei festgestellt wird, dass Fehler gemacht wurden, können Änderungen verworfen werden. Die Testpersonen haben die hierzu gestellte Frage folgendermaßen bewertet:

- Info über zulässige Eingaben Mittelwert: 4,75, Standardabweichung:.0,931.

Die Fehlertoleranz wurde ebenfalls positiv beurteilt und ist somit erfüllt.

- **Individualisierbarkeit**: Über Benutzereinstellungen können Benutzer den Wizard an ihren aktuellen Kenntnisstand anpassen. Dazu sind zwei Modi vorgesehen: Der Experten- und der Laienmodus. Außerdem kann für die Anzeige der Benutzeroberfläche zwischen deutscher und englischer Sprache gewählt werden. Dadurch wurde eine Individualisierbarkeit erreicht. (Im Fragebogen gab es keine Frage zu dieser Anforderung.)
- **Lernförderlichkeit**: Der Wizard leitet den Benutzer beim Erlernen der Nutzung an, indem er eine detaillierte Anleitung anbietet, wie vorzugehen ist. Die Testpersonen haben die hierzu gestellte Frage folgendermaßen bewertet:

  - Hilfreiche Anleitung: Mittelwert: 5,75, Standardabweichung:.0,577.

  Die Anleitung wurde von den Testpersonen positiv bewertet. Da alle Personen anhand des Werkzeuges und der Anleitung innerhalb kürzester Zeit in der Lage waren, die geforderten Aufgaben durchzuführen, lässt sich auch die Anforderung der Lernförderlichkeit als erfüllt betrachten.

Somit lässt sich sagen, dass zum einen alle Anforderungen bei der Entwicklung des Werkzeuges zur Unterstützung von Anpassungsprozessen berücksichtigt wurden. Zum anderen wurden die zum Überprüfen gestellten Fragen von den Testpersonen positiv beantwortet.

8.2.5 Zusammenfassung der Ergebnisse

Als Endergebnis für die Evaluation des Anpassungswerkzeuges lässt sich sagen, dass Personen mit dem Anpassungswerkzeug deutlich schneller arbeiten als mit dem Composer. Außerdem ist bei ihnen die Bewertung des verwendeten Werkzeuges hinsichtlich Aufgabenangemessenheit besser. Hinsichtlich der Transparenz des Programms, sowie der Informationsanordnung auf dem Bildschirm wurden beide Werkzeuge positiv bewertet. Auch bei der Fehleranzahl war kein großer Unterschied zwischen den Werkzeugen festzustellen. Somit lässt sich sagen, dass das Anpassungswerkzeug beim Test in Teilen genauso gut wie das Vergleichswerkzeug abschneidet, in Teilen sogar besser.

Insgesamt sind die Ergebnisse für das Anpassungswerkzeug unabhängig vom Vergleichswerkzeug sehr positiv, da die für die Durchführung der Anpassungen benötigte Zeit sehr gering war, sehr wenige Fehler gemacht wurden und die Bewertung des Werkzeuges insgesamt sehr positiv ausgefallen ist.

Somit lässt sich sagen, dass auch das zweite Hauptziel der vorliegenden Arbeit erfüllt ist: Basierend auf dem in dieser Arbeit vorgestellten Ansatz lassen sich Wizards

erstellen, die Anwender geeignet bei der Durchführung von Anpassungsprozessen unterstützen.

8.3 Zusammenfassung

In diesem Kapitel wurde der letzte Beitrag dieser Dissertation ausführlich vorgestellt: Die Evaluation des vorgestellten Konzeptes, sowie die Evaluation eines mit diesem Konzept erstellten Werkzeuges zur Unterstützung von Anpassungsprozessen.

Dabei wurden zwei Annahmen überprüft:

1. Durch einen Benutzertest wurde nachgewiesen, dass das vorgestellte Konzept und die dafür entwickelten Werkzeuge PIT und WGT sich für die Prozessbeschreibung durch Prozessexperten eignen, unabhängig von deren Vorkenntnissen in Software-Design und -Entwicklung.
2. In einem zweiten Benutzertest wurde gezeigt, dass der mit dem vorgestellten Konzept entwickelte Wizard zur Unterstützung von Anpassungsprozessen die Durchführung der unterstützten Prozesse verbessert. Dazu wurden die Faktoren Zeit, Fehleranzahl und Benutzerzufriedenheit für das Anpassungswerkzeug und den Netscape Composer verglichen. Dabei hat sich gezeigt, dass die Anwender mit dem Anpassungswerkzeug zwar nur geringfügig weniger Fehler als mit dem Netscape Composer machen, aber deutlich schneller sind. Außerdem ist beim Anpassungswerkzeug die Benutzerzufriedenheit deutlich höher. Somit lässt sich sagen, dass das Anpassungswerkzeug die Prozessdurchführung geeignet unterstützt.

Abschließend lässt sich somit sagen, dass sich im Rahmen der Evaluation gezeigt hat, dass beide Annahmen erfüllt sind und somit beide Hauptziele der vorliegenden Arbeit erreicht wurden.

9 Zusammenfassung und Ausblick

9.1 Zusammenfassung

Ein Hauptziel der vorliegenden Arbeit war es, ein Konzept und ein Framework für einen einfach handhabbaren Formalismus zur Beschreibung von Anpassungsprozessen zu erstellen. Die mittels dieses Formalismus erstellten Prozessbeschreibungen sollten als unmittelbare Basis für das zweite Hauptziel dieser Arbeit dienen: Die Erstellung eines Werkzeuges zur Unterstützung der Durchführung der Anpassungsprozesse auch für Personen, die nicht Experten in der Prozessdurchführung sind.

Um dieses Ziel zu erreichen, mussten eine Reihe von Anforderungen berücksichtigt werden: Es sollte ein einfach erlernbarer Formalismus entwickelt werden, der es Prozessexperten ohne Modellierungskenntnissen erlaubt, ihr Wissen zu Anpassungsprozessen formalisiert beschreiben zu können. Das so zur Verfügung gestellte Wissen sollte als Basis für die Entwicklung eines Werkzeuges zur Unterstützung von Anpassungsprozessen dienen. Dieses Werkzeug soll Laien wie Experten durch Anpassungsprozesse führen und ihnen eine schnellere und weniger fehleranfällige Anpassung von Lernressourcen ermöglichen.

Um den Anforderungen gerecht zu werden, wurde ein Konzept entwickelt, das es Prozessexperten ermöglicht, ihr Wissen über Anpassungsprozesse in Form von Pattern-basierten Prozessbeschreibungen verfügbar zu machen. Die zur Prozessbeschreibung verwendeten Patterns beschreiben in natürlicher Sprache, wie Anpassungsprozesse durchzuführen sind. Mit Hilfe des Eingabewerkzeuges PIT können Prozessbeschreibungen in Pattern-Form erfasst werden. Die Prozessbeschreibungen bilden die Basis für die automatische Erzeugung eines lauffähigen prototypischen Wizards.

Der so entstandene Wizard verdeutlicht die Prozessabläufe in Anpassungsprozessen und führt andere Personen schrittweise durch die Prozesse. Dadurch dient er auch als wichtige Kommunikationsgrundlage zwischen Prozessexperten, Software-Designern und -Entwicklern. Außerdem ist er so gestaltet, dass er sich durch ein Hinzufügen automatisierter Funktionalitäten zu einem einsatzfähigen Werkzeug zur Prozessunterstützung für Anpassungsprozesse erweitern lässt. Dieses Werkzeug setzt das zweite Hauptziel der vorliegenden Arbeit um: Es unterstützt Laien und Experten bei der Durchführung von Anpassungsprozessen und bietet die Möglichkeit, diese Prozesse schneller und weniger fehleranfällig durchzuführen.

Damit liefert die vorliegende Arbeit folgende wissenschaftliche Beiträge, die es ermöglicht haben, das erste Hauptziel zu erreichen:

1. Es wurde ein *Konzept zur Einbindung von Prozessexperten in die Erstellung von Software zur Unterstützung von Anpassungsprozessen* entwickelt, das einfach handhabbar ist und alle für die Durchführung der Anpassungsprozesse notwendigen Informationen abbildet.
2. Im Rahmen des Konzeptes wurde ein *Verfahren zur Beschreibung von Anpassungsprozessen* entwickelt. Dieses kann sowohl von Personen mit, als auch von Personen ohne Kenntnisse in der Prozessmodellierung angewandt werden, um

Anpassungsprozesse so zu beschreiben, dass die Prozessbeschreibungen alle für die Entwicklung eines Unterstützungswerkzeuges notwendigen Informationen enthalten.

3. Es wurde eine *für die Beschreibung von Prozessabläufen in Anpassungsprozessen geeignete Pattern-Notation* entwickelt. Diese basiert auf existierenden Pattern-Formalismen und erweitert diese in der Form, dass sie alle für die Beschreibung von Anpassungsprozessen benötigten Elemente enthält.
4. Für die Erstellung von Prozessbeschreibungen in Pattern-Form und zur automatisierten Wizard-Erzeugung basierend auf den Prozessbeschreibungen wurden *zwei Werkzeuge entworfen und anschließend umgesetzt*. Diese Werkzeuge unterstützen Prozessexperten bei der Erstellung von Prozessbeschreibungen im verlangten Format und ermöglichen es, ohne Programmierkenntnisse einen prototypischen Wizard zu erstellen.
5. Es wurde ein *Konzept zur Erweiterung automatisch generierter Wizards* entwickelt. Dieses ermöglicht es, einen automatisch generierten Wizard für Anpassungsprozesse in ein voll funktionsfähiges Werkzeug zur Unterstützung von Anpassungsprozessen zu überführen.
6. Um das Konzept und die Werkzeuge zur Prozessbeschreibung und Wizard-Generierung zu testen, wurde basierend auf dem Konzept mit Hilfe der Werkzeuge *ein voll funktionsfähiges Unterstützungswerkzeug für Anpassungsprozesse entwickelt*.

Das zweite Hauptziel wurde durch folgenden Schritt erreicht:

7. Zur Erweiterung des generierten Unterstützungswerkzeuges wurden *automatisierte Funktionen zur Anpassung von Lernressourcen entworfen und umgesetzt* und im Rahmen des Projektes Content Sharing erprobt.

Zur Überprüfung der Zielerreichung wurden zwei Benutzertests durchgeführt:

8. Zum einen, um das Konzept und die Werkzeuge zur Prozessbeschreibung und Wizard-Generierung von verschiedenen Personengruppen testen zu lassen,
9. und zum anderen, um das Anpassungswerkzeug zu evaluieren.

Darüber hinaus wurde

10. anhand weiterer Prozesse getestet, *ob das in der Motivation genannte Problem unzureichender Prozessunterstützung auch bei Software zur Unterstützung anderer Prozesse zu finden ist*, und ob *sich das Konzept zur Prozessbeschreibung und Wizard-Generierung auch für andere Prozesse einsetzen lässt*.

9.2 Ausblick

Die vorliegende Arbeit stellt ein Konzept zur Entwicklung eines Wizards zur Unterstützung von Anpassungsprozessen vor. Es baut auf Pattern-basierten Prozessbeschreibungen auf. Im Rahmen der Arbeit wurden zwei Werkzeuge entwickelt, die dieses Konzept umsetzen. Sie ermöglichen die stärkere Einbindung von Prozessexperten in die

Entwicklung eines Werkzeugs zur Unterstützung von Anpassungsprozessen. So lässt sich erreichen, dass das Wissen dieser Personen besser in Entwicklungsprojekten berücksichtigt wird und auch anderen Personen zur Verfügung gestellt werden kann.

Ausgehend von dieser Arbeit gibt es noch einige weitere Aspekte, die als Ansatzpunkt für zukünftige Arbeiten interessant sind:

- Im Rahmen dieser Dissertation wurde die Anwendbarkeit des vorgestellten Konzeptes exemplarisch für einige Prozessarten untersucht. Dabei hat sich gezeigt, dass das Konzept auch für diese Prozesse geeignet ist.

  Für eine weitergehende Arbeit wäre es wünschenswert, detailliert zu untersuchen, ob es weitere Prozesse gibt, auf die sich das hier vorgestellte Konzept anwenden lässt. Dabei wäre zu berücksichtigen, ob es Kriterien gibt, die Prozesse erfüllen müssen, damit sie berücksichtigt werden können. Außerdem wäre zu untersuchen, welche Merkmale Prozesse aufweisen, für die das vorgestellte Konzept geeignet ist.

- Patterns erläutern nicht nur, wie ein Problem zu lösen ist, sie geben auch den Kontext an, in dem das jeweilige Problem auftreten kann und die Vorbedingungen, die erfüllt sein müssen, damit die Lösung des Patterns angewandt werden kann. Sie benennen die Anforderungen, denen ein Pattern genügen muss und erläutern die Konsequenzen, die sich aus der Patternausführung ergeben.

  Aktuell sind diese Elemente zwar in den Prozessbeschreibungspatterns enthalten und sie werden im Wizard benannt, aber ihre Verwendung ist nicht exakt spezifiziert. Für die Zukunft wäre es interessant zu untersuchen, wie die exakte Auswirkung dieser Elemente auf den auszuführenden Anpassungsprozess aussieht und inwiefern sie sich stärker in die Unterstützung einbinden lassen. So wäre es beispielsweise vorstellbar, dass Anwender Informationen zu ihrem Kontext und ihre Rahmenparameter angeben können und dies bei der Auswahl von Anpassungsprozessen berücksichtigt wird. Die Auswertung der Konsequenzen könnte für die Ermittlung von Folgeprozessen genutzt werden.

- Es existiert bereits ein Reihe von Patterns, die Prozesse aus den verschiedensten Bereichen beschreiben. Beispiele sind die Security Patterns aus [Sc+06] oder die Patterns zu Web Usability [Gr03]. In diesem Zusammenhang wäre es interessant zu erforschen, ob sich diese Patterns (teil-)automatisiert in die hier eingesetzten Patterns zur Prozessbeschreibung überführen lassen. Dazu müsste festgestellt werden, welche Elemente der verschiedenen Pattern-Notationen auf die Elemente der Pattern-Notation für Prozessbeschreibungen abgebildet werden können und wo keine Abbildung möglich ist. (Für einige Security Patterns wurde das bereits in Abschnitt 7.4.1 in dieser Arbeit untersucht.) Es müsste überlegt werden, welche Informationen ein Anwender zur Verfügung stellen müsste, damit ein Werkzeug große Teile der Transformation automatisiert vornehmen könnte.

- Das Werkzeug CoPE (Collaborative Pattern Editor) von Schobert und Schümmer [SS06] ermöglicht Pattern-Autoren, die Texte eines Patterns und eine grafische Visualisierung der Zusammenhänge mehrerer Patterns, die sogenannten Pattern-Landkarten, zu erstellen. PIT dient ebenfalls der Eingabe von Patterns.

Außerdem visualisiert es die Zusammenhänge der Prozessschritte und Unterschritte. Bisher werden aber keine Zusammenhänge zwischen Patterns dargestellt. Für die Zukunft wäre eine Weiterentwicklung von PIT in diese Richtung anzustreben. Dadurch könnten Pattern-Landkarten erstellt werden, die es Entwicklern erleichtern würden, die Zusammenhänge zwischen verschiedenen Anpassungsprozessen zu überblicken und diese falls nötig, bei der Entwicklung zu berücksichtigen.

- In der vorliegenden Arbeit wurden 15 Anpassungsarten identifiziert. Diese wurden jedoch nicht alle im Rahmen des Anpassungswerkzeuges realisiert. Insbesondere die inhaltlichen Anpassungsprozesse sind fehleranfällig und zeitintensiv. Allerdings gehören viele dieser Prozesse zu den unterstrukturierten, erfahrungsbasierten Anpassungsprozessen und können somit schwer automatisiert unterstützt werden. Hier ist es wünschenswert, die Prozesse noch detaillierter zu analysieren und Möglichkeiten für eine geeignete Unterstützung zu ermitteln.
- Derzeit spielt die Service-orientierte Architektur (SOA) von Anwendungen eine große Rolle in der Forschung. Auch für das hier vorgestellt Konzept zur Automatisierung der mit WGT generierten Wizards wäre es denkbar, Funktionen durch Webservices [Be08] zu realisieren (vergleiche Abschnitt 6.3). So könnte man beispielsweise den bereits vorgestellten Prozess zur Reisebuchung über diese Services realisieren. Auch für die betrachteten Anpassungsprozesse wären diverse Webservices, beispielsweise zur Bildbearbeitung oder zur Übersetzung denkbar. Für die Zukunft wäre daher zu untersuchen, in welchen Fällen Webservices zur Erweiterung der automatisch erzeugten Wizards geeignet sind und wie diese sinnvoll eingebunden werden können.
- Weiterhin wäre eine Weiterentwicklung von PIT und WGT wünschenswert. Dabei sollte insbesondere berücksichtigt werden, dass sich derzeit nicht alle Prozessabläufe auf eine einfache Art und Weise beschreiben lassen. So ist es beispielsweise nur umständlich möglich, Verzweigungen zu beschreiben, die in mehr als nur zwei mögliche Äste verzweigen. Diesbezüglich scheint es für die Zukunft sinnvoll, zu analysieren, wie oft welche bisher nicht in PIT umgesetzten Abläufe vorkommen und wie sich diese beschreiben lassen. Abhängig vom Aufwand der Entwicklung und dem potentiellem Nutzen sollten PIT und WGT entsprechend erweitert werden.

Literaturverzeichnis

[Aa+02] van der Aalst, W.M.P, ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. In: Distributed and Parallel Databases, 14(3), Seite 5-51, 2003.

[Ab+02] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile Software Development Methods: Review and Analysis. VTT Publications, 2002.

[Ac98] Achour, C.B.: Guiding Scenario Authoring. In: Proceedings of the Eighth European Japanese Conference Information Modeling and Knowledge Bases, Seite 181-200, 1998.

[AH02] van der Aalst, W.M.P., van Hee, K.: Workflow Management: Models, Methods, Systems. The MIT Press, 2002.

[AHD05] van der Aalst, W.M.P., ter Hofstede, A.H.M., Dumas, M.: Patterns of Process Modeling. In: Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Process-Aware Information Systems – Bridging People and Software through Process Technology, Wiley-Interscience, Seite 179-203, 2005.

[Al64] Alexander, C.: Notes On The Synthesis Of Form. Oxford University Press, 1964.

[Al+77] Alexander, C., Ishikawa S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language – Towns, Buildings, Construction. Oxford University Press, 1977.

[Al79] Alexander, C.: The Timeless Way of Building. Oxford University Press, 1979.

[AS07] Amelunxen, C., Schürr, A.: Formalizing Model Transformation Rules for UML / MOF 2. In: IET Software Journal, Academic Press, 2007.

[At+03] Ateyeh, K., Klein, M., König-Ries, B., Mülle, J.: A Practical Strategy for the Modularization of Courseware. In: Proceedings of Professionelles Wissensmanagement - Erfahrungen und Visionen Workshop on Adaptive E-Learning and Metadata, 2003.

[Ba98] Balzert, H.: Lehrbuch der Software-Technik. Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Spektrum Akademischer Verlag, 1998.

[Ba04] Baumgartner, P.: The ROI Paradox. Keynote Präsentation bei der Gesellschaft für Medien in den Wissenschaften Konferenz, 2004.

[BC87] Beck, K.; Cunningham, W.: Using Pattern Languages for Object-Oriented Programs. In: OOPSLA-87 Workshop on the Specification and Design for Object-Oriented Programming, 1987.

[Be99] Beck, K. Extreme Programming Explained. Addison-Wesley, 1999.

[Be+01] Beck, K., et al.: Manifesto for Agile Software Development. Agile Alliance, 2001.

[Be+05] Bergsträßer, S., Zimmermann, B., Meyer, M., Rensing, C., Faatz, A., Hildebrandt, T., Steinmetz, R.: Re-Purposing: Motivation, Related Work, and Building Blocks. Technical Report KOM-TR-2005-03, Technische Universität Darmstadt, 2005.

[Be+06] Bergsträßer, S., Faatz, A., Rensing, C., Steinmetz, R.: A semantic content representation supporting re-purposing of learning resources. In: Proceedings of I-KNOW 2006, 2006.

[Be08] Berbner, R.: Dienstgüteunterstützung für Service-orientierte Workflows. Books on Demand Gmbh, 2008.

[BGM99] Braga, R.T.V.; Germano, F.S.R.; Masiero, P.C.: A Pattern Language for Business Resource Management. In: Proceedings of the 6th Conference on Pattern Languages of Programs, Seite 1-34, 1999.

[BGM04] Braga, R.T. V.; Germano, F.S.R.; Masiero, P.C.: System Development using a Pattern Language-based Tool. In: Proceedings of the 6th International Conference on Enterprise Information Systems, ICEIS 2004, Seite 155-162, 2004.

[BM02] Braga, R.T.V.; Masiero, P.C.: A Process for Framework Construction Based on a Pattern Language. In: Proceedings of the 26th Annual International Computer Software and Applications Conference, Seite 615-620, 2002.

[BM03] Braga, R.T.V., Masiero, P.C.: Building a Wizard for Framework Instantiation Based on a Pattern Language. In: OOIS 2003, Lecture Notes in Computer Science, LNCS 2817, Springer, Seite 95–106, 2003.

[Bo79] Boehm, B.W.: Guidelines for Verifying and Validating Software Requirements and Design Specification. In: Proceedings of EURO IFIP 79, Seite 711-719, 1979.

[Bo88] Boehm, B.W.: A Spiral Model of Software Development and Enhancement. In: IEEE Computer, Vol. 21, Ausgabe 5, Seite 61-72, 1988.

[Bo01] Borchers, J.O.: A Pattern Approach to Interaction Design. John Wiley & Sons, 2001.

[Bo05] Bortz, J.: Statistik für Human- und Sozialwissenschaftler, 6. Auflage. Springer, 2005.

[BR02] Bergner, K., Rausch, A.: Process Pattern Test Suite Bootstrapping. White Paper an der TU München, 2002.

[Br07] Brinkkötter, C.J.: Rating im Mittelstand. Eine empirische Analyse qualitativer Kriterien. Eul, 2007.

[Bu+96a] Budinsky, F., Finnie, M., Yu, P., Vlissides, J.: Automatic Code Generation from Design Patterns. In: IBM Systems Journal archive, Volume 35, Issue 2, Seite 151-171, 1996.

[Bu+96b] Buschmann, F.; Meunier, R; Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture – A System of Patterns. Wiley and Sons, 1996.

[Bu01] Bulterman, D.C.A.: SMIL 2.0: Repurposing Broadcast Content for the Web. EBU Technical Review, No. 287, 2001.

[Bu03] Bulterman, D.C.A.: Using SMIL to encode interactive, peer-level multimedia annotations. In: Proceedings of the 2003 ACM symposium on Document engineering. 2003.

[Bu08] Buchholz, S.: Evaluation eines Software-Prototypen zur Prozessbeschreibung als Basis für einen Software-Assistent (zur Prozessunterstützung). Studienarbeit an der Technischen Universität Darmstadt, 2008.

