Christoph Ballhause
(christoph.ballhause@daimler.com)
ist als Senior Consultant für Requirements Engineering bei der Daimler TSS GmbH, einem internen IT-Dienstleister der Daimler AG, tätig. Er beschäftigt sich mit RE-Methoden für Fahrzeugentwicklung und IT-Systeme und ist als Product Owner tätig. Als Trainer vermittelt er Requirements Engineering-Wissen.

Jörg Leuser
(joerg.leuser@daimler.com)
ist als Senior Consultant für Requirements Engineering bei der Daimler TSS GmbH tätig. Er beschäftigt sich mit RE-Methoden für Fahrzeugentwicklung und IT-Systeme sowie der Spezifikation von IT-Systemen und ist als Trainer für RE aktiv.

Anforderungen programmieren – eine domänenspezifische Sprache (DSL) im Praxiseinsatz

Für Fachbereiche stellen textuell beschriebene Anforderungen an IT-Systeme eine gute Lösung dar. Für Requirements Engineers (RE) dagegen sind formalere Notationen effizienter. Da Anforderungen von Fachbereichen validiert werden müssen, sind der Formalisierung in der Praxis zumindest für die anwendernahen Spezifikationsanteile Grenzen gesetzt. Domänenspezifische Sprachen (DSL) verheißen lesbare Anforderungen für Fachbereiche und Formalisierung für die Requirements Engineers. In diesem Beitrag beschreiben wir unsere positiven Erfahrungen mit dem pilothaften Einsatz einer DSL für die Lastenhefterstellung.

Problemstellung

Die Fachbereiche besitzen das fachliche Wissen des zu entwickelnden IT-Systems. Ihnen fehlt jedoch meist tiefgreifendes IT-Fachwissen. Deshalb beauftragen Fachbereiche typischerweise Requirements Engineers mit der Spezifikation „ihrer" IT-Systeme. Die Requirements Engineers bringen dabei das IT- und RE-Wissen ein und ziehen bei Bedarf weitere IT-Fachexperten hinzu.

Zwei typische Herausforderungen der Anforderungsentwicklung in der Praxis sind die begrenzte Zeit, die zur Verfügung steht, sowie die Komplexität und Größe der Spezifikation. Da es kaum automatisierte Unterstützung bezüglich Komplexität und Qualitätssicherung bei natürlichsprachlichen Anforderungen gibt, sind diese nicht optimal.

Bei formalisierten Spezifikationen können automatisierte Checks bei der Bewältigung der Komplexität helfen. Wird jedoch beispielsweise UML mit all seinen Details verwendet, so leidet die Verständlichkeit auf Seiten der Fachbereiche. Da die Anforderungen aber durch die Fachbereiche validiert werden müssen, ist dies problematisch.

Daher sind formale Spezifikationen für anwendernahe Anforderungsdokumente nicht besonders gut geeignet. Für technische Spezifikationen hingegen können for-

```
usecase UC1 „Report anzeigen"
    actors Benutzer, System
    happy flow
        step 1 Wähle Funktion zum Anzeigen von Reports by Benutzer
        step 2 Zeige eine Liste der Reports by System
        step 3 Wähle einen Report aus by Benutzer
            alternative
                if „Report noch nicht generiert"
                    step 1 Generiere gewählten Report by System
                end if
            end alternative
        step 4 Zeige Report an by System
    end flow with endcondition Report angezeigt
end usecase
```

Tabelle UC1: *Report anzeigen*

Schritt	Beschreibung	Akteur
	Happy flow	
1.	Wähle Funktion zum Anzeigen von Reports	Benutzer
2.	Zeige eine Liste der Reports	System
3.	Wähle einen Report aus	Benutzer
4.	Zeige Report an	System
Nachbedingung: Report angezeigt		
	Alternativablauf	
3.	**Falls** „Report noch nicht generiert"	
3.1	Generiere gewählten Report	System
3.2	**Weiter** im *Happy flow* in Schritt 4	

