

Issue Management für agile Projekte mit JIRA

Down under behält den Überblick

■ VON RAPHAEL DRÜNER UND ARNO SCHUMACHER

Die Steuerungen von Prozessen, insbesondere das Management von Entwicklungs-Tasks, Bug Tracking und das Controlling von Test Cases, sind wichtige Aufgaben in jedem Softwareprojekt. Das Issue- und Bug-Tracking-System JIRA des australischen Software-Produktshauses Atlassian verspricht hier Unterstützung, die weit über den Umfang anderer Bug-Tracking-Systeme hinausgeht.

Die Auswahl und die Verwendung geeigneter Tool-Sets kann wesentlich zum Erreichen des Projektzieles beitragen: Compiler, IDEs, Source Control Systems

und Bug Tracker sind wichtig für die Zusammenarbeit in einem Team. In ihrer täglichen Arbeit setzen die Autoren agile und leichtgewichtige Prozesse für die Softwareentwicklung ein. Eine Reihe von Werkzeugen dient dabei der Organisation der zu erledigenden Aufgaben:

tures im Weiteren näher beleuchten. Die Ursprünge von JIRA liegen im Bereich Bug Tracking. Die verwendeten Konzepte und Prinzipien haben sich aber als so elegant und flexibel erwiesen, dass mit JIRA auch weitere Workflow-lastige Prozesse abgebildet werden können. Mit der einfachen und intuitiven Browser-basierten Oberfläche können zusätzliche Workflows für neue Prozesse leicht angelegt sowie der mitgelieferte Standard-Workflow auf die projektspezifischen Bedürfnisse angepasst werden.

Steckbrief

JIRA 3.4.2

Hersteller: Atlassian



Kurzbeschreibung: Issue-Tracking- und Projektmanagement-Software

Plattformen: Java EE (u.a. Windows, Linux, Solaris, AIX)

Unterstützte Browser: IE 6, Firefox, Opera, Safari, Konqueror

Application Server: gebundelter Tomcat 5.5 (empfohlen), ansonsten u.a. Orion, Resin, JBoss, Jetty, Oracle OC4J, WebLogic, WebSphere

Datenbank: Oracle, DB2, MySQL, Firebird, SQL Server 2000, Mckoi, MaxDB, PostgreSQL, Sybase, HSQL (Standalone-Version)

Preis: Standard-Edition US-\$ 1.200, Professional US-\$ 2.400, Enterprise US-\$ 4.800. Testversion verfügbar. Frei für Non-Profit-Organisationen und Open-Source-Projekte.

URL: www.atlassian.com/software/jira

- ein Wiki-System – etwa MoinMoin [1] oder Confluence [2] – als strukturiertes Dokumentationsmedium für Requirements, User Stories, Testfälle, Knowledge Base und ergänzende Dokumentation,
- JIRA [3] als Werkzeug zur Steuerung von Teilprozessen, insbesondere das Management von im Wiki dokumentierten Entwicklungs-Tasks, Bug Tracking und das Controlling von Test Cases, und
- Eclipse [4] als Entwicklungsplattform, aus der mithilfe von Plug-ins direkt Task- und Defektinformationen in Jira gepflegt werden können.

Hier wollen wir uns auf JIRA konzentrieren und dieses Werkzeug und seine Fea-

Issues und Vorgänge

Das effiziente Management von Vorgängen – Issues – ist wichtig für jedes Projekt. Als festen Bestandteil bietet JIRA Unterstützung für die folgenden Vorgangsarten:

- Bugs/Defekte (*Bug*)
- Verbesserungsvorschläge (*Improvement*)
- Aufgaben (*Task*)
- Neue Anforderungen/Eigenschaften (*New Feature*)