[Ch07] Charfi, A.: Aspect Oriented Workflow Languages: AO4BPEL and Applications. Dissertation an der Technischen Universität Darmstadt, 2007.

[Ch+07] Charfi, A., Berbner, R., Mezini, M., Steinmetz, R.: On the Management Requirements of Web Service Compositions. In: Proceedings of the 2nd ECOWS Workshop on Emerging Web Services Technology (WEWST), 2007.

[CM07] Charfi, A., Mezini, M.: AO4BPEL: An Aspect-oriented Extension to BPEL. In: World Wibe Web Journal, 10 (3), Seite 309-344, 2007.

[Co91] Coplien, J.O.: Advanced C++ Programming Styles and idioms. Addison-Wesley, 1991.

[Co92] Coad, P.: Object-Oriented Patterns. Communications of the ACM, 35(9), Seite 152-159, 1992.

[Co94] Coplien, J.O.: A Development Process Generative Pattern Language. In: Proceedings of Pattern Languages of Programs 1994, 1994.

[Co96] Cockburn, A.: A Medical Catalog of Project Management Patterns, In: Proceedings of Pattern Languages of Programs 1996, 1996.

[Co02] Coldewey, J.: Multi-Kulti - Ein Überblick über die agile Entwicklung. In: Objektspektrum 1/2002, 2002.

[CS89] Conell, J.L., Shafer, L.B.: Structured rapid Prototyping. Prentice Hall International, Yourdon Press, 1989.

[CS93] Claus, V., Schwill, A.: Duden Informatik, Ein Sachlexikon für Studium und Praxis, Dudenverlag, 1993.

[DH03] Duval, E., Hodgins, W.: A LOM research agenda. In: Proceedings of the twelfth international conference on world wide web, 2003.

[Di95] DIN EN ISO 9241: Ergonomics of Human System Interaction, 1995.

[Di06] DIN EN ISO 9241 Part 110: Dialogue principles. In: DIN EN ISO 9241: Ergonomics of Human System Interaction, 2006.

[EC00] Eisenecker, U.W., Czarnecki, K.: Generative Programming, Addison-Wesley, 2000.

[EH98] Effelsberg, W., Hornung, C.: Lehre und Lernen im Internet. In: Informationstechnik und Technische Informatik : it +ti, Band 40, Heft 2, 1998.

[EYG97] Eden, A.H., Yehudai, A., Gil, J.: Precise Specification and Automatic Application of Design Patterns. In: Proceedings of the 12th Annual Conference on Automated Software Engineering, 1997.

[ES01] El Saddik, A.: Interactive Multimedia Learning: Shared Reusable Visualizationbased Modules. Springer, 2001.

[Fa04] Faatz, A.: Ein Verfahren zur Anreicherung fachgebietsspezifischer Ontologien durch Begriffsvorschläge. Dissertation an der Technischen Universität Darmstadt, 2004.

[Fl84] Floyd, C.: A systematic look at prototyping. In: Budde, R., Kuhlenkanp, K., Matthiassen, L., Züllighoven, H.: Approaches to prototyping. Springer. 1984.

[Fo96] Fowler, M.: Analysis Patterns: Reusable Object Models. Addison-Wesley, 1996.

[FWB06] Folmer, E.,van Welie M., Bosch, J.: Bridging patterns: An approach to bridge gaps between SE and HCI. In: Information and Software Technology, 48(2), Seite 69-98, 2006.

[Ga+95] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[Gh+08] Ghidini, C., Rospocher, M., Serafini, L., Kump, B, Pammer, V., Faatz, A., Zinnen, A., Guss, J., Lindstaedt, S.: Collaborative Knowledge Engineering via Semantic MediaWiki. In: Proceedings of I-SEMANTICS 2008, 2008.

[Gna+01] Gnatz, M., Marschall, F., Popp, G., Rausch, A., Schwerin, W.: Towards a Living Software Development Process based on Process Patterns. In: Proceedings of the Eight European Workshop on Software Process Technology 2001. Springer. 2001

[Go+08] Godehardt, E., Doehring, M., Faatz, A., Goertz. M.: KAJAL – An Algorithm for Workflow Navigation in Informal Learning. In: Proceedings of I-KNOW 2008, 2008.

[Gr03] Graham, I.: A Pattern Language for Web Usability, Addison-Wesley, 2003.

[Gr04] Gröne, B.: Konzeptionelle Patterns und ihre Darstellung. Dissertation an der Universität Potsdam, 2004.

[Ha99] Harrison, N.B.: The Language of Shepherding - A Pattern Language for Shepherds and Sheep. In: Proceedings of Pattern Languages of Programs 1999, 1999.

[Ha01] Hahsler, M.: Analyse Patterns im Softwareentwicklungsprozeß. Dissertation an der WU Wien, 2001.

[Ha05] Hagen, M.: Definition einer Sprache zur Beschreibung von Prozessmustern zur Unterstützung agiler Softwareentwicklungsprozesse. Dissertation an der Universität Leipzig, 2005.

[He92] Heinen, E.: Einführung in die Betriebswirtschaftslehre. 9., verbesserte Auflage, Gabler, 1992.

[He03] Hesse, W.: Dinosaur Meets Archaeopteryx? Seven Theses on Rational's Unified Process (RUP)? In: Software and Systems Modeling (SoSyM) Vol. 2. No. 4, Seite 240-247, 2003.

[HG04] Hagen, M., Gruhn, V.: PROPEL - Eine Sprache zur Beschreibung von Prozessmustern. In: Proceedings zu Modellierung 2004, 2004.

[HKM00] Holl, A.; Krach, T.; Mnich, R.: Geschäftsprozessdekomposition und Gestalttheorie. In: Britzelmaier, B. et al. (Hrsg.): Information als Erfolgsfaktor. Teubner, Seite 197-209, 2000.

[Ho06] Horneff, A.: Semi-Automatische Erstellung von Druckversionen von E-Learning Inhalten am Beispiel von HTML Kursen. Studienarbeit an der Technischen Universität Darmstadt, 2006.

[Ho07] Horneff, A.: Automatisierte Wizard-Generierung basierend auf Adaptation-Patterns. Diplomarbeit an der Technischen Universität Darmstadt, 2007.

[Hö+05] Hörmann, S., Hildebrandt, T., Rensing, C., Steinmetz, R.: ResourceCenter - A Digital Learning Object Repository with an Integrated Authoring Tool Set. In: Proceedings of EdMedia 2005, Seite 3453-3460, 2005.

[Hö05] Hörmann, S.: Wiederverwendung von Digitalen Lernobjekten in einem auf Aggregation basierenden Autorenprozess. Dissertation an der Technischen Universität Darmstadt, 2005.

[HRS05] Hörmann, S., Rensing, C., Steinmetz, R.: Wiederverwendung von Lernressourcen mittels Authoring by Aggregation im ResourceCenter. In: Proceedings of DeLFI 2005, Seite 153-164, 2005.

[HSW04] Haake, J., Schwabe, G., Wessner, M. (Hrsg.): CSCL-Kompendium. Lehr- und Handbuch zum computerunterstützten kooperativen Lernen. Oldenbourg Verlag, 2004.

[HW99] Haake, J. M., Wang, W.: Flexible Support for Business Processes: Extending Cooperative Hypermedia with Process Support. In: Information and Software Technology 41 (6), Seite 355-366, 1999.

[Ie02] IEEE Learning Technology Standards Committee: IEEE Standard for Learning Object Metadata 1484.12.1, 2002.

[IRC03] Ip, A., Radford, A., Canale, E.: Overcoming the Presentation Mosaic Effect of Multi-Use Sharable Content Objects. In: Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE), volume 1, Seite 256-262, 2003.

[Ja02] Jaquith, A.: The Security of Applications: Not All Are Created Equal. Research Report, @Stake, 2002.

[JB96] Jablonski, S., Bussler, C.: Workflow Management Modeling: Concepts, Architecture and Implementation. International Thomson Computer Press, 1996.

[Ka+03] Klein, M., Ateyeh, K., König-Ries, B., Mülle, J.: Creating, Filling, and Using a Repository of Reusable Objects for Database Courses. In: Proceedings of Datenbanksysteme für Business, Technologie und Web Workshop on Datenbanken und E-Learning, 2003.

[KC97] Kerth, N.L., Cunningham, W.: Using Patterns to Improve Our Architectural Vision. In: IEEE Software 14(1), Seite 53-59. 1997.

[Kr99] Kruchten, P.: Der Rational Unified Process . Eine Einführung. Addison-Wesley 1999.

[La06] Lassmann, W.: Wirtschaftsinformatik - Nachschlagewerk für Studium und Praxis. Gabler, 2006.

[LD02] Lejk, M., Deeks, D.: An Introduction to Systems Analysis Techniques, 2. Ausgabe. Addison-Wesley, 2002.

[Le94] Lea, D.: Christopher Alexander: An Introduction for Object-Oriented Designers. ACM Software Engineering Notes. 1994.

[LT75] Linstone, H., Turoff, M.: The Delphi Method: Techniques and Applications, Addison-Wesley, 1975.

[Mc00] McConnell, S.: 10 Best Influences on Software Engineering. In: Column From the Editor in IEEE Software, January/February 2000. Seite 10-17, 2000.

[Me+06] Meyer, M., Hildebrandt, T., Rensing, C., Steinmetz, R.: Requirements and Architecture for a Multimedia Content Re-purposing Framework. In: Proceedings of the First European Conference Technology Enhanced Learning (EC-TEL), 2006.

[Me+07] Metzger, M., Zimmermann, B., Rensing, C., Steinmetz, R.: Automating Layout Adaptation of Textual-based E-Learning Content. In: Proceedings of EdMedia, 2007.

[Me08] Meyer, M.: Modularization and Multi-Granularity - Reuse of Learning Resources. Dissertation an der Technischen Universität Darmstadt, 2008.

[Mi56] Miller, G.A.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review, vol. 63, Issue 2, Seite 81-97, 1956.

[MRS06] Mühlhäuser, M., Rößling, G., Steinmetz, R.(Hrsg.): DeLFI 2006: 4. E-Learning Fachtagung Informatik. Lecture Notes in Informatics - Gesellschaft für Informatik, 2006.

[MRS07] Meyer, M., Rensing, C., Steinmetz, R.: Using Community-Generated Contents as a Substitute Corpus for Metadata Generation. In: International Journal of Advanced Media and Communication (IJAMC), vol. 2, no. 1, Seite 59-72, 2008.

[Mü07] Müller, S.: Modellbasierte IT-Unterstützung von wissensintensiven Prozessen - Dargestellt am Beispiel medizinischer Forschungsprozesse. Dissertation an der Universität Erlangen - Nürnberg, 2007.

[NGS98] Nurcan, S., Grosz, G., Souveyet, C.: Describing business processes with a guided use case approach. In: Proceedings of the 10th International Conference on Advanced Information Systems Engineering (CAiSE'98), Seite 339-361, 1998.

[OSS04] Obrenovic, Z., Starcevic, D., Selic, B.: A Model-Driven Approach to Content Repurposing. In: IEEE MultiMedia, 11 (1), Seite 62-71, 2004.

[PH95] Pohl, K., Haumer, P.: HYDRA: A Hypertext Model for Structuring Informal Requirements Representations. In: Proceedings of the international Workshop on Requirements Engineering: Foundation of Software Quality, 1995.

[Re+05] Rensing, C. Bergsträßer, S., Hildebrandt, T., Meyer, M., Zimmermann, B., Faatz, A., Lehmann, L., Steinmetz, R.: Re-Use, Re-Authoring, and Re-Purposing of Learning Resources - Definitions and Examples. Technical Report KOM-TR-2005-02, Technische Universität Darmstadt, 2005.

[Re+08] Rensing, C., Zimmermann, B., Meyer, M., Lehmann, L., Steinmetz, R.: Wiederverwendung von multimedialen Lernressourcen im Re-Purposing und Authoring by Aggregation. In: Loos, P., Zimmermann, V., Chikova, P.: Prozessorientiertes Authoring Management: Methoden, Werkzeuge und Anwendungsbeispiele für die Erstellung von Lerninhalten, Logos Verlag, Seite 19-40, 2008.

[RFP05] Rust, M., Flach, G., Petersdorff-Campe, R. : WIESELfederation – Content Sharing Ansatz im Rahmen eines offenen Verbundes von Learning Object Repositories. In: Proceedings of DeLFI 2005, Seite 141-152, 2005.

[Ri98] Rising, L.: The Patterns Handbook: Techniques, Strategies, and Applications. Cambridge University Press, 1998.

[RJ00] Rising, L., Janoff, N.: The Scrum Software Development Process for Small Teams. In: IEEE Software, July/August 2000. Seite 26-32, 2000.

[Ro87] Royce, W.W.: Managing the Development of Large Software Systems. In: Proceedings of the 9th international conference on Software Engineering, Seite 328-338, 1987.

[Ro01] Robertson, S.: Requirements trawling: techniques for discovering requirements. In: International Journal of Human-Computer Studies, Volume 55, Number 4, October 2001, Seite 405-421. 2001.

[Ro04] Robertson, S.: Scenarios in Requirements Discovery. In: Alexander, I. F., Maiden, N. A. M.: Scenarios, Stories and Use Cases, John Wiley, 2004.

[RP02] Rechenberg, P., Pomberger, G. (Herausgeber): Informatik-Handbuch, 3. Auflage, Hanser, 2002.

[Sa+06] Sato, M., Inoue, M., Inoue, T., Yamamura, T.: A Proposal of Requirement Definition Method with Patterns for Element / Network Management. In: Proceedings of Management of Convergence Networks and Services, 2006.

[Sc01] Scheer, A.-W.: ARIS – Modellierungsmethoden, Metamodelle, Anwendungen. Institut für Wirtschaftsinformatik der Universität des Saarlands, 2001.

[Sc03] Schumacher, M.: Security Engineering with Patterns - Origins, Theoretical Model, and New Applications. Springer, 2003.

[Sc+06] Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., Sommerlad, P.: Security Patterns – Integrating Security and Systems Engineering. Wiley & Sons, 2006.

[Sc07] Scheer, A.-W.: CIOs entwickeln sich zu Chief Process Officers. In: CIO, 2007.

[Sc+08] Schulte, S., Eckert, J., Repp, N., Steinmetz, R.: An Approach to Evaluate and Enhance the Retrieval of Semantic Web Services. In: Interoperability in Business Information Systems (IBIS), no. 3, 2008.

[Se02] Seeberg, C.: Life Long Learning: Modulare Wissensbasen für elektronische Lernumgebungen. Springer, 2002.

[Se04] Seeberg, C.: Adaptivität für individuelles Lernen. In: Haake, J., Schwabe, G., Wessner, M. (Hrsg.): CSCL-Kompendium. Lehr- und Handbuch zum computerunterstützten kooperativen Lernen. Oldenbourg Verlag, Seite 166-170, 2004.

[SEL05] Siau, K., Ericksson, J., Lee, L.: Theoretical versus practical complexity: The case of UML. In: Journal of Database Management 16, 3, Seite 40-57, 2005.

[SN02] Steinmetz, R., Nahrstedt, K.: Multimedia-Fundamentals - Media Coding and Content Processing. Volumen 1. Prentice Hall, 2002.

[SN04] Steinmetz, R., Nahrstedt, K.: Multimedia-Applications. Springer, 2004.

[SP05] Shneiderman, B., Plaisant, C.: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Pearson–AddisonWesley, 2005.

[SS06] Schobert, W., Schuemmer, T.: Supporting pattern language visualization with cope. In: Proceedings of EuroPloP 2006, 2006.

[St97] Stapleton, J.: DSDM Dynamic Systems Development method. Addison-Wesley, 1997.

[St02] Steinacker, A.: Medienbausteine für web-basierte Lernsysteme. Dissertation an der Technischen Universität Darmstadt. 2002.

[SV05] Stahl, T., Völter, M.: Modellgetriebene Softwareentwicklung – Techniken, Engineering, Management. Dpunkt.verlag, 2005.

[SVB02] Sturm, T., von Voss, J., Boger, M.: Generating code from UML with velocity templates. In: Proceedings of UML 2002 — The Unified Modeling Language, Seite 150-161, 2002.

[TFR02] Turk, D., France, R., Rumpe, B.: Limitations of Agile Software Processes . In: Proceedings of Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, 2002.

[Tu02] Turau, V.: A framework for automatic generation of web-based data entry applications based on XML, ACM Symposium on Applied Computing, 2002.

[Va03] Vahs, D.: Organisation – Einführung in die Organisationstheorie und -praxis. 4. Auflage. Schäffer-Poeschel Verlag, 2003.

[Ve00] Versteegen, G.: Projektmanagement mit dem Rational Unified Process. Springer, 2000.

[VD07] Verbert, K., Duval, E.: Evaluating the ALOCOM Approach for Scalable Content Repurposing, In: Proceedings of the Second European Conference on Technology Enhanced Learning, 2007.

[VOD08] Verbert, K., Ochoa, X., Duval, E.: The ALOCOM framework: towards scalable content reuse. In: Journal of digital information, 2008.

[Vö05] Völter, M.: Modellgetriebene Softwareentwicklung. In: DatenbankSpektrum, Heft 13, Seite 41-44, 2005.

[WHW02] Wang, W., Haake, J.M., Wessner, M.: Supporting cooperative learning in distributed project teams. In: Proceedings of the Confederated international conferences CoopIS, DOA, and ODBASE 2002. Springer, Seite 266-285, 2002.

[We01] Weidenhaupt, K. L.: Anpassbarkeit von Software-Werkzeugen in prozessintegrierten Entwicklungsumgebungen. Dissertation an der technischen Hochschule Aachen, 2001.

[WH98] Wessner, M., Haake, J. M.: Kooperative Lernumgebungen. In: GMD-Spiegel 28 (2), Seite 32-34, 1998.

[Wi02] Wiley, D.A.: Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In: Wiley, D.A. (Editor): The Instructional Use of Learning Objects. Agency for Instructional Technology and Association for Educational Communication & Training, Seite 3-23, 2002.

[Zi+06] Zimmermann, B., Bergsträßer, S., Rensing, C., Steinmetz, R.: A Requirements Analysis of Adaptations of Re-Usable (E-Learning) Content, In: Proceedings of EdMedia 2006, Seite 2096-2103, 2006.

[ZRS06a] Zimmermann, B., Rensing, C., Steinmetz, R.: Formatübergreifende Anpassungen von elektronischen Lerninhalten. In: Proceedings of DeLFI 2006, Seite 15-26, 2006

[ZRS06b] Zimmermann, B., Rensing, C., Steinmetz, R.: Patterns for Tailoring E-Learning Materials to Make them Suited for Changed Requirement. In: Proceedings of VikingPLoP 2006, 2006.

[ZRS07] Zimmermann, B., Rensing, C., Steinmetz, R.: Patterns towards Making Web Material Accessible. In: Proceedings of EuroPLoP 2007, 2007.

[ZSW98] Zündorf, A., Schürr, A., Winter A. J.: Story Driven Modeling. In: ACM Transactions on Software Engineering and Methodology (TOSEM), Academic Press, 1998.

Online Referenzen

[1] Ambler, S. W.: The Agile Unified Process (AUP), *http://www.ambysoft.com/unifiedprocess/agileUP.html*. Letzter Zugriff am: 08.08.2008.

[2] APOSDLE Projekt, *http://www.aposdle.tugraz.at*. Letzter Zugriff am: 08.08.2008

[3] ARIADNE Foundation for the European Knowledge Pool, *http://ariadne.cs.kuleuven.be*. Letzter Zugriff am: 08.08.2008.

[4] Content Sharing Projekt, *http://www.contentsharing.com*. Letzter Zugriff am: 08.08.2008.

[5] Coplien, J.O.: Software Patterns, *http://hillside.net/patterns/definition.html*. Letzter Zugriff am: 08.08.2008.

[6] Cunningham, W.: Canonical Form, *http://c2.com/cgi/wiki?CanonicalForm*, letzte Änderung: 2005. Letzter Zugriff am: 08.08.2008.

[7] Cunningham, W.: The Portland Pattern Repository, *http://c2.com/cgi/wiki*, Letzter Zugriff am: 08.08.2008.

[8] Eckstein, R.: Java SE Application Design with MVC. *http://java.sun.com/developer/technicalArticles/javase/mvc/index.html*. Letzter Zugriff am: 08.08.2008.

[9] ErgoNorm-Benutzerfragebogen zu "Arbeit & Software", *http://www.ergonomic.de/files/abschlussbericht_de_lang.pdf*. S. 197 ff. Letzter Zugriff am: 08.08.2008.

[10] Fowler, M.: Analysis Patterns 2 - Work in Progress. Verfügbar unter: *http://martinfowler.com/ap2/index.html*. Letzter Zugriff am: 08.08.2008.

[11] IMS Global Learning Consortium Inc.: IMS Digital Repositories v1.0, *http://www.imsglobal.org/digitalrepositories/index.html*. Letzter Zugriff am: 08.08.2008.

[12] Kruchten, P.: What Is the Rational Unified Process? *http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jan01/WhatIstheRationalUnifiedProcessJan01.pdf*. Letzter Zugriff am: 08.08.2008.

[13] Merlot: Multimedia Educational Resource for Learning and Online Teaching. *http://www.merlot.org*. Letzter Zugriff am: 08.08.2008.

[14] Miller, J., Mukerji, J.: MDA Guide 1.0, *http://www.omg.org/docs/omg/03-06-01.pdf*. Letzter Zugriff am: 08.08.2008.

[15] Merriam Webster Dictionary - Online Version, *http://www.merriam-webster.com/dictionary*. Letzter Zugriff am: 08.08.2008.

[16] Object Management Group: OMG Unified Modeling Language Specification, Version 2.1.1, *http://www.omg.org/docs/formal/07-02-05.pdf*. Letzter Zugriff am: 08.08.2008.

[17] Object Management Group: MOF 2.0 / XMI Mapping Specification, Version 2.1, *http://www.omg.org/cgi-bin/apps/doc?formal/07-12-01.pdf*. Letzter Zugriff am: 08.08.2008.

[18] PLML, XML DTD, *http://www.hcipatterns.org/PLML+1.0.html*. Letzter Zugriff am: 08.08.2008.

[19] Prümper, J.: Fragebogen ISONORM 9241/110-S - Beurteilung von Software auf Grundlage der Internationalen Ergonomie-Norm DIN EN ISO 9241-110. *http://www.seikumu.com/de/dok/dok-echtbetrieb/Fragebogen-ISONORM-9241-110-S.pdf* . Letzter Zugriff am: 08.08.2008.

[20] Resource Description Framework (RDF), 1999, *http://www.w3.org/RDF*. Letzter Zugriff am: 08.08.2008.

[21] Reenskaug, T.: MVC XEROX PARC 1978-79, *http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html*. Letzter Zugriff am: 08.08.2008.

[22] Reload Editor, *http://www.reload.ac.uk*. Letzter Zugriff am: 08.08.2008.

[23] Robertson, S.: Requirements Patterns via Events / Use Cases, *http://www.systemsguild.com/GuildSite/SQR/Requirements_Patterns.html*. Letzter Zugriff am: 08.08.2008.

