Anforderungsmodellierung mit UML und dem Unified Process mittels “Enterprise Architect 7.5” von SparxSystems

Regula Gerber
04-723-060

SS 09
Eine Seminararbeit im Rahmen der Veranstaltung „Seminar in Requirements Engineering “
Betreuender Dozent: Prof. Dr. Martin Glinz
Betreuender Assistent: Samuel Fricker

Abstract. Anforderungen sind elementar im Softwareentwicklungsprozess. Funktionale Anforderungen können mittels UML übersichtlich dargestellt werden. Der Unified Process liefert das dazu passende Prozessmodell. Im Enterprise Architect stehen diese beiden Hilfsmittel im Zentrum. Als Modellierungswerkzeug ist das einfach handhabbare, günstige Programm sehr gut geeignet, sofern das Entwicklungsteam die Strukturierungsfunktionen auf organisierte Weise einsetzt.

Keywords: Unified Process, UML, Enterprise Architect, SparxSystems

1 Einführung

In Berlin steht das T-Com Haus – eines der ersten intelligenten Häuser in Europa. Sämtliche Funktionen innerhalb und teilweise auch ausserhalb des Hauses sind mittels eines PDA's steuerbar, von welchem aus die Befehle an das zentrale Steuerungssystem weitergegeben werden [9]. Spätestens angesichts eines solchen Projekts wird klar, dass es unverzichtbar ist, im Voraus die Anforderungen festzulegen. Die Software unterschiedlichster Geräte muss integriert und die Bewohner des Hauses müssten (sofern es sich nicht wie hier um ein Prestige-Projekt zu Demonstrationszwecken handelt) als Stakeholder berücksichtigt werden.
Es gibt zwei Möglichkeiten, um die Anforderungsentwicklung zu unterstützen. Einerseits durch geeignete Befragungs- und Erhebungstechniken, die hier jedoch nicht weiter thematisiert werden sollen. Andererseits durch einen vorgegebenen Prozess, welcher ein flexibles Vorgehen und gleichzeitig eine gleichbleibende Qualität ermöglichen soll. Der Unified Process (UP) ist eines dieser Prozessmodelle. Zur Darstellung der Anforderungen wird bei diesem Prozess hauptsächlich die Unified Modeling Language (UML) verwendet. Dadurch können bei der Anforderungserhebung erstellte Modelle zu einem späteren Zeitpunkt beim Architekturentwurf des Systems weiterverwendet werden. Der Enterprise Architect ist ein Tool, um UML-Modelle zu erstellen und zu verwalten. Solche Tools haben den

Vorteil, dass die digitalen Modelle relativ leicht geändert werden können (im Gegensatz zu Papier- oder mit einem Zeichnungsprogramm erstellten Entwürfen).
In dieser Arbeit soll es vor allem darum gehen, inwiefern der Enterprise Architect das Konzept des Unified Process und die anforderungsbezogene UML-Modellierung unterstützt. Die Resultate setzen sich zusammen aus den Inhalten des Whitepapers und der Webseite von SparxSystems und aus eigenen Erfahrungen, die mittels der selbst erstellten Fallstudie "intelligentes Haus" gemacht wurden. Als Beurteilungsrahmen gelten die Konzepte des UP.

In Kapitel 2 werden zunächst die theoretischen Grundlagen vermittelt, wobei jeweils Kriterien für die Beurteilung des Enterprise Architect abgeleitet werden. Kapitel 2.1 beinhaltet die Begriffsdefinitionen, die Kapitel 2.2 und 2.3 gehen auf den Unified Process und auf UML näher ein. In Kapitel 3 schlussendlich werden die Ergebnisse der Untersuchung präsentiert und in Kapitel 4 folgen deren Diskussion und ein abschliessender Ausblick.

2 Theoretische Grundlagen

2.1 Begriffsdefinitionen

Anforderungen. Es gibt unterschiedliche Sichtweisen auf Anforderungen. Die hier genutzte Sichtweise besagt, dass eine Anforderung eine Bedingung oder Fähigkeit ist, welche ein System oder eine Systemkomponente erfüllen oder besitzen muss, um einen Vertrag, einen Standard, eine Spezifikation oder ein anderes formelles Dokument zu erfüllen [10].

Funktionale Anforderungen. Funktionale Anforderungen definieren, was für ein Verhalten das System zeigen soll [1].

Nicht-funktionale Anforderungen. Diese Art von Anforderungen beschreiben diejenigen Qualitäten des geplanten Systems, welche *nicht* auf die Funktionen des Systems bezogen sind. Dazu gehören beispielsweise Sicherheit, Usability, externe Beschränkungen und Ähnliches. [6]

Anforderungsspezifikation. Dieses Dokument wird von allen am Projekt beteiligten Leuten verwendet, um ein für alle Interessenhalter (Stakeholder) geeignetes Produkt zu schaffen [6]. In diesem Dokument sollten folgende Abschnitte vorhanden sein: funktionale Anforderungen, Performance-Anforderungen, Anforderungen an die grafische Oberfläche, Design-Anforderungen und Entwicklungsstandards [10].

2.2 Der Unified Process

Es gibt mehrere Vorgehensmodelle für den Ablauf von Softwareprojekten, unter anderem der Unified Software Development Process, kurz Unified Process. Der UP wurde als Gegenstück zur UML entwickelt, um einen passenden Prozessablauf zur Hand zu haben [1].

Die drei wichtigsten Kerngedanken des UP sind: (1) Anwendungsfalldiagramme, (2) Architekturzentriert, (3) iterativ und inkrementell. Das *Anwendungsfalldiagramm* wird später in dieser Arbeit behandelt. Im Zentrum des UP steht eine robuste *Systemarchitektur*, die durch schrittweise Verfeinerung erstellt wird. Schon in der ersten Phase wird ein Wegwerf-Prototyp erstellt, um ein frühes Testen zu ermöglichen. *Iterativ* bedeutet, dass jede Phase ein- oder mehrmals wiederholt wird. *Inkrementell* bezieht sich auf ein anfängliches Grobkonzept, das immer stärker verfeinert wird [1].
Der Unified Process unterscheidet zwischen vier zeitlichen Phasen: Beginn, Ausarbeitung, Konstruktion und Umsetzung bzw. Transition (siehe Bild 1). Während dieser Zeitabschnitte stehen bestimmte Tätigkeiten stärker oder schwächer im Vordergrund. Es gibt fünf Bereiche: Anforderungen, Analyse, Design/Entwurf, Implementation und Test. Sie finden phasenübergreifend und meist auch parallel zueinander statt. Ausserdem beeinflussen sie sich gegenseitig, wodurch eine eindeutige zeitliche Trennung nicht möglich ist [1]. Die Bereiche "Anforderungserhebung" (Requirements) und "Anforderungsanalyse" (Analysis) sollen im Folgenden näher beschrieben werden. Die restlichen drei Bereiche hängen nur indirekt mit Anforderungen zusammen und werden daher an dieser Stelle nicht weiter erläutert.

