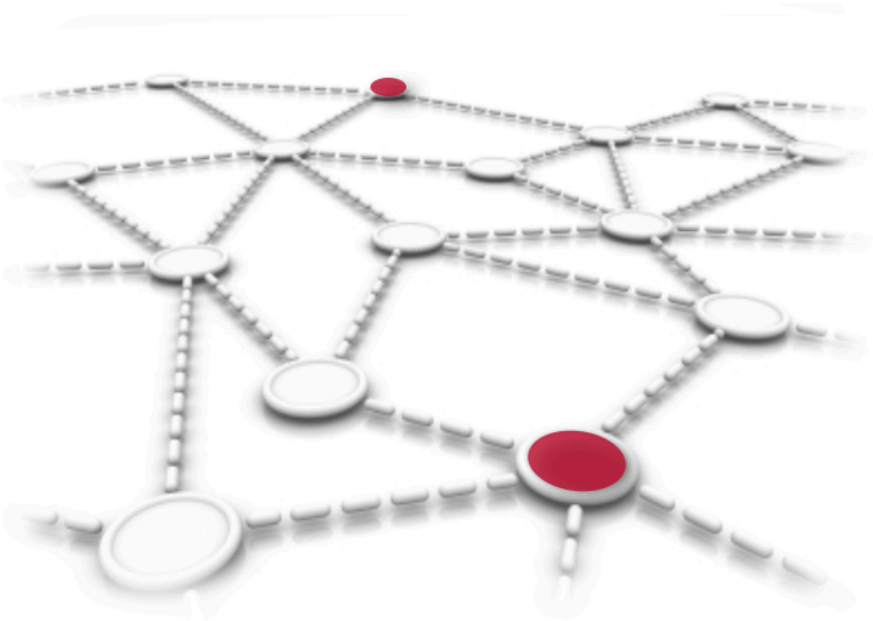
Traceability ist Alles

Old Fashion vs OSLC

Embedded Software Engineering Report Nr 38



Inhalt:

Die modellbasierte Entwicklung von Software mit IBM® Rational Rhapsody® als Modellierungswerkzeug erfordert das Zusammenwirken mit anderen Disziplinen wie zum Beispiel Anforderungs-management, aber auch das Zusammenwirken mehrerer Personen.

Die Auswirkungen werden an unterschiedlichen Vorgehensweisen, aber auch an unterschiedlicher Nutzung des Werkzeugs sichtbar.

Es werden hier sicherlich nicht alle Facetten beleuchtet, aber beispielhaft für das Zusammenwirken von Anforderungen und Modellelementen werden verschiedene Möglichkeiten des Linkens beleuchtet. Sowie zwei Varianten, die das Zusammenwirken mehrerer Personen beleuchten sollen.

Modellbasierte Entwicklung mit IBM® Rational Rhapsody® im Team und mit werkzeugübergreifender Traceability

Referenzierte Werkzeuge

Werkzeuge

IBM® Engineering Requirements Management DOORS® [1], im weiteren Verlauf DOORS [1] genannt

IBM® Engineering Systems Design Rhapsody® [2], im weiteren Verlauf Rhapsody [2] genannt

IBM® Rational Rhapsody® Gateway [3], im weiteren Verlauf Gateway [3] genannt

Willert ReqXChanger [4], im weiteren Verlauf ReqXChanger [4] genannt

IBM® Engineering Requirements Management DOORS® Next [5], im weiteren Verlauf DNG [5] genannt