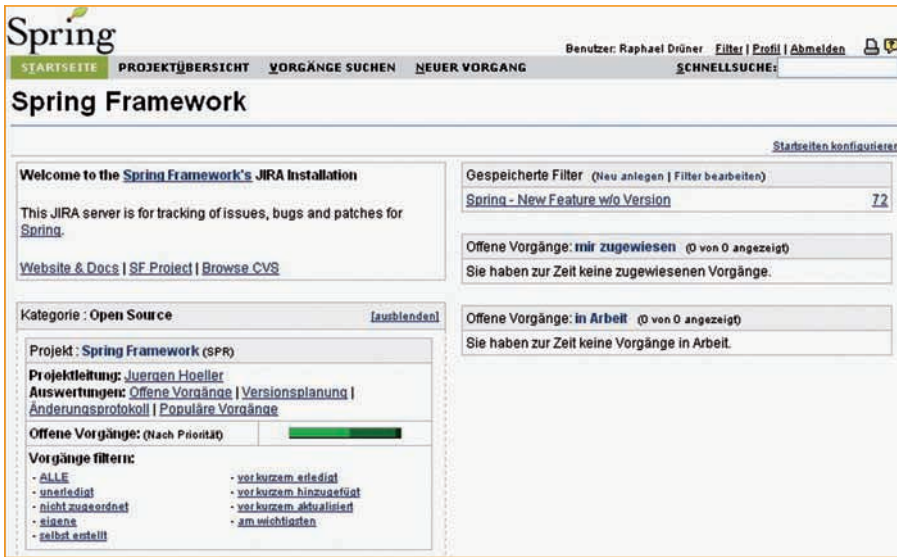


Abb. 1: Dashboard der JIRA-Installation des Spring-Projekts

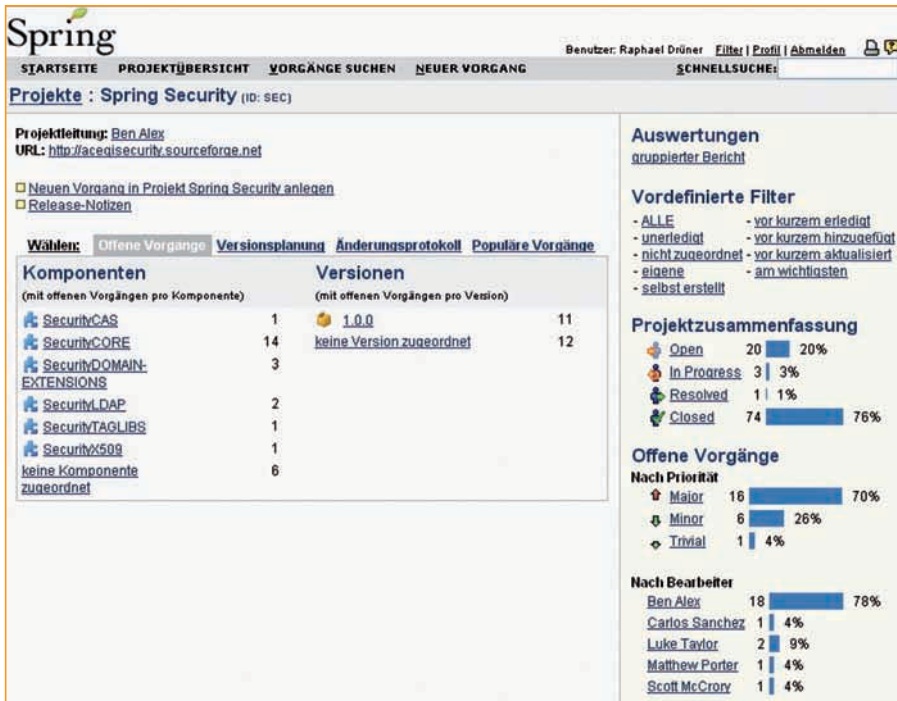


Abb. 2: Projektübersichtsseite von Spring Security

Die Einstellung eines neuen Vorgangs und die Zuweisung an den ersten Bearbeiter erfordern nur geringe Vorarbeiten. Dazu sind außer dem initialen Einrichten der User und eines Projekts keine weiteren Schritte erforderlich. Genügen die Voreinstellungen nicht, so können nahezu alle Elemente an komplexere Anforderungen angepasst werden. Hier zählt sich das von JIRA durchgängig verfolgte Konzept aus: maximale Anpassbarkeit

an spezielle Bedürfnisse bei gleichzeitiger Berücksichtigung eines möglichst einfachen und trotzdem für viele Anwendungsfälle ausreichenden Standardverhaltens.