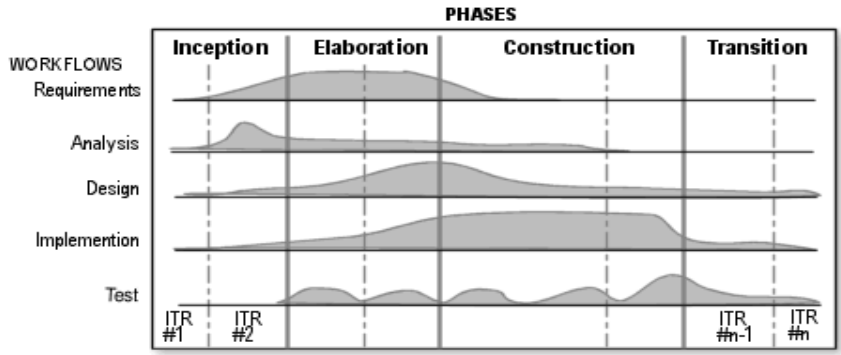

Bild 1: Der Unified Process [11]

Anforderungserhebung. Die Anforderungserhebung findet vor allem in den Zeitabschnitten "Beginn" und "Elaboration" statt. Bei der Anforderungserhebung geht es darum, Anforderungen unterschiedlicher Stakeholders zu erheben und zu priorisieren [1].
Die UML wird in Form von Anwendungsfalldiagrammen eingesetzt. Diese sollen anschliessend mittels Szenarien detailliert werden. Eine persönliche Ergänzung des UP von Arlow und Neustadt besteht darin, in dieser Phase zusätzlich eine ergänzende Anforderungsspezifikation zu erstellen, in der weitere (insbesondere nicht-funktionale) Anforderungen festgehalten werden. Ausserdem soll ein Glossar erstellt werden, um ein gemeinsames Verständnis von grundlegenden Begriffen zu erzielen. Möglich wäre auch bereits ein erster grober Entwurf der Systemarchitektur [1].

Anforderungsanalyse. Die Anforderungsanalyse besitzt ihren zeitlichen Schwerpunkt in der Phase "Elaboration". Erhebung und Analyse überlappen sich teilweise und oft wird die Analyse eingesetzt, um die bestehenden Anforderungen klarer darzustellen, woraus wieder neue Anforderungen entstehen könnten (was eine weitere Iteration der ersten zeitlichen Phase nach sich zieht). Die zusätzliche Leistung der Analyse besteht darin, das Verhalten des Systems und die Analyseklassen (d.h. die Repräsentationen der realweltlichen oder konzeptuellen Objekte) zu modellieren. Das Verhalten zeigt sich in der sogenannten Realisierung der Anwendungsfälle. Diese können nicht nur in Form der zuvor erwähnten Szenarien erfolgen, sondern auch durch Sequenz-, Kollaborations- oder Aktivitätsdiagramme [1]. Eine genauere Beschreibung dieser Modelle folgt unter Punkt 2.3.

Abgeleitete Kriterien für die Untersuchung. Zusätzlich wären im UP in den Phasen "Beginn" und "Elaboration" Dokumente wie die Beschreibung des Geschäftsmodells vorhanden [1]. Diese werden jedoch nicht Bestandteil der Untersuchung sein, da sie in der Regel in einfacher Textform dargestellt werden können.
Im Zentrum des UP steht das iterative und inkrementelle Vorgehen. Wiederholungen und Ergänzungen benötigen gewisse Bedingungen, damit keine negativen Erscheinungen wie z.B. Inkonsistenzen oder Redundanzen auftreten. Zu diesen Bedingungen gehört beispielsweise eine übersichtliche Strukturierung der Informationen (ob in Form von Modellen oder Textdokumenten) (1.1). Traceability wird immer wieder als zentrale Bedingung im Zusammenhang mit Anforderungen genannt. Nur, wenn die Zusammenhänge zwischen Anforderungen und der Ursprung einer Anforderung ersichtlich sind, können Iterationen und Inkremente gefahrlos durchgeführt werden. Als geeignetes Mittel wird hierbei eine sogenannte Traceability-Tabelle empfohlen, d.h. eine Tabelle, welche die Zusammenhänge aller Anforderungen darstellt (1.2) [5]. Zusätzlich sollten Anforderungen in einem standardisierten Textdokument dokumentiert werden können, um in der Modellierung unerfahrene Stakeholder in die Iterationen und später in das Review einbeziehen zu können (1.3).

Unified Process
1 Übersichtlichkeit
2 Traceability
3 Dokumentation

2.3 UML im Unified Process

Definition. Bei der UML handelt es sich um eine grafische Sprachensammlung zur Darstellung von Software-Architektur und Anforderungen. Sie umfasst 13 Modelle, die unterschiedliche Sichtweisen auf ein System ermöglichen [2]. UML vereinigt die Ansätze verschiedener Sprachen und schafft auf diese Weise einen Industriestandard. UML kann von der Konzeptstufe bis hin zu ausführbaren Artefakten eingesetzt werden [3].

UML versus andere Notationsformen. Anforderungen können informal, formal und teilformal dargestellt werden. Die *informale Darstellung* (z.B. natürliche Sprache) ist verständlich, ausdrucksmächtig und benötigt kein spezielles Vorwissen, ist jedoch nicht eindeutig, schwierig zu prüfen und unübersichtlich [4]. Die *formale Darstellung* (z.B. Z-Notation) ist eindeutig und gut prüfbar. Sowohl das Schreiben als auch das Lesen dieser Form ist jedoch schwierig und gewisse Aspekte lassen sich schwer formalisieren [5]. Die *teilformale Darstellung* (modellbasierte Verfahren) stellt Systeme und deren Anforderungen mithilfe von Modellen dar [2]. UML und ADORA zählen dazu. Hierarchisch untergeordnete Modelle klären die Detailfragen (Dekomposition) [1]. Im Gegensatz zur Z-Notation benötigen Modelle minimale Vorkenntnisse und sind übersichtlicher und eindeutiger als Fliesstext [4]. Es sind jedoch trotzdem Vorkenntnisse über die Notation nötig. Zudem können Modelle bei komplexen Sachverhalten unübersichtlich werden, wenn die Möglichkeiten der Dekomposition zu wenig gegeben sind oder zu wenig genutzt werden. Gewisse Aspekte können innerhalb eines UML-Modells nicht dargestellt werden [3]. Die Identifikation dieser Aspekte ist jedoch für die Evaluation des Enterprise Architect unwichtig – wichtig ist vielmehr die Tatsache, dass Modelle manchmal mit anderen Notationsformen (insbesondere Text) ergänzt werden müssen.

Anwendungsfalldiagramm. Ein Anwendungsfalldiagramm besteht aus den Systemgrenzen, den externen Akteuren (die ein Ereignis auslösen) und den internen Anwendungsfällen, (den Aktionen, die als Folge dieses Ereignisses innerhalb des Systems stattfinden). Da das Innere des Systems noch nicht bekannt sein muss und die Interaktion mit *unterschiedlichen* Akteuren (mit unterschiedlichen Bedürfnissen) dargestellt wird, ist es für die anfängliche Anforderungserhebung besonders gut geeignet [1].
Die möglichen konkreten Ereignisabläufe (Szenarien) innerhalb eines Anwendungsfalls werden oftmals zuerst in Textform dargestellt. Später können sie jedoch auch als Interaktionsdiagramme (d.h. Sequenz- oder Kollaborationsdiagramme) dargestellt werden [2].
Anwendung des Modells: Wenn wenige oder gar keine Akteure vorhanden und viele nicht-funktionale Anforderungen zu erwarten sind, ist ein Anwendungsfalldiagramm wenig geeignet. Dann gäbe es keine Interaktionen und/oder wenige funktionale Anforderungen darzustellen [1].

Domänenmodell. Ein als Domänenmodell bezeichnetes Diagramm wird in der UML als Klassenmodell bezeichnet. Ein *Klassenmodell* bildet die (abstrakten) Objekte der realen oder digitalen Welt als Klassen mit Attributen, Operationen und Beziehungen untereinander ab. Beim Domänenmodell werden die Atribute und Operationen weggelassen oder als provisorisch eingestuft [1].

Interaktionsdiagramme. Interaktionsdiagramme (Sequenz- oder Kommunikationsdiagramme) können eingesetzt werden, um einerseits die Dynamik einer bestimmten Menge von Objekten zu visualisieren, anderseits, um einen Anwendungsfall dynamisch darzustellen [2].