[24] SCORE-Projekt (System for Courseware reuse), *http://www.ipd.uka.de/SCORE/de/index.html?inhalte/home.html*. Letzter Zugriff am: 08.08.2008.

[25] SCORM Modell der Advanced Distributed Learning Initiative, *http://www.adlnet.gov/scorm/index.aspx*. Letzter Zugriff am: 08.08.2008.

[26] SMETE Digital Library, *http://www.smete.org*. Letzter Zugriff am: 08.08.2008.

[27] SMIL: W3C Synchronized Multimedia Home page, *http://www.w3.org/AudioVideo*. Letzter Zugriff am: 08.08.2008.

[28] Schröder, U., Rohde, P., Gebhardt, M.: Fallstudie der eLearning Strategie der RWTH Aachen, *http://www.e-teaching.org/projekt/fallstudien/rwth_aachen/Fallstudie_CiL_RWTH_Aachen.pdf*. Letzter Zugriff am: 08.08.2008.

[29] SYSTRAN Übersetzungssoftware, *http://www.systran.de/*. Letzter Zugriff am: 08.08.2008.

[30] Team Training Solutions: TT Knowledge Force, *http://www.tt-s.com/software/tt-knowledge-force.html*. Letzter Zugriff am: 08.08.2008.

[31] The Apache Velocity Project, *http://velocity.apache.org*. Letzter Zugriff am: 08.08.2008.

[32] Internetseite zum V-Modell, *http://v-modell.iabg.de*. Letzter Zugriff am: 08.08.2008.

[33] W3C Working Group Note: Web Services Architecture Requirements, *http://www.w3.org/TR/wsa-reqs*. Letzter Zugriff am: 08.08.2008.

Abkürzungsverzeichnis

API Application Programming Interface

ARIS Architektur integrierter Informationssysteme

AV abhängige Variable

BPMN Business Process Modeling Notation

CASE Computer Aided Software Engineering

CoPE Collaborative Pattern Editor

CWM Common Warehouse Model

DIN Deutsches Institut für Normung e. V.

DOC document, Dateiformat für Microsoft Office Word

DSL Domain Specific Language

DSL Digital Subscriber Line

DSDM Dynamic System Development Method

DTD Document Type Definition

FFT Format-Funktions-Tabelle

FP Format PlugIn

GIF Graphics Interchange Format

GoF Gang of Four

HTML Hypertext Markup Language

ISDN Integrated Services Digital Network

ISO International Organization for Standardization

JPEG Joint Photographic Experts Group, ein Grafikformat

MDA Model Driven Architecture

MDSD Model Driven Software Development

MOF Meta Object Facility

MTE Modification Transaction Engine

OMG Object Management Group

OOCR Object Orientet Content Representation

PCR Physical Content Representation

PDF Portable Document Format

PIM Platform Independent Model

PIT Process description Input Tool

PLML Pattern Language Markup Language

PLoP Pattern Languages of Programs

POSA Pattern-Oriented Software Architecture

PROPEL Process Pattern Description Language

PSM Platform Specific Model

RDF Ressource Description Framework

RUP Rational Unified Process

SCORM Sharable Content Object Reference Model

SCR Sematic Content Representation

SEC Semantic Enrichement Component

SMIL Synchronized Multimedia Integration Language

UML Unified Modeling Language

UV Unabhängige Variable

W3C World Wide Web Consortium

WGT Wizard Generation Tool

WYSIWYG What You See Is What You Get

XMI XML Metadata Interchange

XML Extensible Markup Language

XP Extreme Programming

YAWL Yet Another Workflow Language

Anhang A: Das Pattern zur Anpassung an ein verändertes (Corporate) Design

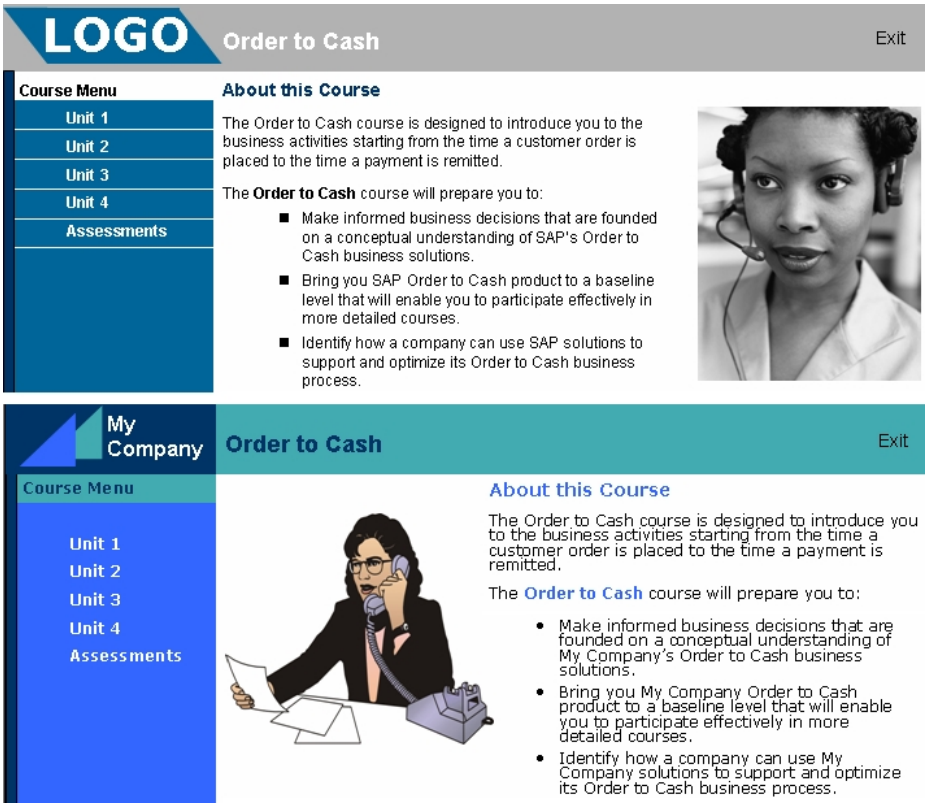
Nachfolgend wird das vollständige Pattern zur Anpassung an ein verändertes (Corporate) Design als ein Beispiel für ein Anpassungspattern vorgestellt. Das Pattern wurde auf der VikingPLoP 2006 [ZRS06b] vorgestellt. Es wird das Original-Pattern verwendet, das in englischer Sprache vorliegt.

Design Adaptation *

Intent: Adapt the design of materials to match incoming requirements.

Context: As for many kinds of content the design is very important for E-Learning content. Therefore you should always take care of a design matching all requirements. If there is a change in design requirements it is necessary to adapt the course to the new requirements. There are several reasons for a change in the requirements, e.g. if a course was originally designed for one company and should be re-used in another company or if the style guide of a company changes.

Problem: You want to adapt your course to changed requirements concerning the (corporate) design. What do you have to do in order to achieve a design that fits the new requirements?



Example for a design adaptation.

The figure above shows a course: First the original version, and then after adapting it to a changed corporate design.

Forces:

- E-Learning courses are normally designed by following a style guide. If this style guide changes for some reason the design of the course has to be adapted to the new guideline.
- The design normally consists of many items, like logos, background images and colors, fonts etc. To adapt the design all elements have to be considered.
- If a style template is used you can change this template (e.g. CSS for HTML or slide master for PPT).
- If no style template is used you have to change the design by changing element by element, page by page and file by file.

Solution: A design adaptation starts by replacing graphical elements that do not meet the requirements (e.g. logos). Therefore you decide for each graphical element if it is conform to your requirements. If it is not you replace it by a conform element. If the new graphical elements have a different size compared to the original ones it might be necessary to resize them. Depending on the file format of the materials the way how you replace the elements is different. E.g. in HTML you replace the target of the element's tag, whereas in DOC you delete the old element and insert a new one.

There might be some graphical elements that occur in places where no elements at all are allowed to occur. You should check for those elements and if you detect some you must delete them.

If you need additional graphical elements in places where no element is provided you have to add those elements. Keep in mind that this has effects on the arrangement of all elements. In the last step you rearrange all elements that are not placed correctly.

Style guides normally define rules how to design the whole layout. If the style guide changes you have to adapt the design accordingly.

If you want to use the course in another company it might be also necessary to change the company name. Be careful if the new name has a different length then the old one. This might has a negative effect on the look of the text blocks, where the name has been changed.

There is no strict order in executing the steps mentioned so far. However the order proposed here seems to be useful. The step "Rearranging text parts and images" should always be executed as the last step. When executing this step you should check that all elements are positioned correctly. All other steps might influence the arrangement of elements. E.g. if a logo is deleted instead of replacing it, this leads to a change in the arrangement of the other elements as well.

Steps needed to execute the solution:

1. Replacing graphical elements
2. Deleting graphical elements
3. Add additional graphical elements
4. Performing changes according to style guides

5. Changing company naming
6. Rearranging text parts and images

Known uses: This pattern is based on the experience of experts in branding and corporate design from several companies, e.g. SAP, as well as on our experiences in changing the design of E-Learning content.

Consequences:

Positive:

- The course is adapted to the required (corporate) design. This generates the required look and feel for this course.

Negative:

- In the rare cases where no style guide is available the adaptation is hard to execute and might be incomplete.
- If you have done adaptations that provide an additional version of the course for parallel use (like translation or printability) you should perform the design changes for these versions as well.

Related patterns: Not known

Connected Patterns:

- Terminology (Often the target group has changed if a layout change has to be executed. A change in the target group might also require an adaptation to a changed terminology.)
- Printability (If you have changed a version that is optimized for printing, you should check that none of your changes conflicts with printability.)

Used Patterns:

- “Correct arrangement of elements” used by “Rearranging text parts and images”
- “Correct length of text blocks” used by “Changing company naming”

Anhang B: Vom Wizard-Generierungswerkzeug verwendete Templates

In Abschnitt 5.1.1 wurde ein Teil der Templates vorgestellt, die vom WGT zur Generierung eines prototypischen Wizards zur Unterstützung von Anpassungsprozessen verwendet werden. An dieser Stelle soll ein vollständiger Überblick über alle bei der Erstellung des Prototyps verwendeten Templates gegeben werden. Dabei werden auch die Templates vorgestellt, die in Abschnitt 5.1.1 nicht erläutert wurden.

Die Templates werden in der Reihenfolge erläutert, in der die auf den Templates basierenden Seiten bei der Ausführung des prototypischen Wizards angezeigt werden.

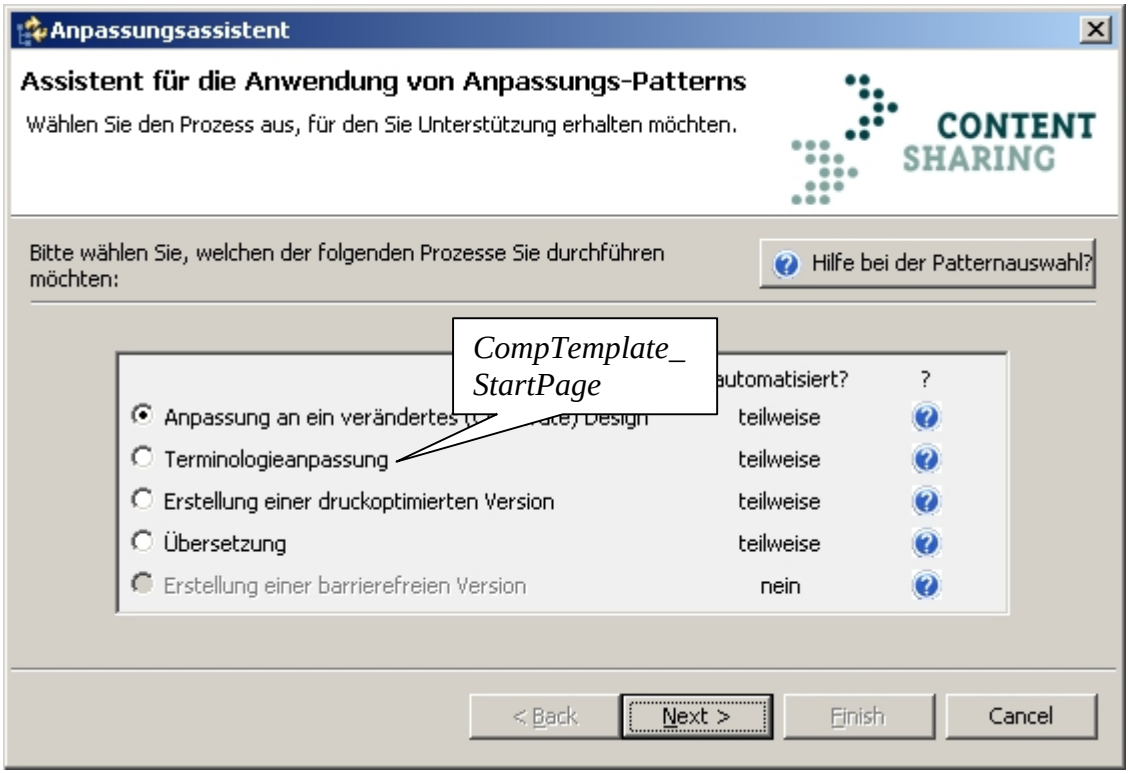


Abbildung 67: Auswahl aus unterstützten Prozessen.

APStartPage

Mit diesem Template wird die Startseite des Prototyps generiert (Abbildung 67). Diese listet die vom Prototyp unterstützten Anpassungsprozesse auf. Der Nutzer kann den durchzuführenden Prozess wählen. Bei der Auswahl hilft zum einen eine HTML-Hilfeseite des Anpassungsprozesses, die den Prozess beschreibt. Sie ist über das blaue Fragezeichen hinter jedem Prozess aufrufbar. Zum anderen gibt es über den entsprechenden Button die Möglichkeit, eine zusätzliche Hilfe bei der Prozessauswahl zu erhalten. Das Drücken dieses Buttons ruft die Seite *ChoosePatternDialog* auf. Außerdem zeigt die Startseite, zu welchem Grad ein Anpassungsprozess automatisiert ist (vgl. Abschnitt 6.2.4). Direkt nach der Prototyp-Generierung sind alle Anpassungsprozesse

durch Anleitungen unterstützt, aber nicht automatisiert, da die Automatisierung erst nach der Prototyp-Generierung durch einen Entwickler vorgenommen wird.

CompTemplate_StartPage

Diese Composites zeigen auf der Startseite (*APStartPage*) alle vom Prototyp beschriebenen Anpassungsprozesse, deren Unterstützungsgrad sowie das blaue Fragezeichen für den Hilfelink an, so dass ein Anwender wählen kann, welchen Prozess er durchführen möchte (Abbildung 67). Das Composite wird für jeden unterstützten Anpassungsprozess einmal erzeugt.

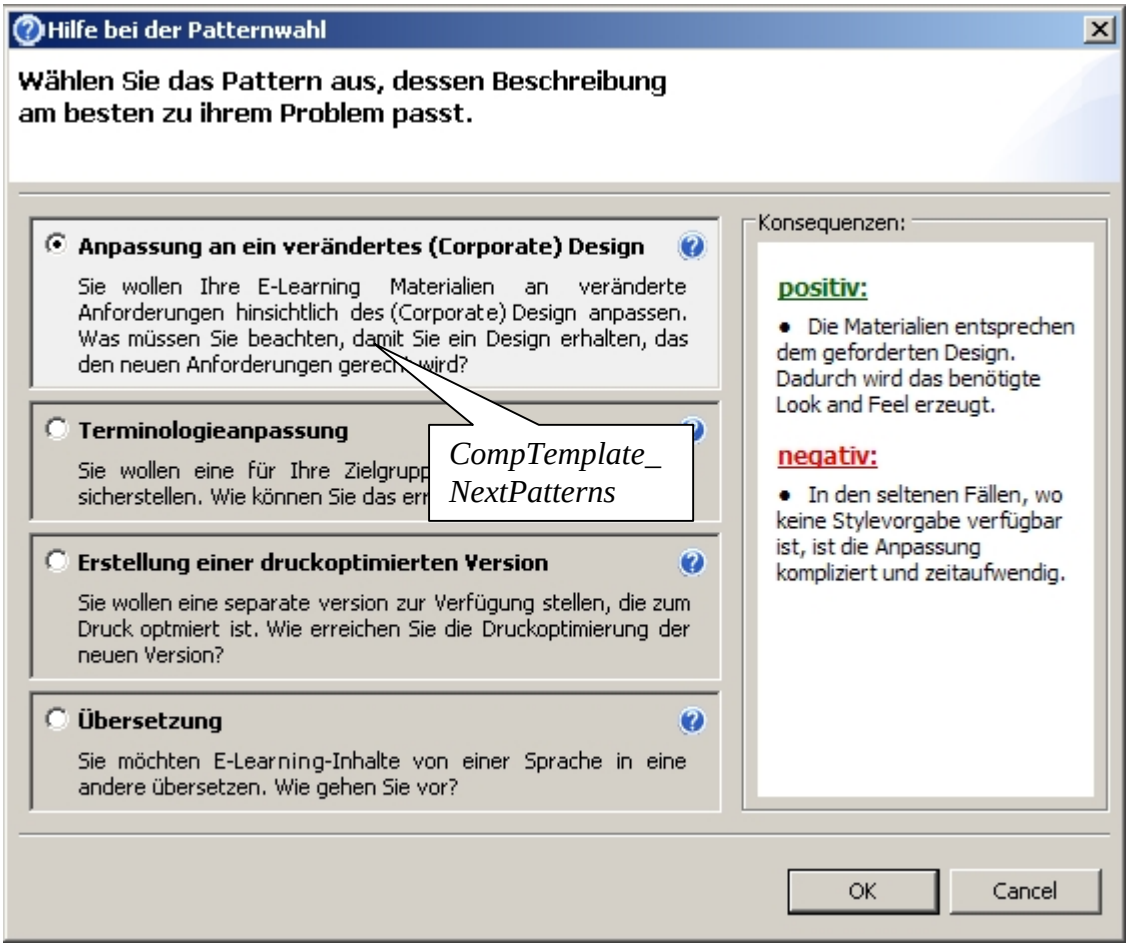


Abbildung 68: Unterstützung bei der Prozessauswahl.

ChoosePatternDialog

Wenn ein Nutzer den Prototyp startet, weiß er gegebenenfalls nicht, welchen Anpassungsprozess er benötigt (Abbildung 68). Deswegen gibt es auf der Startseite die Möglichkeit, zusätzliche Informationen zu den im Prototyp abgebildeten Anpassungsprozessen zu erhalten. Das Template *ChoosePatternDialog* erzeugt die dazu benötigte Seite, die alle Prozesse auflistet und zu jedem Prozess Informationen anzeigt, die aus

Platzgründen auf der Startseite nicht gezeigt werden, die aber bei der Auswahl des richtigen Prozesses helfen können.

CompTemplate_NextPatterns

Dieses Quelltextfragment fügt einen Anpassungsprozess in die Liste der Prozesse in *ChoosePatternDialog* ein (Abbildung 68). Das Composite zeigt Informationen zum Prozess an, aufgrund derer der Anwender in *ChoosePatternDialog* auswählen kann, welchen Prozess er durchführen möchte. Das Composite wird für jeden unterstützten Anpassungsprozess einmal erzeugt.

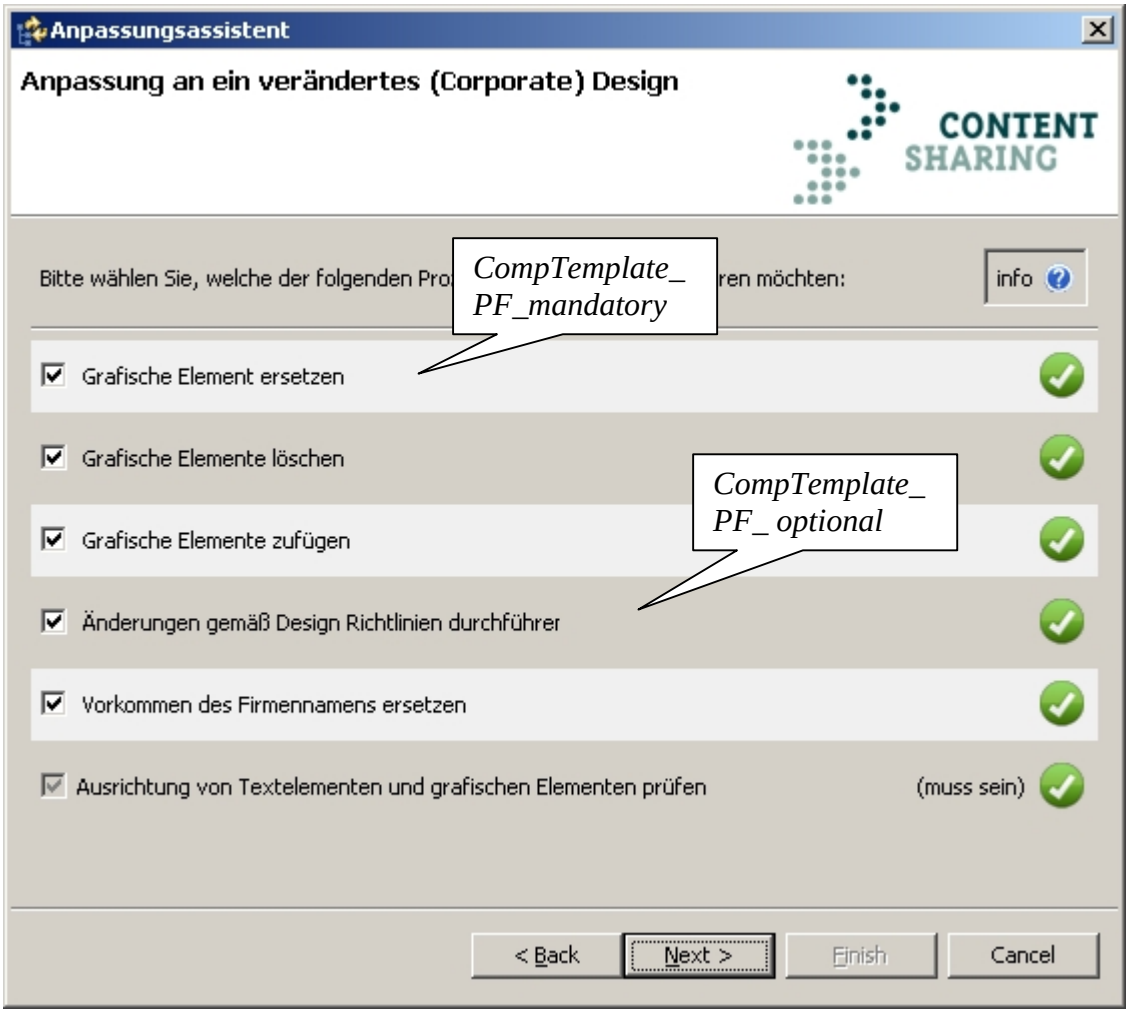


Abbildung 69: In einem Prozess enthaltene Prozessschritte.

