



Koordination von Teamarbeit mit Jazz

Jazz ist eine neuartige Werkzeugplattform, die in Systementwicklungsprojekten die Zusammenarbeit im Team auf aufgabenorientierte Weise unterstützt. Dieser erste Beitrag der Artikelserie zu Jazz führt die Kernkonzepte ein. Weitere Beiträge zum praktischen Einsatz im Projekt sowie zu Sourcecode-Management mittels Jazz folgen in den nächsten Heften.

von Michael Müller, Prof. Dr. Veronika Thurner und Martin Wassermann

Softwaresysteme sind aus dem Alltagsgeschäft der meisten Unternehmen nicht mehr wegzudenken und in vielen Fällen essenzielle Grundlage für den Geschäftserfolg. Entsprechend steigen die Anforderungen an Softwaresysteme stetig an, sowohl hinsichtlich der Funktionalität als auch hinsichtlich Qualität, schneller Verfügbarkeit und flexiblem Einsatz. Um diese Forderungen zu bedienen, wird Software heute fast ausschließlich in Teams entwickelt, in denen Personen mit vielfältigen Fähigkeitsprofilen zusammenarbeiten und die oft auf verschiedene geografische Standorte verteilt sind. Die effektive Koordination der Zusammenarbeit aller Beteiligten wird dabei zum entscheidenden Erfolgsfaktor für die erfolgreiche Durchführung eines Entwicklungsprojekts.

Motivation

Die heute gängigen integrierten Entwicklungsumgebungen erhöhen zwar

die Produktivität der einzelnen Projektbeteiligten erheblich, bieten jedoch nur verhältnismäßig wenig Unterstützung für die koordinierte Zusammenarbeit im Team [1]. Stattdessen kommunizieren Entwicklerteams über eine bunte Mischung aus E-Mails, Kommentaren im Quelltext bzw. in der Sourcecode-Verwaltung, Besprechungen oder Instant-Messaging-Werkzeuge. Darüber hinaus wird eine Vielzahl von Werkzeugen eingesetzt, um die gemeinschaftliche Zusammenarbeit zu koordinieren. Die dabei anfallenden Informationen sind meist auf verschiedene Systeme verteilt, untereinander nicht integriert und oft nur den unmittelbar an der Kommunikation beteiligten Personen zugänglich, auch wenn viele der Informationen auch für die anderen Teammitglieder wissenswert wären. Benötigt werden somit Werkzeuge und Mechanismen, die diese Lücke schließen und speziell die Teamarbeit koordinieren. Um die Motivation und Effizienz der einzelnen Projektbeteiligten zu erhalten und gleichzeitig die Produktivität des Teams als Ganzem zu erhöhen, dürfen die einzelnen Teammitglieder dabei möglichst wenig in ihrer individuellen Arbeitsweise eingeschränkt und nicht mit unnötigen Verwaltungsarbeiten belastet werden.

Eine zentrale Grundvoraussetzung für die effiziente Zusammenarbeit ist Transparenz. Diese wird durch die freie Verfügbarkeit aller relevanten Projektinformationen für alle Beteiligten in einer gemeinschaftlich genutzten Projektumgebung erreicht, auf deren Basis sich dann an die jeweiligen Bedürfnisse angepasste, individuelle Sichten konfigurieren lassen. Ebenfalls notwendig ist ein auf die konkrete Projektkonstellation abgestimmter Prozess, der die Projektabwicklung „so viel wie nötig“ koordiniert, ohne dabei jedoch unnötig einzuschränken [2].

Jazz [3] ist eine von IBM Rational entwickelte Plattform, die diese Punkte umsetzt und die Zusammenarbeit im Team über den gesamten Softwareentwicklungszyklus hinweg auf aufgabenorientierte Weise unterstützt. Dabei verbindet Jazz nicht ausschließlich Softwareingenieure miteinander, sondern explizit alle Projektbeteiligten, insbesondere auch die Spezialisten des Anwendungsbereichs und die späteren Nutzer des zu erstellenden Softwaresystems.