In größeren Projekten kann die Anzahl der Bugs, Change Requests oder Erweiterungen schnell ein unübersichtliches Ausmaß erreichen. JIRA erlaubt auch bei großen Datenbeständen das schnelle und präzise Auffinden eines Vorgangs, indem

neben der reinen Volltextsuche die Treffermenge über Filter sinnvoll eingegrenzt wird. Die Filterdefinitionen lassen sich abspeichern und später wieder verwenden. Filter können global oder nur für eine bestimmte Benutzergruppe freigeschaltet werden. Mit so genannten E-Mail-Berichten können die Ergebnisse eines Filters automatisch einer Benutzergruppe in regelmäßigen Abständen zugeschickt werden.

Um einen Vorgang später wieder zu finden, lässt sich dieser beobachten, d.h., dass bei jeder Änderung des Vorgangs eine Mail an alle Beobachter mit Informationen zu der Statusveränderung geschickt wird. Ist der gefundene Bug für einen selbst kritisch, so kann man ihm in einer Art Abstimmung eine Stimme geben. Dadurch können die Issues von dem Projektteam selbst in Relation zueinander gebracht werden, um dem Bearbeiter oder Projektleiter Hinweise für die Priorisierung zu geben.

Neben der Suchfunktion ist die Ansicht von Vorgängen für eine schnelle Orientierung wichtig. Es nützt nichts, wenn man zwar den Titel schnell gefunden hat, die folgende Detailansicht aber die eigentlich gewünschte Information standardmäßig nicht enthält und erst umständlich in Untermenüs gesucht werden muss. Sinnvoll ist hier, dass JIRA es erlaubt, die Ansichtsmasken zu personalisieren, sodass alle für einen Benutzerkreis relevanten Informationen klar ersichtlich sind.

Dashboard und Projekte

Die JIRA-Startseite (Dashboard) dient als personalisiertes Menü zu den häufigsten Funktionen (Abb. 1). Jeder Benutzer kann sie individuell konfigurieren und vom Dashboard aus direkt eine bestimmte Projektauswertung ausführen oder etwa zu einem aktuell zugewiesenen Vorgang verzweigen. Besonders nützlich ist die Zahl neben den auf dem Dashboard eingblendeten gespeicherten Filter: Sie zeigt sofort an, wie viele Vorgänge die Ausführung dieses Filters zurückliefern wird. So können auf den ersten Blick relevante Änderungen erkannt werden. Jeder neu angelegte Benutzer startet zunächst mit dem „Default Dashboard“, welches auch an unternehmens- oder teamspezi-

fische Gegebenheiten angepasst werden kann. Die einzelnen Informationsblöcke auf der Startseite werden Portlets genannt. Neben den 16 mitgelieferten Portlets können für weitergehende Anforderungen auch eigene Portlets in Java geschrieben und auf dem Dashboard platziert werden.

Wählt man auf dem Dashboard eines der Projekte aus, so gelangt man zu einer Zusammenfassung der wichtigsten Kenngrößen dieses Projekts (Abb. 2). Der Status der Vorgänge wird grafisch hervorgehoben und kann schnell überblickt werden. Detaillierte Auswertungen können von dieser Maske aus angestoßen werden. Sind die Vorgänge bestimmten Versionen zugeordnet worden, so kann in Echtzeit die aktuelle Versionsplanung (Roadmap) oder ein Änderungsprotokoll (Change Log) eingesehen werden. Interessant ist der Reiter POPULÄRE VORGÄNGE: Einen entsprechend großen Anwenderkreis vorausgesetzt, kann über diese Informationsquelle gut abgeschätzt werden, wie wichtig die einzelnen Vorgänge

den Benutzern sind. Diese strukturierte Feedback-Schleife kann wertvolle Hinweise für die weitere Release-Planung geben.

