Entscheidungsunterstützungswerkzeug für Release Management

Bernd Bruegge, Korbinian Herrmann, Ivo Bonev, Florian Schneider

Lehrstuhl für Angewandte Software Technik
Technische Universität München
Boltzmannstraße 3
D-85748Garching
bruegge@in.tum.de
herrmann@in.tum.de
bonev@in.tum.de
schneidf@in.tum.de

Abstract: Die heutige Software-Entwicklung ist durch schnelle Änderungen geprägt. Zum einen kommen neue Produkte immer schneller auf den Markt, zum anderen verkürzen sich die Lebenszyklen bestehender Produkte. Dieser Beitrag schlägt ein Entscheidungsunterstützungswerkzeug für das Release Management vor. Es hält die Nachvollziehbarkeit zwischen Anforderungen, Systemmodellen und Management aufrecht. Das Entscheidungsunterstützungswerkzeug erlaubt dem Manager, verschiedene Releasepläne mit alternativem Lieferumfang unter Zuhilfenahme eines 3D-Modells visuell zu vergleichen.

1 Einführung

In traditionellen Softwarelebenszyklen wird mit einem Release (deutsch: Freigabe) eine im Pflichtenheft festgelegte Funktionalität geliefert. Kann die gesamte Funktionalität bis zur Deadline nicht oder nicht in gewünschter Qualität fertig gestellt werden, werden daraus resultierende Arbeiten in einer an die Lieferung angeschlossene Wartungs-Phase durchgeführt. [Ro98] Diese Vorgehensweise wird heutigen Anforderungen nicht mehr gerecht: Zum einen kommen neue Produkte immer schneller auf den Markt, zum anderen verkürzen sich die Lebenszyklen bestehender Produkte. Oft sind die Anforderungen bei Projektstart noch nicht klar formuliert, sodass ein Releaseplan erst während ihrer Ermittlung entstehen kann. Ein weiteres Beispiel ist die Abhängigkeit von einer neuartigen Technologie, deren Lieferung sich verzögert, was sich auf den Releaseplan des Produktes auswirkt. In jedem dieser Fälle muss auf die Änderung reagiert werden und die dann getroffene Entscheidung in den Releaseplan eingearbeitet werden.

Reguläres Build Management ist ein Versuch, einem Teil dieser Problematik gerecht zu werden: Bei dieser Methode wird Release Management nicht länger als Entwicklungsaktivität am Ende eines Projektes, sondern als Projektfunktion eingeführt. Ein wichtiger

Aspekt dieser Methode ist es, nach jeder Änderung eine lauffähige Version des Systems zu erstellen auch wenn kein Liefertermin anliegt. Reguläre Build Management Methoden können als pragmatische, opportunistische Vorgehensweisen bezeichnet werden, die es erlauben immer etwas liefern zu können. Alle bisher erzeugten Freigaben sind Kandidaten für die endgültige Lieferung an den Kunden.

Agile Methoden erlauben eine noch schnellere Folge von Freigaben, bei denen Änderungen nicht mehr auf die Wartungs-Phase verschoben werden, sondern noch während der Projektlaufzeit beim nächsten anstehenden Release berücksichtigt werden können. XP unterstützt dieses Konzept mit „User Stories“. Scrum benutzt das Konzept des „ potentially shippable product increments“ [Mo06] am Ende eines Sprints. Nicht bearbeitete Sprint Backlog Items werden im nächsten Sprint bearbeitet und beeinflussen somit erst das nächste Release [SB02]. Im Wesentlichen kann man agile Methoden deshalb lediglich als Verbesserung des regulären Build Management ansehen.

Was aber wirklich gebraucht wird, sind Methoden, die die Entscheidungsfindung bei der Releaseplanung unterstützen. Eine solche Methode muss es dem Projektmanager erlauben, die Auswirkungen einer Änderung auf den Releaseplan zu sehen. Wenn beispielsweise ein wichtiger Entwickler das Unternehmen noch während der Projektlaufzeit verlässt, ist eventuell die Lieferung der dem Kunden versprochenen Funktionalität gefährdet. Hier ist es wichtig, dem Projektmanager ein Werkzeug in die Hand zu geben, mit dem er alternative Releasepläne erstellen und miteinander vergleichen kann. Die Verlagerung einer geplanten Funktionalität auf ein späteres Release ist eventuell für den Kunden akzeptierbar, aber nur wenn dadurch keine kritische Kernfunktionalität verletzt wird. Derartige Entscheidungen werden heute immer noch aus dem „Bauch heraus“ getroffen, ohne dass eine fundierte Entscheidungsgrundlage vorliegt.