Architektur

Jazz ist selbst noch kein Produkt, sondern eine modular aufgebaute Technologieplattform, deren logische Architektur in Abbildung 1 dargestellt ist. Die Grundlage für die Integration von

Artikelserie

Teil 1: Koordination von Teamarbeit

Teil 2: Methodischer Einsatz im Projekt

Teil 3: Sourcecode-Management mit Jazz

Daten und Diensten in Jazz bildet Open Services for Lifecycle Collaboration (OSLC), eine Initiative zur Definition einheitlicher Dienste, über die die verschiedenen, entlang des Softwareentwicklungsprozesses eingesetzten Werkzeuge ihre Daten untereinander austauschen und auch auf semantischer Ebene teilen können [4]. Im Zentrum der Jazz Integration Architecture steht der Jazz Team Server, der auf den OSLC-Diensten aufsetzt und mit den so genannten Jazz Foundation Services die Basisdienste als REST-konforme Web Services zur Verfügung stellt. Hierzu zählen unter anderem Dienste für die Verwaltung von Benutzern und Projekten, Sicherheit, Zusammenarbeit oder die Auswertung von Query-Anfragen.

Auf Jazz basierende konkrete Produktimplementierungen können diese Dienste nutzen und weiter ausbauen. Die verschiedenen Werkzeuge kommunizieren dabei untereinander ausschließlich über ihre Verbindung zur Jazz-Plattform. Von IBM Rational sind derzeit Jazz-basierte Werkzeuge für die Aufgabenbereiche Anforderungsdefinition, kollaborative Softwareentwicklung und Qualitätsmanagement verfügbar (Abb. 1). Als weiteren Jazz-basierten Baustein hat IBM Rational unter anderem mit Tara ein Werkzeug für klassische Projektmanagementaufgaben vorgestellt. Auch andere Hersteller setzen bereits auf der Jazz-Plattform auf, beispielsweise Mainsoft mit dem Produkt Mainsoft Document Collaboration for Rational Jazz, das Rational Team Concert auf Dokumentenebene mit bestehenden Infrastrukturen zur Zusammenarbeit im Team integriert.

Hinsichtlich der physischen Verteilung ist Jazz als klassische Client-/Serverarchitektur konzipiert (Abb. 2). Serverseitig läuft dabei eine JEE-Applikation ab, die die Jazz Foundation Services bereitstellt. Diese Basisdienste können von auf der Jazz-Plattform basierenden Werkzeugen genutzt und bei Bedarf entsprechend ergänzt werden. Ebenfalls vom Server verwaltet wird das Jazz Repository, das sich auf eine relationale Datenbank stützt. Für den Zugriff auf das Repository stellt der Team Server verschiedene Dienste zur Verfügung, die von den Clients aus ge-

Abb. 1:
Logische Architektur von Jazz

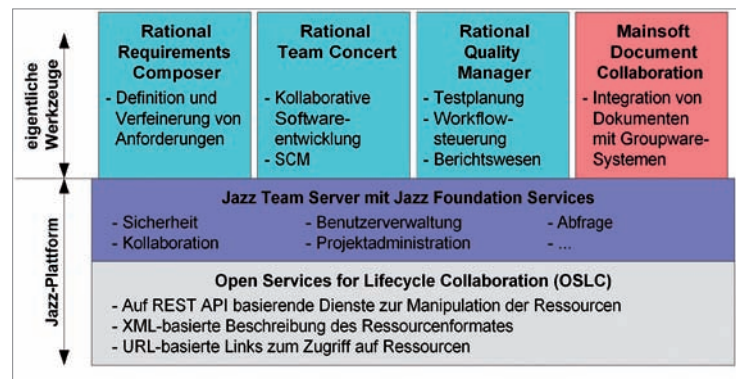
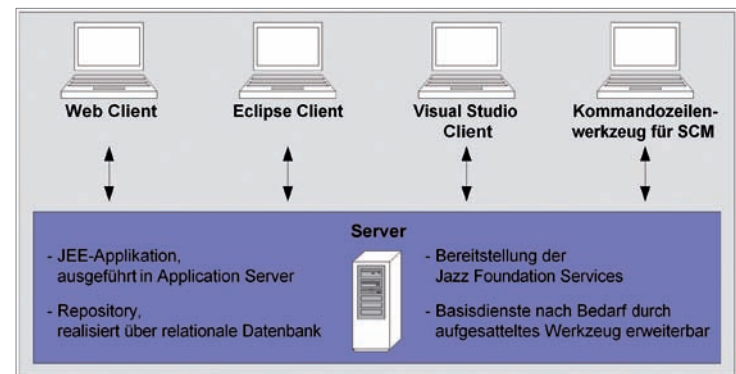


Abb. 2:
Physische Architektur von Jazz



nutzt werden können. Aktuell verfügbar sind ein webbasierter, ein Eclipse- und ein Visual-Studio-Client sowie speziell für den Aufgabenbereich des Sourcecode-Managements ein Kommandozeilenwerkzeug.

