Von der Nutzungsanforderung zur formalen Softwarespezifikation

Modellierung mit dem Werkzeug YAKINDU Requirements

Florian Geyer
itemis AG
Am Brambusch 15–24
44536 Lünen
florian.geyer@itemis.de

Jens Trompeter
itemis AG
Am Brambusch 15–24
44536 Lünen
jens.trompeter@itemis.de

Michael Jendryschik
itemis AG
Am Brambusch 15–24
44536 Lünen
michael.jendryschik@itemis.de

Abstract
Die Analyse und Spezifikation von Software-Anforderungen ist eine komplexe Aufgabe, die als Grundlage jedes Softwareentwicklungsprojekts für den Erfolg oder Misserfolg maßgeblich ist. Oft bleiben jedoch Nutzungsanforderungen auf dem Weg zur Implementierung aufgrund einer mangelnden Integration in formale technische Spezifikationen auf der Strecke. Dieses Tutorial stellt einen werkzeugbasierten Ansatz zur Spezifikation komplexer interaktiver Systeme mit Hilfe des Werkzeugs YAKINDU Requirements vor. Das Werkzeug ermöglicht nicht nur eine Prozessunterstützung für die formale Spezifikation von Software-Anforderungen durch eine Verknüpfung verschiedener Prozessphasen und Modelle (Traceability), sondern bringt dabei interdisziplinäre Stakeholder wie Usability Professionals, Requirements Engineers, Systemarchitekten und Entwickler durch die Verwendung einer gemeinsamen Modellierungssprache zusammen. Das Tutorial demonstriert die Funktion und den Nutzen des Ansatzes an einfachen Beispielen und richtet sich dabei an Usability Professionals, die an einer formalen Integration von Nutzungsanforderungen, User-Interface-Entwürfen und Interaktionsabläufen in komplexe Softwareprojekte interessiert sind.

Keywords:
/// Anforderungsspezifikation
/// Werkzeuge
/// Requirements Engineering
/// Usability Engineering
/// Modellierung

1. Motivation

Eine detaillierte und widerspruchsfreie Spezifikation von Anforderungen ist häufig die zentrale Grundlage erfolgreicher Softwareentwicklungsprojekte. Dabei sind nicht nur technische Anforderungen für den späteren Erfolg oder Misserfolg maßgeblich, sondern auch die konsequente Berücksichtigung von Anforderungen aus Anwenderperspektive (Hartson & Pyla, 2012). Effektives Anforderungsmanagement stellt daher das systematische Ermitteln, Dokumentieren, Prüfen und Verwalten von Nutzungsanforderungen im Zusammenspiel mit technischen Rahmenbedingungen sicher (Rupp, 2009; Robertson & Robertson, 2012). Auf Basis unserer Erfahrungen gibt es dabei zwei zentrale Hindernisse bei der Spezifikation und dem Management von Anforderungen: die Integration interdisziplinärer Zusammenarbeit und die kontinuierliche Änderung und Nachverfolgbarkeit von Anforderungen über inkrementelle Entwicklungsstufen.

1.1. Interdisziplinäre Spezifikation

Im Rahmen des Anforderungsmanagements üben Stakeholder aus unterschiedlichen Domänen Einfluss auf die Anforderungsspezifikation aus. Neben Requirements Engineers, Produktverantwortlichen und dem Management bringen auch Usability Engineers Nutzungsanforderungen ein oder übersetzen technische Anforderungen in Interaktionsabläufe und User-Interface-Entwürfe. Schließlich sind auch Fachexperten oder Kaufleute genauso auf eine verständliche und konsistente Spezifikation angewiesen, wie Projektmanager, Systemarchitekten und Entwickler (Nyßen, 2009). Oft kommt es jedoch vor, dass die beteiligten Personen aus verschiedenen Domänen unterschiedliche „Sprachen" sprechen, was zu Missverständnissen oder Fehldeutungen führen kann (Gross & Hess, 2011). Eine besondere Herausforderung dabei ist der Unterschied zwischen den Modellen und Sprachen, in denen Anforderungen von den Stakeholdern formuliert werden. So erfordert eine formale technische Spezifikation eine exakte und eindeutige Formulierung von Anforderungen (z.B. Use Cases, Flowcharts oder UML-Diagramme). Dies steht oft im Kontrast zu vielen informelleren Artefakten, wie sie im Interaktionsdesign verwendet werden (z.B. Skizzen, Mockups oder Storyboards). Mangels Integration dieser verschiedenen Ausdrucksformen in formalen Spezifikationen bleiben wichtige Nutzungsanforderungen auf dem Weg zur Implementierung auf der Strecke. Um eine bessere Verknüpfung zu erreichen, ist daher eine gemeinsame, interdisziplinäre Basis für die Anforderungsspezifikation wünschenswert.