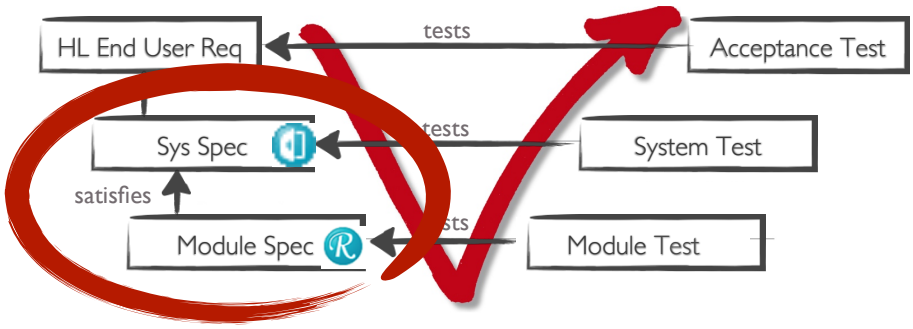
IBM® Engineering Systems Design Rhapsody® - Model Manager [6], im weiteren Verlauf Model Manager [6] genannt

Plattform

Jazz Server [7]

Thema Traceability:
Anforderungen in DOORS, Modell in Rhapsody

In einem sehr stark vereinfachten V-Modell betrachten wir lediglich die beiden in der Graphik gezeigten Teile „Sys Spec“ und „Module Spec“, wobei die System Spezifikation in Form von Anforderungen im Werkzeug DOORS [1] vorliegt, und als Modulspezifikation ein Modell in Rhapsody [2] dienen soll. Dieses Bild soll ohne Bezug zu empfohlenen Entwicklungsumgebungen lediglich dazu beitragen, die notwendigen Schritte zu verdeutlichen, eine geforderte Traceability zwischen Anforderungen und Modellelementen herzustellen.



Nutzt man die beiden Programme DOORS [1] und Rhapsody [2] in ihren ursprünglichen Varianten, so werden die in der „Sys Spec“ vorhandenen Informationen in der DOORS [1] Datenbank und die in der „Module Spec“ vorhandenen Informationen im Dateisystem in Dateien abgespeichert. Per se sind es also getrennte Speicherorte. Um eine Verlinkung zu erstellen, sind die Anforderungen als Modellelement „Requirement“ im Modell notwendig. Erst dann können sie per Diagramm oder Matrix mit einer „Satisfy“-Dependency von einem „normalen“ Modellelement verknüpft werden. Ist dieser Vorgang erfolgt, sind die in Normen geforderten Analysen in einem der beiden Werkzeuge möglich. In Rhapsody [2], da ja in diesem Werkzeug die Verlinkung erfolgt ist; in DOORS [1] allerdings nur, falls die Verlinkungsinformationen dorthin zurück transportiert worden sind.

Zusammengefasst wird man folgende Aktionen durchführen:

- Erstellen einer Anforderung der Sys Spec in DOORS [1]
- Kopie dieser Anforderung nach Rhapsody [2]
- Erstellen entsprechender Modellelemente für die Module Spec in Rhapsody [2]
- Erstellen einer Satisfy-Dependency in Rhapsody [2]
- Kopie dieser Dependency nach DOORS [1]
- Durchführung einer Traceability-Analyse in DOORS [1] oder auch in Rhapsody [2].

Und dieses war erst der einfache Teil in der Entwicklung. Bekanntermaßen ändern sich ja Anforderungen, es kommen welche dazu, manchmal werden auch einige obsolet. Dann beginnt obiger Kreislauf von Neuem. Und jetzt die Gretchenfrage: wer übernimmt das Kopieren? Vorzugsweise ein Programm, beispielsweise das Gateway [3] oder der ReqXChanger [4], allerdings haben wir jetzt 3-fache Datenhaltung, die entsprechenden Anforderungen sind in DOORS [1], im Gateway [3] und im Modell, die Verlinkung im Modell, im Gateway [3] und in DOORS [1]. Eine konsistente Datenhaltung zu gewährleisten erfordert in einer solchen Umgebung eine automatisiert ablaufende Synchronisierung der Daten.