Projekt und Prozess

Die Inhalte des Jazz Repositories sind in *Project Areas* gegliedert. In einer solchen *Project Area* sind alle Artefakte abgelegt, die zu diesem Jazz-Projekt gehören. Jedem Projekt ist dabei ein Prozess zugeordnet, der die Abwicklung des Projekts steuert und dessen Rollen- und Rechtemodell, Koordinationsstrukturen, Iterationsplanung und sonstige Prozessregeln über Jazz abgebildet werden. Die Konfigurationseinstellungen des für ein Projekt gewählten Prozesses werden als *Project Process Settings* in der *Project Area* abgelegt (Abb. 3). Aktuell sind vordefinierte Prozess-Templates beispielsweise für Eclipse Way und für Scrum verfügbar.

Eine *Project Area* ist in eine Menge von *Team Areas* untergliedert. Jede *Team Area* beinhaltet alle Konfigurationseinstellungen und Entwicklungsartefakte des zugehörigen Teams. Ein Team wird aus einer Menge von Benutzern gebildet,

die als Teammitglieder der *Team Area* zugeordnet sind. Jedes Mitglied nimmt dabei innerhalb des Teams eine oder mehrere der Rollen ein, die über die Prozessdefinition festgelegt sind. Alternativ kann ein Benutzer auch direkt einem Projekt zugeordnet sein. Jedes Team kann die von seinem übergeordneten Projekt vorgegebene Prozessdefinition an seine Bedürfnisse anpassen. Diese teamspezifischen Prozesseinstellungen werden als *Team Process Settings* in der *Team Area* abgelegt. *Team Areas* sind hierarchisch verfeinerbar.

Zur Definition des Verhaltens, das im Rahmen des definierten Prozesses von einem Projektteam erwartet wird, stehen so genannte *Preconditions* und *Followup Actions* zur Verfügung. Eine *Precondition* definiert dabei eine Vorbedingung, die erfüllt sein muss, bevor ein bestimmter Prozessschritt ausgeführt werden darf. Beispielsweise könnte gefordert werden, dass eine Quelltextdatei fehlerfrei übersetzen muss, bevor sie an das zentrale Repository übergeben wird. Korrespondierend definiert eine *Followup Action*, welche Prozessschritte automatisch auszulösen sind, nachdem ein bestimmter anderer Prozessschritt durchgeführt wurde. Es könnte z. B.