1.2. Spezifikationsdokumente

Traditionelle Vorgehensmodelle wie das Wasserfallmodell und das V-Modell (Boehm, 1981) machen umfangreiche Vorgaben in Bezug auf die für das Requirements Engineering zu erstellenden Dokumente und Artefakte. Dies gilt ebenso für

einige iterativ-inkrementelle Softwareentwicklungsprozesse wie den Rational Unified Process (RUP) (Jacobson et al., 1999). Agile Methoden unterscheiden sich hier deutlich vom klassischen Wasserfall-Modell und dessen Varianten. Statt in einer zu Beginn festgelegten Abfolge aus Spezifikation, Konstruktion und Umsetzung wird das Projekt in sehr enger, fortlaufender und direkter Zusammenarbeit mit dem Auftraggeber realisiert (Beck et al., 2001). Die Spezifikation erfolgt daher sukzessive während der Umsetzung (Paetsch et al., 2003). Das erlaubt zeitnahes Feedback und schnelle Reaktionen auf sich ergebende Veränderungen. Es gilt die Regel, dass neue und geänderte Anforderungen zu jeder Zeit willkommen sind. Diese Vorgehensweise erfordert jedoch Verzicht auf umfangreiche Spezifikationsdokumente oder deren kontinuierliche Anpassung und Erweiterung. Ein gänzlicher Verzicht bedeutet dabei jedoch auch die Aufgabe von Vorteilen formalisierter Vorgehensweisen, wie etwa Strukturierbarkeit, Suchmöglichkeiten, Versionierung und Protokollierung von Änderungen sowie Auswertungsmöglichkeiten. Kontinuierliche Anpassung hingegen erfordert zusätzliche Aufwände, um eine Reihe auf sich aufbauende Anforderungen, Modelle und Gestaltungsartefakte anzupassen. Oft ist dies auch mit der Verwendung unterschiedlicher Werkzeuge verbunden (z.B. Requirements Management Tools, UML Tools, UI Mockup Tools), deren Ergebnisse schließlich in ein umfangreiches Dokument konsistent eingepflegt werden müssen. Diese Werkzeugketten erschweren die Nachverfolgbarkeit und eine zeitnahe und effiziente Änderung von Anforderungen.

2. Werkzeug-basierte Spezifikation mit YAKINDU Requirements

Dieses Tutorial stellt einen Ansatz zur Spezifikation komplexer interaktiver Systeme mit Hilfe des Tools YAKINDU Requirements (www.yakindu.de) vor. Das Spezifikationswerkzeug bietet eine, auch für agile Prozesse sinnvolle Formalisierung und ist damit eine Alternative zu schwergewichtigen und komplexen Werkzeugketten. Es erlaubt Anwendungsfälle (Use Cases), Abläufe, Masken/Maskenfolgen, Fachobjekte und Geschäftsregeln in textueller Notation formal zu beschreiben. Aus dem resultierenden konsistenten und prüfbaren Gesamtmodell lassen sich automatisch Dokumente wie Diagramme, Lasten- und Pflichtenhefte, Testfälle und Schätztabellen generieren. Der zentrale Vorteil dieses Ansatzes liegt darin, dass Nutzer einfach und schnell formale Anforderungsmodelle erstellen und diese auch in verteilten Teams effizient teilen und anpassen können. Durch die Verknüpfung verschiedener prozessübergreifender Modelle und Artefakte unterschiedlicher Domänen bringt das Werkzeug interdisziplinäre Stakeholder wie Usability Professionals, Requirements Engineers, Systemarchitekten und Entwickler