*Sequenzdiagramm***.** Das Sequenzdiagramm modelliert die zeitliche Abfolge von Nachrichten [2]. Es besteht aus Akteuren, Objekten und deren gegenseitigen Methodenaufrufen [1].
*Kollaborationsdiagramm***.** Das Kollaborationsdiagramm zeigt ebenfalls die gegenseitigen Methodenaufrufe von Akteuren und/oder Objekten, jedoch können die Objekte frei angeordnet werden und es ist auch eine Anzeige von Beziehungen der Objekte (z.B. Aggregation) möglich. Zudem können hier auch einzelne Instanzen von Objekten dargestellt werden [1].

Aktivitätsdiagramm. Im Unterschied zu Interaktionsdiagrammen werden bei einem Aktivitätsdiagramm die Arbeitsflüsse bzw. die verschiedenen Aktivitäten dargestellt. Dadurch ist der zeitliche und logische Zusammenhang ersichtlich [2]. Hauptbestandteile von Aktivitätsdiagrammen sind Aktivitäten, die Transitionen (von einer Aktivität zur nächsten), einen Start- und einen Endzustand, und Entscheidungsknoten. Die Aktivitäten können durch vertikale Trennlinien und einer entsprechenden Anordnung den einzelnen Akteuren zugeordnet werden [1].

Abgeleitete Kriterien für die Untersuchung. Grundsätzlich sollte der Enterprise Architect sämtliche UML-Modelle unterstützen, für eine sinnvolle Umsetzung des UP sind insbesondere die oben genannten Modelle relevant (1). Für einen flexiblen Prozess ist es zudem unerlässlich, dass Modelle leicht geändert werden können. Da – wie bereits erwähnt – UML-Modelle nicht sämtliche Aspekte darstellen können, wäre es wünschenswert, sie mit weiteren Symbolen oder Notationsformen (wie z.B. Text) zu ergänzen (2). Das Tool sollte zudem eine Möglichkeit zur Dekomposition anbieten, damit zu komplexe Modelle vermieden werden können (3). Ein wichtiger Bestandteil der Modellierung ist auch, dass die entsprechenden Modelle validiert werden können [5]. Es wäre zeitsparend, wenn das Tool zumindest grundlegende Gültigkeitsregeln vollautomatisch überprüfen könnte (4). Zudem soll überprüft werden, ob die gegebene Reihenfolge Anwendungsfalldiagramm – Szenario – Interaktions-/ Aktivitätsdiagramme in irgendeiner Weise vorgegeben ist (5).

UML im UP
1 Unterstützung UML-Modelle
2 Änderbarkeit/Ergänzbarkeit
3 Dekomposition
4 Validierung
5 UP-typische Modellabfolge

3 Untersuchung

3.1 Kurzbeschreibung des Enterprise Architect

Der Schwerpunkt des Enterprise Architect liegt bei der UML-Modellierung. Dies beginnt bei der Modellierung des Business Prozesses und geht über die Anforderungen bis hin zur Systemarchitektur. Daneben gibt es die Möglichkeit, aus

den Modellen Code zu generieren oder das Programm via Plugins z.B. mit Eclipse zu verbinden, um so die nachgelagerten Prozesse nahtlos anschliessen zu können. Das Ziel ist dabei die Entwicklung eines Softwareprodukts mithilfe von UML, ohne dass Medienbrüche (auch notationstechnischer Art) entstehen.
Zusätzlich sind Features wie beispielsweise weitere Modellierungsformen (z.B. Mindmap) vorhanden, diese spielen jedoch in Zusammenhang mit der Anforderungserhebung und -analyse eine untergeordnete Rolle und wären Ausgangspunkt für eine weitere Untersuchung.
Die Unterteilung der folgenden beiden Kapitel ist rein formaler Natur, da sich die Kriterien teilweise auch auf die jeweils andere Thematik beziehen. In der Diskussion werden die Kriterien dann in einer Gesamtbeurteilung integriert.

3.2 Kriterien zum Unified Process

Übersichtlichkeit. Um die Übersicht im EA zu behalten, gibt es vier zentrale Mittel. Im *Project Browser* (siehe Bild 2) wird die Struktur der Ordner und der darin enthaltenen Diagramme angezeigt. Zusätzlich werden bestimmte Modellelemente als einzelne Punkte unterhalb eines Modells aufgeführt [8]. Es zeigte sich bei der praktischen Durchführung, dass die Übersicht über die Packages und deren Inhalte zu jedem Zeitpunkt gegeben ist. Voraussetzung dafür ist eine sinnvolle Struktur. Diese bleibt jedoch dem Nutzer überlassen, sofern nicht eine vorgegebene Struktur genutzt wird (siehe “UP-typische Modellabfolge”). Als Negativpunkt wäre zu erwähnen, dass die Packages “Anforderungserhebung” und “Anforderungsanalyse” ursprünglich in der zeitlichen Abfolge erstellt wurden, sie aber automatisch alphabetisch einsortiert wurden. Der Autorin scheint eine thematisch-zeitliche Anordnung sinnvoller als eine alphabetische, insbesondere, wenn zwischen sehr vielen Ordnern hin und her navigiert werden muss.

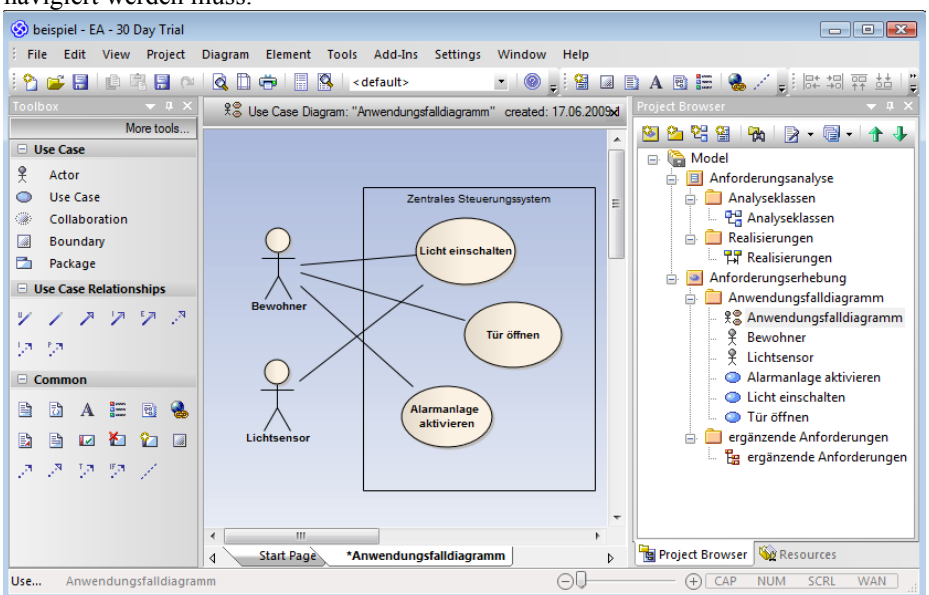

Bild 2: Totalansicht mit Project Browser (rechts)

Durch das sogenannte *Pan and Zoom* lässt sich anzeigen, welchen Ausschnitt eines Modells der Nutzer auf der mittleren Arbeitsfläche gerade sieht. Durch Herumschieben des Vierecks lässt sich durch das Modell navigieren. Zudem kann die gewünschte Zoomstufe eingestellt werden (siehe Bild 3) [8]. Bei dem in der Fallstudie verwendeten (wenig komplexen) Modell war der Nutzen dieser Funktionalität gering. Bei komplexen Modellen (eventuell in Verbindung mit einem kleinen Bildschirm-Display) wirkt sich dies jedoch stark positiv auf die Übersichtlichkeit innerhalb eines Modells aus. Wobei sich die Frage stellt, inwiefern die Vorteile eins Modells erhalten bleiben, wenn es diese Komplexitätsstufe erreicht und ob die Mittel zur Komposition in einem solchen Fall zur Genüge ausgeschöpft wurden.