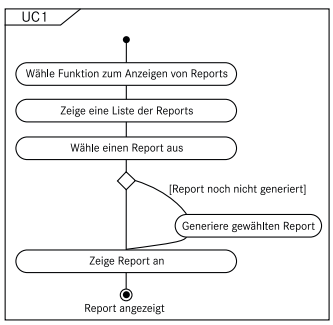

Abb. 1: Transformation der DSL in fachbereichstaugliche Darstellungsformen

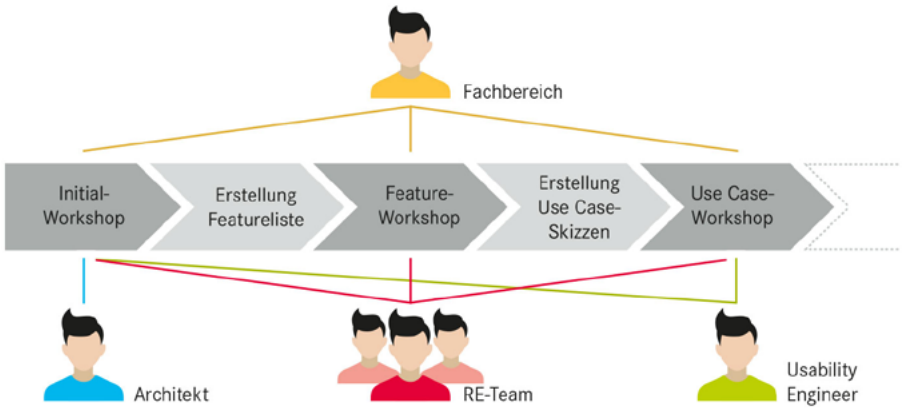

Abb. 2: Skizze des Vorgehensmodells zur Anforderungserhebung

male Methoden einen Mehrwert liefern (vgl. [Sie14, New15]). Als Mittelweg zwischen natürlichsprachlichen Anforderungen und formalen Spezifikationen liefern domänenspezifische Sprachen die Basis dafür, dem RE-Team die Arbeit effizienter zu gestalten ohne die Verständlichkeit für den Fachbereich zu reduzieren.

Was sind DSLs

Domänenspezifische Sprachen (Domain Specific Languages, DSL) sind formale Sprachen für ein spezifisches Anwendungsgebiet. Für das Requirements Engineering sind insbesondere DSLs interessant, die alle Artefakte eines Anforderungsdokuments abbilden können. Dies sind beispielsweise textuelle Anforderungen, Use Cases oder auch ein konzeptionelles Datenmodell.

Wie bei Programmiersprachen gibt es eine Syntax, mit deren Hilfe die Fachlichkeit beschrieben werden kann (siehe Abbildung 1). DSLs können ausgewertet, interpretiert und durch automatisierte Checks überprüft werden. Beispielsweise kann die Konsistenz von Referenzen geprüft werden. Weiterhin können die textuell beschriebenen Sachverhalte automatisiert auf verschiedene Arten präsentiert werden, z. B. als tabellarischer Use Case, als Aktivitäts- oder Use Case-Diagramm (siehe Abbildung 1).

Unser Anwendungsfall

Theoretisch liefern DSLs eine Lösung für das Spannungsfeld zwischen formalisierten Anforderungen und gleichzeitiger Verständlichkeit der Spezifikation für die Fachbereiche. Um zu überprüfen, ob ein DSL-basierter Ansatz auch in der Praxis funktioniert, wurde ein Lastenheft per DSL erstellt. Dabei kam das Werkzeug YAKINDU Requirements [YAKINDU] in der Team-Edition inklusive der unveränderten, mitgelieferten DSL für Anforderungen zum Einsatz.

Ausgewählt wurde ein neues, von der Größe her überschaubares IT-System (rund 80 Features), für welches im Auftrag des Fachbereichs ein Lastenheft zu erstellen war. Das System stellte eine Neuentwicklung ohne Vorgängersystem dar, weshalb keine Alt-Spezifikation existierte. Es wurde als Pilotprojekt ausgewählt, da das vorgesehene Zwei-Personen-RE-Team bereits gute Vorkenntnisse der Fachlichkeit sowie Erfahrung mit Systemen wie LaTeX hatte. In der Spezifikationsphase wurden zusätzlich sowohl ein Architekt als auch ein Usability Engineer zeitweise eingebunden.

Vorgehen

Das Vorgehen bezüglich Anforderungserhebung und -validierung war stark Workshop-getrieben (siehe Abbildung 2). In Richtung Fachbereich war keine Änderung des üblichen Vorgehens notwendig. In verschiedenen Workshops wurden einzelne Themenaspekte in unterschiedlicher Besetzung bearbeitet. Die Ergebnisse der Workshops wurden dokumentiert und im Falle der RE-Artefakte nach den Workshops in der gewählten DSL dokumentiert.

Ein direktes Dokumentieren in der DSL während der Workshops hat sich als nicht praktikabel herausgestellt. Bei den Validierungs-Workshops wurden aus der DSL generierte Artefakte als Basis verwendet. Da die DSL Möglichkeiten bot, Benutzerschnittstellen-Mockups einzubinden und genauer zu beschreiben, wurde diese Möglichkeit genutzt. Der Usability Engineer konnte sein Tool der Wahl für die Erstellung der Mockups verwenden, die genaue Beschreibung der Masken fand jedoch in der DSL statt.