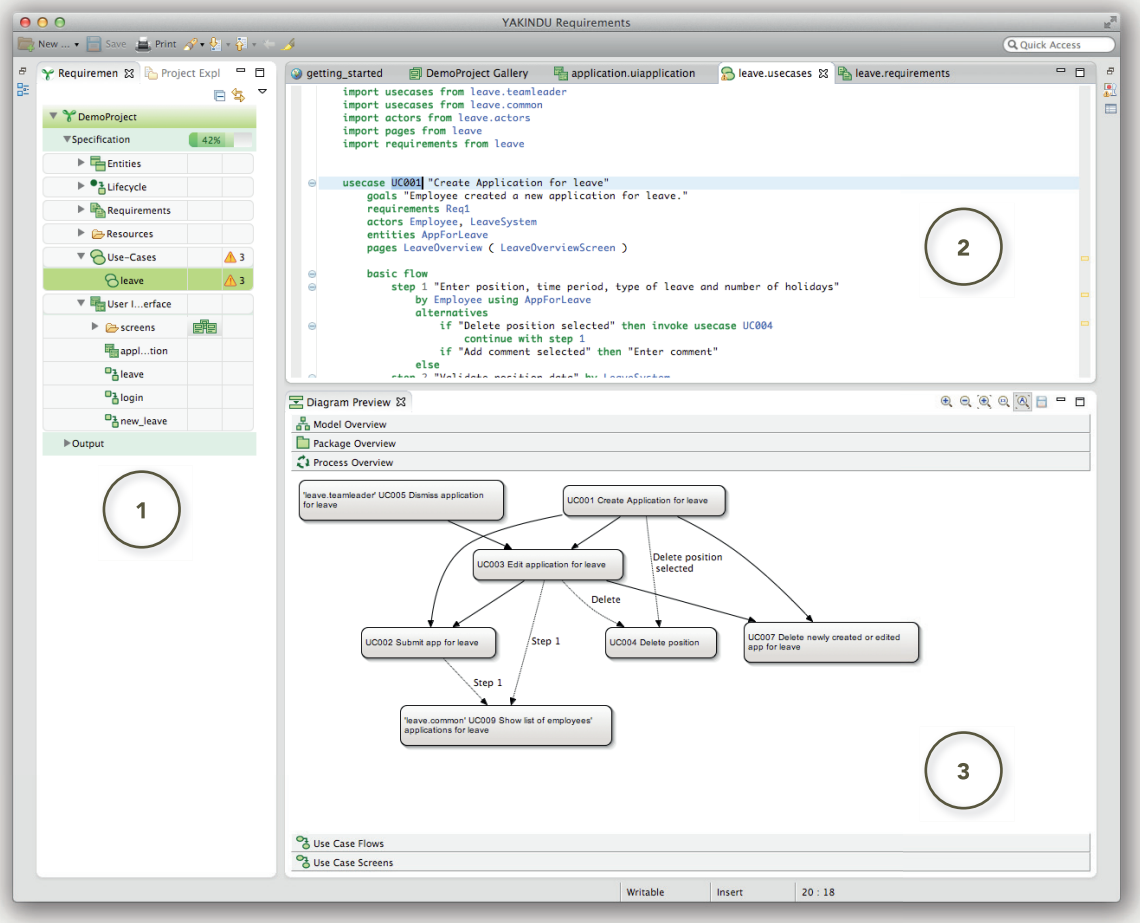

Abb. 1.
Die integrierte Modellierungsumgebung YAKINDU Requirements.

zusammen. Die Verbindung der Anforderungen und Entwicklungsartefakte wird mit einer gemeinsamen Modellierungssprache ermöglicht, welche die Präzision und Eindeutigkeit formaler Spezifikationen mit einfach verständlichen, visuellen Repräsentationen kombiniert. Als einer der wichtigsten Bestandteile des Requirements Managements ermöglicht dies eine bidirektionale Navigierbarkeit, mit dem Ziel, Anforderungen verfolgen und nachvollziehen zu können. Jedes Entwicklungsartefakt lässt sich so auf Anforderungen und Modelle zurückführen und hilft eventuelle Wechselwirkungen von Änderungen zu verstehen.

2.1. Integrierte Modellierungsumgebung

YAKINDU Requirements ist eine integrierte Modellierungsumgebung basierend auf der offenen Plattform Eclipse (www.eclipse.org). Es kombiniert eine textuelle Notation mit grafischen Modellen und Editoren (siehe Abb. 1). Die grundlegende Modellierungsumgebung ist aufgebaut auf einen Navigationsbereich (1), einen textueller Editor (2) und eine grafische Visualisierung (3).