APProcessFragmentsPage

Seiten, die mit diesem Template erzeugt werden, listen alle in einem Anpassungsprozess enthaltenen Prozessschritte auf (Abbildung 69). Außerdem wird es ermöglicht, zu

wählen, welche der freiwilligen Prozessschritte man ausführen möchte. Die Seite wird für jeden im Wizard enthaltenen Anpassungsprozess einmal erzeugt.

CompTemplate_UsedPatternButton

Falls es zu einem Prozessschritt eine ausführliche Beschreibung gibt, kann auf der Seite dieses Prozessschrittes ein Button zugefügt werden, der eine HTML-Datei öffnet, die die zugehörige Beschreibung enthält (Abbildung 69). Das Composite wird einmal für jede existierende ausführliche Prozessschritt-Beschreibung erzeugt.

CompTemplate_PF_mandatory

Dieses Composite wird verwendet, um auf den Seiten, die auf *APProcessFragmentsPage* beruhen, die Prozessschritte anzuzeigen, deren Ausführung verpflichtend ist (Abbildung 69). Das Composite wird für alle in den berücksichtigten Anpassungsprozessen enthaltenen verpflichtenden Prozessschritte einmal erzeugt.

CompTemplate_PF_optional

Dieses Composite wird verwendet, um auf den Seiten, die auf *APProcessFragmentsPage* beruhen, die Prozessschritte anzuzeigen, deren Ausführung freiwillig ist (Abbildung 69). Das Composite wird für alle in den berücksichtigten Anpassungsprozessen enthaltenen freiwilligen Prozessschritte einmal erzeugt.

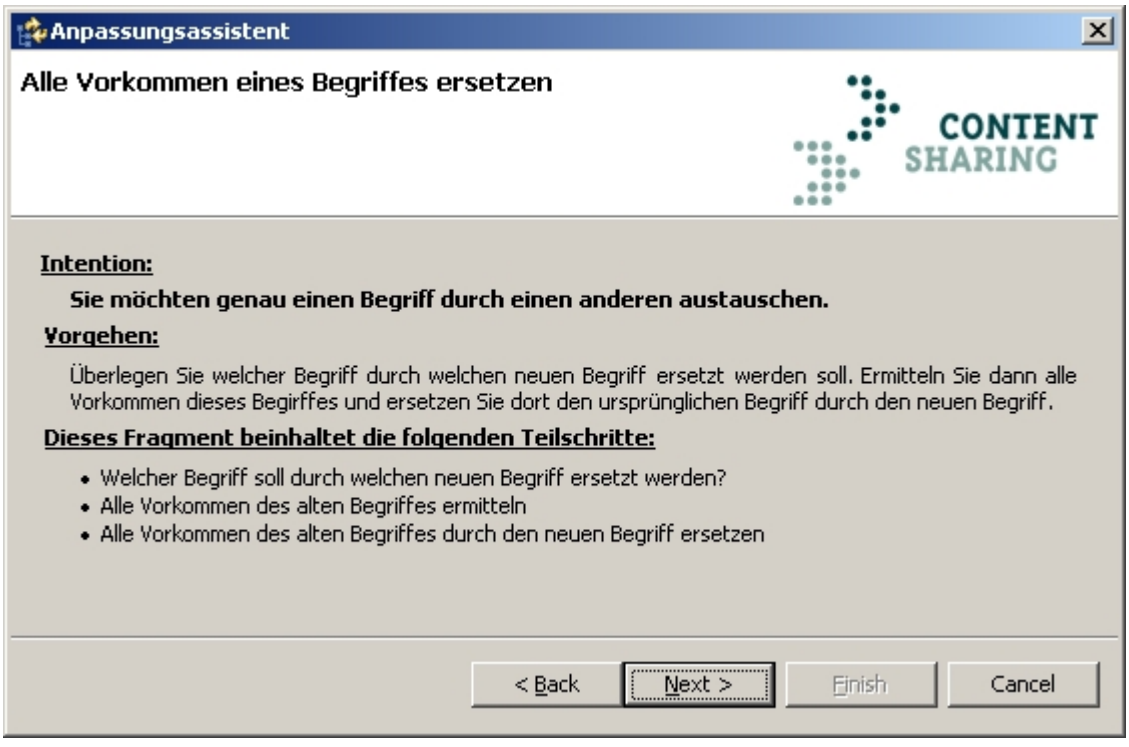


Abbildung 70: Überblicksseite für einen Prozessschritt.

APFragmentWizardPage

Die mit diesem Template angelegte Seite gibt einen Überblick über einen Prozessschritt und benennt die darin enthaltenen Unterschritte (Abbildung 70). So kann sich ein Anwender einen Überblick verschaffen, was im Prozessschritt geschieht. Diese Seite wird für alle Prozessschritte, die in den vom Wizard unterstützten Anpassungsprozessen enthalten sind, einmal angelegt.

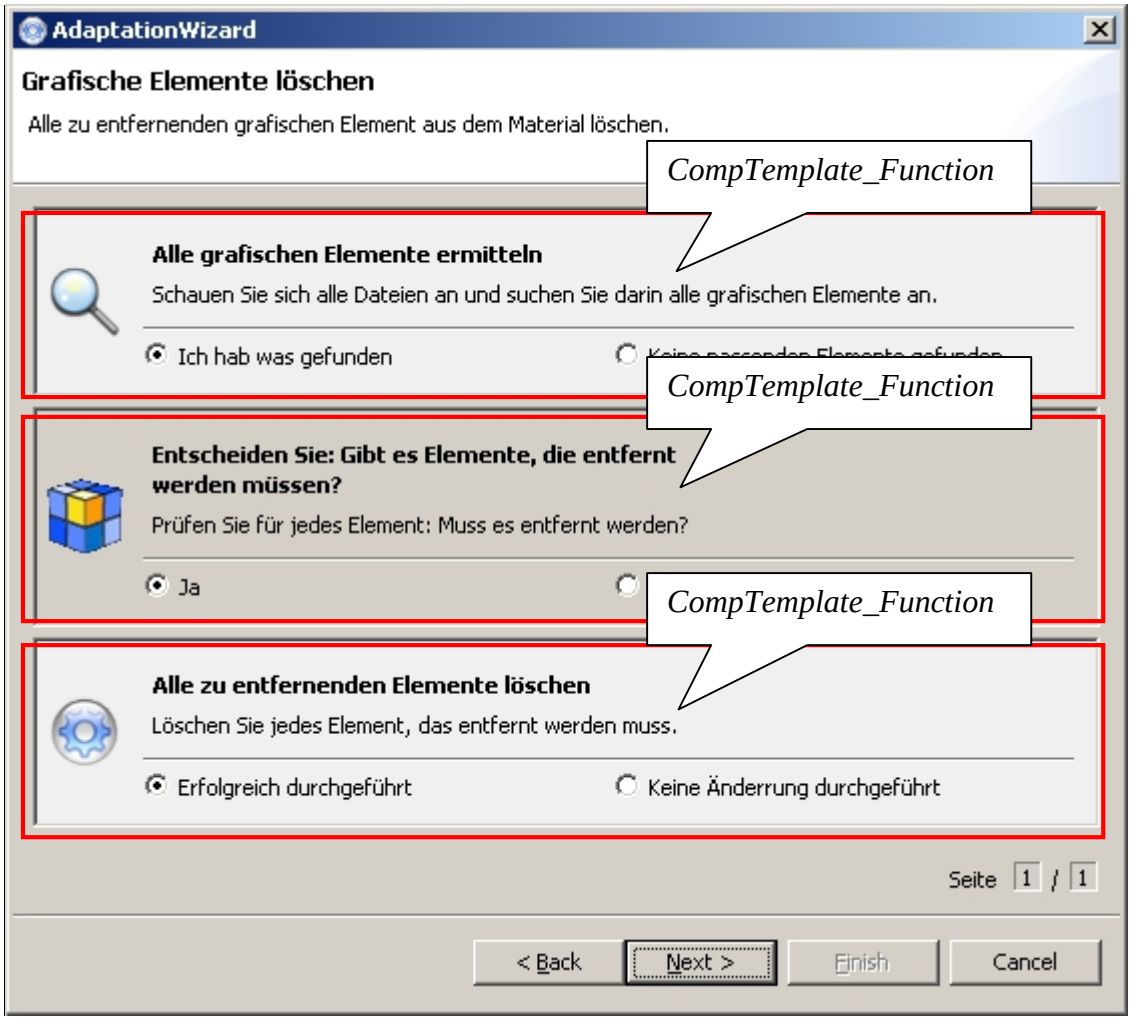


Abbildung 71: Anzeige einer Gruppe von Unterschritten.

APFragmentFunctionsPage

Dieses Template wird genutzt, um eine Seite anzulegen, die eine Gruppe von atomaren Unterschritten anzeigt (Abbildung 71). Die einzelnen Unterschritte werden durch Composites realisiert, die mit dem Template *CompTemplate_Function* angelegt wurden. Basierend auf dem Template werden alle Seiten erzeugt, die Unterschritte anzeigen.

Abbildung 71 zeigt eine Prozessschrittseite vor der Ergänzung automatisierter Funktionen. Die Informationen zu den atomaren Unterschritten werden aus der Prozessbeschreibung gewonnen. Um die Ablauflogik zu steuern, muss der Nutzer des Prototyps außerdem die Ergebnisse der Durchführung jedes Unterschrittes rückmelden. Dazu wählt er beispielsweise aus, dass er Elemente gefunden hat (vergleiche Abbildung 71). Erst dann wird der nächste Schritt aktiviert, im Beispiel ist das die Entscheidung darüber, ob es zu löschende Elemente gibt. Welche Rückmeldung möglich ist, hängt vom Typ des Unterschrittes ab. Der Prototyp reagiert also entsprechend des Unterschritttyps und des zugrunde liegenden Modells auf die Rückmeldungen des Anwenders.

CompTemplate_Function

Hier wird ein Quelltextfragment für ein Composite zur Verfügung gestellt. Für jeden auf einer Seite enthaltenen Unterschritt wird der zugehörigen Seite, die auf *APFragmentFunctions_Page* basiert, ein neues Composite zugefügt, das die Angaben zum Unterschritt enthält (Abbildung 71). Das Composite wird für jeden im Prozess enthaltenen Unterschritt einmal erzeugt. Abhängig davon, ob der Unterschritt vom Typ Ermittlung, Entscheidung oder Ausführung ist, wird das Composite unterschiedlich gestaltet. Abbildung 71 zeigt drei dieser Composites.

CompTemplate_PageCounter

Die Anzahl der Seiten, die für die Durchführung eines Prozessschrittes benötigt werden, wird gezählt, damit Anwender einen Überblick über die noch ausstehenden Seiten erhalten (Abbildung 71, rechts unten). Dies wird durch das Composite-Template für den PageCounter realisiert, das den Seiten, die auf *APFragmentFunctions_Page* beruhen, zugeordnet ist. Das Composite wird für jede *APFragmentFunctions_Page* einmal erzeugt.

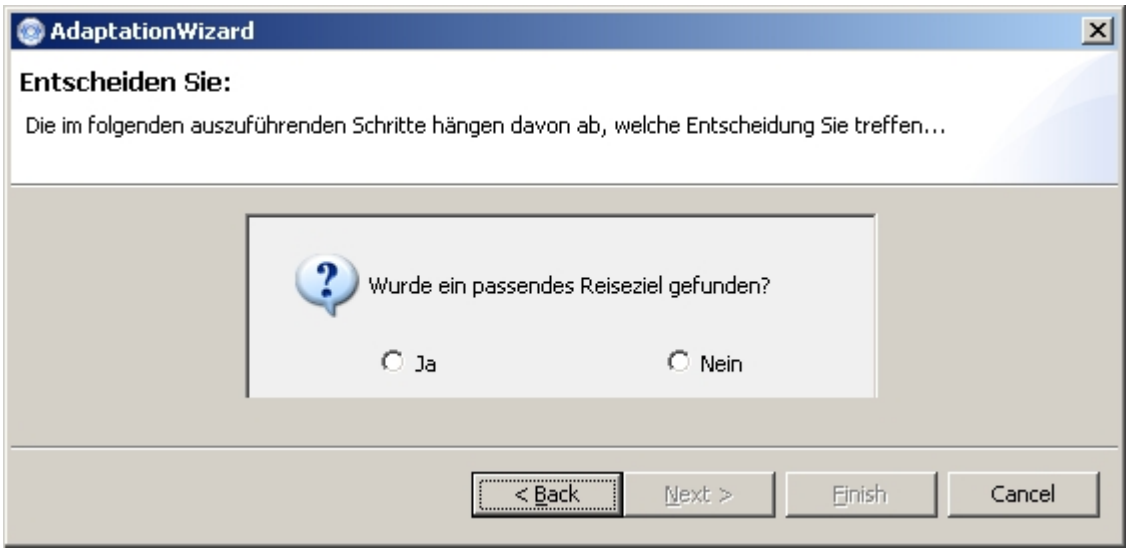


Abbildung 72: Anzeige von Entscheidungen.

APDecisionWizardPage

Es gibt Fälle, wo ein Prozessschritt nur ausgeführt wird, wenn der Anwender vorher eine bestimmte Entscheidung mit „ja“ oder „nein“ beantwortet hat. Die mit diesem Template erzeugte Seite präsentiert die Entscheidungsfrage und zeigt abhängig von der Entscheidung des Anwenders die nächste Seite (Abbildung 72). Basierend auf dem Template werden für alle entsprechenden Entscheidungen die Seiten erzeugt.

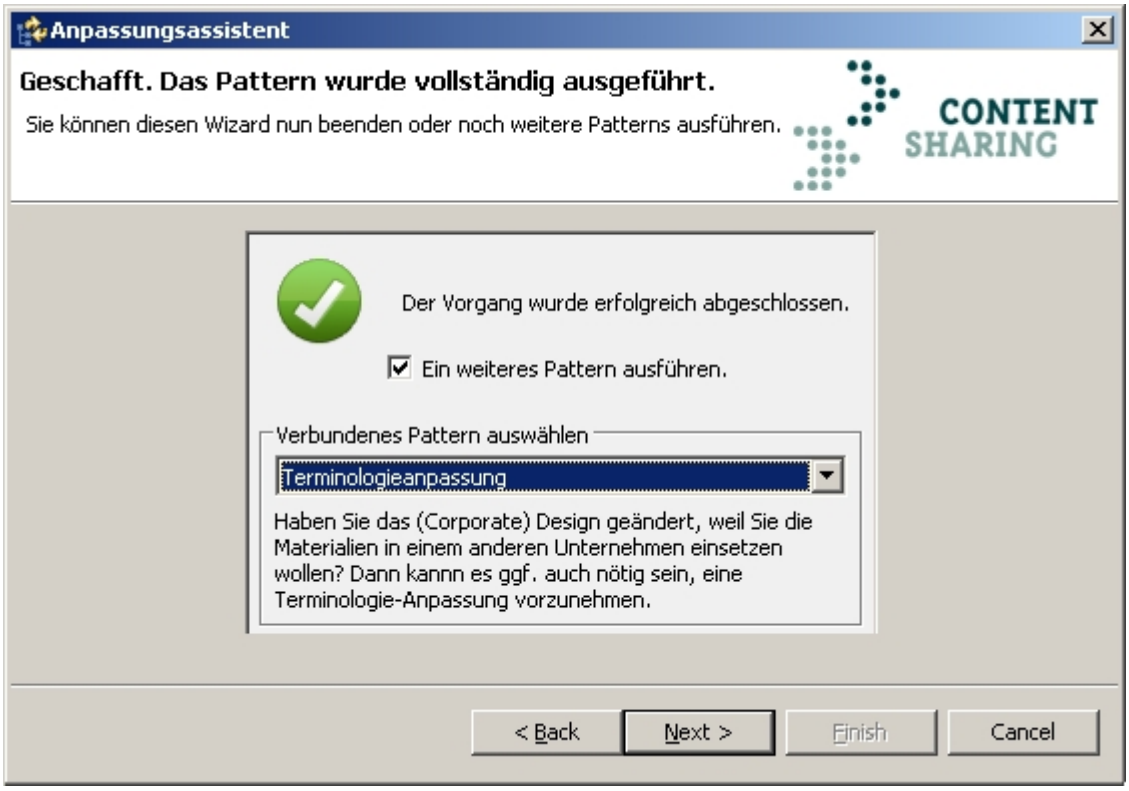


Abbildung 73: Letzte Seite des Prototyps.

APFinalWizardPage

Diese Klasse stellt die letzte Seite des Prototyps dar (Abbildung 73). Sie bietet dem Anwender die Möglichkeit, den Prototyp zu beenden, oder bei Bedarf einen weiteren Anpassungsprozess zu starten. Sie gibt auch Hinweise, falls im Anschluss an den aktuellen Anpassungsprozess andere Prozesse ausgeführt werden sollten.

Nachdem nun alle Templates betrachtet wurden, die mehrfach erzeugt werden können, werden nachfolgend die Templates vorgestellt, die bei der Erzeugung eines prototypischen Wizards genau einmal erstellt werden. Die Templates werden alphabetisch geordnet aufgelistet.

AdaptationWizard

Dieses Template erweitert die Java-Klasse `Wizard`. Die basierend auf dem Template erstellte Klasse enthält eine Liste aller Seiten des Prototyps.

APAutomater

ist ein Interface, das bei der Automatisierung von Funktionen zu implementieren ist. Dadurch wird sichergestellt, dass im basierend auf dem Prototyp erstellten Wizard alle benötigten Daten einer Funktion zur Verfügung stehen.

APWizardAutomate

Diese Klasse wird vom Controller benötigt. Er verwendet sie, um eine Liste aller automatisiert verfügbaren Unterschritte zu generieren.

APWizardStarter

Wie der Name bereits sagt, wird diese Klasse zum Starten des Prototyps benötigt. Sie enthält die main-Methode und ruft die Startseite auf.

Controller

Diese Klasse realisiert den in Kapitel 5 erläuterten Controller der Anwendung.

Dependency

Diese Klasse wird verwendet, um die Abhängigkeiten und Vorbedingungen eines *WizardNode* zu beschreiben.

GUF

Bei der Klasse GUF (Graphical Utility Functions) handelt es sich um eine Hilfsklasse, die immer wieder benötigte Funktionen der grafischen Benutzeroberfläche bereitstellt.

Texts und **Texts_EN**

Die Klassen stellen die deutschen und englischen Texte, die in allen Prototypen gleich sind, für die Beschriftung aller Elemente zur Verfügung. Hierzu zählen beispielsweise alle Buttons.

WGT-WizardPage

Alle Seiten im Prototyp erben von dieser abstrakten Klasse. Sie stellt die Methoden zur Verfügung, die auf allen Seiten benötigt werden.

WizardNode

Diese Klasse wird für die Implementierung der Knoten des Prozessgraphen verwendet.

Anhang C: Erweiterung des WGT-Prototyps zum Anpassungswerkzeug

Erläuterung zum Vorgehen

In Abschnitt 6.2.4 wurde erläutert, wie der vom WGT für die Anpassungspatterns erzeugte prototypische Wizard in die Repurposing Suite integriert und durch automatisierte Funktionalitäten erweitert wurde. Nachfolgend wird diese Erweiterung noch einmal deutlich detaillierter erklärt.

Übersichtlichkeit gewinnen:

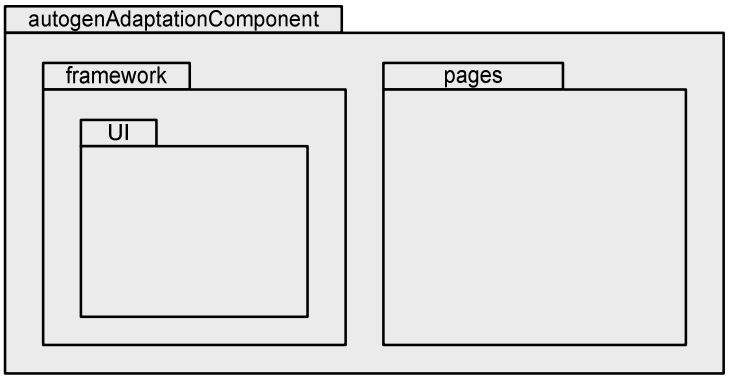


Abbildung 74: Package-Struktur für den erweiterten Anpassungs-Wizard

Der erste Schritt der Integration des generierten prototypischen Wizards in die Repurposing Suite war die Aufteilung der erzeugten Klassen in vier Pakete, die die Klassen zu logischen Einheiten zusammenfassen und die Übersichtlichkeit erhöhen (siehe Abbildung 74). Dies war notwendig, da während der Automatisierung noch eine Reihe weiterer Klassen zugefügt wurden.

Als oberstes Paket wurde durch das WGT ein `defaultpackage` erzeugt. Es enthielt direkt nach der Erzeugung alle automatisch generierten Klassen. Da es nicht den Namenskonventionen im Moduleditor gerecht wurde, wurde stattdessen ein neues Paket `autogenAdaptationComonent` erzeugt. Ziel war, dass dieses Paket nur die zentralen, von anderen Klassen benötigten Klassen enthält, wie beispielsweise den Wizard-Controller. Deswegen wurden innerhalb dieses Paketes zwei weitere Pakete angelegt: `framework` und `pages`.

Das Paket `framework` enthält alle Klassen, die die für den Wizard benötigten Funktionen oder Informationen zur Verfügung stellen. Beispielsweise findet sich hier die Klasse `WizardNode`, die die Knoten des Prozessgraphs darstellt, sowie die Klasse `Dependency`, die Abhängigkeiten und Vorbedingungen repräsentiert. Das Paket `pages` bündelt alle im Wizard verwendeten Seiten. Innerhalb von `framework` wurde noch ein zusätzliches Paket `UI` angelegt, das für Klassen gedacht ist, die für die korrekte Anzeige der

Benutzeroberfläche genutzt werden. Das `defaultpackage` blieb nach dem Verschieben aller Klassen leer.

Nachfolgend werden alle weiteren Schritte, die nach der Erzeugung der Pakete durchgeführt wurden, aufgezählt. Hierbei wird auch erläutert, wann welche Klassen in welche Pakete verschoben wurden.

Allgemeine Wizard-Klassen transferieren:

Das WGT erzeugt bei der Wizard-Generierung eine Reihe von Klassen, die nicht vom Inhalt des jeweiligen prototypischen Wizards abhängen und somit für alle erzeugten Wizards gleich aussehen. Diese Seiten wurden an die gewünschte Stelle in der Paketstruktur verschoben. Dabei war zu berücksichtigen, dass in den verschobenen Klassen alle Pfade korrekt gepflegt sind. Betroffen waren hiervon die folgenden Klassen:

- `APAutomater`, `APWizardAutomate`, `Dependency`, `GUF`, `WizardNode` und der `Controller` wurden ins Paket `autogenAdaptationComponent` verschoben.
- `APFinalWizardPage`, `WGT_WizardPage` wurden ins Paket `pages` verschoben.
- `Texts` und `Texts_EN` wurden ins Paket `UI` verschoben.