Erfahrungen

Für das RE-Team war der Einsatz der DSL gleich zu Beginn leicht. Durch sinnvoll benannte Schlüsselwörter und Unterstützung im Editor konnte die DSL schnell verstanden und verwendet werden. Eine explizite Schulung war nicht notwendig.

Insbesondere bei komplexeren Anforderungsartefakten wie z. B. Use Cases führte der Einsatz der DSL zu deutlichem Effizienzgewinn. Rein textuelle Anforderungen (z. B. die Features) haben vom Einsatz der DSL erwartungsgemäß nicht profitiert.

Weil die Struktur bereits durch die DSL vorgegeben war und kein Aufwand in die Formatierung eines Use Cases gesteckt werden musste, konnten die Use Cases schneller als gewohnt durch das RE-Team erstellt werden. Die Verwendung bereits definierter Akteure steigerte zum einen die Geschwindigkeit bei der Erstellung, führte aber auch zu einer verbesserten Konsistenz der Spezifikation.

In der Praxis war hilfreich, dass zu jeder Zeit sowohl eine vollständige Darstellung der Use Cases in tabellarischer Form als auch dazu konsistent eine übersichtliche grafische Darstellung in Form von Aktivitätsdiagrammen existierte, die bei jeder inhaltlichen Änderung automatisch vom Werkzeug aktualisiert wurden.

Die Tatsache, dass auch das konzeptionelle Datenmodell in der DSL abgebildet werden konnte, war ein deutlicher Vorteil – insbesondere, wenn Änderungen anstanden. Auch sonst wird das konzeptionelle Datenmodell textuell beschrieben, doch meist wird es durch eine grafische Darstellung (ein einfaches UML-Klassendiagramm) ergänzt. Dieses wird manuell erzeugt und als Grafik in die Spezifikation eingebunden. Bei Änderungen sind sowohl Text als auch die Grafik manuell anzupassen. Dieser Anpassungsschritt entfiel durch die direkte Beschreibung mithilfe der DSL und der Tatsache, dass die Grafik automatisiert erstellt wurde.

Da das Referenzieren von benannten Elementen (z. B. Anforderungen, Use Cases, Daten) direkt textuell an vielen Stellen möglich war, wurden mehr Abhängigkeiten explizit dokumentiert als dies üblicherweise beispielsweise in IBM Rational DOORS [DOORS] geschehen würde. Der Aufwand einer Referenzierung war aufgrund der Auto-Vervollständigung so gering, dass eine Referenz auch dort dokumentiert wurde, wo man sonst ggf. auf eine explizite Dokumentation verzichtet hätte.

Überraschenderweise war es bei paralleler Bearbeitung der Spezifikation sehr hilfreich, dass Referenzen nicht zu jeder Zeit korrekt sein mussten und erst bei Bedarf wieder korrigiert werden konnten. So konnte beispielsweise schon auf einen Use Case referenziert werden, bevor dieser im zentralen Repository verfügbar war.

Vorteilhaft gegenüber rein textuellen Spezifikationen war, dass benannte Elemente umbenannt werden konnten und alle zugehörigen Nennungen ebenfalls automatisch geändert wurden. Die Gefahr, durch simples „Suchen-und-Ersetzen" auch falsche Elemente umzubenennen, war dadurch deutlich reduziert.

Durch die Generierung des Lastenheftes war sichergestellt, dass alle Darstellungen im kompletten Dokument konsistent sind. Dadurch entfiel der Aufwand, nach der inhaltlichen Konsistenzsicherung auch die Darstellung konsistent zu machen.

Zusammenarbeit mit anderen Disziplinen

Im Projekt waren für die Spezifikationsphase sowohl ein Architekt als auch ein Usability Engineer (UE) eingebunden. Beide kamen mit der DSL nicht direkt in Kontakt, sondern jeweils nur mit aus der DSL erzeugten Artefakten (Dokumente bzw. Tabellen, siehe Abbildung 3). Während die gewählte DSL auch Beschreibungsmittel für die grafische Oberfläche bereitstellte, enthielt sie keine spezifischen Inhalte für die Dokumentation der Systemarchitektur.

Die Zusammenarbeit mit dem Architekten hat sich durch den Einsatz der DSL nicht verändert. Das Schnittstellen-Artefakt zwischen Anforderungen und Architektur namens „Systemkontextdiagramm" wurde zwar mithilfe der DSL erzeugt, doch hier war kein Mehrwehrt gegenüber klassischer Dokumentation zu erkennen. Die Architektur wurde weiterhin in einem separaten Dokument festgehalten.