Ein hierarchischer Projektexplorer ermöglicht es dem Nutzer eine klare Struktur zu erstellen und zwischen Anforderungen, Modellen und Artefakten der Spezifikation zu navigieren (siehe Abb. 1, 1). Mittels des textuellen Editors (siehe Abb. 1, 2) können Modelle formal spezifiziert werden. YAKINDU Requirements verwendet hierfür eine einfache, schnell erlernbare Sprache, die Vorteile wie Textvervollständigung, Textbausteine und Templates, sowie Fehlerkorrekturen und Konsistenzprüfungen integriert. Aufgrund des textbasierten Datenmodells ist es zudem möglich, durch Copy & Paste, Refactoring und der Verwendung von Diff- und Merge-Werkzeugen Änderungen oder Erweiterungen effizient zu verwalten. Zusätzlich bietet die Sprache eine durchgängige Referenzierung und bidirektionale Verknüpfung von Inhalten und kann um individuelle Konzepte und domänenspezifische Konzepte erweitert werden. Diese formale Spezifikation wird ergänzt durch eine grafische Aufbereitung der Modelle in einem Vorschau-Bereich (siehe Abb. 1, 3). Der Vorschaubereich stellt eine automatisch generierte grafische Repräsentation des textuell beschriebenen Modells dar und ermöglicht so dem Nutzer, einen Überblick über abstrakte Konzepte zu erhalten. Zusätzlich kann die grafische Darstellung ebenfalls zur Navigation innerhalb und zwischen Modellen und verknüpften Artefakten genutzt werden. **[Abb. 1]**

YAKINDU unterstützt derzeit folgende, miteinander verknüpfte Modelle zur Spezifikation von Softwareprojekten:

- Requirements-Modell – Anforderungen, deren Metadaten und Attribute
- Use-Case-Modell – Anwendungsfälle, Akteure und deren Zusammenhänge
- Entity-Modell – Objekte und deren Typisierungen und Relationen
- Lifecycle-Modell – Prozessmodell der Anwendung und der Daten
- UI-Modell – Masken und Komponenten der Benutzerschnittstelle
- UI-Flow-Modell – Verhalten der Benutzerschnittstelle und der Komponenten

```
usecase UC002 "Submit app for leave"
    goals "The application for leave is sent to the superior for approval."
    requirements Req1
    actors Employee
    entities AppForLeave
    pages leave.LeaveOverview ( LeaveOverviewScreen )
    preconditions "Newly created app for leave is available." requires UC001
        "Edited app for leave is available." requires UC003

    basic flow
        step 1 "Show list of created or edited apps for leave" invokes usecase UC009
        step 2 "Select application for leave" by Employee
        step 3 "Check automatically the plausibility of the application"
            alternatives
                if "Problems occured during checking" then "Show problems"
                    continue with step 1
            else
        step 4 "Send E-Mail to teamlead"
    end flow with postcondition "App for leave submitted"

end usecase
```

Abb. 2.
Beschreibung eines Use Cases in der YAKINDU Modellierungssprache.

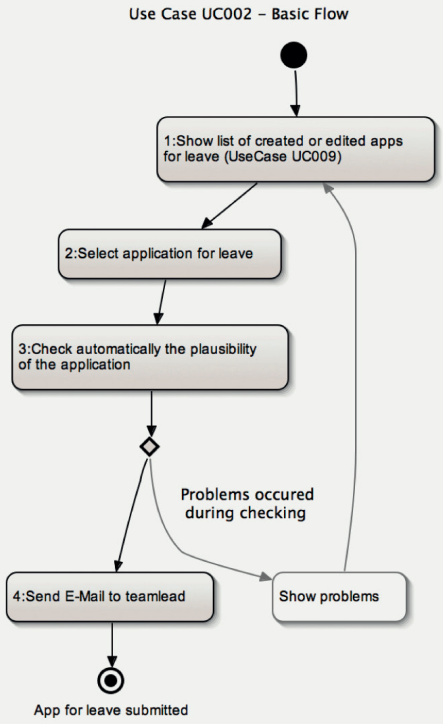

Abb. 3.
Aus der textuellen Beschreibung automatisch generiertes Use Case Diagramm.