Einbindung des Anpassungswizards in die Repurposing Suite

Abbildung 75 zeigt alle zur Einbindung des Anpassungswizards benötigten Aktivitäten.

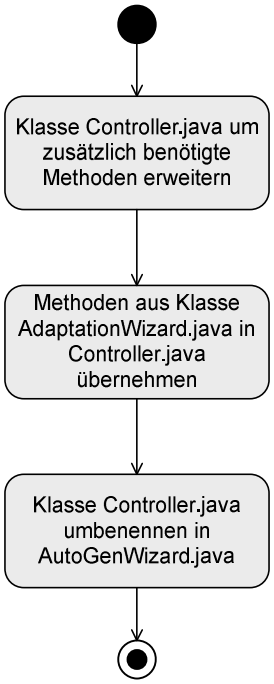


Abbildung 75: Aktivitätsdiagramm der Einbindung des Anpassungswizards.

Um den automatisch erzeugten Wizard in das Repurposing-Framework einzubinden, wurde der vom WGT erzeugte Wizard-Controller um Methoden erweitert, die die MTE aufrufen, um vom Anwender gewünschte Modifikationen an die zuständigen PlugIns weiterzugeben. Das ist in der ersten Aktion des Aktivitätsdiagramms in Abbildung 75 gezeigt.

Weiterhin enthielt der automatisch erzeugte, prototypische Wizard nur eine recht einfache Steuerung, wann zur Folgeseite und zurück gesprungen werden darf. Um hier eine Ablaufsteuerung zu ermöglichen, die auch die Ergebnisse der automatisierten Funktionen berücksichtigt, wurden entsprechende Methoden zum Vor- und Zurückspringen in der Klasse `Controller.java` zugefügt.

Die Anpassungskomponente nutzt Informationen aus dem SCR. Nimmt der Anwender Änderungen vor, so werden diese im OOCR vorgenommen und dann ins SCR überführt. Es kann vorkommen, dass der Benutzer bereits durchgeführte Änderungen rückgängig machen möchte. Um in diesem Fall OOCR und SCR in ihren ursprünglichen Zustand zurückzuversetzen, wurde in `Controller.java` eine Methode `performCancel()`implementiert, die es ermöglicht, OOCR und SCR zu bereinigen, falls der Nutzer eine Änderung verwerfen möchte. Außerdem wurde eine Methode `performFinish()` eingefügt, die zur Laufzeit dafür sorgt, dass gemachte Änderungen an die MTE weitergegeben werden, wenn der Nutzer dies veranlasst.

Da der Wizard-Controller die gesamte Steuerung der Ablauflogik überwachen soll, wurden auch die Methoden der Klasse `AdaptationWizard.java` in den Wizard-Controller aufgenommen (Aktion 2 im Aktivitätsdiagramm) und die ursprünglich erzeugte Klasse `AdaptationWizard.java` wurde gelöscht. Die Klasse `Controller.java` wurde umbenannt in `AutoGenWizard.java`, um Verwechslungen mit dem Controller des Moduleditors zu vermeiden (Aktion 3 im Aktivitätsdiagramm).

Anpassung und Verschieben Wizard-spezifischer Klassen:

Die Seiten des prototypischen Wizards, der Prozessgraph, sowie die Hilfeseiten für Prozesse und Prozessschritte enthalten prozessspezifische Informationen und unterscheiden sich somit für jeden erzeugten Wizard. Daher werden sie vom WGT für jeden Prototyp neu erzeugt und während der Erzeugung mit den prozessspezifischen Informationen befüllt. Bevor diese Seiten an die gewünschte Stelle in der Paketstruktur verschoben werden konnten, mussten einige Anpassungen vorgenommen werden:

Die Namen der Klassen, die für die Anzeige der Seiten des Wizards benötigt werden, entsprechen den vom PIT erzeugten IDs der Prozesselemente. Dies ist notwendig, um die Seiten eindeutig den Knoten des Prozessgraphen zuordnen zu können. Allerdings erschweren die kryptischen Bezeichnungen die Arbeit des Entwicklers. Deswegen wurde eine Klasse `AutoGenHelper.java` erstellt, die die vom PIT generierten XML-Dateien (die Strukturbeschreibung des Wizards und die textuellen Prozessbeschreibungen) einliest und daraus zwei Text-Dateien erzeugt:

1. Die erste Text-Datei enthält eine Liste aller IDs und den für die jeweilige ID im PIT angegebenen Namen des zugehörigen Elementes. Erzeugt der Entwickler

aus dieser Datei eine Java-Klasse `PageID.java`, so erhält er die Möglichkeit, alle WizardKnoten über ihren Namen anzusprechen.

2. Weiterhin wird eine Datei erzeugt, die die Deklaration aller Wizard-Knoten mit IDs und zugehörigen Namen enthält. Der Inhalt dieser Datei ist in die Klasse `AutoGenAdaptationComponentAction.java` an Stelle der automatisch erzeugten Deklarationen einzufügen, die nur die IDs enthalten. Dadurch können die in `PageID.java` aufgeführten Namen statt der IDs im Wizard verwendet werden.

Die Klasse `AutoGenAdaptationComponentAction.java` ist durch Erweiterung der ursprünglichen Klasse `APWizardStarter.java` (eine Implementierung des Interfaces `IRepurposingAction)` entstanden. Sie wurde umbenannt, um den Namenskonventionen im Moduleditor gerecht zu werden. Somit wurde sichergestellt, dass der Wizard ein Eclipse PlugIn realisiert und zur Laufzeit im Moduleditor registriert wird. Die ursprünglich enthaltene `main()`-Klasse wurde entfernt, da diese in PlugIns nicht benötigt wird.

Abschließend wurden alle dynamisch erzeugten Seiten an die korrekten Stellen in der Paketstruktur verschoben. Nach Ausführung dieser Schritte war der Wizard bereits innerhalb der Repurposing Suite aufrufbar und lauffähig. Allerdings enthielt er noch keine automatisierten Funktionen. Wie diese zugefügt wurden, wird im nächsten Abschnitt erläutert.

Erweiterung des Wizards

Um automatisiert Modifikationen an den Lernressourcen ausführen zu können, musste der erzeugte Wizard erweitert werden. Dazu wurden die in Kapitel 6.1 vorgestellten Möglichkeiten genutzt. Abbildung 76 erläutert das Vorgehen.

Zuerst wurden im Paket `framework` die Klassen erstellt, die für die Durchführung der automatisierten Funktionen zuständig sind. Das entspricht der Realisierung des bereits erwähnten Funktionspools. Die Klassen greifen auf das SCR zu und lösen Aktionen der SECs aus. Beispielsweise finden sich hier Klassen zum Sammeln von Informationen über Hintergrundfarben und -bilder oder über Fonts, sowie Klassen zum Auswählen bestimmter Elemente, wie Bilder.

Als nächstes war zu überlegen, welche atomaren Unterschritte durch die zur Verfügung stehenden Funktionen automatisiert werden. Die vorher erzeugte Datei `PageID.java` enthält die Abbildung von Namen auf IDs. Mit ihrer Hilfe konnte vom Entwickler ermittelt werden, welche Prozessschritte und somit welche Seiten im Wizard die gesuchten atomaren Unterschritte enthalten. Zusätzlich findet man hier auch die IDs der Unterschritte. Die Seiten enthalten für alle angezeigten atomaren Unterschritte Composites, die durch die ID des jeweiligen Unterschrittes gekennzeichnet sind. Die Composites beginnen immer mit dem Kommentar `//Function-Composite...` gefolgt vom Typ des jeweiligen atomaren Unterschrittes, z.B. `Query`. Danach beginnt das Composite. Standardmäßig ist hier abhängig vom Typ des Unterschrittes ein vom WGT erzeugtes Composite eingefügt. Dieses ist umgeben von einer if-Abfrage, die zur Laufzeit prüft, ob für den Unterschritt eine Automatisierung vorhanden ist. Ist dies nicht der Fall, wird

das Standard-Composite angezeigt, andernfalls wird das Composite der automatisierten Funktion abgefragt und angezeigt.

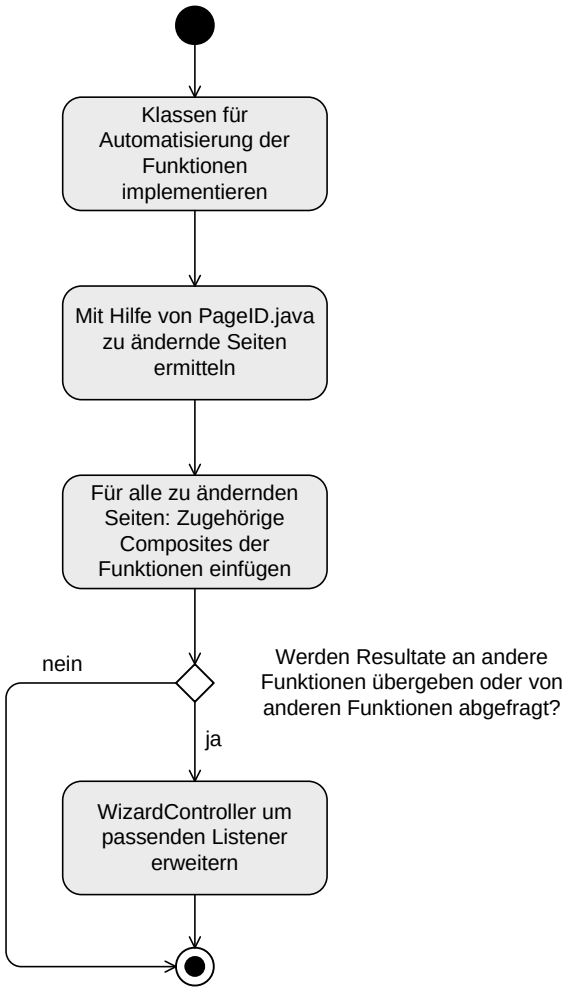


Abbildung 76: Vorgehen bei der Erweiterung des Wizards.

Im Falle der Anpassungskomponente wurden die Composites für die Funktionen nicht in den jeweiligen Funktionsklassen abgelegt. Hintergrund ist, dass je nach Verwendung der Funktion die Anzeige differiert. So ist beispielsweise die Auswahl von Bildern aus der Menge der gefundenen Bilder beim Löschen durch die gleiche Funktion realisiert wie beim Ersetzen. Die Anzeige unterscheidet sich aber in den beiden Fällen. Daher wurde bei der Automatisierung der Anpassungskomponente davon abgesehen, die Composites in den Klassen der entsprechenden Funktionen abzulegen. Stattdessen wurden sie direkt in die jeweilige Klasse der jeweiligen Seite integriert, die die Anzeige des atomaren Unterschrittes enthält. Dazu wurde die Abfrage auf Automatisierung, sowie das darin enthaltene Standard-Composite entfernt. An deren Stelle wurde das passende Composite für die Anzeige der automatisierten Funktion eingefügt.

Da Funktionen oft Resultate aus anderen Funktionen benötigen bzw. Resultate für andere Funktionen weitergeben, wurde der Wizard-Controller (`AutoGenWizard.java`)

zusätzlich um Methoden erweitert, die es erlauben, verschiedene Listener zu registrieren. Dadurch kann für jede Funktion zur Laufzeit der passende Listener im Wizard-Controller registriert werden. So lässt sich sicherstellen, dass der Wizard-Controller zur Laufzeit alle Resultate von Funktionen sammelt und diese bei Bedarf weitergibt und so die Ablaufsteuerung kontrolliert.

Im nachfolgenden Abschnitt wird ein Quelltext-Beispiel vorgestellt: einmal vor dem Zufügen automatisierter Funktionen und einmal danach.

Quelltext-Beispiel für die Wizard-Erweiterung

Nachfolgend wird ein Ausschnitt aus dem automatisch generierten Codefragment für die Auswahl zu löschender graphischer Elemente gezeigt (Listing 4). Im Vergleich dazu ist in Listing 5 das entsprechende Codefragment nach dem Zufügen des Composites für den automatisierten Unterschritt gezeigt. (Der Quelltext wurde gekürzt: Anweisungen, die die Anzeige des Composites auf der Seite betreffen, wurden der Übersichtlichkeit wegen entfernt.) Abbildung 77 zeigt das Composite zu Listing 4 vor der Automatisierung des zugehörigen atomaren Unterschrittes. Abbildung 78 zeigt das Composite zu Listing 5 nach der Automatisierung des Unterschrittes.

```
// Function-Composite... DECISION

if (myWizard().getAutomater("Decision_atomic_12184611_1275821162") ==
    null) {  (A)
  compForFun = new Composite(composite, SWT.BORDER);
  lab = new Label(compForFun, SWT.NONE);
  register("Decision_atomic_12184611_1275821162", lab);                (B)

  {
   Composite innerFunBox = new Composite(compForFun, SWT.NONE);
   Composite firstLine = new Composite(innerFunBox, SWT.NONE);
   styledText = new StyledText(firstLine, SWT.WRAP);
   styledText.setText("Entscheiden Sie: Gibt es Elemente, die entfernt
       werden müssen?");
   register("Decision_atomic_12184611_1275821162", styledText);

       if (myWizard().getNodeFromRef("Decision_atomic_12184611_1275821162")
       .isMandatory()) {
      lab = new Label(firstLine, SWT.LEFT);
   } else {
      button = new Button(firstLine, SWT.CHECK); // SKIP-button
      // sets the function's state to SKIP or SCHEDULED, based on the
            selection
      button.addListener(SWT.Selection,
            new FES_Listener("Decision_atomic_12184611_1275821162",
         AutoGenWizard.STATE_SKIP, AutoGenWizard.STATE_SCHEDULED));
      register("Decision_atomic_12184611_1275821162", button);
   }

   // description
   lab = new Label(firstLine, SWT.LEFT | SWT.WRAP);
```

```
   lab.setText("Prüfen Sie für jedes Element: Muss es entfernt werden?");
   register("Decision_atomic_12184611_1275821162", lab);

   /* choice: yes, sets the function's result to true */                (C.1)
   button = new Button(innerFunBox, SWT.RADIO);
   register("Decision_atomic_12184611_1275821162", button);
   button.addListener(SWT.Selection,
       new FES_Listener("Decision_atomic_12184611_1275821162",
      AutoGenWizard.STATE_DONE, true));                                 (E)

   /* choice: no, sets the function's result to false */                (C.2)
   button = new Button(innerFunBox, SWT.RADIO);
   register("Decision_atomic_12184611_1275821162", button);
   button.addListener(SWT.Selection, new
  FES_Listener("Decision_atomic_12184611_1275821162",
  AutoGenWizard.STATE_DONE, false));                                    (E)
   }
}
```

Listing 4: Teil eines automatisch vom WGT erzeugten Composite.

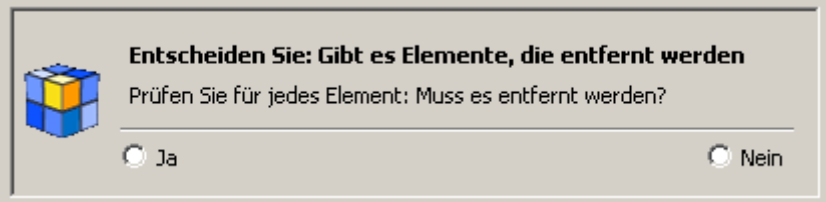


Abbildung 77: Composite vor Automatisierung des zugehörigen Unterschrittes.

Das automatisch erzeugte Composite beginnt mit dem Kommentar, der anzeigt, dass hier ein automatisch erzeugtes Composite vom Typ `Decision` beginnt. Danach folgt eine if-Abfrage, die prüft, ob es eine Automatisierung für den Unterschritt gibt und dementsprechend das Composite aus der automatisierten Klasse angezeigt werden muss **(A)**. Dieser Teil des Quelltextes wird im unteren Listing nicht mehr gezeigt.

Die FunktionsID **(B)** ist auch nach der Automatisierung dieselbe geblieben, jedoch wird sie nicht mehr explizit ausgeschrieben, sondern als Konstante aus der PageID-Klasse bezogen **(b)**.

Die zum Unterschritt gehörigen GUI-Komponenten werden sowohl vor als auch nach der Automatisierung auf der Seite registriert (`register(…)`). (Das wird benötigt, damit der Controller den Zustand der Seiten überwachen kann. So hat der Controller Zugriff auf alle Elemente einer Seite.) Statt mit den beiden Ja/Nein-Buttons **(C.1,2)** wird die Funktion nach der Automatisierung aber mit einer Tabelle **(c.1)** visualisiert – hinzu kommt ein Button zum Anzeigen des in der Tabelle selektierten Bildes **(c.2)**.

Zur Manipulation des Funktionsergebnisses durch den Benutzer muss die Tabelle Interaktionen unterstützen, wofür im unteren Listing der Abschnitt „GUI-user-interaction“ **(d)** eingefügt wurde. Der Ergebnistyp des Unterschrittes ändert sich dabei von einem booleschen Wert **(E)** zu einer Liste mit SCR-IDs (der zum Löschen ausgewählten

Bilder) **(e)**. (Beim automatisch erzeugten Prototyp wird der Nutzer gefragt: „Gibt es Elemente, die gelöscht werden sollen?“ Und der Nutzer kann mit „Ja“ bzw. „Nein“ antworten. Bei der automatisierten Variante kann der Nutzer die zu löschenden Elemente in der Tabelle auswählen. Dabei werden die SCR-IDs der Elemente vorgehalten, um diese später der MTE übergeben zu können, damit diese das Löschen der entsprechenden Elemente veranlassen kann.)

Die Einbindung des automatisierten Unterschrittes in die Ablaufsteuerung und den Datenfluss des Wizards wird in der Sektion „`integrate function into the information-flow`“ **(f)** realisiert. Als Eingabe benötigt der automatisierte Unterschritt das Ergebnis des vorangegangenen, ebenfalls automatisierten Unterschrittes („Alle graphischen Elemente ermitteln“ **(g.1)**) – die von diesem Unterschritt ermittelten Bilder müssen in die Tabelle übernommen werden **(g.2)**. Da es sich nicht an einer GUI-Interaktion festmachen lässt, wann der Benutzer seine Auswahl abgeschlossen hat, wird die Funktion nach dem Aktivieren automatisch als erledigt gewertet **(h)**. Damit kann der Benutzer den Next-Button wählen, sobald die Tabelle alle grafischen Elemente des zur Anpassung gewählten Moduls anzeigt. (Das ist insofern wichtig, da es vorkommen kann, dass der Nutzer beim Prüfen feststellt, dass keine Elemente gelöscht werden müssen. Dennoch will er aber gegebenenfalls zum nächsten Prozessschritt des Anpassungsprozesses weitergehen und dazu muss der Next-Button aktiviert sein.)

```
final String funID =
PageID.STEP_ENTSCHEIDEN_SIE_GIBT_ES_ELEMENTE_DIE_ENTFERNT_WERDEN_MUESSEN_;
(b)

Group graphicalElemsGroup = new Group(parent, SWT.NONE);
register(funID, graphicalElemsGroup);

// create & config image-table
final Table elemTable = new Table (...)                                    (c.1)
...
register(funID, elemTable);
clearOnFinish(elemTable);

final Button viewImageButton = GUF.createPushButton(...);                  (c.2)
viewImageButton.setEnabled(false);

/*
 * GUI-user-interaction                                                    (d)
 */
// opens the selected image onClick
viewImageButton.addSelectionListener(new CmdRunner(elemTable, 1));

elemTable.getColumn(0).addSelectionListener(new AGAC_SelectionAdapter() {
   public void widgetSelected(SelectionEvent e) {
        // update table-selection
        ...
        // set the result
        myWizard().setResult(funID, generateResult(elemTable));            (e)
        updateControls();
        setPageComplete(checkCompleteness());
```

```
      }
   }
});
elemTable.addSelectionListener(new AGAC_SelectionAdapter(){
   public void widgetSelected(SelectionEvent e) {
         // update table-selection
         ...
         if (selectionMapper.get(selectedItem) != null) {
            // set the result
            myWizard().setResult(funID, generateResult(elemTable));        (e)
            updateControls();
            setPageComplete(checkCompleteness());
         }
      }
   }
});

/*
 * integrate function into the information-flow                            (f)
 */
// listen to results of previous function
myWizard().addResultListener(
   PageID.
STEP_ALLE_GRAFISCHEN_ELEMENTE_ERMITTELN_Query_atomic_32477808_1275821162,
                                                                           (g.1)
      new ResultListener() {
         public void reportNewResult(String id, Object value) {
            // observed function detects all images
            addResultsToTable(elemTable, viewImageButton, value);          (g.2)
            updateControls();
            setPageComplete(checkCompleteness());
         }}
      );
// regard function as DONE by default
myWizard().addStateChangeListener(funID, new StateChangeListener() {
   @Override
   public void reportNewState(String id, int state) {
      if (state == AutoGenWizard.STATE_SCHEDULED) {
         myWizard().setResult(id, generateResult(elemTable));
         myWizard().setState(id, AutoGenWizard.STATE_DONE);                (h)

         updateControls();
         setPageComplete(checkCompleteness());
      }
   }
});
```

Listing 5: Teil eines Composites nach Automatisierung des atomaren Unterschrittes.

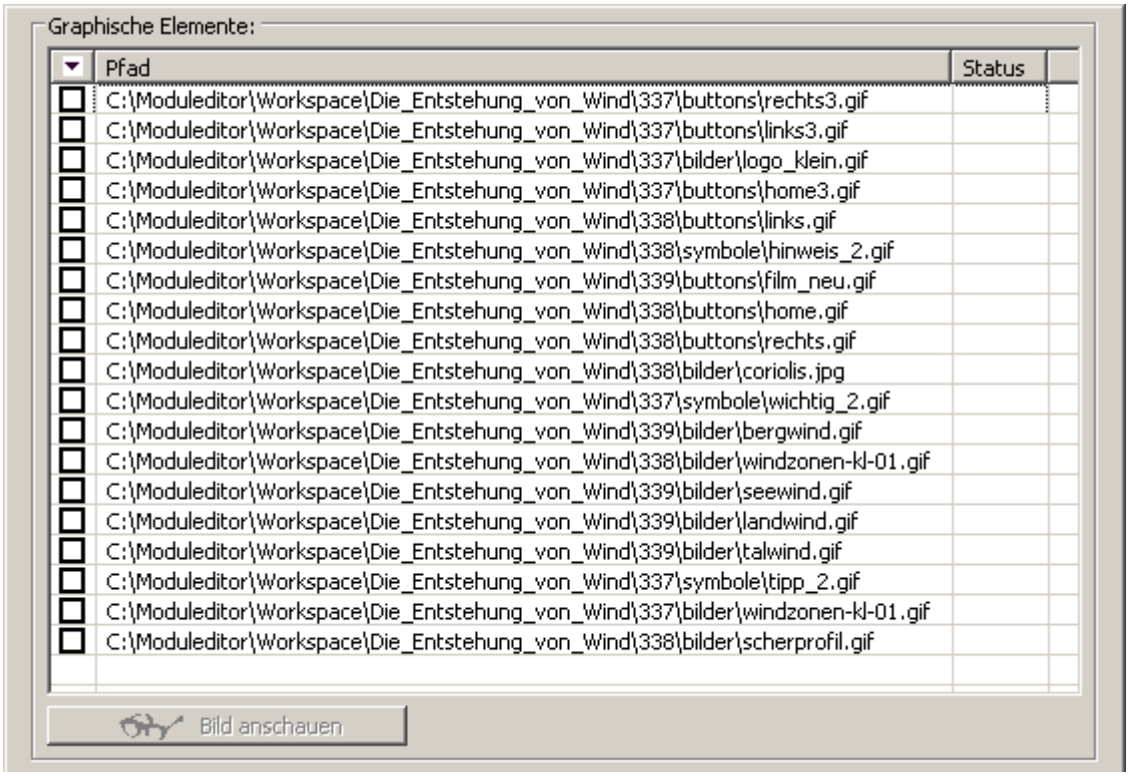


Abbildung 78: Composite nach Automatisierung des zugehörigen Unterschrittes.