Das Ziel dieses Beitrags ist es, ein Framework für die Unterstützung von Release Management zu präsentieren, das dem Projektmanager erlaubt, derartige Entscheidungen wohlinformiert zu treffen. Eine Nachverfolgbarkeit der Anforderungen unterstützt den Projektmanager: Wenn eine bestimmte Anforderung nicht geliefert werden kann, werden sofort die Auswirkungen auf die in der Releaseplanung zu berücksichtigenden Faktoren aufgezeigt. Beispiele für solche Faktoren sind die Anforderungen selbst, ihre Implementierung, Dringlichkeit für den Kunden und Risikomanagement. [Ru05] Um die Abhängigkeiten zwischen den Anforderungen und der Implementierung aufzuzeigen, wird die Modellierung des Systems auf verschiedenen Abstraktionsniveaus in Abhängigkeit von den Phasen des eingesetzten Produktlebens-Zyklus erlaubt. Damit die zu berücksichtigenden Faktoren aktuell sind, erfolgt eine nahtlose Integration des Release Managements mit Aufgaben des Projektmanagents und der Systementwicklung. Zur Unterstützung der Entscheidung können mehrere alternative Releasepläne mit ihren Auswirkungen auf das Gesamtsystem verglichen werden. Um Entscheidungen, die in der Vergangenheit getroffen worden sind, nachschlagen zu können, unterstützt das vorgeschlagene Werkzeug die Betrachtung der zeitlichen Entwicklung der Releasepläne. Schließlich wird eine Visualisierung vorgeschlagen, die es erlaubt, alternative Releasepläne zu vergleichen und deren Auswirkungen auf das System zu betrachten.

Bestehende Release Management-Werkzeuge, wie der Release Planner Prototype [Ca02] oder der Release Planner [Ru05], unterstützen nicht die Anforderungsermittlung. In der Folge ermittelt man die Anforderungen mit Hilfe eines weiteren Werkzeuges, beispielsweise DOORS [Te06]. Deshalb kommen derzeit in einem einzelnen Software-Projekt viele Werkzeuge zum Einsatz. Diese Werkzeuge sind unabhängig von einander und haben ihr eigenes Repository, um ihre Projekt-Daten zu verwalten. Es fehlen Notifikationsmechanismen, die es erlauben, dass Änderungen in einem Repository auch zu Änderungen in anderen Repositories führen. In der Folge treffen Projektmanager auf Grund von veralterten und nicht konsistenten Daten falsche Entscheidungen.

Im nächsten Abschnitt soll anhand eines visionären Szenarios gezeigt werden, wie ein Projektmanager optimal beim Release Management unterstützt wird. Im Anschluss daran wird das Sysiphus-Framework vorgestellt, das Nachvollziehbarkeit erlaubt. Hierauf aufbauend wird ein Konzept eingeführt, das mehrere alternative Freigaben managen kann.

2 Szenario: Entwicklung eines Multimedia-Players

In diesem Abschnitt wird gezeigt, wie ein Projektmanager bei seinen Entscheidungen unterstützt wird, einen Releaseplan während der Erfassung der Anforderungen aufzustellen sowie bei Änderungen anzupassen.

2.1 Release Management bei Änderungen in den Anforderungen

Als Beispiel dient ein Projekt zur Entwicklung eines Multimedia-Players, der Songs, Podcasts, TV-Shows und Fußballspiele abspielen kann. Die Anforderungen werden in 2 Iterationen ermittelt: In einer ersten Iteration wird zunächst ein Anwendungsfall „SongsAbspielen“ modelliert. Der Projektmanager erstellt einen Plan für Release 1 und weist diesem Release den Anwendungsfall “SongsAbspielen“ zu. An ihm partizipieren die Analyseobjekte „Product“, „Audio Stream“ und „Song“. Abbildung 1 zeigt das Analyseobjektmodell für Release 1 mit dem zugehörigen Releaseplan.

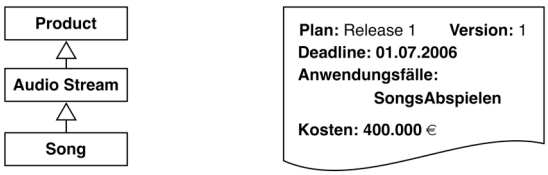

Abbildung 1: Analyseobjektmodell und Releaseplan für das Release 1 eines Multimedia-Players

Als nächstes wird der Anwendungsfall „TV-ShowsAbspielen“ identifiziert. Der Projektmanager entscheidet, diese Anforderung vorerst nicht zu liefern und plant sie für Release 2 ein. Dazu erstellt er einen Releaseplan für Release 2; Release 2 umfasst den Anwendungsfall „SongsAbspielen“ aus Release 1 und „TV-ShowsAbspielen (siehe Abbildung 2). Der Anwendungsfall „TV-ShowsAbspielen“ hat die partizipierenden

Klassen „Product“, „Video-Stream“ und „TV-Shows“. Abbildung 2 zeigt das Analyseobjektmodell für Release 2. Ihm sind alle identifizierten Analyseobjekte aus Release 1 und die neu identifizierten Analyseobjekte aus Release 2 zugeordnet.