2.2. Beispiel: Use-Case-Modellierung

Abbildung 2 zeigt an einem Beispiel, wie mit Hilfe von einfachen Schlüsselwörtern Use Cases spezifiziert werden können. Der beispielhafte Anwendungsfall „submit app for leave“ enthält einen Ablauf (basic flow) mit mehreren Schritten und Alternativen, aus denen YAKINDU Requirements automatisch visuelle Diagramme erzeugt und im Vorschaubereich anzeigt (siehe Abb. 3). Zudem enthält das Beispiel eine Reihe von Verknüpfungen zu anderen Artefakten, wie zu verwandten Anwendungsfällen (requires, invokes), zu relevanten Anforderungen (requirements), Akteuren (actors), einem Datenmodell (entities) und zu Entwürfen der Benutzerschnittstelle (pages). Diese

blau dargestellten, interaktiven Links ermöglichen dem Nutzer die Nachverfolgbarkeit von Veränderungen und erleichtern die Navigation zwischen den Modellen. YAKINDU Requirements nutzt diese Verknüpfungen jedoch auch, um die Konsistenz und Validität der Spezifikation sicher zu stellen. So wird der Nutzer beispielsweise benachrichtigt, falls ein bestimmter Akteur in keinem der spezifizierten Use Cases referenziert wurde. Das System unterstützt den Nutzer aktiv dabei, die formale Spezifikation auf Validität und Konsistenz zu prüfen – eine Aufgabe, die mit traditionellen Spezifikationsdokumenten nur sehr schwer zu realisieren ist. **[Abb. 2], [Abb. 3]**

2.3. Beispiel: User-Interface-Modellierung

Die Definition der Benutzerschnittstelle einer Anwendung wird mit Ablaufdiagrammen unterstützt (vgl. Page Flows, Storyboards). Abbildung 4 zeigt beispielhaft die Modellierung eines Login-Vorgangs. YAKINDU Requirements verfolgt hier einen Ansatz der visuellen Modellierung, der für Usability Engineers vertraut ist. Über einen grafischen Editor, der einem Ablaufdiagramm ähnelt, werden dynamische Veränderungen von Komponenten und Übergänge zwischen Masken modelliert (siehe Abb. 4, 1). So kann hier formal spezifiziert werden, wie das Verhalten der Anwendung durch Nutzerinteraktion und Programmlogik visualisiert wird. Der Editor verwendet hierfür das Konzept der „Screens", um dynamische Veränderungen in Zustände abzubilden. Neben einer Verknüpfung zu dem dahinter liegenden Design Rationale (Use Cases, Nutzungsanforderungen), können diese Zustände zudem mit Skizzen oder Mockups verknüpft werden (siehe Abb. 4, 2). Durch die Möglichkeit, Bilddateien per Drag & Drop zu integrieren, können auf einfache Weise beliebige Prototyping oder Mockup-Tools für die Visualisierung von Screen-Designs verwendet werden (auch gescannte oder abfotografierte Handskizzen). Durch die konsistente Verlinkung der Modelle ist sicher gestellt, dass sich Designentscheidungen vom User-Interface-Entwurf bis hin zu den Requirements zurückverfolgen lassen. Die potentiellen Auswirkungen von Änderungen sind daher jederzeit kontrollierbar, egal ob sie durch Feedback von Nutzertests (top-down) oder durch neue Produktanforderungen oder veränderte technische Rahmenbedingungen entstanden sind (bottom-up). **[Abb. 4]**

2.4. Automatische Generierung von Spezifikationsdokumenten

Durch den Ansatz einer integrierten Modellierungsumgebung ermöglicht YAKINDU Requirements die Verknüpfung von Modellen zu einer umfassenden interaktiven Spezifikation, die während des Entwicklungsprozesses leicht aktualisiert und angepasst werden kann (Change Management & Traceability). Alle Stakeholder – seien es Entwickler, Software-Architekten oder Usability Engineers – können interaktiv durch die hierarchisch organisierte und durch Querverbindungen verdichtete