Projekte können zeitlich in Versionen und strukturell in Projektkategorien und Projektkomponenten geordnet werden. Die Unterteilung dient nicht nur der besseren Übersicht, denn JIRA nutzt z.B. das Meta-Wissen über die Projektversionen zur automatischen Erzeugung von Release Notes.

Kommunikation und Sicherheit

Über die reine Verwaltung von Bugs hinaus enthält JIRA viele Funktionen, die die Kommunikation in verteilten Teams erleichtern. Interne Ereignisse eines Issue-Management-Tools, wie die Erzeugung oder Veränderung von Vorgängen, sollten nicht intern bleiben, sondern an die richtigen Adressaten kommuniziert werden. So ist es selbstverständlich, dass JIRA per E-Mail auf neue Issues hinweist. Ungewöhnlicher ist hingegen die Flexibilität von JIRA: Selbst die

Kommentierung eines Vorgangs kann über Gruppen- und Rollenmanagement zu genau definierbaren Benachrichtigungen führen. Die Konfiguration selbst muss nicht für jedes Projekt neu erstellt werden, sondern kann in Form eines Notification Scheme wieder verwendet werden. Bei so viel Mitteilungsbedarf stellt sich die Frage der Sicherheit, denn

JIRA ist ein flexibles und erweiterbares System rund um das Management von Issues und gleichzeitig ein einfach zu bedienendes Werkzeug, das sich schnell in der täglichen Projektarbeit bewährt.

interne Anwendungen, die wie JIRA über Abteilungs- und Hierarchieebenen hinweg eingesetzt werden, sind in dieser Hinsicht besonders heikel. Bestimmte Informationen und Aktionen sollen nicht allgemein zugänglich sein und müssen entsprechend geschützt werden. JIRA bietet dafür ein fein gra-

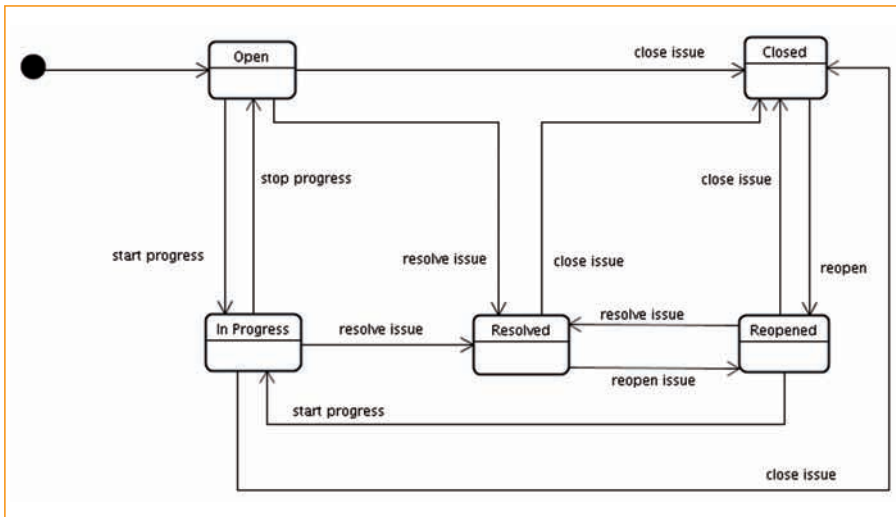


Abb. 3: JIRA-Standard-Workflow

nulares Berechtigungskonzept, mit dem auf verschiedenen Ebenen (global, pro Projekt, pro Vorgang) bestimmt werden kann, wer was tun und sehen darf. Einzelne Rechte werden dazu entweder einer Benutzergruppe, einer Rolle oder auch direkt einem individuellen User zugeordnet. Über ein Permission Scheme

wird die Wiederverwendbarkeit der Konfigurationsarbeit gewährleistet.