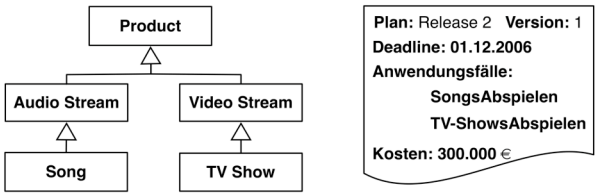

Abbildung 2: Analyseobjektmodell und Releaseplan für das Release 2 eines Multimedia-Players

Zuletzt wird der Anwendungsfall „FußballspieleAbspielen“ ermittelt. Da seine Priorität am geringsten ist, erstellt der Projektmanager einen neuen Plan für Release 3 und ordnet ihm diesen Anwendungsfall zu. Somit verwirklicht das Release 3 alle drei Anwendungsfälle. An dem Anwendungsfall „TV-ShowsAbspielen“ partizipiert die Klasse „Football Game“ als Unterklasse von „Video Stream“. Das Analyseobjektmodell für Release 3 enthält die partizipierenden Objekte aller Anwendungsfälle, die es umsetzt (siehe Abbildung 3).

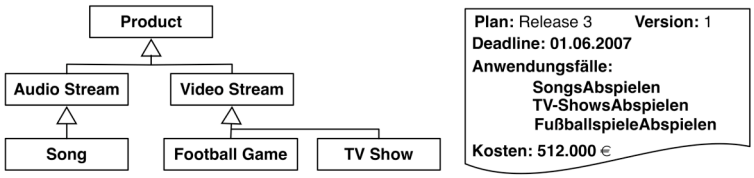

Abbildung 3: Analyseobjektmodell und Releaseplan für das Release 3 eines Multimedia-Players

Noch bevor Release 1 fertig gestellt ist, kommt die neue Technologie Podcasts auf. So wird in der zweiten Iteration der Anforderungsanalyse ein Anwendungsfall „Podcasts-Abspielen“ erstellt. Der Projektmanager entscheidet, diese neue Technologie schnell zu unterstützen und plant sie für Release 2 ein, auch wenn dadurch die Kosten für das Release um 100.000 € steigen. Daher wird das Analyseobjektmodell für Release 2 um die Klasse „Podcast“ als Subklasse von „Audio-Stream“ ergänzt (siehe Abbildung 4).

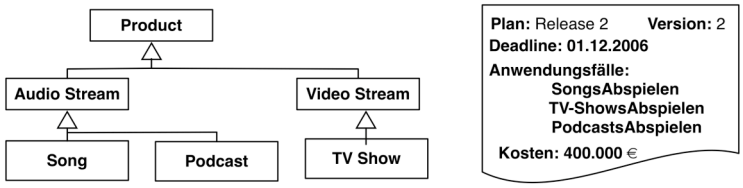

Abbildung 4: Analyseobjektmodell und Releaseplan für das Release 2 eines Multimedia-Players nach Aufkommen der Podcast-Technologie

Der Releaseplan und das Analysemodell von Release 3 werden ergänzt, da in diesem Release „PodcastsAbspielen“ auch verfügbar sein soll (siehe Abbildung 5).

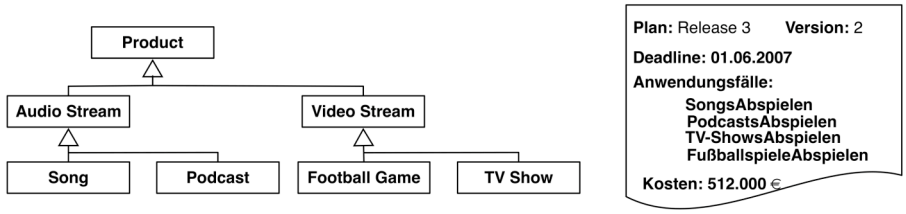

Abbildung 5: Analyseobjektmodell und Releaseplan für das Release 3 eines Multimedia-Players nach Aufkommen der Podcast-Technologie

2.2 Release Management bei Änderungen im Systementwurf

Während des Systementwurfs von Release 1, wird eine Klasse „MP3-Storage“ zur effizienten Ablage von Multimedia Dateien eingeführt. Abbildung 6 zeigt das Objektmodell des Systementwurfs von Release 1.