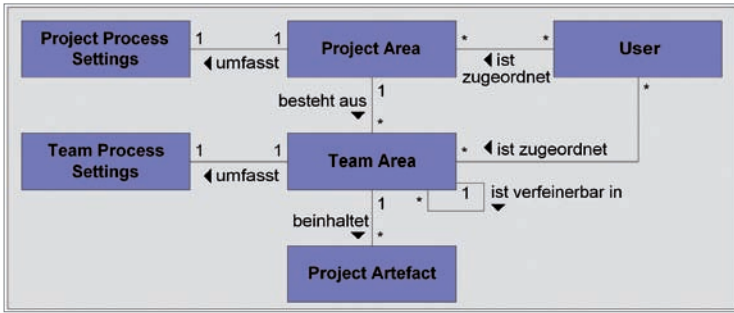


Abb. 3: Repository-Struktur in Jazz

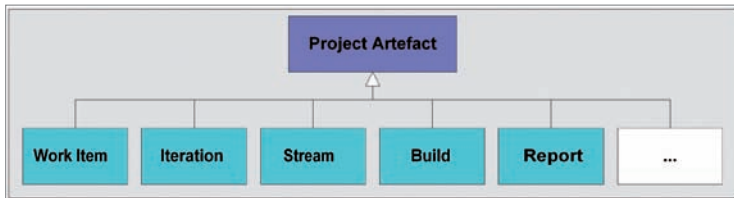


Abb. 4: Projektartefakte

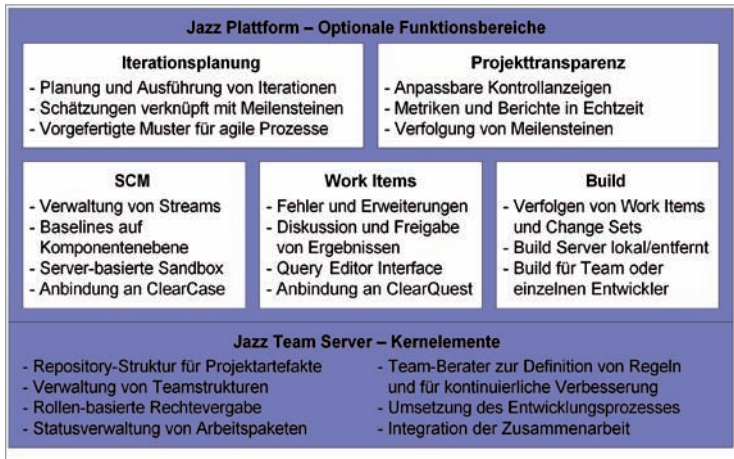


Abb. 5: Funktionsbereiche der Jazz-Plattform

festgelegt werden, dass nach der Integration eines neuen Projektartefakts in das zentrale Repository passende Verifikationstätigkeiten auszuführen sind. Followup Actions können auch an den *Event-Handling-Mechanismus* von Jazz gekoppelt werden. Dieser Mechanismus erzeugt Ereignisse, die über bestimmte Arten von Änderungen informieren. In der Prozesskonfiguration wird festgelegt, auf welche Arten von Ereignissen mit welchen Followup Actions reagiert werden soll. Tritt ein derartiges Ereignis ein, werden die zugehörigen Followup Actions automatisch angestoßen.

Projektartefakte

Neben den bisher aufgeführten Konfigurationseinstellungen beinhaltet eine Team Area als *Project Artefacts* auch alle Entwicklungsartefakte, die das Team im Verlauf des Entwicklungsprozesses erstellt hat. In Rational Team Concert, das

derzeit als Jazz-basiertes Koordinationswerkzeug für die Softwareentwicklung verfügbar ist, sind fünf vordefinierte Arten von Projektartefakten verfügbar (Abb. 4). Andere auf Jazz basierende Werkzeuge können ergänzend je nach Bedarf und Aufgabenbereich eigene Artefakttypen definieren.

Die in einem Projekt durchzuführenden Arbeitsschritte werden als sogenannte *Work Items* geplant. Welche Arten von Arbeitsschritten möglich sind, ist auch hier wieder über die zugrunde liegende Prozessdefinition festgelegt. Jazz setzt voraus, dass die Prozessdefinition die Work-Item-Typen *Defect* für Tätigkeiten zur Fehlerkorrektur und *Task* für die Entwicklung eines neuen Artefakts vorsieht. Weitere Work-Item-Typen können je nach Bedarf ergänzt werden. Beispielsweise erlaubt das Prozess-template für Eclipse Way zusätzlich den Work-Item-Typ *Enhance-*

ment für die Weiterentwicklung eines bestehenden Artefakts, während das Prozess-Template für Scrum ergänzend den Work-Item-Typ *Story* vorsieht, um mehrere inhaltlich zusammenhängende Work Items zu einer Einheit zu bündeln. Jeder Work-Item-Typ ist mit spezifischen Eigenschaften und Zustandsmodellen hinterlegt, die ebenfalls in der zugrunde liegenden Prozessdefinition vordefiniert sind und bei Bedarf angepasst und erweitert werden können. Das Zustandsmodell definiert dabei die verschiedenen möglichen Entwicklungsstadien eines Work Items und legt fest, durch welche Aktionen eine Instanz eines Work Items in einen neuen Zustand übergeht.