Die Zusammenarbeit mit dem Usability Engineer hingegen hat vom Einsatz der DSL profitiert. Zum einen war es den REs möglich, identifizierte Masken frühzeitig ohne großen Aufwand initial referenzierbar zu definieren, zum anderen führte die höhere Formalisierung der Oberflächenbeschreibung zu reduziertem Dokumentationsaufwand bei gleichzeitig erhöhter Qualität.

Da es in der DSL bequem (d. h. einfach und schnell) möglich war, auf Masken bzw. Elemente auf den Masken zu verweisen, wurde stärker als sonst üblich auf die Masken referenziert. Die Ergebnisse des UEs sind zum Ende des Projektes vollständig im Lastenheft aufgegangen.

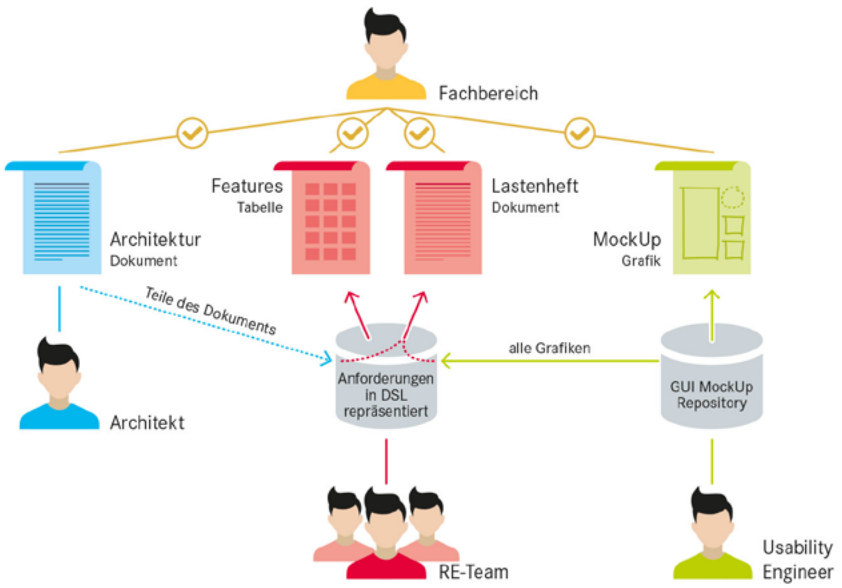

Abb. 3: Erstellte Dokumente inkl. Verantwortlichkeiten

Anforderungsverifikation und -validierung

Die Verifikation der Anforderungen war im Vergleich zu rein textuellen Spezifikationen deutlich schneller, da einige formale Fehler bereits durch die DSL ausgeschlossen waren. Gegenüber einer Spezifikation in DOORS waren die Vorteile bei der Verifikation geringer, da auch DOORS z. B. die Einhaltung von Attributsdefinitionen sicherstellt. Solange die DSL eingehalten war (was das eingesetzte Werkzeug sicherstellte) war klar, dass z. B. Use Cases richtig strukturiert waren, Referenzen valide waren und grafische Darstellungen konsistent zur textuellen/tabellarischen Darstellung waren.

Bei der fachlichen Validierung konnte das DSL-basierte Vorgehen jedoch noch deutlicher punkten. Die Möglichkeit, die beschriebenen Inhalte auf verschiedene Arten ohne manuellen Aufwand auszugeben, war sehr hilfreich.

Für die frühzeitige Validierung der textuell beschriebenen Features hat es sich bewährt, diese als Excel-Tabelle zu generieren und dem Fachbereich zum Review zur Verfügung zu stellen. Die Use Cases – als zentrales Element der Spezifikation – ließen sich gut anhand der automatisch generierten Aktivitätsdiagramme mit dem Fachbereich in Workshops validieren.

Dabei war auffällig, dass die generierten Diagramme zwar inhaltlich korrekt waren, aber die Inhalte nicht immer so dargestellt waren, wie man sich dies für die Übersichtlichkeit und Erklärbarkeit gewünscht hätte. Die finale Validierung des vollständigen Lastenheftes konnte über ein generiertes Dokument erfolgen. Die im Dokument als Hyperlink enthaltenen Referenzen stellten sich dabei als sehr hilfreich heraus.

Werkzeugunterstützung

Die von uns eingesetzte Werkzeugversion sowie die zugehörige DSL stellten sich im Nachhinein als nicht ganz optimal heraus. Während Use Cases sehr gut mittels der DSL beschrieben werden konnten, stellte sich im Praxiseinsatz beispielsweise beim Datenmodell heraus, dass die Mächtigkeit der mitgelieferten DSL für unseren Anwendungsfall nicht ausreichend war.