Abbildung 6: Objektmodell des Systementwurfs von Release 1 eines Multimedia-Players

Der Systementwurf für Release 2 führt eine Schnittstelle „Multimedia-Storage“ ein, welche die Klassen „MP3-Storage“ und „MP4-Storage“ implementieren. Die neue Klasse „MP4-Storage“ dient der Ablage von TV-Shows. Abbildung 7 zeigt das Objekt-modell des Systementwurfs für Release 2. Es enthält zudem die Klassen des Analyseobjektmodells aus Release 2, das der Systementwurf in Richtung eines

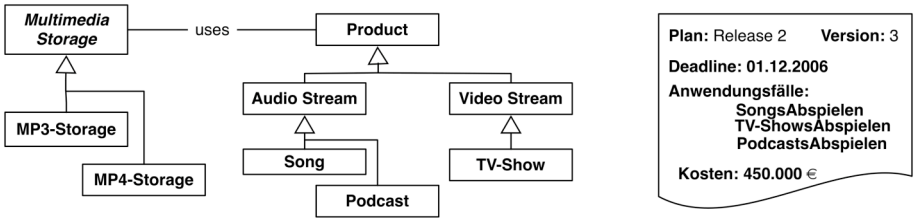

Abbildung 7: Objektmodell für den Systementwurf und Releaseplan für Release 2 eines Multimedia-Players

ausführbaren Systems verfeinert. Zudem wird festgestellt, dass zur Codierung in MP4 eine zusätzliche Komponente gekauft werden muss und sich deshalb die Kosten für das Release um 50.000 € erhöhen. Der Releaseplan wird dementsprechend angepasst (siehe Abb. 7).

2.3 Release Management bei Änderungen in der Organisation

Während der Entwicklung von Release 2 verlassen zwei Mitarbeiter das Projekt. In der Folge stehen nicht mehr genügend Kapazitäten zur Verfügung, um rechtzeitig beide Anwendungsfälle fertig zu stellen. Der Projektmanager muss nun die Releaseplanung ändern und sich zwischen zwei Alternativen entscheiden: Entweder können Podcasts oder TV-Shows realisiert werden. Zum Vergleich dieser beiden Alternativen legt er für jede von ihnen je einen Releaseplan an: Die Release 2-Alternative Podcast und die Release 2-Alternative TV-Show. In der Release 2-Alternative Podcast fällt der Anwendungsfall „TV-ShowsAbspielen“ weg, wodurch die Kosten für Release 2 um 50.000 € für die MP4-Codierung sinken; dafür erhöhen sich die Kosten für Release 3 bei Auswahl dieser Release 2 - Alternative. Für die Release 2-Alternative Podcast wird ein Analyseobjektmodell erstellt, das die Objekte „Product“, „Audio Stream“, „Song“ und „Podcast“ enthält (siehe Abbildung 8).

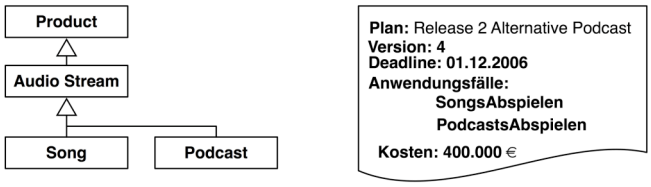

Abbildung 8: Analyseobjektmodell und Releaseplan für die Alternative Podcast des Release 2

Die Release 2-Alternative TV-Show sieht die Lieferung der Anwendungsfälle „SongsAbspielen“ und „TV-ShowsAbspielen“ vor. Durch den Wegfall von Podcasts reduzieren sich die Kosten um 100.000 €. Das Objektmodell dieser Alternative besteht aus den Klassen „Product“, „Audio Stream“, „Song“, „Video Stream“ und „TV-Show“ (siehe Abbildung 9).

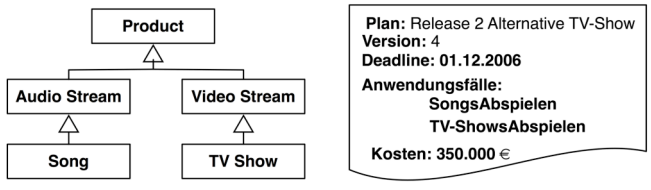

Abbildung 9: Analyseobjektmodell und Releaseplan für die Alternative TV-Show des Release 2

Dies stellt eine Entscheidungsgrundlage für den Projektmanager dar. Er sieht, dass im Analyseobjektmodell der Release 2-Alternative TV-Show zusätzlich eine neue Klasse

zur Unterstützung eines Video Streams hinzugefügt werden muss. Da der Zeitplan drängt, wählt er die sichere Release 2-Alternative Podcast und verschiebt die Realisierung von TV-Shows auf Release 3, obwohl diese Alternative kostengünstiger wäre.