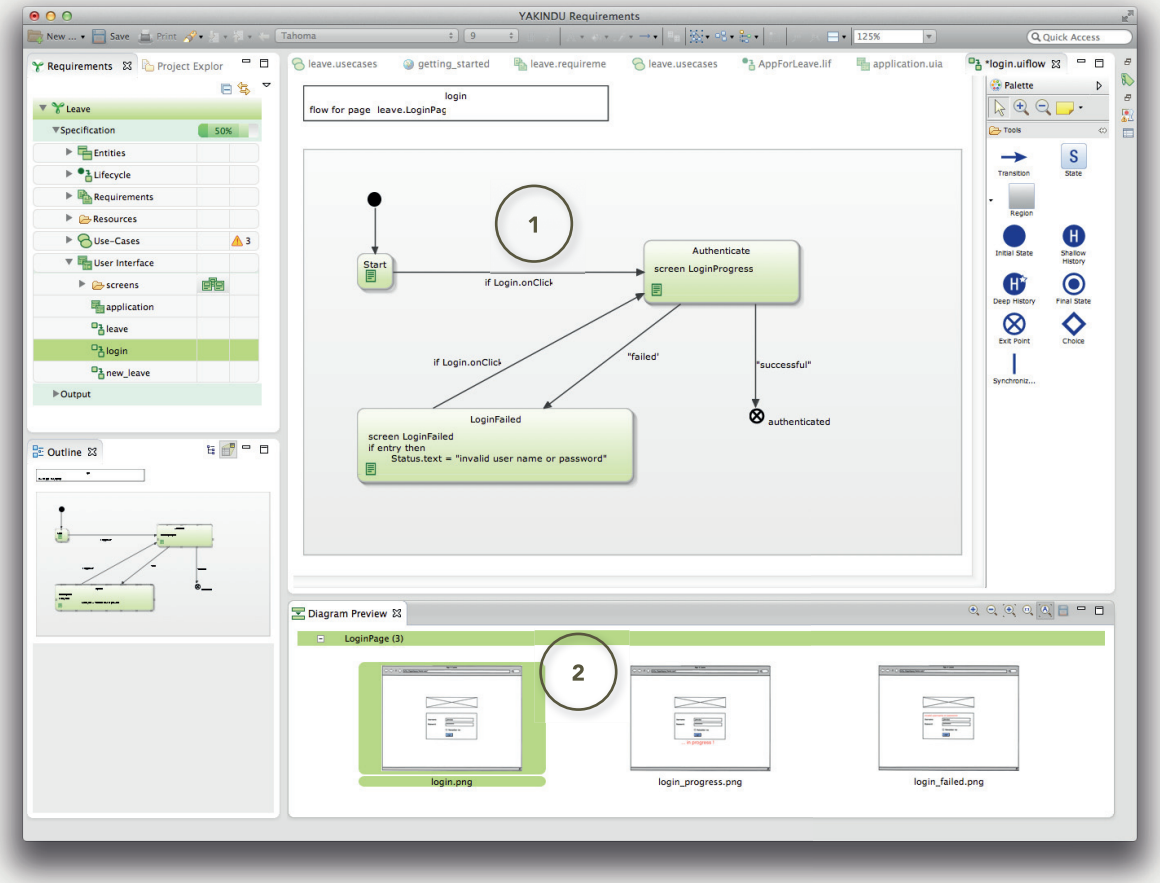

Abb. 4.
Spezifikation einer Benutzerschnittstelle (UI Flow Modell).

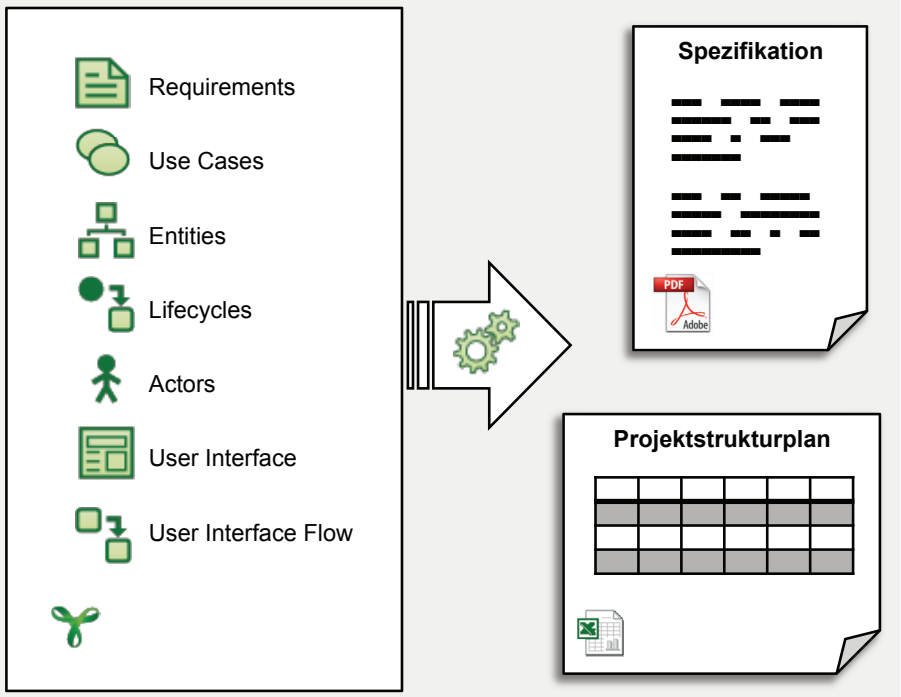

Abb. 5.
Automatische Erzeugung von Dokumenten aus den Modellen der interaktiven Spezifikation.