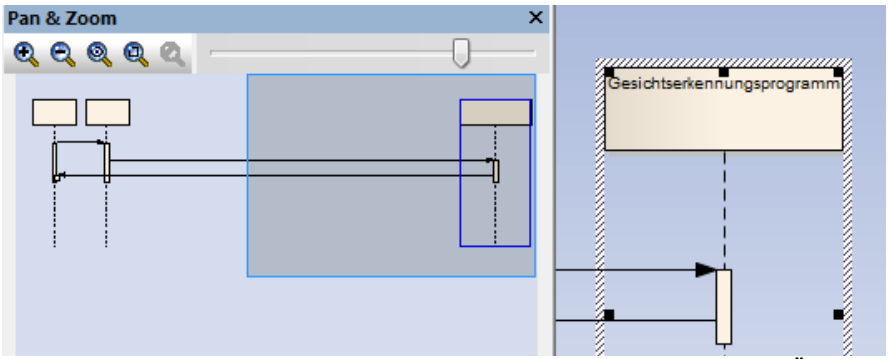

Bild 3: Die Pan & Zoom Ansicht (rechts das Modell, links die verkleinerte Übersicht)

In der *Elementenliste* können sämtliche Elemente eines Modells mit ihren Eigenschaften als Liste dargestellt und dort geändert, hinzugefügt oder gelöscht werden (siehe Bild 1 im Appendix). Dazu ergänzend gibt es die *Suche* nach Elementen, die typ- oder namenbezogen durchgeführt werden kann (siehe Bild 2 im Appendix) [8]. Die Elementenliste ermöglicht eine Übersicht über mehrere Modelle. Die Liste kann dabei z.B. nach Typ oder Status sortiert werden, was ein Auffinden bestimmter Aspekte wie z.B. aller Akteure oder aller implementierten Anforderungen einfach macht. Die Suche nach Elementen bringt dann den grössten Nutzen, wenn der Name eines Elements bekannt ist, dieses aber wegen einer grossen Anzahl Elemente nicht mehr auffindbar ist. Ansonsten bietet sie denselben Nutzen wie die Elementenliste.

Traceability. Um die Funktionen bezüglich der Traceability nutzen zu können, muss der User zwischen den zusammenhängenden Anforderungen oder Elementen sogenannte Relationships erstellen. Diese Beziehungen der Elemente untereinander können beispielsweise in der *Relationship-Matrix* (siehe Bild 4) (die Entsprechung der unter 2.2 erwähnten Traceability-Tabelle) oder in der *Hierarchy-View* (siehe Bild 5) angezeigt werden (letztere lässt sich auch in der weiter unten erwähnten Elementenliste anzeigen). In der Relationship-Matrix können Beziehungen auch geändert, hinzugefügt oder gelöscht werden. [8]. Bild 4 zeigt eine verhältnismässig kleine Matrix und ein Fenster, in welchem die Beziehung zwischen Akteur und Anwendungsfall geändert werden könnten. Am Kopf der Tabelle können die zu

anzeigenden Beziehungen ausgewählt werden. Dies ist insbesondere dann wichtig, wenn sehr viele Beziehungen vorhanden sind und die Tabelle unübersichtlich zu werden droht. Die Hierarchy-View zeigt die Zusammenhänge auf der Detailebene und macht insbesondere dann Sinn, wenn die Beziehungen eines einzelnen Elements im Kernpunkt des Interesses stehen. Diese Ansicht ergänzt somit die Relationship-Matrix und ermöglicht einen schnelleren Zugang zu Detailinformationen.

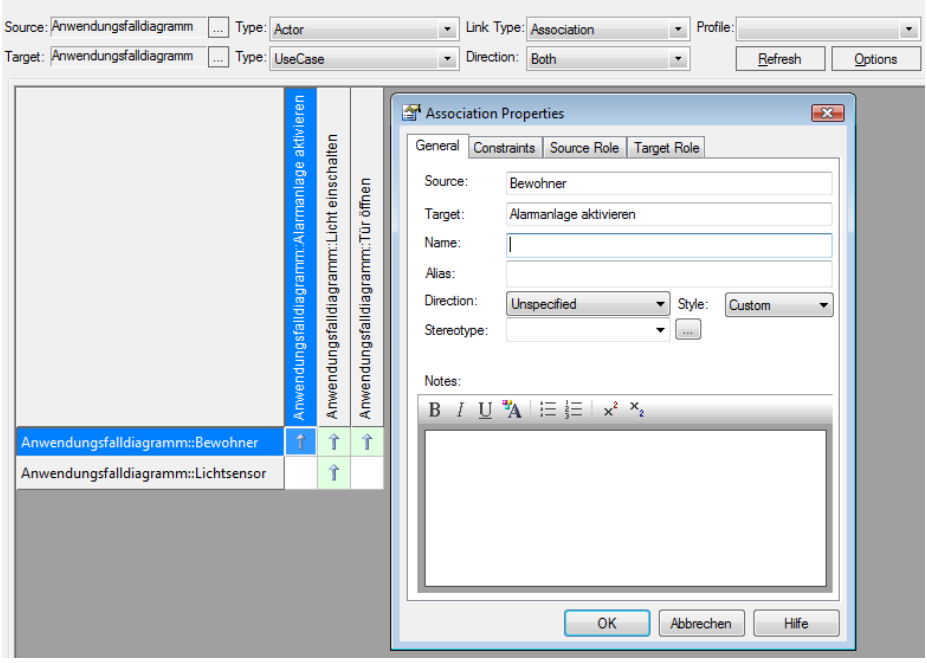

Bild 4: Relationship-Matrix

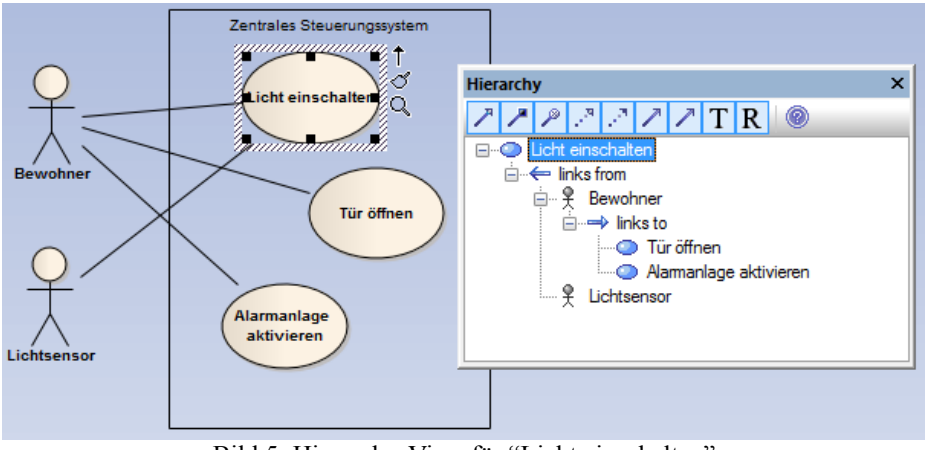

Bild 5: Hierarchy-View für “Licht einschalten”