3 Release Management mit Knowledge Nuggets in Sysiphus

Als Wissen eines Software-Projekts bezeichnen wir das Systemmodell – gewöhnlich in UML beschrieben-, das Begründungsmodell [Du06] sowie das Organisationsmodell des Projekts. Im Allgemeinen werden diese Wissensquellen mit unterschiedlichen Methoden erstellt und verarbeitet. Sysiphus ist ein Framework, das auf einem einheitlichen Metamodell zur Repräsentation dieser Wissensquellen aufbaut. [BDW06]. Dieses Metamodell stellt das Projektwissen als Modellelemente dar. Beispielsweise gibt es Modellelemente für Anwendungsfälle, Klassen, Risiken und Terminaufgaben. (siehe Abbildung 10)

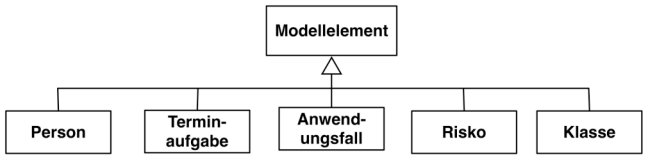

Abbildung 10: Metamodell zur Vereinheitlichung von Projektwissen

Instanzen von Modellelementen können durch Links verbunden werden [BDW06]: Abbildung 11 zeigt als Beispiel den Anwendungsfall „SongsAbspielen“ des Multimedia-Players mit den partizipierenden Klassen „Product“, “Audio Stream“ und „Song“. Der Anwendungsfall Song ist der Person Huber zugewiesen. Es gibt ein Risiko, dass der Anwendungsfall in Release 1 nicht lieferbar ist. Um dieses Risiko zu vermeiden, sollen die zu unterstützenden Datenformate untersucht werden. Diese Terminaufgabe ist der Person Mayer zugewiesen.

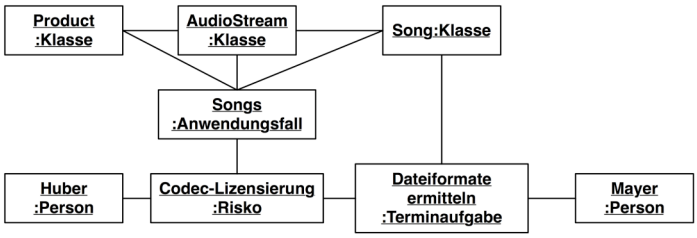

Abbildung 11: Das Sysiphus Framework verbindet unterschiedliche Modellelemente in einem Projektgraph

Die Verlinkung von Instanzen von Modellelementen resultiert in einem Projektgraph (siehe Abbildung 11), der das Wissen eines Software-Projekts vernetzt und so Nachvollziehbarkeit erlaubt. Ein Projektmanager kann entlang der Links durch die Modellelemente eines Projekts navigieren und bestehende Zusammenhänge nachvollziehen. Beispielsweise können alle offenen Terminaufgaben, die den Anwendungsfall „SongsAbspielen“ betreffen, aufgelistet werden.

Der Projektgraph enthält das Projekt-Wissen für alle Freigaben. Um ein spezifisches Release zu managen ist jedoch nur ein Teil dieses Wissens von Bedeutung. Ein Knowledge Nugget ist ein Subgraph des Projektgraphen. Der Knowledge Nugget enthält das Modellierungswissen, das ein Release betrifft. Releasepläne verwenden die Nachvollziehbarkeit, um die Systemmodelle aus den Knowledge Nuggets mit dem Begründungs- und Organisationsmodell zu verbinden. Sysiphus ist um die Modellelemente Knowledge Nuggets und Releaseplan erweitert worden.

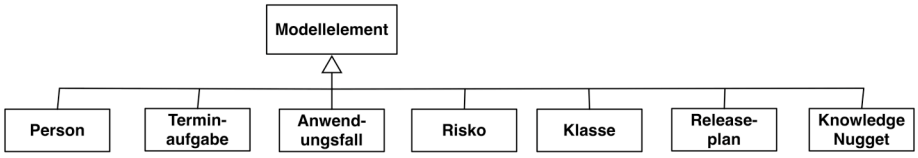

Abbildung 12: Das Sysiphus Framework wurde um die Modellelemente Releaseplan und Knowledge Nugget erweitert.