Struktur der Spezifikation navigieren. Ein entscheidender Vorteil liegt dabei darin, dass alle am Spezifikationsprozess aktiv beteiligten Personen dasselbe Werkzeug und dieselbe konsistente Datenbasis verwenden. Schnittstellen auf der Ebene von Anforderungen (Requirements Interchange Format ReqIF, www.omg.org/spec/ReqIF/) und User-Interface-Entwürfen erlauben zudem eine Integration von domänenspezifischen Werkzeugen. Darüber hinaus bietet YAKINDU Requirements umfangreiche Exportfunktionen, die es ermöglichen, automatisiert zielgruppengerechte Dokumente zu erzeugen (siehe Abb. 5). **[Abb. 5]**

Das Werkzeug lässt den Benutzer entscheiden, wie der Dokumentenexport aussehen soll und welchen Umfang und Vollständigkeit die erzeugten Dokumente besitzen. YAKINDU Requirements verwendet hierfür die Business-Intelligence Software BIRT, ein Projekt der Eclipse Foundation (www.eclipse.org/birt). So ist es möglich aus der Datenbasis für unterschiedliche Zielgruppen geeignete Dokumente zu erzeugen, wie beispielsweise ein ausführliches Lastenheft, das die Gesamtheit der Anforderungen eines Auftraggebers umfasst, oder einen Projektstrukturplan (Work Breakdown Structure), der einen Überblick über die Komplexität der zu entwickelnden Komponenten für das Projektmanagement bietet (McConnell, 2006). Individuelle Export-Definitionen ermöglichen darüber hinaus viele weitere flexible Einsatzszenarien. Weitere Einstellungen für den Dokumentenexport umfassen unterschiedliche Sprachen (deutsch, englisch), Formate (PDF, HTML, Office) und individuelle Layout- und Formatierungsregeln (CSS Stylesheets). Zusätzlich lassen sich die erzeugten Dokumente auch manuell anpassen und erweitern, falls dies erforderlich ist.

Automatisch erstellte Anforderungsspezifikationen haben einen entscheidenden Vorteil gegenüber traditionellen, dokumentenbasierten Spezifikationen: Bislang in Handarbeit durchgeführten Auswertungen, beispielsweise wie viele Anwendungsfälle ein bestimmtes Objekt nutzen, werden automatisch erledigt. Diagramme und Visualisierungen wie Anwendungsfalldiagramme und Zustandsautomaten werden vollautomatisch aus den erfassten Beschreibungen erzeugt, sodass Änderungen über den gesamten Prozess gepflegt werden können – von der Architektur über Design, Implementierung bis hin zu den Tests und umgekehrt. Das Resultat ist eine, auf Knopfdruck erzeugte Spezifikation, die stets auf dem aktuellen Stand ist und dabei unterschiedliche Perspektiven berücksichtigt.

3. Ausblick

YAKINDU Requirements hat seinen Ursprung in der Domäne des Requirements Engineering und orientierte sich dabei im Kern am Ansatz des „Use Case Modeling" (Bittner & Spence, 2002). Die aus dem Software Engineering stammende Methode richtet sich vor allem an Stakeholder mit einem technischen Hintergrundwissen, deren primäre Aufgabe die Spezifikation und das Management von Software-Anforderungen ist. Von diesem Fundament aus wurde die Modellierungssprache von YAKINDU Requirements kontinuierlich erweitert, um auch Stakeholder aus anderen Domänen aktiv in den Spezifikationsprozess einzubinden. Das Ziel des Werkzeuges ist es die Probleme interdisziplinärer Zusammenarbeit und die Schwerfälligkeit von umfangreichen Dokumenten und Werkzeugketten zu adressieren. Mit dem Ansatz einer interdisziplinären Modellierungsumgebung unterstützt die Software eine inkrementell-iterative Spezifikation und erleichtert es damit auch effizient mit der kontinuierlichen Änderung von Anforderungen in einem kollaborativen Umfeld umzugehen. Deshalb bietet es gerade bei agilen Prozessen, die trotzdem formalen Ansprüchen genügen müssen, eine angemessene Methodik. Dabei ermöglicht es der Ansatz auch, nichtfunktionale Anforderungen mit einer formalen, funktionalen Spezifikation zu verknüpfen und deren Abhängigkeiten und Querbezüge sichtbar zu machen.