Eine weitere Möglichkeit, die Verlinkung zu nutzen, ist die Verlinkung von Text- oder anderen Dateien mit Elementen (siehe Bild 3 Appendix). Die entsprechenden Dateien können direkt vom Formular aus gestartet werden [8]. Sofern dies konsequent für sämtliche Dateien durchgeführt wird, ist dies eine nützliche Methode, um schnellstmöglich auf eine Datei zugreifen zu können, die bspw. den Grund für eine Anforderung enthält. In der Praxis zeigte sich jedoch, dass die Verlinkung nicht mehr funktioniert, wenn eine Datei verschoben wird.
Auch der Project Browser (siehe Bild 2) kann laut SparxSystems genutzt werden, um die Traceability zu unterstützen. Hierbei wird empfohlen, eine Struktur zu erstellen, die voneinander abgeleitete Packages (z.B. Ansprüche aller Stakeholder und der daraus resultierende Kompromiss) direkt untereinander platziert [8]. Dies funktionierte im Fallbeispiel gut, da die Realisierungen von den Anwendungsfällen abgeleitet waren. Wenn jedoch das Projekt an Komplexität zunimmt, sind Mehrfachzusammenhänge zwischen unterschiedlichen Packages zu erwarten. Diese Zusammenhänge sind dann nicht mehr in einer Ordnerstruktur darstellbar. Als alleiniges Mittel zur Erhaltung der Traceability reicht der Project Browser daher nicht aus. Er stellt jedoch ein gutes Mittel dar, um die Traceability auf der Übersichtsebene darzustellen, die mutmasslicherweise eher bei linear ablaufenden Projekten gegeben sein dürfte.

Dokumentation. Zur Dokumentation können mehrere Arten von Reports generiert werden. Hierfür sind jeweils Templates vorhanden, beispielsweise für die Szenarien der Anwendungsfälle (siehe Bild 6) [8]. Diese Funktion sorgt einerseits für wahlweise ein RTF- oder HTML-Dokument, welches auf einfache Weise mit Personen ausgetauscht werden kann, welche den EA nicht installiert haben oder sich mit UML-Modellen wenig auskennen. Es benötigt zudem keinen grossen Aufwand, um den Report zu erstellen, da die Generierung automatisch passiert.

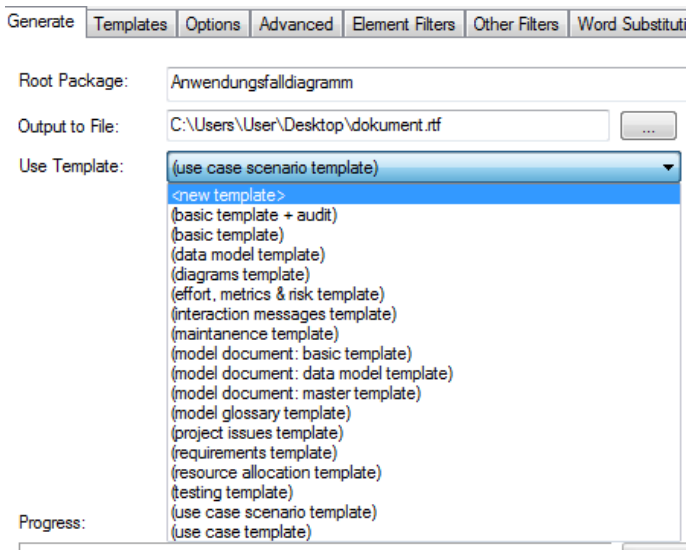

Bild 6: Auswahl an Report-Templates

Die Werte in der folgenden Textbox bedeuten: -- (sehr negativ), - (negativ), = (neutral), + (positiv), ++ (sehr positiv)

1 Übersichtlichkeit + 2 Traceability ++ 3 Dokumentation ++

3.3 Kriterien zu UML im Unified Process

Unterstützung der Modelle. Laut SparxSystems werden vom EA sämtliche UML-Modelle unterstützt [7]. Diese Aussage bestätigte sich, es standen alle bekannten Modelle zur Auswahl (siehe Bild 4 und 5 im Appendix).

Änderbarkeit und Ergänzbarkeit. Änderungen erfolgen per Drag and Drop, die Verbindungen zwischen zwei Elementen bleiben dabei bestehen. Ergänzen lassen sich die Modelle einerseits mit Symbolen verschiedenster Notationsformen (siehe Bild 10 im Appendix), andererseits mit Text-Boxen. Zudem lässt sich mit Doppelklick auf ein Element das dazugehörige Formular öffnen, welches weitere Informationen enthält (siehe "UP-typische Modellabfolge" weiter unten in diesem Text) [8]. Bei komplexeren Modellen als jenen in der Fallstudie würden sich Textboxen eher störend auswirken, da sie die Komplexität zusätzlich steigern.
Für eine Ersterfassung der Daten sind die Formulare sinnvoll, damit sämtliche Informationen tatsächlich im EA festgehalten werden (siehe Bild 6 im Appendix). Wenn eine grosse Anzahl von Eigenschaften von Elementen geändert werden sollen, empfiehlt sich jedoch ein direkter Zugang über die Elementenliste, da dort z.B. der Name direkt in der Liste (ohne den Umweg über ein Formular) geändert werden kann. In den Formularen sind teilweise Default-Werte voreingestellt (z.B. Status "proposed" für eine neue Anforderung). Diese könnten unter Umständen zu Fehlinformationen führen wenn der Nutzer bei der Erstellung eines Elements vergisst, sie entsprechend anzupassen (während ein leeres Feld unübersehbar auf eine fehlende Information hindeuten würde).

Dekomposition. Über- und untergeordnete Modelle können einerseits durch die Ordnerstruktur im Project Browser dargestellt werden. Andererseits besteht die Möglichkeit, Modelle und Elemente so miteinander zu verknüpfen, dass der Nutzer mit einem Doppelklick auf das entsprechende Element direkt zur Ansicht des untergeordneten Modells geleitet wird [8].
Als zentraler Negativpunkt wäre zu erwähnen, dass Modelle unterhalb von anderen Modellen nicht eingerückt dargestellt werden können, um einen Dekompositions-Zusammenhang auszudrücken. Die Modelle müssten einzeln in einen Ordner verpackt werden, um eine solche Darstellung zu ermöglichen.
Die Fortbewegung durch die Dekompositionsebenen mittels Doppelklick ist eine zeitsparende Funktion, um verschiedene Zusammenhänge zu erfassen. Wenn Ansichten wie die Relationship-Matrix oder die Hierarchy-View hinzugezogen werden, ist eine Dekomposition mit einer relativ guten Übersichtlichkeit möglich.

Validierung. Es gibt eine Möglichkeit, Modelle vollautomatisch zu validieren. Es können einzelne Elemente und deren Kindelemente, ganze Diagramme oder sogar die Inhalte von Packages auf einen korrekten UML-Syntax hin validiert werden. Bei nicht valider Darstellung werden Fehlermeldungen angezeigt [8]. Bild 7 zeigt ein Anwendungsfalldiagramm, in welchem nicht zugelassene Symbole enthalten sind. Die Validierung zeigt jedoch keine Fehler an. Fehler werden jedoch innerhalb eines Modells angezeigt, wenn beispielsweise ein Startzustand keine Verbindung zu einem nachgelagerten Zustand besitzt. UML stellt einen Standard dar und die Modelle sollten daher nur die vorgesehenen Symbolnotationen verwenden. Daher scheint es der Autorin sinnvoller, als Default-Einstellung modellfremde Symbole als Fehler ausgeben zu lassen. Die Regel kann jederzeit in den Einstellungen deaktiviert werden.