Liste automatisiert zur Verfügung stehender Unterschritte

Das Anpassungswerkzeug unterstützt Anwender bei der Anpassung existierender Lernressourcen. Da diese üblicherweise aus einer Reihe von Dateien in unterschiedlichen Formaten bestehen, werden sogenannte Format PlugIns verwendet, um automatisierte Unterschritte formatspezifisch umzusetzen (vergleiche Abschnitt 6.2.2) Derzeit sind drei Format PlugIns für die Automatisierung der atomaren Unterschritte des Anpassungswerkzeuges für HTML, CSS und ein XML-basiertes Format zuständig. Die Wahl fiel auf diese drei Formate, da HTML in vielen Lernressourcen verwendet wird. Meist wird das Layout der HTML-Ressourcen über Stylesheet-Dateien im Format CSS geregelt. XML wurde gewählt, das sehr viele Produzenten von Lerninhalten planen, in der Zukunft XML-basierte Formate für ihre Lerninhalte einzusetzen. Bei XML ist allerdings zu beachten, dass viele Hersteller von Lerninhalten ihre eigenen XSDs oder DTDs verwenden und somit die XML-Ressourcen verschiedener Hersteller unterschiedlich aufgebaut sind.

Im Rahmen der Erweiterung des automatisch vom WGT erzeugten Prototyps für die Anpassungsunterstützung zu einem einsatzfähigen Werkzeug wurden die atomaren Unterschritte von einigen Prozessschritten automatisiert. Nachfolgend wird aufgezählt, für welche Prozessschritte welche atomaren Unterschritte automatisiert zur Verfügung stehen.

Anpassung an ein verändertes (Corporate) Design:

Prozessschritt: Grafische Elemente ersetzen

Atomarer Unterschritt: Finden aller in einer Lernressource enthaltenen grafischen Elemente: Dieser Unterschritt dient dem Ermitteln aller grafischen Elemente (Bilder, Diagramme, Fotos, …) einer Lernressource. Die ermittelten Elemente werden dem Anwender in einer Tabelle angezeigt.

Atomarer Unterschritt: Auswahl der zu ersetzenden grafischen Elemente aus allen gefundenen Elementen: Der Anwender kann in der Tabelle die Elemente auswählen, die ersetzt werden sollen. Um die Auswahl zu erleichtern, ist hier eine Vorschau auf jedes Element verfügbar.

Atomarer Unterschritt: Ersetzen von ausgewählten Elementen durch ein anderes Element: Die Elemente, die der Anwender zur Ersetzung ausgewählt hat, können durch ein anderes Elemente ersetzt werden. Dabei wählt der Anwender aus, durch welches Element das ursprüngliche Element ersetzt werden soll. Ist das neue Element größer als das ursprüngliche, kann der Anwender direkt die Größe anpassen. Es werden alle Vorkommen des Elementes in der gesamten Lernressource ersetzt. Bevor die eigentliche Ersetzung stattfindet, kann der Anwender sich eine Vorschau auf die Änderungen anzeigen lassen. Dabei ist es möglich, einzelne Ersetzungen gezielt rückgängig zu machen.

Prozessschritt: Grafische Elemente löschen

Atomarer Unterschritt: Finden aller in einer Lernressource enthaltenen grafischen Elemente: Dieser Unterschritt dient dem Ermitteln aller grafischen Elemente (Bilder, Diagramme, Fotos, …) einer Lernressource. Die ermittelten Elemente werden dem Anwender in einer Tabelle angezeigt.

Atomarer Unterschritt: Auswahl der zu löschenden grafischen Elemente aus allen gefundenen Elementen: Der Anwender kann in der Tabelle die Elemente auswählen, die gelöscht werden sollen. Um die Auswahl zu erleichtern, ist hier eine Vorschau auf jedes Element verfügbar.

Atomarer Unterschritt: Ersetzen von ausgewählten Elementen durch ein anderes Element: Die Elemente, die der Anwender zum Löschen ausgewählt hat, können gelöscht werden. Es werden alle Vorkommen des Elementes in der gesamten Lernressource gelöscht. Bevor das eigentliche Löschen stattfindet, kann der Anwender sich eine Vorschau auf die Veränderungen anzeigen lassen. Dabei ist es möglich, einzelne Elemente gezielt nicht löschen zu lassen.

Prozessschritt: Änderungen gemäß Design-Richtlinien durchführen

Sub-Prozessschritt: Fonts ändern

Atomarer Unterschritt: Alle benutzten Fonts ermitteln: Dieser Unterschritt dient dem Ermitteln aller Fontangaben einer Lernressource. Dabei werden Schriftart, Schriftgröße und Schriftfarbe ermittelt. Die ermittelten Angaben werden dem Anwender in einer Tabelle angezeigt.

Atomarer Unterschritt: Auswahl der zu ändernden Fontangaben aus allen gefundenen Fontangaben: Der Anwender kann in der Tabelle die Fontangaben auswählen, die geändert werden sollen.

Atomarer Unterschritt: Fontangaben ändern: Alle zum Ändern ausgewählten Angaben können geändert werden. Bevor die eigentliche Änderung stattfindet, kann der Anwender sich eine Vorschau auf die Änderungen anzeigen lassen. Dabei ist es möglich, einzelne Fontangaben gezielt nicht zu ändern.

Sub-Prozessschritt: Hintergrundfarben ändern

Atomarer Unterschritt: Alle benutzten Hintergrundfarben ermitteln: Dieser Unterschritt dient dem Ermitteln aller Hintergrundfarben einer Lernressource. Die ermittelten Angaben werden dem Anwender in einer Tabelle angezeigt.

Atomarer Unterschritt: Auswahl der zu ändernden Hintergrundfarben aus allen gefundenen Fontangaben: Der Anwender kann in der Tabelle die Hintergrundfarben auswählen, die geändert werden sollen.

Atomarer Unterschritt: Hintergrundfarben ändern: Alle zum Ändern ausgewählten Hintergrundfarben können geändert werden. Bevor die eigentliche Änderung stattfindet, kann der Anwender sich eine Vorschau auf die Änderungen anzeigen lassen. Dabei ist es möglich, einzelne Hintergrundfarben gezielt nicht zu ändern.

Sub-Prozessschritt: Hintergrundbilder ändern

Atomarer Unterschritt: Alle benutzten Hintergrundbilder ermitteln: Dieser Unterschritt dient dem Ermitteln aller Hintergrundbilder einer Lernressource. Die ermittelten Angaben werden dem Anwender in einer Tabelle angezeigt.

Atomarer Unterschritt: Auswahl der zu ändernden Hintergrundbilder aus allen gefundenen Fontangaben: Der Anwender kann in der Tabelle die Hintergrundbilder auswählen, die geändert werden sollen.

Atomarer Unterschritt: Hintergrundbilder ändern: Alle zum Ändern ausgewählten Hintergrundbilder können geändert werden. Bevor die eigentliche Änderung stattfindet, kann der Anwender sich eine Vorschau auf die Änderungen anzeigen lassen. Dabei ist es möglich, einzelne Hintergrundbilder gezielt nicht zu ändern.

Sub-Prozessschritt: Bildgrößen ändern

Atomarer Unterschritt: Finden aller in einer Lernressource enthaltenen grafischen Elemente: Dieser Unterschritt dient dem Ermitteln aller grafischen Elemente (Bilder, Diagramme, Fotos, …) einer Lernressource. Die ermittelten Elemente werden dem Anwender in einer Tabelle angezeigt.

Atomarer Unterschritt: Auswahl der grafischen Elemente, deren Größe geändert werden soll, aus allen gefundenen Elementen: Der Anwender kann in der Tabelle Element für Element alle Elemente auswählen, deren Größe geändert werden soll. Um die Auswahl zu erleichtern, ist hier eine Vorschau auf jedes Element verfügbar.

Atomarer Unterschritt: Ändern der Größe der ausgewählten Elemente: Für jedes Element, das der Anwender zur Ersetzung ausgewählt hat, kann er die neue Größe festlegen. Eine Vorschau zeigt die Auswirkung der Änderungen. So wird es ermöglicht, nur die Elemente zu ändern, bei denen die Größenänderung wie gewünscht durchgeführt werden kann.

Anpassung an eine veränderte Terminologie:

Prozessschritt: Alle Vorkommen eines Begriffes ersetzen

Atomarer Unterschritt: Festlegen, welcher Begriff durch welchen neuen Begriff ersetzt werden soll: Dieser Unterschritt erlaubt es dem Anwender, festzulegen, welcher Begriff ersetzt werden soll und was der neue Begriff ist.

Atomarer Unterschritt: Suchen aller Vorkommen des Begriffes: Es werden alle Vorkommen des Begriffes ermittelt. Dem Anwender werden alle Textausschnitte, die den Begriff enthalten, in einer Tabelle angezeigt.

Atomarer Unterschritt: Ersetzung aller Vorkommen des zu ändernden Begriffes: Alle Vorkommen des zu ändernden Begriffes werden durch den neuen Begriff ersetzt. Eine Vorschau zeigt die Auswirkung der Änderungen. So kann der Anwender gezielt einzelne Änderungen nicht vornehmen lassen.

Prozessschritt: Alle Vorkommen mehrerer Begriffe ersetzen

Atomarer Unterschritt: Festlegen, welche Begriffe durch welche neuen Begriffe ersetzt werden soll: Dieser Unterschritt erlaubt es dem Anwender, eine Liste aller zu ersetzenden Begriffe sowie deren Ersetzung festzulegen. Der Anwender kann alle in der Liste enthaltenen Begriffe sehen, er kann Begriffe zufügen, ändern oder löschen.

Atomarer Unterschritt: Suchen aller Vorkommen der Begriffe: Es werden alle Vorkommen der zu ersetzenden Begriffe ermittelt. Dem Anwender werden alle Textausschnitte, die die Begriffe enthalten, in einer Tabelle angezeigt.

Atomarer Unterschritt: Ersetzung aller Vorkommen des zu ändernden Begriffes: Alle Vorkommen der zu ändernden Begriffe werden durch die neuen Begriffe ersetzt. Eine Vorschau zeigt die Auswirkung der Änderungen. So kann der Anwender gezielt einzelne Änderungen nicht durchführen lassen.

Anpassung zum Erzeugen einer Druckversion:

Prozessschritt: Prüfen ob alle Elemente auf ein gewähltes Papierformat passen

Atomarer Unterschritt: Festlegen des Seitenformates: Hier kann der Anwender das gewünschte Papierformat (z.B. DIN A4) festlegen und die Seitenausrichtung (Hochformat oder Querformat).

Atomarer Unterschritt: Überprüfen der teilbaren Elemente: Für alle Elemente, die sich auf zwei Seiten aufteilen lassen (z.B. Tabellen), wird geprüft, ob sie auf das gewünschte Zielformat passen. Die Elemente, die nicht passen, werden dem Benutzer in einer Tabelle angezeigt.

Atomarer Unterschritt: Auswählen der teilbaren Elemente, die angepasst werden sollen: Aus allen Elementen, die nicht passen, können die zu verändernden Elemente ausgewählt werden.

Atomarer Unterschritt: Überprüfen der unteilbaren Elemente: Für alle Elemente, die sich nicht auf zwei Seiten aufteilen lassen (z.B. Bilder), wird geprüft, ob sie auf das gewünschte Zielformat passen. Die Elemente, die nicht passen, werden dem Benutzer in einer Tabelle angezeigt.

Atomarer Unterschritt: Auswählen der teilbaren Elemente, die angepasst werden sollen: Aus allen Elementen, die nicht passen, können die zu verändernden Elemente ausgewählt werden.

Atomarer Unterschritt: Korrigieren der teilbaren Elemente, die angepasst werden sollen: Bei Elementen, die auf zwei Seiten aufgeteilt werden können, muss nur die Breite geändert werden. Die Höhe kann auf zwei Seiten verteilt werden. Der Nutzer bekommt gesagt, wie breit das Element derzeit ist und welche Breite maximal erlaubt ist.

Atomarer Unterschritt: Korrigieren der unteilbaren Elemente, die angepasst werden sollen: Bei Elementen, die nicht auf zwei Seiten aufgeteilt werden können, müssen Breite und Höhe geändert werden. Der Nutzer bekommt gesagt, wie breit und hoch das Element derzeit ist und welche Breite und Höhe maximal erlaubt sind.

Atomarer Unterschritt: Anpassungen durchführen: Alle gewählten Größenänderungen werden vom Werkzeug durchgeführt. Bevor die eigentlichen Änderungen stattfinden, kann der Anwender sich eine Vorschau auf die Änderungen anzeigen lassen. Dabei ist es möglich, einzelne Änderungen gezielt nicht durchführen zu lassen.

Übersetzung:

Prozessschritt: Sprache bestimmen

Atomarer Unterschritt: Sprache erkennen: Hier wird über eine semantische Anreicherungskomponente ermittelt, welche Sprachen in der Lernressource vorkommen. Die ermittelten Sprachen werden dem Anwender in einer Tabelle gezeigt.

Atomarer Unterschritt: Sprache auswählen: Der Anwender kann aus der Tabelle eine Sprache wählen.

Atomarer Unterschritt: Textstellen in der gewählten Sprache anzeigen lassen: Zur in der Tabelle gewählten Sprache werden alle Textstellen in dieser Sprache gezeigt.

Prozessschritt: Texte exportieren

Atomarer Unterschritt: Texte exportieren: Der Anwender kann sich eine Lernressource zerlegt in einzelne Textstellen in eine csv-Datei exportieren lassen. Diese Datei kann in andere Werkzeuge oder in das Anpassungswerkzeug importiert werden, um mit dieser Datei die Übersetzung manuell durchführen zu können.

Prozessschritt: Texte übersetzen

Atomarer Unterschritt: Textdatei importieren: Die vorher exportierte csv-Datei kann in das Anpassungswerkzeug importiert werden. Es werden alle Textstellen der Datei angezeigt.

Atomarer Unterschritt: Durch die Textdatei navigieren: Der Anwender kann Textstelle für Textstelle durch die Datei navigieren. Existiert zu der Textstelle bereits eine Übersetzung, wird diese angezeigt.

Atomarer Unterschritt: Übersetzung angeben: Existiert für eine Textstelle noch keine Übersetzung, kann der Nutzer eine Übersetzung angeben.

Atomarer Unterschritt: Übersetzungen in Textdatei übernehmen: Alle Übersetzungen werden in die csv-Datei übernommen und gespeichert.

Prozessschritt: *Übersetzte Texte einfügen*

Atomarer Unterschritt: Übersetzungsdatei importieren: Die in einer csv-Datei enthaltenen Übersetzungen können in das Anpassungswerkzeug importiert werden.

Atomarer Unterschritt: Änderungen vornehmen: Die in der csv-Datei enthaltenen Übersetzungen werden in die Lernressource übernommen.

Anhang D: Beispiel einer Prozessbeschreibung zur Reisebuchung

Nachfolgend wird zuerst das XML-Dokument der Prozessbeschreibung gezeigt, danach der Prozessgraph. Anschließend werden einige Screenshots des automatisch mit dem WGT erzeugten Wizards gezeigt.

Der Prozess der Reisebuchung wurde auch in diesem Beispiel vereinfacht beschrieben: Es sollte nur eine Pauschalreise gebucht werden. Dabei wurden aber Prozessschritte und die darin enthaltenen atomaren Unterschritte detailliert beschrieben.

```
<?xml version="1.0" encoding="UTF-8"?>
<patterns>
   <pattern id="process_pattern$16197143" level-of-confidence="0"
            name="Pauschalreise buchen">
      <intent>Eine Pauschalreise ermitteln und buchen</intent>
      <context/>
      <problem>Wie geht man vor, um eine Pauschalreise zu buchen?
      </problem>
      <example_illustration/>
      <example_explanation/>
      <forces/>
      <solution>Um eine Pauschalreise zu buchen, muss man zunächst
         das Reiseziel aussuchen. Wenn das Reiseziel feststeht, müssen
         mögliche Pauschalreisen ermittelt werden. Dann muss man sich
         für eine Reise entscheiden und diese buchen. Evtl. kann man
         dann noch einen Mietwagen buchen.
      </solution>
      <process_steps>
         <process_step name="Reiseziel aussuchen" mandatory="true"/>
         <process_step name="Wurde ein passendes Reiseziel gefunden?"
         mandatory="true"/>
         <process_step name="Mögliche Pauschalreisen ermitteln"
         mandatory="true"/>
         <process_step name="Wurde ein passendes Reiseangebot
         gefunden?” mandatory="true"/>
         <process_step name="Reise buchen" mandatory="true"/>
      </process_steps>
      <known_uses/>
      <consequences>
         <positive_consequence name="ein Reiseziel ausgesucht"/>
         <positive_consequence name="eine Pauschalreise gefunden"/>
         <positive_consequence name="Reise gebucht"/>
         <negative_consequence name="Zeit investiert"/>
         <negative_consequence name="Geld ausgegeben"/>
      </consequences>
      <related_patterns/>
      <connected_patterns/>
      <used_patterns/>
      <additional_information/>
   </pattern>
   <processFragment id="Prozess_Schritt$48380865"
                    name="Mögliche Pauschalreisen ermitteln">
```

```
      <intent>Unter verschiedenen Angeboten soll das gefunden werden,
         das am ehesten den eigenen Bedürfnissen entspricht</intent>
      <proceeding>In ein Reisebüro gehen, Kataloge mitnehmen, Kataloge
      durchsuchen und für ein bestimmtes Ziel entscheiden
      </proceeding>
      <sub_steps>
         <sub_step name="In ein Reisebüro gehen" mandatory="false"/>
         <sub_step name="Nach Katalogen fragen" mandatory="true"/>
         <sub_step name="Kataloge mitnehmen" mandatory="true"/>
         <sub_step name="In Katalogen nach passenden Angeboten suchen"
         mandatory="true"/>
         <sub_step name="Ein Internetportal öffnen" mandatory=
         "false"/>
         <sub_step name="Im Internetportal nach einem passenden Ange-
         bot suchen" mandatory="true" />
         <sub_step name="Sich für ein Ziel entscheiden" mandatory=
         "true"/>
      </sub_steps>
   </processFragment>
   <processFragment id="Prozess_Schritt$47431278"
                    name="Reiseziel aussuchen">
      <intent>Den Ort ermitteln, zu dem die Reise führen soll</intent>
      <proceeding>Ich überlege: welches Land wollte ich immer schon
         einmal besuchen? Dann besorge ich mir Reiseführer über dieses
         Land und suche nach einem Ort, zu dem ich gerne reisen würde.
         Danach entscheide ich mich für einen bestimmten Ort.
      </proceeding>
      <sub_steps>
         <sub_step name="Reiseland ermitteln" mandatory="true"/>
         <sub_step name="Reiseführer besorgen" mandatory="false"/>
         <sub_step name="Sich für einen bestimmten Ort entscheiden"
         mandatory="true"/>
      </sub_steps>
   </processFragment>
   <processFragment id="Prozess_Schritt$34514792" name="Reise buchen">
      <intent>Eine ausgewählte Reise soll gebucht werden </intent>
      <proceeding>Zum Reisebüro gehen die ausgewählte Reise buchen
      </proceeding>
      <sub_steps>
         <sub_step name="Zum Reisebüro gehen" mandatory="false"/>
         <sub_step name="Nach der ausgewählten Reise fragen"
         mandatory="true"/>
         <sub_step name="Das Internetportal, in dem man die Reise
         gefunden hat, öffnen" mandatory="false"/>
         <sub_step name="Die ausgewählte Reise auswählen" mandatory=
         "true"/>
         <sub_step name="Ist die gewünschte Reise buchbar?" mandatory=
         "true"/>
         <sub_step name="Die Reise buchen" mandatory="true"/>
         <sub_step name="Eine Anzahlung leisten" mandatory="false"/>
      </sub_steps>
   </processFragment>
   <atomicOperation id="Ausführung$22770687"
                    name="In Katalogen nach passenden Angeboten suchen">
      <description>Zu Hause die Kataloge nach möglichen Reisen
         durchsuchen</description>
```

```
</atomicOperation>
<atomicOperation id="Ausführung$38187869"
                 name="Eine Anzahlung leisten">
   <description>Wenn erforderlich, eine Anzahlung auf die gebuchte
      Reise leisten</description>
</atomicOperation>
<atomicOperation id="Ausführung$43459788"
                 name="Nach der ausgewählten Reise fragen">
   <description>Nach dem ausgewählten Reiseziel
      fragen</description>
</atomicOperation>
<atomicOperation id="Ermittlung$26531635"
                 name="Im Internetportal nach einem passenden
                 Angebot suchen">
   <description>Im Internetportal nach passenden Angeboten
      suchen</description>
</atomicOperation>
<atomicOperation id="Entscheidung$29324487"
                 name="Die Reise buchen">
   <description>Die Reise buchen lassen und die entsprechenden
      Unterlagen aushändigen lassen</description>
</atomicOperation>
<atomicOperation id="Ausführung$46953601"
                 name="Zum Reisebüro gehen">
   <description>Zum Reisebüro meiner Wahl gehen</description>
</atomicOperation>
<atomicOperation id="Entscheidung$32714404"
                 name="Reiseland ermitteln">
   <description>Überlegen,in welches Land die Reise gehen soll
   </description>
</atomicOperation>
<atomicOperation id="Ermittlung$28321328"
                 name="Die ausgewählte Reise auswählen">
   <description>Die gewählte Reise erneut aufrufen</description>
</atomicOperation>
<atomicOperation id="Entscheidung$36133453"
                 name="Ist die gewünschte Reise buchbar?">
   <description>Ist die Reise, die man gern buchen möchte, noch
      buchbar?</description>
</atomicOperation>
<atomicOperation id="Ausführung$48278416"
                 name="Ein Internetportal öffnen">
   <description>Im Internet ein Reiseportal, mit dem man gut
      umgehen kann, öffnen</description>
</atomicOperation>
<atomicOperation id="Entscheidung$1809534391"
                 name="Wurde ein passendes Reiseziel gefunden?">
   <description>Wurde ein passendes Reiseziel
     gefunden?</description>
</atomicOperation>
<atomicOperation id="Ermittlung$26018484" name="Reiseführer
 besorgen">
   <description>In einer Bibliothek alle verfügbaren Reiseführer
      über das Land ausleihen oder in einer Buchhandlung die
      Reiseführer kaufen, die einem zusagen</description>
</atomicOperation>
```

```
   <atomicOperation id="Ausführung$53475378"
                    name="Das Internetportal, in dem man die Reise
                    gefunden hat, öffnen">
      <description>Das Internetportal, das man bereits zum ermitteln
         der Angebote geöffnet hatte, erneut öffnen</description>
   </atomicOperation>
   <atomicOperation id="Entscheidung$30946577"
                    name="Sich für ein Ziel entscheiden">
      <description>Sich für ein Reiseziel und für eine bestimmte Reise
         entscheiden</description>
   </atomicOperation>
   <atomicOperation id="Entscheidung$30385245"
                    name="Sich für einen bestimmten Ort entscheiden">
      <description>Sich mit Hilfe des Reiseführers für einen
         bestimmten Ort entscheiden</description>
   </atomicOperation>
   <atomicOperation id="Ausführung$44272749"
                    name="In ein Reisebüro gehen">
        <description>In ein Reisebüro meiner Wahl gehen</description>
   </atomicOperation>
   <atomicOperation id="Entscheidung$723661569"
                    name="Wurde ein passendes Reiseangebot gefunden?">
      <description>Wurde ein passendes Reiseangebot gefunden?
      </description>
   </atomicOperation>
   <atomicOperation id="Ermittlung$37764746"
                    name="Nach Katalogen fragen">
      <description>Nach Katalogen über das ausgewählte Land und über
         den ausgewählten Ort fragen</description>
   </atomicOperation>
   <atomicOperation id="Ausführung$26300639"
                    name="Kataloge mitnehmen">
      <description>Kataloge einpacken und mit nach Hause nehmen
   </description>
   </atomicOperation>
</patterns>
```

Listing 6: XML-Datei für die Prozessbeschreibung „Reise buchen“.