Workflow und Konfiguration

Der Standard-Workflow von JIRA deckt die typischen Fälle der Issue-Bearbeitung ab und reicht für kleinere Projekte oft aus. Dennoch werden im Laufe der

Arbeit mit jedem Issue-Tracking-System sicherlich Anforderungen entstehen, die nicht vom Hersteller vorausgesehen werden (können). Dafür bietet JIRA ab der Professional-Version an, über einen web-basierten Editor den mitgelieferten Standard-Workflow anzupassen bzw. neue Workflows anzulegen (Enterprise). So können ein neuer Status hinzugefügt und Regeln hinterlegt werden. Benötigt man bei einem Übergang neue Eingabedaten, die nicht von JIRA vorgesehen werden, so können diese mitsamt passender Eingabemasken direkt in JIRA angelegt werden. Neben neuen Statusdefinitionen können auch die vorgegebenen Prioritäten erweitert oder auch komplett neue Vorgangstypen mit eigenen Masken eingerichtet werden.

Die Erweiterbarkeit und Konfigurierbarkeit von JIRA sind sicherlich zwei seiner großen Vorteile gegenüber anderen Lösungen. So können neben den bereits angesprochenen Konfigurationsmöglichkeiten über die Weboberfläche selbst auch mittels programmatischer Erweiterungen neue

JIRA-Technologie

JIRA ist eine Java-Webanwendung, die auf einem Java EE Application Server oder einer Servlet Engine deployt wird. Diese müssen als Minimalanforderung den J2EE 1.3-Standard bzw. die Servlet-Spezifikation 2.3 implementieren. Offiziell werden von Atlassian unter anderem JBoss 4.0, WebLogic 7 und 8.1, Tomcat 5.0 und 5.5 unterstützt. Neben einer Distribution als WAR-Archiv kann JIRA auch als Standalone-Anwendung mit integriertem Tomcat 5.5 und einer HSQL-Datenbank bezogen werden. Mit dieser lässt sich JIRA innerhalb weniger Minuten installieren. Jedoch empfiehlt es sich, für produktiv genutzte Systeme die vorkonfigurierte HSQL-Datenbank gegen ein externes relationales Datenbank-System, etwa MySQL, PostgreSQL oder Oracle, auszutauschen. Die modulare und flexible Architektur erlaubt es, JIRA auf einfache Art und Weise durch Konfiguration anzupassen. Darüber hinaus gehende Anforderungen lassen sich durch eigene Plug-ins realisieren. Innerhalb von JIRA wird PicoContainer [5] von Codehaus zum Management von Modulen, Komponenten und Plug-ins verwendet. Ein Plug-in besteht aus einem XML Deployment Descriptor und Java-Klassen, welche in einem JAR-Archiv gebündelt werden. Stellt ein Plug-in zudem noch Erweiterung der Benutzeroberfläche bereit, dann finden sich auch Velocity Templates

innerhalb des Archivs. Ein Plug-in wird deployt, indem das JAR-Archiv in den *WEB-INF/lib*-Folder der JIRA-Webapplikation abgelegt wird. In der JIRA-Administrationsoberfläche besteht die Möglichkeit, Plug-ins zu aktivieren, deaktivieren oder zu konfigurieren. Unter anderem werden von JIRA folgende Plug-in Typen unterstützt:

- *Component Plug-in*: eine beliebige POJO-Komponente, welche im PicoContainer registriert werden kann und anderen Plug-ins Services zur Verfügung stellt.
- *Custom Field Plug-in*: Sollte die Funktionalität der ausgelieferten Feldtypen oder Felder zur Bearbeitung von Issues nicht ausreichen, so können mit einem Custom Field Plug-in neue Felder bereitgestellt werden, die in beliebigen Issue Screens benutzt werden können.
- *Workflow Plug-in*: realisieren Validators, Functions und Conditions zur Erweiterung der JIRA Workflow Engine.
- *Portlet Plug-in*: deployt neue Report-Typen.

OSWorkflow [6] von OpenSymphony kommt als Workflow Engine zum Tragen. Workflow Steps, Transitions, Validators, Functions und Conditions können mithilfe des JIRA Workflow-Editors konfiguriert oder direkt in XML editiert werden.