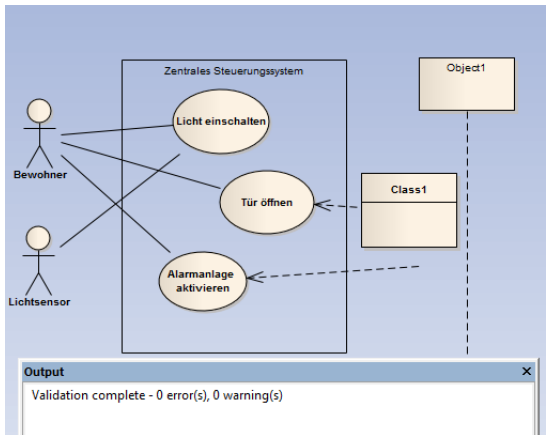

Bild 7: Validierung eines nicht validen UML-Modells

UP-typische Modellabfolge. Bei Anwendungsfalldiagrammen öffnet sich bei Klick auf einen Anwendungsfall ein Formular (siehe Bild 6 im Appendix), welches neben der Definition weiterer Eigenschaften auch erlaubt, Szenarien zu definieren. Aus diesen Szenarien kann ein Report generiert werden (siehe Bild 8 im Appendix) [8]. Diese beiden Funktionen wurden zuvor bereits erwähnt.

Interaktions- und Aktivitätsdiagramme lassen sich durch eine realize-Beziehung mit den entsprechenden Anwendungsfällen verbinden (siehe Bild 7 im Appendix) [8]. Die Beziehungen von Elementen innerhalb eines Diagramms sind sichtbar und ergeben sich häufig aus dem Sachverhalt. Bei Querverweisen von Elementen unterschiedlicher Modelle jedoch besteht die Gefahr, dass Beziehungen übersehen werden. Insbesondere, wenn für einen Anwendungsfall mehrere Realisierungen zur Verfügung stehen. Dies setzt eine sorgfältige Arbeitsweise voraus, da dies nicht automatisiert abläuft.

Bei der Erstellung eines neuen Projekts kann eine vorgegebene *Ordnerstruktur* ausgewählt werden, unter anderem eine, die für den Unified Process angepasst wurde (siehe Bild 8, siehe Bild 9 im Appendix) [8]. Die notwendigen Bestandteile der Anforderungserhebung und -analyse sind vorhanden. Die von Arlow und Neustadt empfohlenen zusätzlichen Anforderungen sind in der Struktur nicht enthalten, können aber problemlos hinzugefügt werden. Prinzipiell liesse sich darüber streiten, ob das

Domänenmodell dem Designmodell nicht vorangestellt werden müsste, da es bereits in der Phase der Anforderungsanalyse erstellt wird und daher zeitlich vor dem Architekturentwurf liegt, und ob eine erste Aufteilung nach einzelnen Phasen (z.B. Anforderungserhebung) nicht sinnvoller wäre.

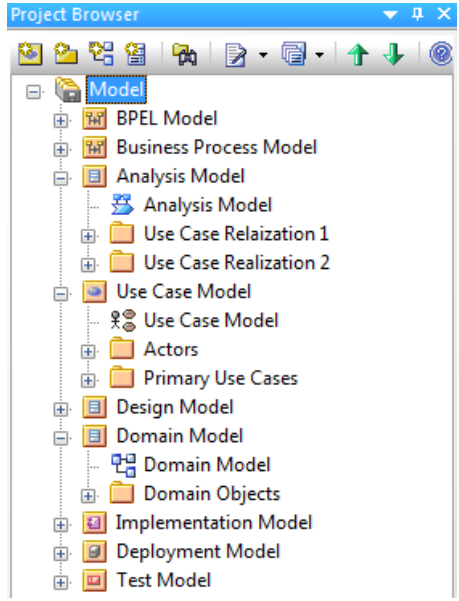

Bild 8: Vorgegebene Ordnerstruktur zum Unified Process

Die Werte in der folgenden Textbox bedeuten: -- (sehr negativ), - (negativ), = (neutral), + (positiv), ++ (sehr positiv)

1 Unterstützung UML-Modelle ++
2 Änderbarkeit/Ergänzbarkeit ++
3 Dekomposition +
4 Validierung =
5 UP-typische Modellabfolge +

4 Diskussion und Ausblick

4.1 Diskussion

Die Ausgangsfragestellung dieser Arbeit war, inwiefern der EA den Unified Process und insbesondere die darin verwendeten UML-Modelle unterstützt. Der iterative und inkrementelle Charakter des UP fordert eine übersichtliche, flexible Darstellung, in welcher die Zusammenhänge gut ersichtlich sind.
Die Übersichtlichkeit des EA ist als positiv zu bewerten, auch wenn sich hinsichtlich des Project Browsers noch einige Dinge verbessern liessen. Die Traceability und die

Änder- bzw. Ergänzbarkeit sind durch mehrfache Funktionen gegeben und werden sinnvoll unterstützt. Dadurch ist die Durchführung von Iterationen und Inkrementen ohne Informationsverlust möglich.
Ein weiterer Beurteilungspunkt ist der konkrete Ablauf des Unified Process, das heisst, die Bestandteile der Phasen “Anforderungserhebung” und “-analyse”.
Der EA ermöglicht sämtliche Modelle und auch eine Szenarienbeschreibung von Anwendungsfällen. Das Tool bietet desweiteren eine vorgegebene Ordnerstruktur an, welche für den Unified Process genutzt werden kann und welche sämtliche notwendigen Bestandteile für die ersten beiden Phasen enthält.
Zusätzlich gab es Kriterien auf der Modell-Ebene wie die Dekomposition, die Validierung und die Dokumentation. Die Dekomposition könnte nach Meinung der Autorin hinsichtlich des Project Browsers verbessert werden. Die Validierung funktioniert innerhalb der Modellnotation, ermöglicht jedoch ein Umgehen des UML-Standards, was zur schleichenden Entwicklung von Notationsformen führen könnte, welche den persönlichen Vorlieben statt dem Standard entsprechen. Die Dokumentation ist ein adäquates Mittel, um modellunerfahrene Personen in den Prozess einbeziehen zu können.
Insgesamt wird der Unfied Process durch geeignete Mittel gut unterstützt. Ob gewisse bemängelte Funktionen sich im Arbeitsalltag auswirken, hängt von der Arbeitsweise und den Notationsstandards eines Unternehmens ab. Hier muss jedoch auch das Preis-Leistungs-Verhältnis beachtet werden. Mit dem Preis von umgerechnet rund 300 Franken ist der EA als sehr günstig zu bezeichnen. Visio von Microsoft kostet bereits als Standard-Edition rund die Hälfte mehr und bietet dabei nur die Modellierungsfunktion. Der EA ist Editor, Modellierungs-, (zum Teil) Projekt-planungs-, Prototypingwerkzeug und Web-Browser in einem [8]. Das Laden und die Bearbeitung grösserer Modelle geht (durch persönliche Erfahrung bestätigt) schneller vonstatten als bei Visio.
Der Enterprise Architect ist daher insbesondere für Unternehmen mit einem kleinen Budget sehr empfehlenswert.

4.2 Ausblick

In der vorliegenden Arbeit wurden ausschliesslich die anforderungsbezogenen Aspekte untersucht. Da der Unified Process jedoch noch mehr Bestandteile enthält, welche sich auf die UML-Modellierung beziehen, wäre es in einer weiteren Untersuchung interessant, inwiefern diese unterstützt werden und ob sie nahtlos an die Anforderungserhebung und -analyse angeschlossen werden können. Auch die Funktionen von Plugins wie z.B. die Verbindung zu Eclipse konnten im Rahmen dieser Arbeit nicht untersucht werden. Um einen möglichen Einsatz des Enterprise Architect im Softwareentwicklungsprozess umfassend beurteilen zu können, müssten jedoch auch diese fehlenden Informationen zur Verfügung stehen. Die vorliegende Untersuchung kann daher nur als teilweise aussagekräftig betrachtet werden, kann jedoch als Grundlage für weitere ergänzende Untersuchungen dienen.