Eine gutes Beispiel dafür ist nachlesbar in einer unserer Referenz-Studien:

"Höchster Automatisierungsgrad durch Modeling-Kompetenz"

www.willert.de/embedded-software-engineering/anwenderberichte

Thema Zusammenarbeit im Werkzeug Rhapsody

Schauen wir uns an, wie Rhapsody [2] Daten auf der Festplatte abspeichert. Im Werkzeug existiert der Begriff einer „Unit“. Eine Unit ist ein Teil des Modells, per default ist ein Package des Modells eine Unit. Der Nutzer kann die default Einstellung ändern, er kann auch andere Modellelemente zu einer Unit machen, oder er kann auch bestehende Units wieder auflösen. Jede Unit wird im Dateisystem als Datei gespeichert.

Variante 1, Zusammenarbeit durch personenbezogene Units

Diese Art des Zusammenarbeitens in einem Team mag in einem kleinen Modell oder auch in einem kleinen Team, das eng zusammenarbeitet und gut miteinander kommuniziert, funktionieren, ist aber nicht zu empfehlen. Die Units des Modells werden hier so festgelegt, dass jeweils nur eine Person in einer Unit arbeitet. Zum Beispiel kann man die Package-Struktur des Modells so wählen, dass für alle Inhalte eines Packages jeweils genau eine Person „zuständig“ ist.

Variante 2, Nutzung eines Configuration Management Systems

Weit verbreitet ist die mehr oder weniger tiefe Integration zwischen Rhapsody [2] und einem CM-Wekzeug. Grundsätzlich unterliegen die Dateien (Units) einer Versionsverwaltung, unterschiedlich ist lediglich die Art, wie gewisse Aktionen der Versionsverwaltung ausgeführt werden. Ein „Check-In“ oder „Check-Out“ kann man sich in Rhapsody durchgeführt denken oder aber im CM-Werkzeug, in dem man die der Unit entsprechenden Datei ein- oder auscheckt. In dieser Variante muss aber genau wie in Variante 1 auf eine vernünftige Granularität geachtet werden, schließlich möchte man weitestgehend die Entstehung von Konflikten vermeiden. Das Lösen eines Konflikts erfordert Zeit, diese hat man ja nicht im Überfluss in einem Projekt der heutigen Komplexität.

Variante 3, Nutzung personenbezogener Teil-Modelle

Jede Person des Teams bearbeitet einen Teil des Gesamt-Modells in einem eigenen kleineren Modell, und zwar so, dass notwendige Teile anderer durch Refenzierung der entsprechenden Units lesbar, aber nicht veränderbar sind. Das Gesamt-Modell wird dann durch Zusammenfügen aller Teile gebildet. Hier muss lediglich beachtet werden, dass Änderungen an Teilen des Modells, die von allen anderen benötigt werden, in kurzer Zeit fehlerfrei durchgeführt werden müssen. Zu empfehlen ist hier natürlich auch die Verwendung eines CM-Systems.

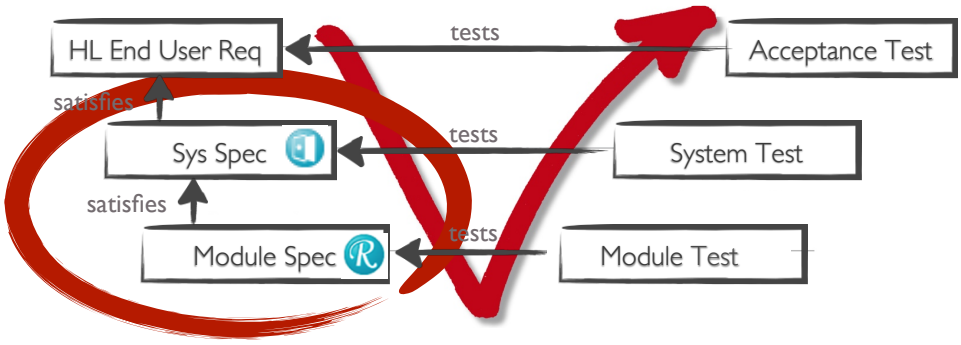
Resümee

Zu der im ersten Kapitel vorgestellten Vorgehensweise mit Anforderungen in DOORS [1], einem Modell in Rhapsody [2] und der Erstellung der Satisfy-Beziehungen zwischen Anforderungen und Modellelemente durch Kopieren, wollen wir die Variante 2 des Zusammenarbeitens innerhalb des Modell mit Rhapsody [2] nutzen.