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definitions>
   <process id="process_pattern$16197143" process-pattern="true">
      <requires fragmentRef="Prozess_Schritt$47431278"
      mandatory="true"/>
      <requires functionRef="Entscheidung$1809534391"
      mandatory="true">
         <precondition fragmentRef="Prozess_Schritt$47431278"/>
      </requires>
      <requires fragmentRef="Prozess_Schritt$48380865"
      mandatory="true">
         <precondition functionRef="Entscheidung$1809534391"
      value="true"/>
      </requires>
      <requires functionRef="Entscheidung$723661569" mandatory="true">
         <precondition fragmentRef="Prozess_Schritt$48380865"/>
      </requires>
```

```
      <requires fragmentRef="Prozess_Schritt$34514792"
      mandatory="true">
         <precondition functionRef="Entscheidung$723661569"
      value="true"/>
      </requires>
   </process>
   <process-fragment id="Prozess_Schritt$47431278">
      <requires functionRef="Entscheidung$32714404" mandatory="true"/>
      <requires functionRef="Ermittlung$26018484" mandatory="false">
         <precondition functionRef="Entscheidung$32714404"/>
      </requires>
      <requires functionRef="Entscheidung$30385245" mandatory="true">
         <precondition functionRef="Entscheidung$32714404"/>
      </requires>
   </process-fragment>
   <process-fragment id="Prozess_Schritt$48380865">
      <requires functionRef="Ausführung$44272749" mandatory="false"/>
      <requires functionRef="Ermittlung$37764746" mandatory="true">
         <precondition functionRef="Ausführung$44272749"/>
      </requires>
      <requires functionRef="Ausführung$26300639" mandatory="true">
         <precondition functionRef="Ermittlung$37764746"/>
      </requires>
      <requires functionRef="Ausführung$22770687" mandatory="true">
         <precondition functionRef="Ausführung$26300639"/>
      </requires>
      <requires functionRef="Ausführung$48278416" mandatory="false"/>
      <requires functionRef="Ermittlung$26531635" mandatory="true">
         <precondition functionRef="Ausführung$48278416"/>
      </requires>
      <requires functionRef="Entscheidung$30946577" mandatory="true">
         <precondition functionRef="Ausführung$22770687"/>
         <precondition functionRef="Ermittlung$26531635"/>
      </requires>
   </process-fragment>
   <process-fragment id="Prozess_Schritt$34514792">
      <requires functionRef="Ausführung$46953601" mandatory="false"/>
      <requires functionRef="Ausführung$43459788" mandatory="true">
         <precondition functionRef="Ausführung$46953601"/>
      </requires>
      <requires functionRef="Ausführung$53475378" mandatory="false"/>
      <requires functionRef="Ermittlung$28321328" mandatory="true">
         <precondition functionRef="Ausführung$53475378"/>
      </requires>
      <requires functionRef="Entscheidung$36133453" mandatory="true">
         <precondition functionRef="Ausführung$43459788"/>
         <precondition functionRef="Ermittlung$28321328"/>
      </requires>
      <requires functionRef="Entscheidung$29324487" mandatory="true">
         <precondition functionRef="Entscheidung$36133453"
         value="true"/>
      </requires>
      <requires functionRef="Ausführung$38187869" mandatory="false"/>
   </process-fragment>
   <function fftKey="" id="Entscheidung$32714404" type="decision"/>
   <function fftKey="" id="Ermittlung$26018484" type="query"/>
```