Das Modellierungs-Wissen wird von Release zu Release anders sein: Beispielsweise kommen neue Anwendungsfälle hinzu, bestehende werden geändert oder fallen weg oder neue Technologien werden eingeführt. Im Beispiel des Multimedia-Players kam die neue Technologie Podcasts auf. Es gibt beispielsweise zwei Vorgehensweisen mit solchen Änderungen umzugehen: Eine von visionären Szenarios getriebene Vorgehensweise fügt den Anwendungsfall hinzu, bevor die Technologie verfügbar ist, und wartet dann ab, bis es sie wirklich gibt. Die zweite Vorgehensweise reagiert auf das Herauskommen einer neuen Technologie, indem sie einen Anwendungsfall spezifiziert. Solche Änderungen am Wissen zwischen Freigaben entsprechen einem Homomorphismus. [HB06]

Abbildung 13: Es gibt zwei Möglichkeiten von Release 1 zu Release 2’ zu kommen: Die visionärszenario- getriebene Vorgehensweise, die zunächst einen Anwendungsfall erstellt und die technologiegetriebene Vorgehensweise, die erst nach Aufkommen der Technologie den Anwendungsfall hinzufügt.

Das Szenario sieht die Modellierung der Freigaben auf den Abstraktionsniveaus Analyse und Systementwurf vor. Für die Freigaben 1, 2 und 3 werden insgesamt sechs

Knowledge Nugget Instanzen $N_{i,j}$ angelegt, wobei i das Release und j das Abstraktionsniveau beschreibt.

Werden die Systemmodelle eines Knowledge Nugget geändert, müssen die Modelle anderer Nuggets aktualisiert werden. Um zu klären, welche Nuggets von einer Änderung betroffen sind, wird eine partielle Ordnung auf den Instanzen der Knowledge Nuggets eingeführt:

$$N_{\text{Release1, Analyse}} < N_{\text{Release 2, Analyse}} < N_{\text{Release3, Analyse}} \text{ und}$$

$$N_{\text{Release1, Systementwurf}} < N_{\text{Release2, Systementwurf}} < N_{\text{Release3, Systementwurf}} \text{ und}$$

$$N_{\text{Release1, Analyse}} < N_{\text{Release1, Systementwurf}} \text{ und}$$

$$N_{\text{Release2, Analyse}} < N_{\text{Release2, Systementwurf}} \text{ und}$$

$$N_{\text{Release 3, Analyse}} < N_{\text{Release3, Systementwurf}}$$

Entlang dieser Ordnung wird das Modellierungswissen weitergeleitet. Abbildung 14 zeigt die Knowledge Nuggets, wobei die Pfeile zeigen, wie das Wissen weitergeleitet wird.

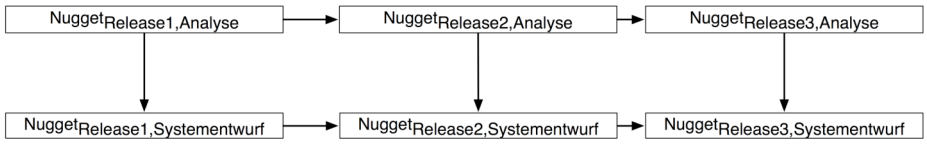

Abbildung 14: Knowledge Nuggets für die Realisierung des Multimedia-Players

4 Visualisierung

Das Rationale-based Analysis Tool (RAT) von [WD04] ermöglicht bisher nur die Darstellung eines Modellelements zu einem bestimmten Zeitpunkt. Ein Releaseplaner sollte jedoch mehrere alternative Releasepläne zusammen ihren zugehörigen Modellen miteinander vergleichen können. Deshalb schlagen wir eine neue 2D Ansicht vor, die den direkten Vergleich alternativer Releasepläne und deren korrespondierenden Modellen zulässt. Abbildung 15 zeigt im oberen Teil die Releasepläne für die Release 2-Alternativen Podcast und TV-Show des Multimedia-Players. Im unteren Teil wird das zugehörige Analyseobjektmodell gezeigt.

Die zu vergleichenden Releasepläne und Modellelemente wählt der Projektmanager unter Verwendung einer 3D-Navigation aus. In der 3D-Navigation können die x- und die z-Achse frei belegt werden, während die y-Achse stets zur Darstellung von Alternativen verwendet wird. Zur Auswahl von Releaseplänen bietet es sich an, auf der x-Achse die einzelnen Freigaben darzustellen und auf der y-Achse unterschiedliche Versionen des Konfigurationsmanagements zu positionieren. (siehe Abbildung 16).