Weiterhin lassen sich eigene Issue Listener bereitstellen. Bei einem Übergang eines Issue in einen neuen Zustand werden die registrierte *Issue Listener* benachrichtigt. Beispielsweise könnte beim Übergang eines Issue in den Zustand *resolved* ein Issue Listener über die JMX-Schnittstelle einen neuen Build auf einem CruiseControl-System [7] anstoßen. JIRA verwendet Quartz [8] als Scheduler für asynchrone Jobs. Das regelmäßige Backup der JIRA-Daten wird durch einen mitgelieferten Quartz-Job gestartet. Auch eigene asynchrone Services lassen sich in JIRA deployen. Über ein *ComponentManager* Singleton können diese Services auf die PicoContainer Registry zugreifen und damit alle JIRA-Komponenten und Module ansprechen. Als Persistenz-Framework wird OfBiz [9] verwendet. Das von JIRA eingesetzte Schema ist sehr generisch, sodass nur selten Änderung am DB Schema zur Bereitstellung von Erweiterungen vorzunehmen sind. Die Basis für die JIRA-Benutzeroberfläche ist Velocity und WebWork [10]. Atlassian liefert den Sourcecode bei Erwerb einer JIRA-Lizenz mit aus. Für eigene Erweiterungen steht ein Fundus an Beispielen bereit [11].

Funktionalitäten hinzugefügt werden. Dafür stehen sowohl Java-APIs als auch eine Web-Service-Schnittstelle zur Verfügung.

Issue Management vs. Projektmanagement

Die Entwicklung von JIRA scheint zunehmend in Richtung eines umfassenden Projektmanagement Tools rund um die Issue-Bearbeitung zu gehen. So hat JIRA inzwischen auch ein Zeitverfolgungs-Feature (Time Tracking), das zwar (noch) nicht ein separates Zeiterfassungs-Tool ersetzen kann, mit dem man aber zu jedem Vorgang einen Planaufwand und tatsächliche Ist-Zeiten erfassen kann. Bei jeder Zeiteingabe wird man automatisch gefragt, ob der Gesamtaufwand angepasst werden muss, um die Aufwandsdaten konsistent zu halten. JIRA verfügt zur Auswertung dieser Zahlen bereits über einige Reports.

Allgemein sind die Daten, die sich während eines Projekts in JIRA ansammeln, in besonderem Maß für die Projektleitung von Interesse. Das gilt vor allem für agil geführte Projekte, bei welchen ein möglichst zeitnahes Feedback zum Fertigstellungsgrad notwendig ist, um den Fortschritt der Iterationen festzustellen und dem Auftraggeber mitteilen zu können. Neben den eingebauten Auswertungen von JIRA können dank des flexiblen Plug-in-APIs eigene Reports hinzugefügt

werden. In der JIRA-Community gibt es bereits nützliche Erweiterungen. Ist die direkte Integration der Auswertung in die JIRA-Oberfläche nicht unbedingt nötig, so lassen sich die Rohdaten nach Excel exportieren, um dieses Tool für Reports verwenden zu können.

JIRA-Editionen

Atlassian bietet JIRA in drei unterschiedlichen Versionen an: Standard, Professional und Enterprise. In der Standard- und Professional-Version sind die Möglichkeiten, JIRA auf projektspezifische Bedürfnisse anzupassen, eingeschränkt. In der Standard-Version ist etwa der Workflow fest verdrahtet, während sich in der Professional-Version nur ein globaler Workflow für alle Vorgangs-Typen definieren lässt. Diese Einschränkungen entfallen bei der Enterprise-Version. Ähnliche Restriktionen bezüglich der Konfigurationsmöglichkeit gibt es für die Standard- und Professional-Version auch für Informationen und Attribute, die innerhalb eines Workflow gepflegt werden können. Die Kosten für eine Lizenz sind moderat und liegen zwischen US-Dollar 1.200 und 4.800.

JIRA selber ausprobieren

Atlassian stellt eine zeitlich limitierte Testversion von JIRA zum Ausprobieren

bereit. Zudem bieten die Installationen von Open-Source-Organisationen eine Möglichkeit, JIRA zu testen. Atlassian stellt JIRA Open-Source-Projekten und Non-Profit-Organisationen ohne Lizenzkosten zur Verfügung. Namhafte Open-Source-Projekte setzen deshalb auf JIRA als ihr Issue-Tracking- und Management-System.