So musste für das konzeptionelle Datenmodell auf einige Workarounds zurückgegriffen werden, um dieses fachlich korrekt zu beschreiben. Die DSL erlaubte beispielsweise nicht, Multiplizitäten von Assoziationen vollständig zu beschreiben.

Neben einigen Kinderkrankheiten war das erzeugte Dokument für uns der größte Stolperstein des eingesetzten Werkzeuges. Zwar war die Darstellung inhaltlich korrekt und konsistent, doch war die Darstellungsqualität aus unserer Sicht nicht durchgängig kundentauglich. Unschöne Inhaltsverzeichnisse, unerklärliche Seitenumbrüche und sehr ungünstig getrennte Tabellen führten dazu, dass das RE-Team das generierte Dokument noch manuell überarbeiten musste. Diese Nacharbeiten wären eventuell mit der von uns nicht ge-

testeten Enterprise-Edition von YAKINDU Requirements aufgrund des dort enthaltenen Report-Designers nicht notwendig gewesen.

Fazit und nächste Schritte

Mittels DSLs können in der Praxis effizient hochwertige Spezifikationen erstellt werden. Die Fachbereiche profitieren von der erhöhten Qualität und Geschwindigkeit ohne Verlust an Verständlichkeit. Da das erstellte Dokument für den Fachbereich gut verständlich war und seinen Qualitätsansprüchen genügte, sprach aus Sicht des Fachbereiches nichts gegen die Verwendung der DSL durch das RE-Team.

Die Zusammenarbeit mit anderen Disziplinen profitiert vom Einsatz der DSL aufgrund der von vorne herein höheren Qualität der RE-Artefakte. Die Einbindung von UE-Artefakten in die DSL hat sich als sinnvoll erwiesen, da es inhaltlich viele Überschneidungen mit RE gibt. Die Zusammenarbeit mit dem Architekten hat weniger vom Einsatz der DSL profitiert. Da es weniger Überschneidungen mit RE gibt, werden wir hier auch in Zukunft keine stärkere Integration in die DSL anstreben.

In den Workshops wäre ein grafischer Editor hilfreich gewesen, mit dem man direkt in der DSL dokumentieren bzw. vorhandene Inhalte schnell editieren kann (z. B. Schritt-Reihenfolge in Use Cases ändern). Die dadurch hinfällige Transformation der Workshop-Ergebnisse in die DSL hätte die Effizienz weiter gesteigert. Der an der Universität Zürich entwickelte Editor FlexiSketch (siehe [Wue15]) zeigt, dass eine solche Flexibilität möglich ist, es fehlt ihm aber die Anbindung an die von uns verwendete DSL.

Obwohl wir überzeugt davon sind, dass sich eine allgemeingültige DSL für Requirements Engineering erstellen lässt (siehe SysML/UML), denken wir, dass der größte Effizienzgewinn durch Erstellung einer firmenspezifische RE-Sprache erreicht wird. Wir planen daher, eine vollständige DSL zu erstellen, die alle bei uns typischen RE-Artefakte abdeckt. Durch geeignete Dokumentengenerierung sollen fachbereichstaugliche Dokumente erzeugt werden.

Schon im ersten Pilotprojekt konnte trotz für unseren Anwendungsfall unvollständiger DSL und teilweise verbesserungswürdiger Werkzeugunterstützung ein Effizienz- und Qualitätsgewinn festgestellt werden. Das final an den Kunden-Fachbereich gelieferte Lastenheft wurde unseren Qualitätsansprüchen gerecht. Daher planen wir auch in Zukunft ausgewählte Projekte mittels DSL zu spezifizieren. ■

Referenzen

[DOORS] IBM Rational DOORS, http://www-03.ibm.com/software/products/de/ratidoor

[New15] C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, M. Deardueff, How Amazon Web Services Uses Formal Methods, Communications of the ACM, Vol. 58, No. 4, April 2015

[Sie14] S. Siegl, T. Wöhrl, Von natürlichsprachlichen zu formalen Anforderungen – zwei Werkzeuge im Praxistest, OBJEKTspektrum, RE/2014

[Wue15] Dustin Wüest, Norbert Seyff, Martin Glinz, FLEXISKETCH TEAM: Collaborative Sketching and Notation Creation on the Fly, In: 37th International Conference on Software Engineering, ACM/IEEE, 2015.

[YAKINDU] YAKINDU Requirements, Team Edition 1.1.1, http://www.yakindu.de/requirements/