Optional kann die Arbeit innerhalb eines Projekts in mehrere Entwicklungslinien aufgeteilt werden. Die zeitliche Abwicklung einer solchen Entwicklungslinie ist in *Iterationen* gegliedert. Iterationen werden jeweils durch ein Start- und ein Enddatum begrenzt und folgen sequenziell aufeinander. Entsprechend ist in einem laufenden Projekt zu jedem Zeitpunkt maximal eine Iteration pro Entwicklungslinie aktiv. Die zeitliche Abwicklung der Work Items wird organisiert, indem ein zu erledigender Work Item einer bestimmten Iteration zugewiesen wird. Optional kann die Abarbeitung der Work Items, die einer bestimmten Iteration zugeordnet sind, mittels eines *Iteration Plans* organisiert werden. Je nach Bedarf können verschiedene Arten von Iterationen definiert und mit spezifischen Prozessregeln hinterlegt werden. Nützlich ist beispielsweise eine Unterscheidung zwischen Entwicklungs- und Stabilisierungsiteration. Während im ersten Fall das Hauptaugenmerk auf der Erstellung neuer Artefakte liegt, fokussiert die Stabilisierungsiteration den Feinschliff des zu erstellenden Systems, typischerweise kurz vor Auslieferung. Entsprechend werden hier über die iterationspezifischen Prozesseinstellungen enge Regeln der Qualitätssicherung erzwungen, um sicherzustellen, dass nur ausgereifte, gut in das Gesamtsystem integrierte Artefakte und Änderungen eingepflegt werden können.

Entwicklungsartefakte können unmittelbar über Jazz versionsverwaltet

Impressum

Verlag:
Software & Support Verlag GmbH

Anschrift der Redaktion:
Java Magazin
Software & Support Verlag GmbH
Geleitsstraße 14
D-60599 Frankfurt am Main
Tel. +49 (0) 69 6300890
Fax. +49 (0) 69 63008989
redaktion@javamagazin.de
www.javamagazin.de

Chefredakteur: Sebastian Meyen
Redaktion: Claudia Fröhling, Hartmut Schlosser, Mirko Schrempf
Chefin vom Dienst: Nicole Bechtel
Schlussredaktion: Nicole Bechtel, Katharina Klassen, Frauke Pesch
Leitung Grafik & Produktion: Jens Mainz
Layout, Titel: Daniela Albert, Kristin Brockmann, Pöpporn Fischer, Karolina Gaspar, Melanie Hahn, Katharina Ochsenhirt, Maria Rudi, Patricia Schwesinger
CD/DVD-Erstellung: Daniel Zuzek

Autoren dieser Ausgabe:
Rodion Alukhanov, Pavlo Baron, Adam Bien, Thilo Frotscher, Oliver Gierke, Kai Glahn, Arno Haase, Alexander Hanschke, Thomas Krautgartner, Klaus Kreft, Angelika Langer, Berthold Maier, Christian Marquardt, Florian Müller, Michael Müller, Hajo Normann, Mirko Novakovic, Alois Reitbauer, Stefan Rook, Lars Röwekamp, Christian Schneider, Michael Seemann, Markus Stäuble, Veronika Thurner, Dalibor Topic, Bernd Trops, Clemens Utschig-Utschig, Martin Wassermann, Matthias Weißendorf, Torsten Winterberg, Stefan Zörner

Anzeigenverkauf:
Software & Support Verlag GmbH
Patrik Baumann
Tel. +49 (0) 69 63008 90
Fax. +49 (0) 69 630089 89
pbaumann@javamagazin.de

Es gilt die Anzeigenpreisliste Nummer 12

Pressevertrieb:
DPV Network
Tel.+49 (0) 40 378456261,
www.dpv-network.de

Druck: PVA Landau
ISSN: 1619-795X

Abo-Service:
Software & Support Verlag GmbH
Tel. +49 (0) 69 6300890
Fax +49 (0) 69 63008989
www.javamagazin.de/service/

Abonnementpreise der Zeitschrift:

Inland:	12 Ausgaben	€ 79,-
Europ. Ausland:	12 Ausgaben	€ 89,-
Studentenpreis (Inland)	12 Ausgaben	€ 69,-
Studentenpreis (Ausland):	12 Ausgaben	€ 79,-

Einzelverkaufspreis:

Deutschland:	€ 7,50
Österreich:	€ 8,60
Schweiz:	sFr 15,80

Erscheinungsweise: monatlich

© Software & Support Verlag GmbH

Alle Rechte, auch für Übersetzungen, sind vorbehalten. Reproduktionen jeglicher Art (Fotokopie, Nachdruck, Mikrofilm oder Erfassung auf elektronischen Datenträgern) nur mit schriftlicher Genehmigung des Verlages. Jegliche Software auf der Begleit-CD zum *Java Magazin* unterliegt den Bestimmungen des jeweiligen Herstellers. Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden. Honorierter Artikel gehen in das Verfügungsrecht des Verlages über. Mit der Übergabe der Manuskripte und Abbildungen an den Verlag erteilt der Verfasser dem Herausgeber das Exklusivitätsrecht zur Veröffentlichung. Für unverlangt eingeschickte Manuskripte, Fotos und Abbildungen keine Gewähr. Java™ ist ein eingetragenes Warenzeichen der Sun Microsystems Inc.