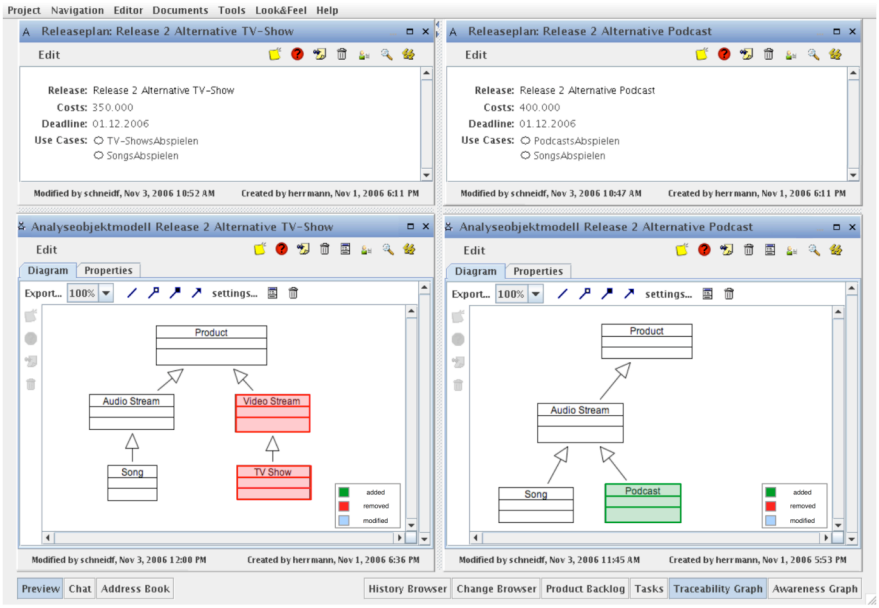

Abbildung 15: Visualisierung alternativer Releasepläne mit zugehörigem Analyseobjektmodell in RAT

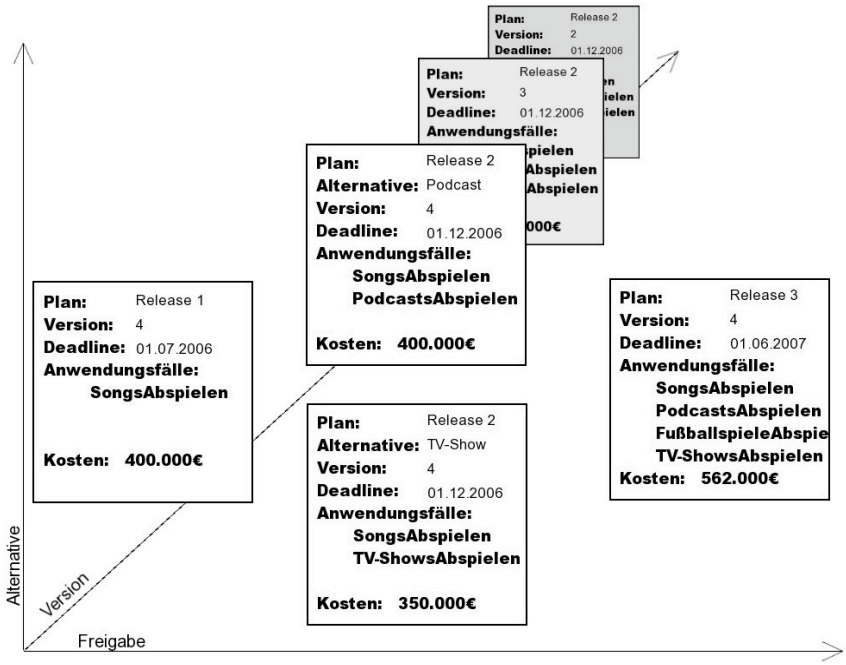

Abbildung 16: 3D-Navigation durch die Releasepläne des Multimedia-Players

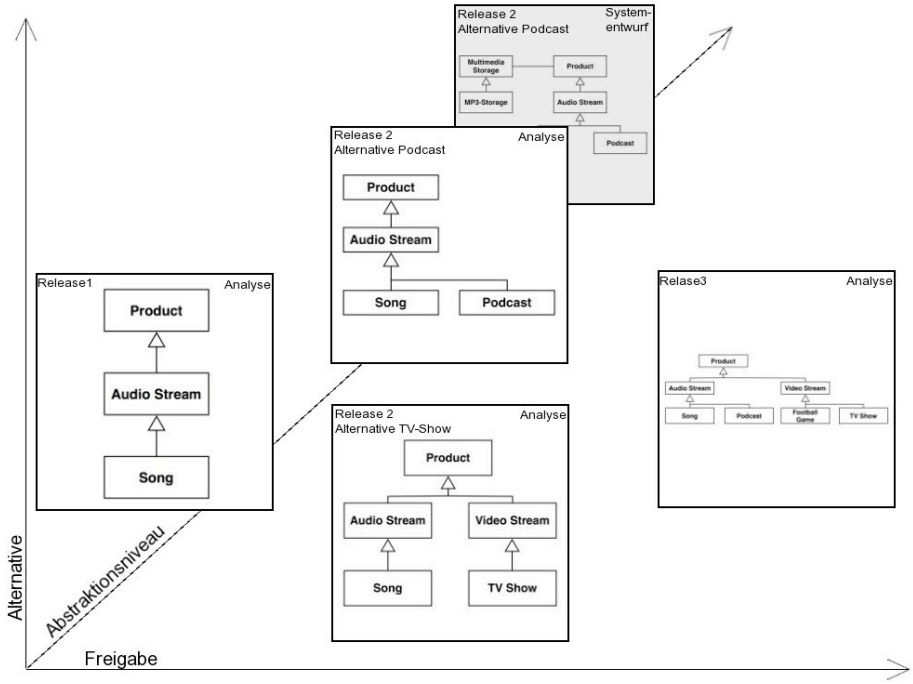

Abbildung 17: 3D-Navigation durch die Objektmodelle der Freigaben des Multimedia-Players