Und dabei nutzen wir eine automatisierte Synchronisation zwischen DOORS [1] und Rhapsody [2], und darin die Anbindung an ein CM-System.

Thema Traceability: Anforderungen in DOORS NG, Modell in Rhapsody, gespeichert im Model Manager

Wir ändern das Scenario, allerdings nicht die Prinzipien, wir haben weiterhin das auf das Wesentliche reduzierte V-Modell, mit Anforderungen, einem Modell und einer Traceability vom Modell zu den Anforderungen. Aber DOORS [1] wird ersetzt durch DOORS NG [5] und hinzugefügt der Model Manager [6]. Die größte Änderung erfolgt durch eine andere Form der Datenhaltung, es sind nicht mehr 3 unterschiedliche Datenquellen, sondern es handelt sich um einen Speicherort: der Jazz Server [7].



Die auf dem Jazz Server [7] installierten Programme sind Web-Anwendungen. Das hat zur Folge, dass DOORS NG [5] in einem Web-Browser arbeitet. Für einige Anwendungen wird allerdings auch ein lokaler Eclipse-Client genutzt, zum Beispiel für den Model Manager. [6]. Dieser wirkt dann zusammen mit der üblichen Rhapsody [2] Oberfläche, um die Datenspeicherung der Modelldaten auf dem Jazz Server [7] durchführen zu können.

Eine weitere grundlegende Änderung ist die Art der Verlinkung, bisher handelte es sich in DOORS [1] um interne DOORS-Links und im Modell um Satisfy-Dependencies. Jetzt wird beides ersetzt durch OSLC Links. OSLC steht für „Open Services Lifecycle Collaboration“, es handelt sich um eine offene Initiative, die 2008 ins Leben gerufen wurde, um Spezifikationen für die Integration von Werkzeugen auszuarbeiten, mit denen Software entwickelt wird. Die Integration geschieht dadurch, dass Ressourcen sowie die Relationen zwischen ihnen jeweils einen Uniform Resource Identifier (URI) erhalten. Weiterhin ist festgelegt, wie man für einen Benutzer Vorschau, Erzeugung und Auswahl von Links zur Verfügung stellt.

Das Bild hat sich geändert, die in der „Sys Spec“ vorhandenen Informationen sind jetzt in einem Requirements Management Projekt auf dem Jazz Server [7] und die in der „Module Spec“ vorhandenen Informationen in einem Architecturemanagement Projekt auf dem Jazz Server [7] abgespeichert. Das Modell wird mit Hilfe des Eclipse Clients des Model Managers [6] in die Rhapsody Oberfläche [2] geladen. Dort hat man dann die Möglichkeit, die entsprechenden Requirements der „Sys Spec“ direkt per OSLC Link mit Modellelemente zu verlinken.

Zusammengefasst wird man folgende Aktionen durchführen:

- Erstellen einer Anforderung der Sys Spec in DOORS NG [5]
- Erstellen entsprechender Modellelemente für die Module Spec in Rhapsody [2]
- Erstellen einer Satisfy-Dependency in Rhapsody [2]
- Durchführung einer Traceability-Analyse in DOORS NG [5] oder auch in Rhapsody [2].

Es entsteht keine mehrfache Datenhaltung mehr, man benötigt keinen Transporteur, um Daten automatisiert zwischen dem Anforderungs-Managementwerkzeug und dem Modellierungswerkzeug durchzuführen.

Und das ist noch nicht alles, man erhält so ganz nebenbei die zusätzlichen Funktionalitäten eines Change und Configuration Managementwerkzeugs.