5 Literaturverzeichnis

1. Arlow, Jim/Neustadt, Ila: UML and the Unified Process. Practical object-oriented analysis & design. Edinburgh (2002)

2. Booch, Grady/Rumbaugh, James/Jacobson, Ivar: Das UML Benutzerhandbuch. Aktuell zur Version 2.0. München (2005)

3. Glinz, Martin: Problems and Deficiencies of UML as a Requirements Specification Language. In: Proceedings of the 10th International Workshop on Software Specification and Design, IEEE Computer Society. S. 11-22 (2000)

4. Glinz, Martin: Folien zur Vorlesung “Software Engineering” http://www.ifi.uzh.ch/rerg/fileadmin/downloads/teaching/courses/software_engineering_hs08/folien/Kapitel_04_Anf_Spez.pdf (2008)

5. Kotonya, Gerald/Sommerville, Ian: Requirements Engineering. Processes and Techniques. Sussex (1998)

6. Oestereich, Bernd: Objektorientierte Softwareentwicklung. Analyse und Design mit der UML 2.0. München (2004)

7. SparxSystems: Enterprise Architect 7.1. Reviewer's Guide. (2008)

8. SparxSystems: http://www.sparxsystems.com

9. http://www.golem.de/0502/36040.html

10. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00159342

11. http://www.datascovery.com/images/unifiedprocess.gif

6 Appendix

Name	Alias	Status	Type	Difficulty	Priority	Phase	Version	Created	Modified	Stereotype
Bewohner		Proposed	Actor			1.0	1.0	17.06.2...	17.06.2...	
Lichtsensor		Proposed	Actor			1.0	1.0	17.06.2...	17.06.2...	
Zentrales...		Proposed	Boundary			1.0	1.0	17.06.2...	17.06.2...	
Tür öffnen		Proposed	UseCase			1.0	1.0	17.06.2...	17.06.2...	
Alarmanla...		Proposed	UseCase			1.0	1.0	17.06.2...	17.06.2...	
Licht eins...		Proposed	UseCase			1.0	1.0	17.06.2...	17.06.2...	