Abbildung 17 zeigt eine 3D-Navigation durch die Objektmodelle. Die x-Achse zeigt Freigaben, die y-Achse Alternativen und die z-Achse das Abstraktionsniveau.

5 Zusammenfassung und zukünftige Arbeiten

In diesem Beitrag schlagen wir ein Entscheidungsunterstützungswerkzeug für Release Management vor. Dieses geht Hand in Hand mit der Entwicklung der Software, so dass der Releaseplan bei Änderungen in den Anforderungen, der Organisation oder der Systementwicklung angepasst werden kann. Grundlage des Entscheidungsunterstützungswerkzeugs ist das Sysiphus Framework, das das Wissen eines Software-Projekts in einem einheitlichen Repository speichert. Knowledge Nuggets halten das Modellierungswissen einer Freigabe zusammen. Dadurch kann das Wissen verschiedener Freigaben unterschieden werden. Knowledge Nuggets und die Nachverfolgbarkeit des Sysiphus-Frameworks ermöglichen die Aufstellung eines Releaseplans. Wir unterstützen Projektmanager bei der Auswahl von Releaseplänen durch eine 3D-Navigation. Sie berücksichtigt neben Alternativen auch die Versionsgeschichte eines Releaseplans. Ausgewählte Releasepläne können zusammen mit ihren UML-Modellen einander gegenübergestellt werden.

Sysiphus und RAT sind bereits seit mehreren Jahren im Einsatz und in zahlreichen Studentenprojekten evaluiert worden. Eine industrielle Fallstudie mit Sysiphus ist geplant. Knowledge Nuggets und Releasepläne sind bereits in Sysiphus implementiert. Die 3D-Visualisierung ist prototypisch realisiert und soll in den nächsten Wochen an Sysiphus angebunden werden. Dann kann eine Evaluierung des Entscheidungsunterstützungswerkzeugs stattfinden.

Literaturverzeichnis

[BDW06] Bernd Bruegge, Allen Dutoit, Timo Wolf: Sysiphus: Enabling informal collaboration in global software development. In Proceedings of the First International Conference on Global Software Engineering, Costão do Santinho, Florianópolis, Brazil, October 16-19 2006

[Ca02] Pär Carlshamre: Release Planning in Market-driven Software Product Development - Provoking an Understanding. In Requirements Engineering Journal, 7(3), Springer, London, 2002, pp. 139-151.

[CS97] Michael A. Cusumano, Richard W. Selby: How Microsoft Builds Software. Communications of the ACM, Vol. 40, No 6, pp. 53-61,1997.

[Du06] Allen H. Dutoit, Raymond McCall, Ivan Mistrík, Barbara Paech: Rationale Management in Software Engineering. Springer, Heidelberg, 2006.

[HB06] Korbinian Herrmann, Bernd Bruegge: Visualization of Release Planing. In Proceedings of First International Workshop on Requirements Engineering Visualization (REV'06), IEEE Requirements Engineering 2006, September 11 - 15, 2006, Minneapolis-St.Paul, Minnesota, USA. Zu erscheinen in IEEE Digital Library. Verfügbar unter http://swen.uwaterloo.ca/~chpkim/re06/.

[Mo06] Mountain Goat Software: „The Scrum Development Process“. http://www.mountaingoatsoftware.com/scrum/ [02.11.2006]

[Ro98] Walker Royce: Software Project Management – A Unified Framework, Addison-Wesley, Massachusetts, 1998.

[Ru05] G. Ruhe: Software Release Planning. In Handbook of Software Engineering and Knowledge Engineering, Vol. 3 (S.K. Chang, Ed.), World Scientific 2005, pp 365-394.

[WD04] Timo Wolf, Allen Dutoit: An Rationale-based Analysis Tool. Proc. Of the 3th International Conference on Intelligent & Adaptive Systems and Software Engineering, Nice, 2004. pp. 209-214.

[SB02] Ken Schwaber, Mike Beedle: Agile Software Development with SCRUM. Prentice Hall, Upper Saddle River(NJ), 2002.

[Sy06] http://sysiphus.in.tum.de [02.03.2006]

[Te06] http://www.telelogic.com/products/doors/index.cfm [02.11.2006]