Zusammenarbeit im Team

Das Modell wird nicht direkt auf dem Jazz Server [7] bearbeitet, man erzeugt erst einmal einen persönlichen Arbeitsbereich auf dem Jazz Server [7], und anschließend lädt man es daraus in einen Arbeitsbereich auf den lokalen Arbeitsrechner. Sind Änderungen erfolgt, wird man das Modell speichern und durch ein „Check-In“ vom lokalen Rechner auf den persönlichen Arbeitsbereich des Jazz Servers [7] hochladen. Als letzte Aktion können dann die gemachten Änderungen verteilt oder ausgeliefert werden, damit sie dann von allen anderen im Team gesehen werden können. Natürlich ist bei dieser Art des Arbeitens die Möglichkeit eines Konflikts gegeben, dieser muss dann manuell gelöst werden. Vermindern kann man die Menge an Konflikten durch explizites Sperren von Teilen des Modells oder auch durch eine kleinere Portionierung der zu erstellenden Änderungen.

Sonstiges

Man erhält weitere Vorteile en passant. Sie sind hier lediglich kurz erwähnt und nicht eigentliches Thema. Es handelt sich um Änderungsmanagement und damit eng verknüpft um Versionsmanagement, sowie um Variantenmanagement.

Änderungsmanagement

Der Eclipse Client des Model Managers auf dem lokalen Rechner bietet Vorteile eines kontrollierten Änderungsmanagements. Es existiert eine Planungskomponente, man kann Tasks, Defects und andere Workitems erstellen. Man kann ebenso die notwendigen Änderungen am Modell in definierte Change Sets verpacken. Sie werden bei Bedarf auf den persönlichen Arbeitsbereich des Jazz Servers [7] hochgeladen. Als letzte Aktion können dann die gemachten Änderungen verteilt oder ausgeliefert werden, damit sie von allen anderen im Team gesehen werden können. Natürlich ist bei dieser Art des Arbeitens die Möglichkeit eines Konflikts gegeben, dieser ist per Diff/Merge innerhalb von Rhapsody zu lösen.

Auf diese Weise wird die Integration zwischen den Disziplinen Modellierung und Änderungsmanagement überflüssig, sie ist ja bereits im Werkzeug mit vorhanden.

Versionsmanagement

Paralleles Arbeiten und Erstellen von Branches sind durch Nutzung der Arbeitsbereiche möglich. Sie werden auf dem Jazz Server [7] verwaltet und ermöglichen die Speicherung unterschiedlicher Ausprägungen der Modellelemente. Arbeitsbereiche können bei Bedarf miteinander verglichen werden, man kann Teile aus einem Arbeitsbereich in einen anderen ausliefern; idealerweise in den bereits erwähnten Change Sets. Auf diese Weise wird die Integration zwischen den Disziplinen Modellierung und Versionsmanagement überflüssig, sie ist ja ebenso bereits im Werkzeug mit vorhanden.

Variantenmanagement

Unterschiedliche Arbeitsbereiche erlauben paralleles Arbeiten (siehe voriges Kapitel), dieses kann genutzt werden, Varianten eines Produktes zu erstellen.

Resümee

In der im zweiten Kapitel vorgestellten Vorgehensweise mit Anforderungen in DOORS NG [5], einem Modell in Rhapsody [2] im Model Manager [2] haben wir keine Daten zu kopieren, bzw. zu synchronisieren, sondern erstellen wir die Traceability-Beziehungen direkt per OSLC-Link. Als Zugabe erhalten wir ein direktes Versionsmanagement, Änderungsmanagement sowie ein Variantenmanagement.

Autor:

WOLFGANG SONNTAG

Herausgeber:

WILLERT SOFTWARE TOOLS GMBH

Hannoversche Straße 21

31675 Bückeburg

info@sodiuswillert.com

Tel.: +49 5722 9678 - 60

willert.de
sodiuswillert.com

Willert Software Tools ist Teil der Unternehmensgruppe SODIUS WILLERT