Fazit

JIRA überzeugt auf zwei Ebenen: Es ist ein hoch flexibles und erweiterbares System rund um das Management von Issues und gleichzeitig ein einfach zu bedienendes Werkzeug, das sich schnell in der täglichen Projektarbeit bewährt. In agilen Projekten sind diese Eigenschaften besonders wichtig, da eine zeitnahe Produktivität der eingesetzten Tools ebenso wie ihre langfristige Perspektive eine entscheidende Rolle spielt.

Die häufige Interaktion mit dem Kunden führt zwangsläufig zu Anforderungsänderungen während der Projektlaufzeit, die verwaltet werden müssen. JIRA unterstützt dabei das ganze Team, in der Fülle der zu lösenden Probleme den Überblick zu bewahren, Dinge zu priorisieren und Issues bis zur vollständigen Lösung nicht aus den Augen zu verlieren.



Raphael Drüner und Arno Schumacher sind Softwareberater bei Avono, einem Software-Projekthaus mit dem Schwerpunkt unternehmenskritische Anwendungen auf Java-Basis. Kontakt: raphael.druener@avono.de, arno.schumacher@avono.de.

JIRA-Standard-Workflow

Innerhalb von JIRA befindet sich jeder Defekt oder Vorgang zu einem Zeitpunkt in einem Verarbeitungszustand. Wird ein neuer Defekt erfasst, dann ist dieser zunächst im Zustand *open*. Im Laufe der weiteren Verarbeitung des Vorgangs wird der Defekt über den Zustand *resolved* – der Defekt wurde korrigiert – in den Zustand *closed* überführt. Der Zustand *closed* wird dann erreicht, wenn etwa der Erfasser des Defektes die vorgenommenen Korrekturen erfolgreich nachgetestet hat.

Innerhalb des Lebenszyklus eines Vorgangs kann dieser eine Reihe weiterer Zustandsübergänge durchschreiten. In Abbildung 3 sind die Zustände eines Vorgangs und ihre Übergänge schematisch dargestellt.

- **open:** Initial ist jeder Vorgang im Status *open*.
- **in progress:** Wenn der Vorgang einem Bearbeiter oder Entwickler zugewiesen wurde und

sich dieser dem Vorgang widmet, dann wechselt der Vorgang *in progress*.

- **resolved:** Der Vorgang wird vom Bearbeiter in den Zustand *resolved* überführt, wenn eine Lösung für den Defekt oder den Vorgang gefunden und umgesetzt wurde.
- **reopen:** In der Regel wird die vom Bearbeiter des Defektes gefundene Lösung nochmals von einer weiteren Person, etwa vom Erfasser des Defektes, nachgetestet. Treten bei dieser Nachprüfung Probleme auf oder ist der Erfasser mit der gefundenen Lösung nicht zufrieden, so kann dieser den Vorgang *reopen* setzen und damit eine erneute Bearbeitung anstoßen.
- **closed:** Wenn die gefundene Lösung akzeptabel ist, dann wird der Vorgang mit *closed* markiert und damit abgeschlossen.

Links & Literatur

- [1] MoinMoin: moinmoin.wikiwikiweb.de
- [2] Confluence: atlassian.com/software/confluence
- [3] JIRA: atlassian.com/software/jira
- [4] Eclipse: www.eclipse.org
- [5] PicoContainer: www.picocontainer.org
- [6] OSWorkflow: www.opensymphony.com/osworkflow/
- [7] CruiseControl: cruisecontrol.sourceforge.net
- [8] Quartz: www.opensymphony.com/quartz/
- [9] Entity Engine: www.ofbiz.org/docs/entityconfig.html
- [10] WebWork: www.opensymphony.com/webwork/
- [11] JIRA Developer Site: confluence.atlassian.com/display/JIRA/JIRA+Development+Hub
- [12] The Agile Alliance: www.agilealliance.org