Bild 1: Die Elementenliste

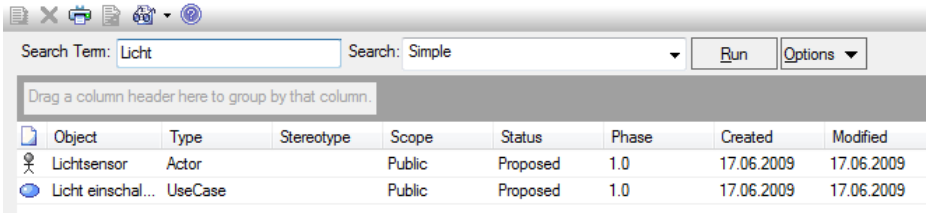

Bild 2: Die Elementensuche

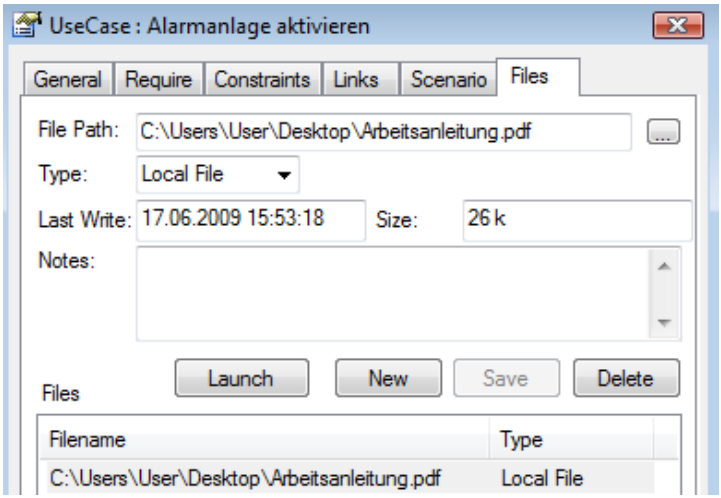

Bild 3: Datei mit einem Element verlinken

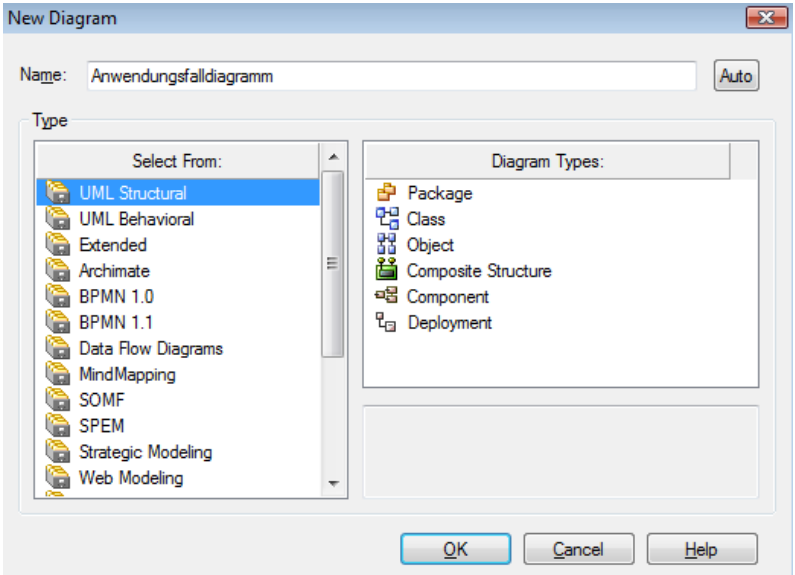

Bild 4: Strukturelle UML-Modelle

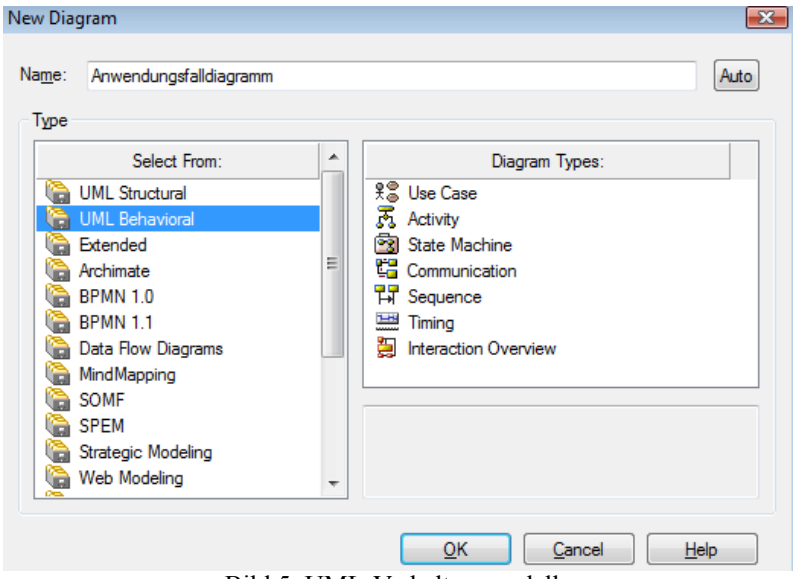

Bild 5: UML-Verhaltensmodelle

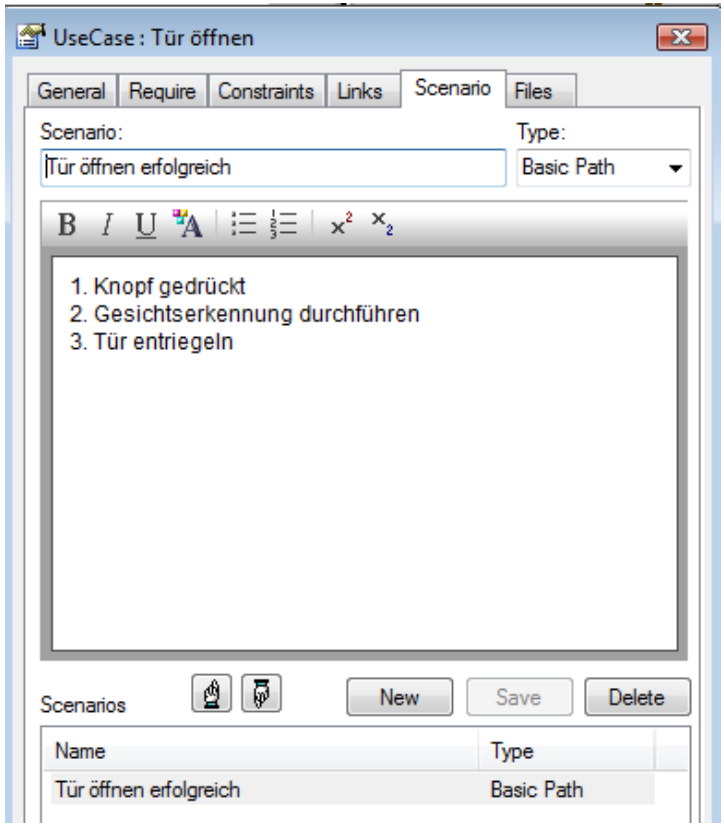

Bild 6: Formular (zugehörig zu einem Element)

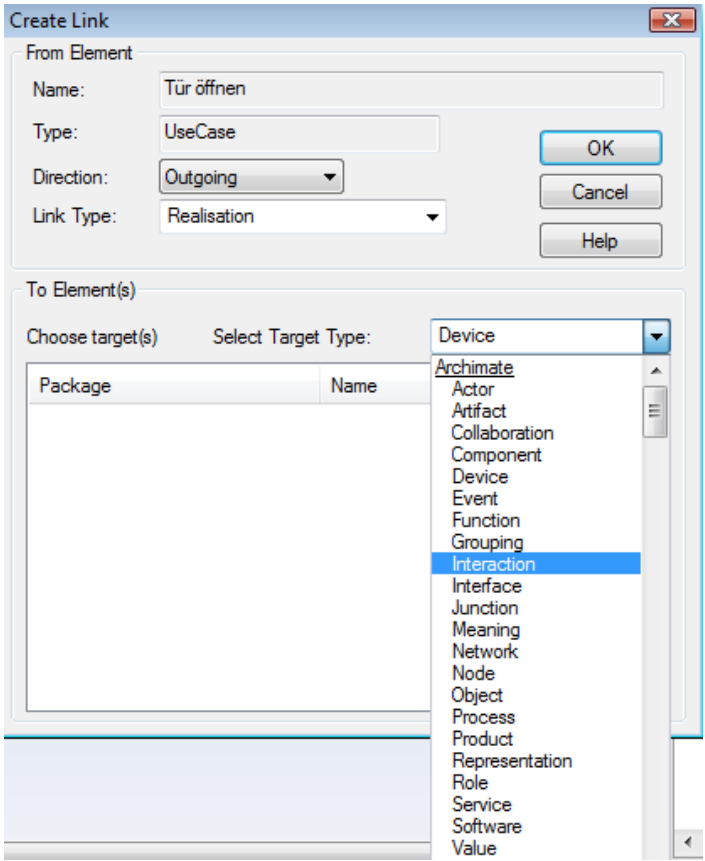

Bild 7: Erstellen einer Beziehung (hier: Realisierungs-Beziehung)

Use Case Name:	Tür öffnen		
Status:	Proposed	Version: 1.0	Phase: 1.0
Author:			
Created on:	17.06.2009	Modified on:	17.06.2009
Notes:			
Scenarios:			
Basic Path	Tür öffnen erfolgreich	1. Knopf gedrückt 2. Gesichtserkennung durchführen 3. Tür entriegeln	
Constraints:			

Bild 8: Ausschnitt aus einem Szenario-Report

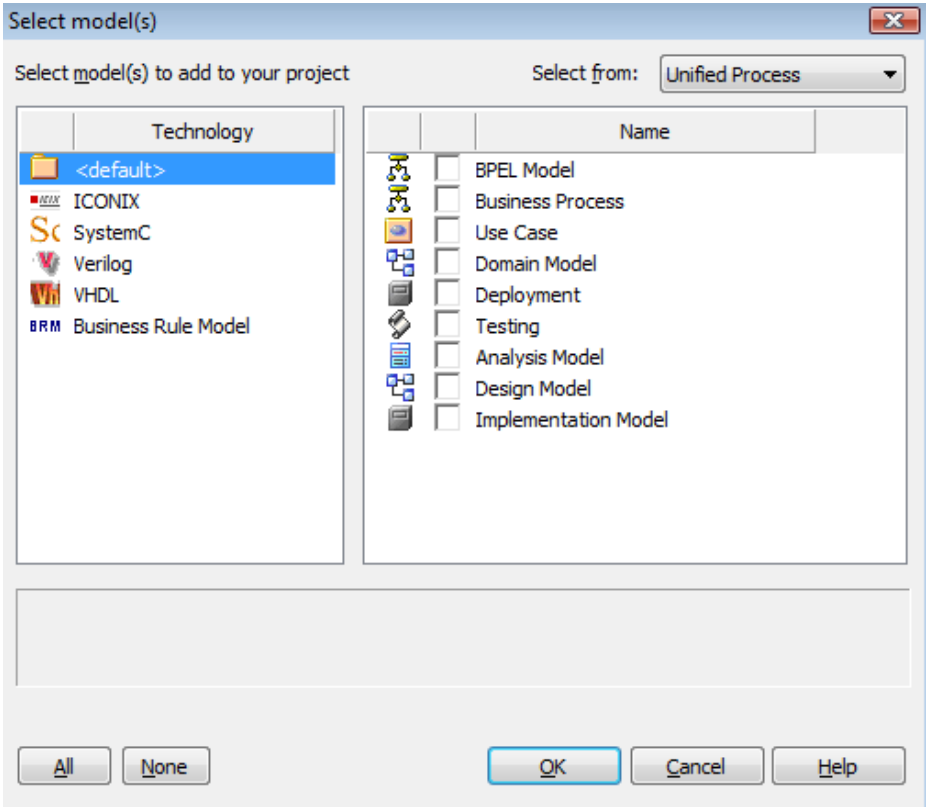

Bild 9: Auswahl an vorgegebenen Ordnern für den Unified Process

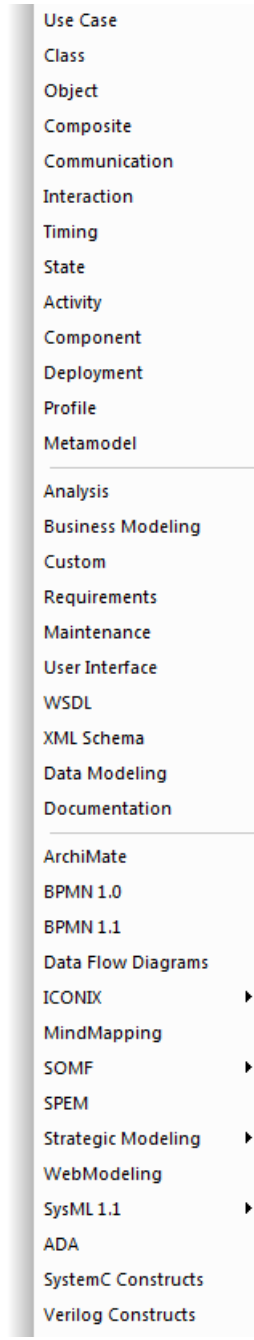

Bild 10: Vorhandene Notationsfomen