```
   <function fftKey="" id="Entscheidung$30385245" type="decision"/>
   <function fftKey="" id="Entscheidung$1809534391" type="decision"/>
   <function fftKey="" id="Ausführung$44272749" type="execution"/>
   <function fftKey="" id="Ermittling$37764746" type="query"/>
   <function fftKey="" id="Ausführung$26300639" type="execution"/>
   <function fftKey="" id="Ausführung$22770687" type="execution "/>
   <function fftKey="" id="Ausführung$48278416" type="execution"/>
   <function fftKey="" id="Ermittlung$26531635" type="query"/>
   <function fftKey="" id="Entscheidung$30946577" type="decision"/>
   <function fftKey="" id="Entscheidung$723661569" type="decision"/>
   <function fftKey="" id="Ausführung$46953601" type="execution"/>
   <function fftKey="" id="Ausführung$43459788" type="execution "/>
   <function fftKey="" id="Ausführung$53475378" type="execution"/>
   <function fftKey="" id="Ermittlung$28321328" type="query"/>
   <function fftKey="" id="Entscheidung$36133453" type="decision"/>
   <function fftKey="" id="Entscheidung$29324487" type="decision"/>
   <function fftKey="" id="Ausführung$38187869" type="execution"/>
</process-definitions>
```

Listing 7: Prozessgraph für die Prozessbeschreibung „Reise buchen“.

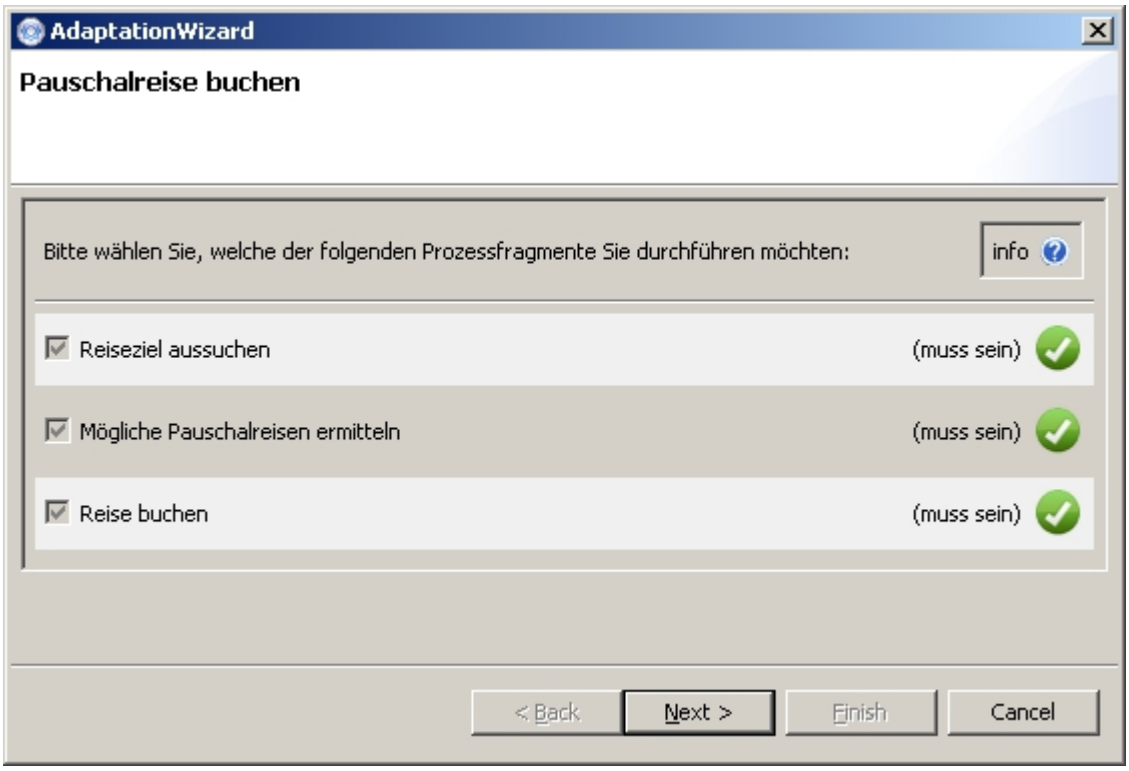


Abbildung 79: Startseite des prototypischen Wizards für den Prozess „Reise buchen“.

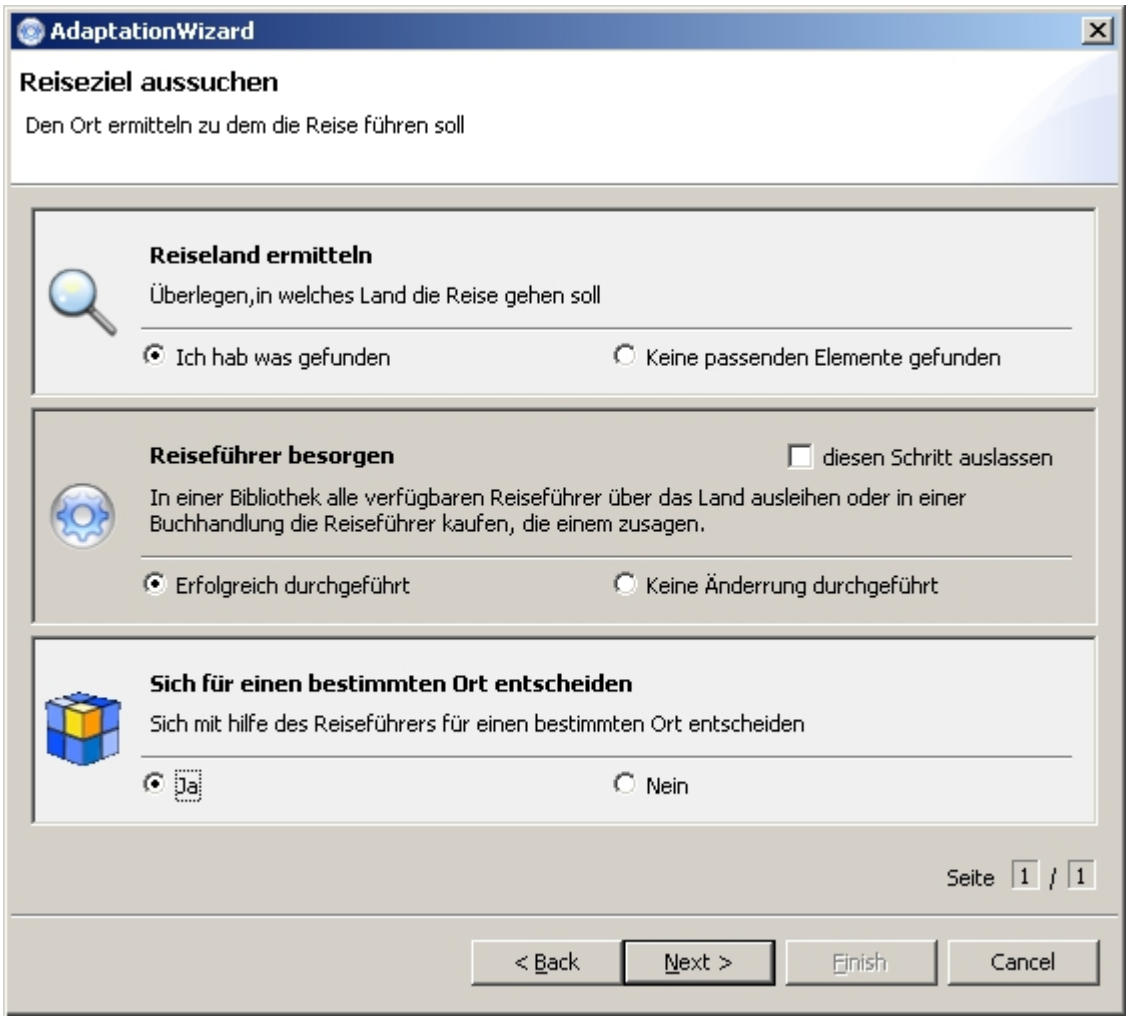


Abbildung 80: Erster Prozessschritt: „Reiseziel aussuchen“.

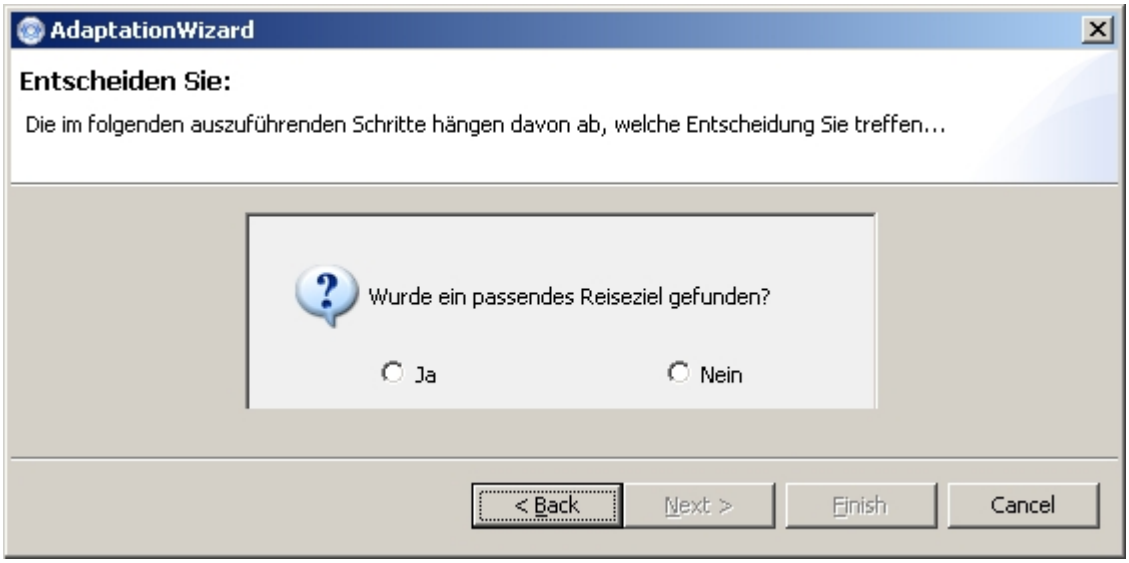


Abbildung 81: Entscheidung: „Reiseziel gefunden?“.

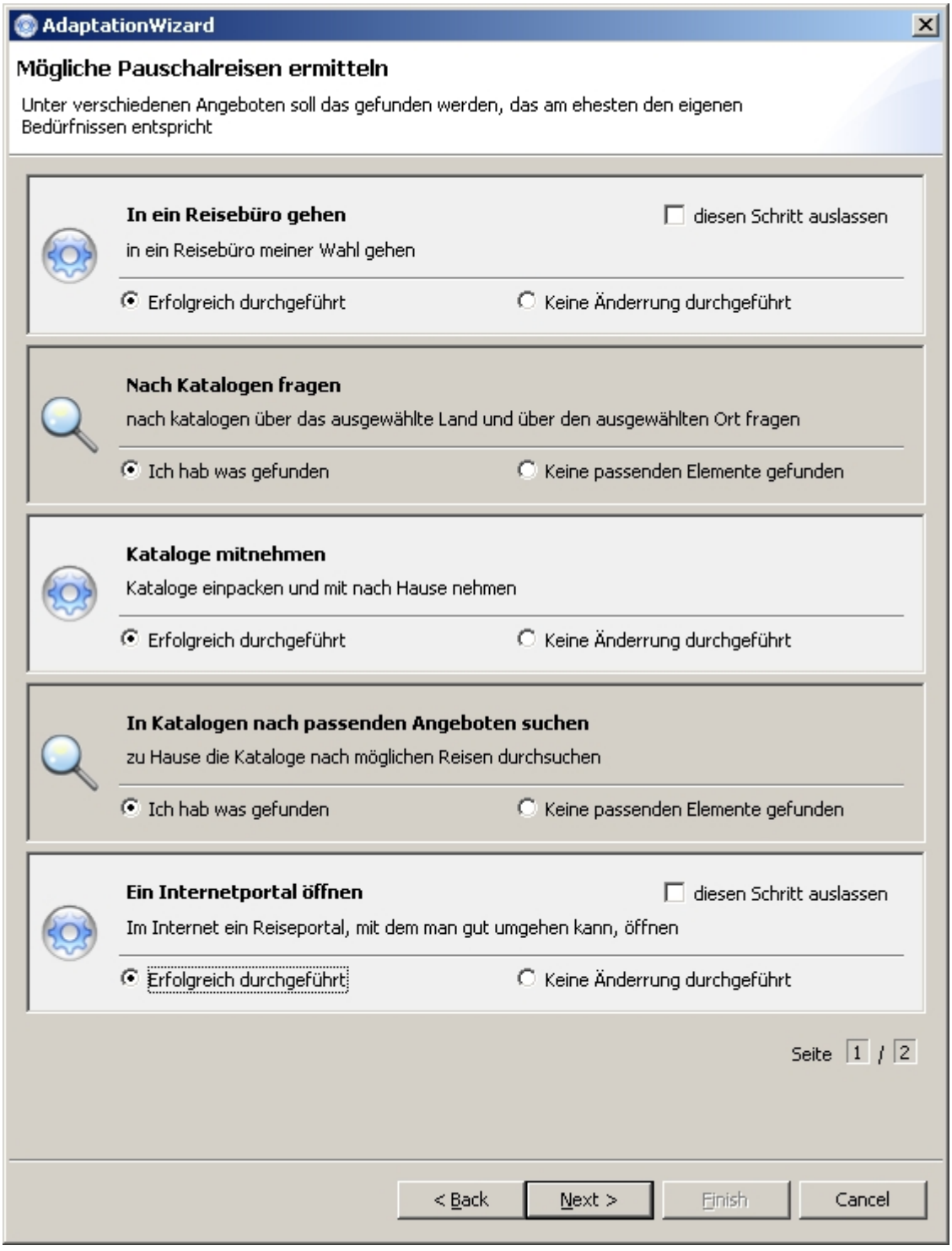


Abbildung 82: Zweiter Prozessschritt: „Mögliche Pauschalreise ermitteln“.

Anhang E: Zusätzliche Informationen zur Evaluation

Fragebogen zum Test des PIT

Um von den Testpersonen deren Einschätzung des Werkzeuges PIT zu erfragen, wurden die Testeilnehmer nach dem Test gebeten, einen Fragebogen auszufüllen (vergleiche Abschnitt 8.1). Der Fragebogen war online verfügbar. Er teilte sich in insgesamt vier Teile auf, die nachfolgend gezeigt werden.

1. Teil: Persönliche Daten

Frage	Pflicht-feld?	Antworttyp	Mögliche Antworten
Identifikationsnummer	Ja	Freitext	
Wie alt sind Sie?	Ja	Freitext	
Ihr Geschlecht	Ja	Auswahl	Weiblich / Männlich
Welchen Schulabschluss besitzen Sie?	Ja	Mehrfachauswahl	Hauptschul- oder Volksschulabschluss Mittlere Reife Fachhochschulreife Allgemeine Hochschulreife
Berufliche Ausbildung	Ja	Mehrfachauswahl mit Möglichkeit Freitext anzugeben	Keine Ausbildung Ausbildung als Studium in Fach Promotion in Fach Sonstiges (mit Freitextfeld)
Aktuelle berufliche Tätigkeit	Ja	Freitext	

Tabelle 17: Erster Teil des Fragebogens.

2. Teil: PC - Vorkenntnisse

Frage	Pflicht -feld?	Antworttyp	Mögliche Antworten
Wie viel Zeit verbringen Sie täglich an Ihrem	Ja	Auswahl	Weniger als 1 Stunde 1 bis 3 Stunden

PC?			3 bis 5 Stunden Mehr als 5 Stunden
Wofür nutzen Sie den PC?	Nein	Mehrfachauswahl	Textverarbeitung (z.B. Word) Tabellenkalkulation (z.B. Excel) Umfrageauswertung (z.B. SPSS) Präsentationen (z.B. PowerPoint) Internet E-mail Spielen Sonstiges (mit Freitextfeld)
Haben Sie schon einmal mit Prozessmodellierung (z.B. UML oder ARIS) gearbeitet?	Ja	Auswahl	Ja / Nein
Haben Sie schon einmal programmiert?	Ja	Auswahl	Ja / Nein

Tabelle 18: Zweiter Teil des Fragebogens.

3. Teil: Bewertung PIT

Frage	**Pflicht-feld?**	**Antworttyp**	**Mögliche Antworten**
Wie hilfreich war die Anleitung für die Arbeit mit PIT?	Ja	Auswahl	Skala von 6 (Sehr hilfreich) bis 1 (Gar nicht hilfreich)
War die Anordnung der Informationen auf dem Bildschirm übersichtlich?	Ja	Auswahl	Skala von 6 (Sehr übersichtlich) bis 1 (Gar nicht übersichtlich)
Hat sich das Programm immer so verhalten, wie Sie das erwartet haben?	Ja	Auswahl	Skala von 6 (Immer) bis 1 (Nie)
War die Bedienung von PIT verständlich?	Ja	Auswahl	Skala von 6 (Sehr verständlich) bis 1 (Gar nicht verständlich)

Wenn die Bedienung für Sie nicht verständlich war: Was war für Sie unverständlich?	Nein	Freitext	
Enthält das Programm PIT alle für die Aufgabe benötigten Funktionen?	Ja	Auswahl	Skala von 6 (Alle benötigten Funktionen vorhanden) bis 1 (Keine benötigten Funktionen vorhanden)
Falls PIT nicht alle benötigten Funktionen enthält: Welche haben Ihnen gefehlt?	Nein	Freitext	
Hatten Sie das Gefühl bei der Arbeit mit PIT, Eingaben gemacht zu haben, welche Ihnen eigentlich überflüssig erschienen?	Ja	Auswahl	Skala von 6 (Keine überflüssigen Eingaben) 1 (Viele überflüssigen Eingaben)
Falls Ihnen Eingaben als überflüssig erschienen sind: Welche waren das?	Nein	Freitext	
Wie hoch war der Aufwand zum Beseitigen von Fehlern?	Ja	Auswahl	Skala von 6 (Kein größerer Aufwand) bis 1 (Sehr großer Aufwand)
Haben Sie generelle Anmerkungen und Anregungen zum Werkzeug PIT?	Nein	Freitext	

Tabelle 19: Dritter Teil des Fragebogens.

4. Teil: Allgemeine Fragen zum Test

Frage	**Pflicht-feld?**	**Antworttyp**	**Mögliche Antworten**
Haben Sie während der Arbeit mit PIT Eingaben gemacht, welche Sie im Nachhinein noch mal ändern würden?	Ja	Auswahl	Ja / Nein
Falls Sie noch Dinge ändern würden: Welche sind das?	Nein	Freitext	

Wollen Sie noch allgemeine Anmerkungen zur Aufgabe mit PIT oder zur Durchführung machen?	Nein	Freitext	
Haben Sie Anmerkungen zum erzeugten Wizard?	Nein	Freitext	

Tabelle 20: Vierter Teil des Fragebogens.

Auswertung des Fragebogens zum PIT

In Abschnitt 8.1 wurde bereits die Faktorenanalyse für den Fragebogen und die Auswertung der Fragebögen im Hinblick auf die beiden Faktoren (Nutzerzufriedenheit und Bewertung der Funktionalität) vorgestellt. An dieser Stelle wird die Auswertung der Fragebögen in Bezug auf die Auswahl-Fragen des dritten Teils gezeigt. Dabei werden jeweils Mittelwert, Standardabweichung, Minimum und Maximum für beide Gruppen (ITler und Nicht-ITler) getrennt aufgelistet.

Frage zu:	**Gruppe**	**Typ des Wertes**	**Wert**
Hilfreiche Anleitung	ITler	Mittelwert	5,19
		Standardabweichung	0,834
		Minimum	4
		Maximum	6
	Nicht- ITler	Mittelwert	4,81
		Standardabweichung	0,911
		Minimum	3
		Maximum	6
Übersichtliche Informationsanordnung	ITler	Mittelwert	5,00
		Standardabweichung	0,730
		Minimum	4
		Maximum	6
	Nicht- ITler	Mittelwert	4,94
		Standardabweichung	0,998
		Minimum	3

		Maximum	6
Verständliche Bedienung	ITler	Mittelwert	5,13
		Standardabweichung	0,719
		Minimum	4
		Maximum	6
	Nicht- ITler	Mittelwert	4,88
		Standardabweichung	0,885
		Minimum	3
		Maximum	6
Vorhersehbares Programmverhalten	ITler	Mittelwert	4,75
		Standardabweichung	0,856
		Minimum	3
		Maximum	6
	Nicht- ITler	Mittelwert	4,81
		Standardabweichung	1,223
		Minimum	3
		Maximum	6
Aufwand Fehlerbeseitigung	ITler	Mittelwert	5,00
		Standardabweichung	0,816
		Minimum	4
		Maximum	6
	Nicht- ITler	Mittelwert	4,94
		Standardabweichung	1,289
		Minimum	2
		Maximum	6
Enthält benötigte Funktionen	ITler	Mittelwert	5,31
		Standardabweichung	1,352
		Minimum	2
		Maximum	6
	Nicht- ITler	Mittelwert	5,69
		Standardabweichung	0,602

		Minimum	4
		Maximum	6
Überflüssige Eingaben	ITler	Mittelwert	4,50
		Standardabweichung	1,366
		Minimum	3
		Maximum	6
	Nicht-ITler	Mittelwert	4,81
		Standardabweichung	1,047
		Minimum	3
		Maximum	6

Tabelle 21: Auswertung der Auswahlfragen im dritten Teil des Fragebogens.

Nachfolgend werden weiterhin die Kreuztabellen für die Auswahl-Fragen zu Teil 3 gezeigt. Aus diesen Tabellen lässt sich ablesen, wie viele der Personen der beiden Testgruppen (ITler und Nicht-ITler) welche Antworten genannt haben. (Die Antwort mit der Ziffer 6 entspricht der besten Bewertung, die mit der Ziffer 1 der schlechtesten.)

Frage	**Gewählte Antwort**	**Anzahl Nennungen**		
		ITler	**Nicht-ITler**	**Gesamt**
Hilfreiche Anleitung	3	1	0	1
	4	2	7	9
	5	6	5	11
	6 sehr hilfreich	7	4	11
Übersichtliche Informationsanordnung	3	1	1	2
	4	2	4	6
	5	8	7	15
	6 sehr übersichtlich	5	4	9
Verständliche Bedienung	3	0	1	1
	4	1	6	7
	5	10	5	15
	6 sehr verständlich	5	4	9
Vorhersehbares Programmverhal-	3	0	5	5
	4	3	3	6

ten	5	8	4	12
	6 immer wie erwartet	5	4	9
Aufwand Fehlerbeseitigung	2	0	1	1
	3	1	1	2
	4	3	3	6
	2	0	1	1
	6 kein großer Aufwand	6	6	12
Enthält benötigte Funktionen	2	0	1	1
	3	0	2	2
	4	0	1	1
	5	2	2	4
	6 keine weiteren Funktionen nötig	14	10	24
Überflüssige Eingaben	3	4	5	9
	4	2	1	3
	5	4	6	10
	6 keine überflüssigen Eingaben	6	4	10

Tabelle 22: Kreuztabelle zu den Auswahlfragen im dritten Teil des Fragebogens.

Fragebogen zum Test des Anpassungswerkzeugs / Netscape Composers

Um von den Testpersonen deren Einschätzung des für die Anpassung verwendeten Werkzeuges (Anpassungswerkzeug bzw. Netscape Composer) zu erfragen, wurden die Testeilnehmer nach dem Test gebeten, einen Fragebogen auszufüllen (vergleiche Abschnitt 8.2). Der Fragebogen war online verfügbar. Er teilte sich in insgesamt vier Teile auf, die nachfolgend gezeigt werden.

In den Fragen zu Teil 3 finden Sie hier immer den Text „Anpassungswerkzeug bzw. Composer“. In dem Fragebogen, den die Versuchsteilnehmer erhalten hatten, wurde an dieser Stelle jeweils das Werkzeug genannt, mit dem der Teilnehmer gearbeitet hatte.

1. Teil: Persönliche Daten

Frage	Pflicht-feld?	Antworttyp	Mögliche Antworten
Identifikationsnummer	Ja	Freitext	
Wie alt sind Sie?	Ja	Freitext	
Ihr Geschlecht	Ja	Auswahl	Weiblich / Männlich
Welchen Schulabschluss besitzen Sie?	Ja	Mehrfachauswahl	Hauptschul- oder Volks-schulabschluss Mittlere Reife Fachhochschulreife Allgemeine Hochschul-reife
Berufliche Ausbildung	Ja	Mehrfachauswahl mit Möglichkeit Freitext anzuge-ben	Keine Ausbildung Ausbildung als Studium in Fach Promotion in Fach Sonstiges (mit Freitext-feld)
Aktuelle berufliche Tä-tigkeit	Ja	Freitext	

Tabelle 23: Erster Teil des Fragebogens.

2. Teil: PC - Vorkenntnisse

Frage	Pflicht-feld?	Antworttyp	Mögliche Antworten
Wie viel Zeit verbringen Sie täglich an Ihrem PC?	Ja	Auswahl	Weniger als 1 Stunde 1 bis 3 Stunden 3 bis 5 Stunden Mehr als 5 Stunden
Wofür nutzen Sie den PC?	Nein	Mehrfachauswahl	Textverarbeitung (z.B. Word) Tabellenkalkulation (z.B. Excel) Umfrageauswertung (z.B. SPSS)

			Präsentationen (z.B. Power-Point) Internet E-mail Spielen Sonstiges (mit Freitext-feld)
Haben Sie schon einmal programmiert?	Ja	Auswahl	Ja / Nein

Tabelle 24: Zweiter Teil des Fragebogens.

3. Teil: Bewertung Anpassungswerkzeug bzw. Composer

Frage	**Pflicht-feld?**	**Antworttyp**	**Mögliche Antworten**
Wie hilfreich war die Anleitung für die Arbeit mit dem Anpassungswerkzeug bzw. Composer?	Ja	Auswahl	Skala von 6 (Sehr hilfreich) bis 1 (Gar nicht hilfreich)
Hat das Anpassungswerkzeug bzw. der Composer Sie informiert, welche Eingaben zulässig oder nötig sind?	Ja	Auswahl	Skala von 6 (Hat ausreichend informiert) bis 1 (Hat nicht ausreichend informiert)
War der erforderliche Aufwand für Ihr Arbeitsergebnis angemessen?	Ja	Auswahl	Skala von 6 (Sehr angemessen) bis 1 (Unangemessen)
Wie war das Anpassungswerkzeug bzw. der Composer auf die Anforderungen der Arbeit zugeschnitten?	Ja	Auswahl	Skala von 6 (Sehr gut) bis 1 (Sehr schlecht)
Haben Sie sich im Anpassungswerkzeug bzw. Composer zusätzliche Funktionen zur Bearbeitung der Aufgabe gewünscht?	Nein	Auswahl	Skala von 6 (Keine weiteren Funktionen gewünscht) bis 1 (Weitere Funktionen gewünscht)
Falls Sie sich weitere	Nein	Freitext	

Funktionen wünschen: Welche sind das?			
Hat sich das Anpassungswerkzeug bzw. der Composer so verhalten, wie Sie es erwartet haben?	Ja	Auswahl	Skala von 6 (Immer) bis 1 (Nie)
War die Bedienung des Anpassungswerkzeuges bzw. Composers gut verständlich?	Ja	Auswahl	Skala von 6 (Sehr verständlich) bis 1 (Gar nicht verständlich)
Wenn die Bedienung für Sie unverständlich war: Was war unverständlich?	Nein	Freitext	
War die Anordnung der Informationen auf dem Bildschirm übersichtlich?	Ja	Auswahl	Skala von 6 (Sehr übersichtlich) bis 1 (Gar nicht übersichtlich)
Haben Sie Fehler gemacht, die Sie noch mal korrigieren mussten?	Ja	Auswahl	Ja / Nein
Falls Sie Fehler gemacht haben, welche Fehler waren das?	Nein	Freitext	
Falls Sie Fehler korrigieren mussten: Wie groß war der Aufwand, um diese umzuändern?	Nein	Auswahl	Skala von 6 (Kein größerer Aufwand) bis 1 (Sehr großer Aufwand)
Haben Sie generelle Anmerkungen und Anregungen zum Anpassungswerkzeug bzw. Composer?	Nein	Freitext	

Tabelle 25: Dritter Teil des Fragebogens.

4. Teil: Allgemeine Fragen zum Test

Frage	**Pflichtfeld?**	**Antworttyp**	**Mögliche Antworten**
Allgemeine Anmerkungen zum Test	Nein	Freitext	

Tabelle 26: Vierter Teil des Fragebogens.

Auswertung des Fragebogens zum Anpassungswerkzeug / Netscape Composer

In Abschnitt 8.2 wurde bereits die Faktorenanalyse für den Fragebogen und die Auswertung der Fragebögen im Hinblick auf die beiden Faktoren (Aufgabenangemessenheit und Transparenz des Programms), sowie die nicht zuordenbare Frage nach der übersichtlichen Informationsanordnung auf dem Bildschirm vorgestellt. An dieser Stelle wird die Auswertung der Fragebögen in Bezug auf die Auswahl-Fragen des dritten Teils gezeigt. Dabei werden jeweils Mittelwert, Standardabweichung, Minimum und Maximum für beide Gruppen (Anpassungswerkzeug und Netscape Composer) getrennt aufgelistet.

Frage zu:	Gruppe	Typ des Wertes	Wert
Hilfreiche Anleitung	Composer	Mittelwert	5,38
		Standardabweichung	1,088
		Minimum	2
		Maximum	6
	Anpassungs-werkzeug	Mittelwert	5,75
		Standardabweichung	0,577
		Minimum	4
		Maximum	6
Information über zulässige Eingaben	Composer	Mittelwert	4,06
		Standardabweichung	1,731
		Minimum	1
		Maximum	6
	Anpassungs-werkzeug	Mittelwert	4,75
		Standardabweichung	0,931
		Minimum	3
		Maximum	6
Aufwand angemessen	Composer	Mittelwert	4,00
		Standardabweichung	1,897
		Minimum	1
		Maximum	6
	Anpassungs-werkzeug	Mittelwert	5,69
		Standardabweichung	0,479

		Minimum	5
		Maximum	6
Auf Anforderung zugeschnitten	Composer	Mittelwert	4,00
		Standardabweichung	1,897
		Minimum	1
		Maximum	6
	Anpassungs-werkzeug	Mittelwert	5,56
		Standardabweichung	0,512
		Minimum	5
		Maximum	6
Zusätzliche Funktionen gewünscht	Composer	Mittelwert	3,25
		Standardabweichung	2,113
		Minimum	1
		Maximum	6
	Anpassungs-werkzeug	Mittelwert	5,25
		Standardabweichung	1,065
		Minimum	3
		Maximum	6
Vorhersehbares Programmverhalten	Composer	Mittelwert	5,38
		Standardabweichung	0,957
		Minimum	3
		Maximum	6
	Anpassungs-werkzeug	Mittelwert	5,13
		Standardabweichung	1,025
		Minimum	3
		Maximum	6
Verständliche Bedienung	Composer	Mittelwert	5,63
		Standardabweichung	0,719
		Minimum	4
		Maximum	6
	Anpassungs-	Mittelwert	5,38

	werkzeug	Standardabweichung	0,719
		Minimum	4
		Maximum	6
Übersichtliche Informationsanordnung	Composer	Mittelwert	5,31
		Standardabweichung	0,704
		Minimum	4
		Maximum	6
	Anpassungswerkzeug	Mittelwert	5,69
		Standardabweichung	0,602
		Minimum	4
		Maximum	6

Tabelle 27: Auswertung der Auswahlfragen im dritten Teil des Fragebogens.

Nachfolgend werden weiterhin die Kreuztabellen für die Auswahl-Fragen zu Teil 3 gezeigt. Aus diesen Tabellen lässt sich ablesen, wie oft welche Antworten in den beiden Testgruppen (Anpassungswerkzeug und Netscape Composer) genannt wurden. (Die Antwort mit der Ziffer 6 entspricht der besten Bewertung, die mit der Ziffer 1 der schlechtesten.)

		Anzahl Nennungen		
Frage	**Gewählte Antwort**	**Composer**	**Anpassungswerkzeug**	**Gesamt**
Hilfreiche Anleitung	2	1	0	1
	4	1	1	2
	5	4	2	6
	6 sehr hilfreich	10	13	23
Information über zulässige Eingaben	1	1	0	1
	2	3	0	3
	3	2	2	4
	4	3	3	6
	5	2	8	10
	6 ausreichend informiert	5	3	8
Aufwand angemessen	1	2	0	2
	2	3	0	3

	3	1	0	1
	4	2	0	2
	5	3	5	8
	6 sehr angemessen	5	11	16
Auf Anforderung zugeschnitten	1	2	0	2
	2	3	0	3
	3	1	0	1
	4	2	0	2
	5	3	7	10
	6 sehr gut	5	9	14
Zusätzliche Funktionen gewünscht	1	5	0	5
	2	2	0	2
	3	3	2	5
	4	1	1	2
	5	0	4	4
	6 keine gewünscht	5	9	14
Vorhersehbares Programmverhalten	3	1	1	2
	4	2	4	6
	5	3	3	6
	6 immer vorhersehbar	10	8	18
Verständliche Bedienung	4	2	2	4
	5	2	6	8
	6 sehr verständlich	12	8	20
Übersichtliche Informationsanordnung	4	2	1	3
	5	7	3	10
	6 sehr übersichtlich	7	12	19

Tabelle 28: Kreuztabelle zu den Auswahlfragen im dritten Teil des Fragebogens.

Anhang F: Publikationen der Verfasserin

Wissenschaftliche Veröffentlichungen

Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: Experiences in Using Patterns to Support Process Experts in Wizard Creation. Akzeptiert zur Publikation in den Proceedings der EuroPLoP 2008, Juli 2008.

Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: Making Expert Knowledge of Adaptations of E-Learning Material Available with Patterns. In: Klaus Tochtermann, Hermann Maurer (Hrsg.): Proceedings of the 7th International Conference on Knowledge Management, Seite 79-86, September 2007.

Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: Ein Werkzeug zur Unterstützung der Anpassung existierender E-Learning Materialien. In: Christian Eibl, Johannes Magenheim, Sigrid Schubert, Martin Wessner (Hrsg.): DeLFI 2007: 5. e-Learning Fachtagung Informatik, Lecture Notes in Informatics (LNI), Seite 293-294, September 2007.

Birgit Zimmermann, Marek Meyer, Christoph Rensing, Ralf Steinmetz: Improving Retrieval of Reusable Learning Resources by Estimating Adaptation Effort. In: Proceedings of the First International Workshop on Learning Object Discovery & Exchange (LODE-2007), Seite 46-53, September 2007.

Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: Patterns towards Making Web Material Accessible. In: Proceedings of EuroPLoP 2007, Juli 2007.

Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: Patterns for Tailoring E-Learning Materials to Make them Suited for Changed Requirements. In: Proceedings of VikingPLoP 2006, Oktober 2006.

Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: Format-übergreifende Anpassungen von elektronischen Lerninhalten. In: Max Mühlhäuser; Guido Rößling, Ralf Steinmetz (Hrsg.): DeLFI 2006 - die 4. e-Learning Fachtagung Informatik 2006, GI Lecture Notes in Informatics, Seite 15-26, September 2006.

Birgit Zimmermann, Sonja Bergsträßer, Christoph Rensing, Ralf Steinmetz: A Requirements Analysis of Adaptations of Re-Usable (E-Learning) Content. In: Piet Kommers and Griff Richards (Hrsg.): Proceedings of World Conference on Educational Multimedia, Hypermedia, Seite 2096-2103, Juni 2006.

Birgit Zimmermann, Melanie Gnasa, Karin Harbusch: Modeling A Corporate Information System to improve Knowledge Management. In: Proceedings of the Second International Workshop on Multimedia Data Document Engineering (MDDE'02), Springer, Seite 435-449, Januar 2002.

Patentanmeldungen

Birgit Zimmermann, Marek Meyer: Patentanmeldung: Invention in the area of e-Learning & Knowledge Management. Oktober 2005.

Mitautorenschaft bei Veröffentlichungen

Sonja Bergsträßer, Christoph Rensing, Birgit Zimmermann, Ralf Steinmetz: Building a Representation of Learning Resources to Support their Re-Purposing. In: Proceedings of World Conference on Educational Multimedia, Hypermedia, Juni 2007.

Michael Metzger, Birgit Zimmermann, Sonja Bergsträßer, Christoph Rensing, Ralf Steinmetz: Automating Layout Adaptation of Textual-based E-Learning Content. In: Proceedings of World Conference on Educational Multimedia, Hypermedia, Juni 2007.

Marek Meyer, Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: An Interactive Tool for Supporting Modularization of SCORM-Based Learning Resources. In: Proceedings of World Conference on Educational Multimedia, Hypermedia, Juni 2007.

Marek Meyer, Sonja Bergsträßer, Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: Modeling Modifications of Multimedia Learning Resources Using Ontology-Based Representations. In: Tat-Jen Cham and Jianfei Cai and Chitra Dorai and Deepu Rajan and Tat-Seng Chua and Liang-Tien Chia (Hrsg.): Advances in Multimedia Modeling, LNCS 4351, Springer, Seite 34-43, Januar 2007.

Weitere Veröffentlichungen

Christoph Rensing, Birgit Zimmermann, Marek Meyer, Lasse Lehmann, Ralf Steinmetz: Wiederverwendung von multimedialen Lernressourcen im Re-Purposing und Authoring by Aggregation. In: Peter Loos, Volker Zimmermann, Pavlina Chikova (Hrsg.): Prozessorientiertes Authoring Management: Methoden, Werkzeuge und Anwendungsbeispiele für die Erstellung von Lerninhalten, Logos Verlag, Seite 19-40, Januar 2008.

Technical Reports

Sonja Bergsträßer, Birgit Zimmermann, Marek Meyer, Christoph Rensing, Andreas Faatz, Tomas Hildebrandt, Ralf Steinmetz: Re-Purposing: Motivation, Related Work, and Building Blocks. KOM-TR-2005-03, Dezember 2005.

Christoph Rensing, Sonja Bergsträßer, Tomas Hildebrandt, Marek Meyer, Birgit Zimmermann, Andreas Faatz, Lasse Lehmann, Ralf Steinmetz: Re-Use and Re-Authoring of Learning Resources - Definitions and Examples. KOM-TR-2005-02, November 2005.

Lebenslauf

Persönliche Daten

Name:	Birgit Zimmermann
Geburtstag:	22. August 1975
Geburtsort:	Freiburg im Breisgau
Nationalität:	Deutsch

Schulausbildung:

1995	Abschluss der Schule mit der allgemeinen Hochschulreife
1986 – 1995:	Stefan-George-Gymnasium Bingen
1985 – 1986:	Grundschule Weiler bei Bingen
1982 – 1985:	Viktor-Gollancz-Grundschule Berlin-Frohnau

Studium:

Dezember 2001:	Diplom in Informatik, Note: Sehr gut
1996 – 2001:	Studium der Informatik mit Vertiefungsfach Computerlinguistik an der Universität Koblenz-Landau
1995 – 1996:	Studium der Grundschulpädagogik an der Universität Koblenz-Landau

Berufliche Tätigkeit:

Seit 2008:	Researcherin bei SAP Research
2005 - 2008:	Research Associate bei SAP Research
2005 - 2008:	Wissenschaftliche Mitarbeiterin am Fachgebiet Multimedia Kommunikation der Technischen Universität Darmstadt
2004 – 2005:	Projektmanagerin bei Siemens VDO
2002 – 2004:	Technologieberaterin bei SAP LGD
1999 – 2001:	Tätigkeit als studentische Hilfskraft im Zentrum für Fernstudien und Weiterbildung in Koblenz

Betreute Studien- und Diplomarbeiten:

Michael Metzger: Layout Anpassungen von E-Learning Inhalten. Diplomarbeit an der Technischen Universität Darmstadt, 2006.

Alexander Horneff: Semi-Automatische Erstellung von Druckversionen von E-Learning Inhalten am Beispiel von HTML Kursen. Studienarbeit an der Technischen Universität Darmstadt, 2006.

Cheng Peng: Accessibility of E-Learning Materials. Diplomarbeit an der Technischen Universität Darmstadt, 2007.

Alexander Horneff: Automatisierte Wizard-Generierung basierend auf Adaptation-Patterns. Diplomarbeit an der Technischen Universität Darmstadt, 2007.