Der Modellkatalog von YAKINDU Requirements kann auf einfache Weise mit domänenspezifischen Modellen erweitert werden. Momentan wird die Modellierungssprache auf Anforderungsbeschreibungsmodelle des Usability Engineerings ausdehnt. Unsere Erfahrungen haben gezeigt, dass es aufgrund einer fehlenden Verknüpfung von Usability Artefakten beispielsweise oft nicht möglich ist, Designentscheidungen bis zu den ursprünglich erhobenen Erfordernissen aus einer Kontextanalyse zurückzuverfolgen. So sollen in Zukunft Modelle wie Nutzungsszenarien (Rosson & Caroll, 2001), Personas (Cooper et al., 2007), und Essential Use Cases (Constantine & Lockwood, 1999) sowie allgemeine Gestaltungsrichtlinien und Standards (z.B. DIN EN ISO 9241–110) als Vorlagen und Templates in den Katalog aufgenommen werden. Durch eine Verknüpfung dieser Modelle mit der formalen funktionalen Spezifikation rücken die Disziplinen Usability Engineering und Software Engineering stärker zusammen. So werden sich etwa Nutzungsszenarien mit technischen Use Cases verbinden lassen, um den Kontext der Nutzung im Spezifikationsdokument abzubilden. Auf ähnliche Art und Weise soll es auch möglich sein Personas mit Akteuren zu verbinden, um den Einfluss von Nutzereigenschaften auf die Systemgestaltung festzuhalten. Schließlich soll eine Verknüpfung von allgemeinen oder systemspezifischen Richtlinien dafür sorgen, die Wahl einer bestimmten Gestaltungslösung für alle Stakeholder nachvollziehbar zu machen.

Literatur

1. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mallor, S., Schwaber, K., und Sutherland, J. (2001) „The agile manifesto". The Agile Alliance.
2. Bittner, K. und Spence, I. (2002). „Use Case Modeling". Addison-Wesley.
3. Boehm, B. W. (1981). „Software Engineering Economics". Prentice-Hall.
4. Constantine, L. L. und Lockwood, L. D. (1999). „Software for Use: A Practical Guide to the Methods of Usage-Centered Design". ACM Press.
5. Cooper, A., Reimann, R. und Cronin, D. (2007). „About Face 3. The Essentials of Interaction Design. John Wiley & Sons.
6. DIN EN ISO 9241–110 (2011). „Ergonomie der Mensch-System-Interaktion. Teil 110: Grundsätze der Dialoggestaltung". International Organization for Standardization.
7. Gross A. und Hess S. (2011). „UX meets RE – Hohe User Experience durch bedarfsgerechte Anforderungsspezifikation". Tagungsband Usability Professionals 2011, German UPA, 24–29.
8. Hartson, R. und Pyla, P. (2012). The UX Book: Process and Guidelines for Ensuring a Quality User Experience". Morgan Kaufmann.
9. Jacobson, I., Booch, G., und Rumbaugh, J. (1999). „The Unified Software Development Process". IBM.
10. McConnell, S. (2006). „Software Estimation: Demystifying the Black Art". Microsoft Press.
11. Nyßen, A. (2009). „Model-based construction of embedded and real-time software: a methodology for small devices". Dissertation. Universitätsbibliothek Aachen.
12. Paetsch, F., Eberlein, A. und Maurer, F. (2003) „Requirements Engineering and Agile Software Development". Proceedings on the Twelth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). pp. 308–313.
13. Robertson, S., und Robertson, J. (2012). „Mastering the Requirements Process: Getting Requirements Right". Addison-Wesley Longman.
14. Rosson, M. B. und Caroll, J. M (2001). „Usability Engineering: Scenario-Based Development of Human-Computer Interaction". Morgan Kaufmann.
15. Rupp, C. (2009). „Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis". Carl Hanser Verlag.