werden. Jedes Teammitglied verfügt dazu über einen persönlichen Repository Workspace. Lokal weiterentwickelte Artefakte lädt ein Teammitglied in seinen persönlichen Arbeitsbereich des Repositories. Für das restliche Team werden diese Änderungen erst sichtbar, wenn der Bearbeiter sie explizit freigibt. So ist es möglich, auch Zwischenstände zentral zu sichern, ohne dabei die Integrität des aktuellen Projektstands zu gefährden. Zu jeder Projektdatei eines Teams wird in einem *Stream* eine Masterkopie gespeichert. Alle persönlichen Repository-Arbeitsbereiche enthalten lediglich jeweils eine Kopie dieser Masterdatei aus dem Stream. Eigene Änderungen können aus dem persönlichen Arbeitsbereich an die Masterkopie übergeben werden. Analog kann ein Teammitglied die Änderungen, die andere Teammitglieder an der Masterkopie durchgeführt haben, akzeptieren, um sie in seinen eigenen persönlichen Arbeitsbereich zu übernehmen und damit seine eigene Kopie der Entwicklungsartefakte wieder auf den aktuellen Stand zu bringen. Der letzte Beitrag dieser Artikelserie zu Jazz wird die Jazz-spezifischen Mechanismen des Sourcecode-Managements im Detail fokussieren.

Jedes Team kann zur Erstellung des lauffähigen Programms einen team-spezifischen *Build* definieren. Die Build-Definition gibt vor, in welchen Zeitabständen welches Build Script auszuführen ist und aus welchen Arbeitsbereichen des Repositories die benötigten Quelldateien entnommen werden.

Insgesamt verwaltet das Jazz Data Warehouse eine sehr große Menge an Informationen zu einem Projekt. Über entsprechende Queries lässt sich diese Datenbasis auswerten, um bedarfsgemäße Sichten darauf zu erzeugen. Das Ergebnis einer solchen Query wird in Form eines BIRT *Reports* erstellt. In Jazz sind bereits einige Report-Typen neben der zugehörigen Queries vordefiniert. Diese können bei Bedarf mittels BIRT um eigene Definitionen erweitert werden.

Zusammenfassung

Jazz bietet eine Plattform mit Konzepten, Repositories und Basisdiensten, die die koordinierte Zusammenarbeit aller

an einem Software- bzw. Systementwicklungsprojekt beteiligten Personen effektiv unterstützen. Abbildung 5 visualisiert die von der Jazz-Plattform bereitgestellten Funktionsbereiche.

Basierend auf dem hier vermittelten Überblick über die Architektur und einer Einführung in die grundlegenden Konzepte von Jazz, erläutert der nächste Beitrag dieser Artikelserie den methodischen Einsatz von Jazz in einem konkreten Projekt. Abschließend fokussiert der dritte Beitrag die Konzepte und Mechanismen, die Jazz für das Sourcecode-Management bereitstellt. ■



Michael Müller und **Martin Wassermann**

arbeiten als Berater bei der ARS Computer und Consulting GmbH in München im Umfeld von Enterprise Java, Web Services und Softwarequalität. **Prof. Dr. Veronika Thurner** forscht und lehrt an der Hochschule München in den Themenbereichen Software Engineering und Geschäftsprozesse.



Links & Literatur

- [1] Erich Gamma, Interview on the Jazz Project: www.theserverside.com/tt/talks/videos/ErichGammaText/interview.tss, März 2007
- [2] Barnett, Liz: „The IBM Rational Jazz Strategy for Collaborative Application Lifecycle Management“, EZ Insight, Inc., Juli 2008
- [3] IBM Rational, Jazz Overview: <http://www-01.ibm.com/software/rational/jazz/>, April 2009
- [4] Open Services for Lifecycle Collaboration (OSLC) Wiki: open-services.net, April 2009
- [5] Jazz Community Site: www.jazz.net, April 2009