# Protokoll der Arbeitspakete

## Dokumentation

|  |  |
| --- | --- |
| Datum | Beschreibung |
| 27.02.2012 | Start der Dokumentation |
| 06.03.2012 | Überarbeitung der Dokumentation |
| 11.04.2012 | Update Dokumentation |
|  |  |

## Planung

|  |  |
| --- | --- |
| Datum | Beschreibung |
| 27.02.2012 | Literaturrecherche und Strukturierung der Arbeit |
| 29.02.2012 | Anfrage bei Prof. Blakowski als Betreuer |
| 16.04.2012 | Detailplanung Masterarbeit, Überarbeitung der Planung im Exposé, Wochenplanung |
| 17.04.2012 | Überarbeitung Forschungsfrage: Die Gliederung der Masterarbeit ist noch nicht zufriedenstellend und muss überarbeitet werden. Der schlüssige Aufbau der Forschungsfrage sowie der sich hieraus ergebenden Unterfragen muss überarbeitet werden und in die Gliederung einfließen. |
| 07.05.2012 | Arbeitstitel kann so nicht stehen bleiben, da es derzeit nicht abzusehen ist, dass es ein reales Projekt geben wird, bei dem CD implementiert werden kann. |
| 22.05.2012 | Überarbeitung Projektplanung, Anpassung der Zeitplanung an aktuelle Gegebenheiten (Zeitbedarf bei der Evaluierung von GO höher als erwartet) |
| 08.06.2012 | Anmeldungsbogen Master-Thesis ausgefüllt |

## Meeting

|  |  |
| --- | --- |
| Datum | Beschreibung |
| 27.02.2012 | Pilottreffen mit Eberhard: Bedeutung von CD und Zielvorgabe für die MA |
| 29.02.2012 | Telko mit Eberhard: Exposé ist okay |
| 09.03.2012 | Telko Prof. Blakowski: Inhaltlicher Aufbau des Exposé ist okay, allerdings sollte der Aufbau der Arbeit noch einmal überprüft werden. |
| 19.03.2012 | Pilottreffen mit Prof. Blakowski: Formale Rahmenbedingungen der Masterarbeit, Vorgehensempfehlung für die Masterarbeit. |
| 11.04.2012 | Pilottreffen mit Dieter: Erläuterung des Themas und Besprechung der nächsten Schritte. |
| 25.04.2012 | Meeting (Dieter, André):  Aktueller Stand der Evaluierung von GO:   * Build ist mit Ant möglich und wird von der Pipeline angestoßen * Alle Jobs laufen über Ant (u.a.) bzw. über Shell-Skripte und –Kommandos. * Test-Stage fehlt noch * Deployment in Server fehlt noch   Deployment-Pipeline in GO sollte folgende Komponenten abdecken können:   * Build und Unit-Test (Unit-Test fehlt noch) * Test-Stage: Deployment und Akzeptance-Test (Deployment über Catalina-Ant.jar, Test mit JMeter) * Produktiv-Umgebung: Manueller angestoßenes Deployment wie im SB-Laden, der Kunde kann so z.B. sich ein Build selber in seine Test-Umgebung deployen und es abnehmen und es danach in seine Produktiv-Umgebung deployen. |
| 21.05.2012 | Gespräch mit Dieter planen |
| 23.05.2012 | Meeting (Dieter, André)  Entwicklung seit letzten Treffen:   * Deployment-Pipeline GO abgeschlossen * SB-Bereich fehlt noch, da kein LDAP für Benutzerverwaltung installiert wurde * Performance: 46h Server-Deployment, 8,5h Infrastruktur, 24h JMeter-Integration, 9h, JUnit-Integration, 24h Dokumentation, 14h JWebUnit   Planung:   * Evaluation Deployinator: 25.05. – 14.06. * Evaluation Dreadnot: 15.06. – 03.07. * Entwicklungsstand adesso: 04.07. – 09.08. * Ausformulierung: 10.08. – 20.09.   Anmeldung Master-Thesis:   * Dieter würde Rolle des Zweitgutachters übernehmen, vorausgesetzt Prof. Blakowski lässt ihn zu. |
| 11.06.2012 | Treffen mit Prof. Blakowski bezüglich der Anmeldung der Master-Thesis.  Folgende Punkte haben wir besprochen:   * Wissenschaftliche Tiefe der Master-Thesis beachten, wenn Produkte auf triviale Weise verglichen werden. * Nicht nur die eine quantitative Betrachtung, sondern die qualitative Betrachtung wird gefordert. * Wie würden sich die Produkte in die bestehende Prozess- und Infrastrukturlandschaft eingliedern? * Wie aufwändig ist die Einarbeitung in die Werkzeuge? * Wie aufwändig ist die Erstellung einer Pipeline? * Kritische Auseinandersetzung durch eine Kosten- / Nutzenübersicht betrachten. * Technologischer Hintergrund und Entwicklung von CD. |
| 12.06.2012 | Meeting (Dieter, André)   * Aktueller Stand der Entwicklung bei Deployinator * Probleme bei der Installation von Deployinator (Pakete, Abhängigkeiten) * Zeitproblem durch Nebenprojekte * Information über das Gespräch mit Prof. Blakowski * Qualitative Betrachtung der Werkzeuge   + Aufwand für die Erstellung einer Pipeline   + Mgl. Projektkosten vergleichen zu manuellen Ausliefern (Testaufwand)   + Annahme, dass ein Produkt eingeführt werden soll, wie würde sich dieses in die bestehenden Prozesse und Infrastruktur eingliedern lassen. |

## Exposé

|  |  |
| --- | --- |
| Datum | Beschreibung |
| 20.02.2012 | Einleitung, Fragestellung, Methodik und Vorgehensweise, Arbeits- und Zeitplanung, Gliederung erstellt. |
| 23.02.2012 | Leichte Überarbeitung |
| 27.02.2012 | Feinschliff |
| 28.02.2012 | Feinschliff |
| 18.04.2012 | Forschungsfrage und Gliederung überarbeitet |

## Werkzeuge

|  |  |
| --- | --- |
| Datum | Beschreibung |
| 23.02.2012 | Toolliste nach „DevOps in the Cloud” und Schematischer Ablauf einer Deployment Pipeline |
| 28.02.2012 | Ermittlung von Evaluierungskriterien: Funktionalitäten für Continuous Delivery (Building, Deployment, Testing, Reporting, Infrastruktur, Workflow, Versionsverwaltung / Repository, Reporting) Grobbetrachtung von Tools: Jenkins Build Pipeline Plugin für einen Deployment-Workflow, ThoughtWorks Go, Etsy Deployinator, Racker Dreadnot |
| 29.02.2012 | Qualitätsbeurteilung von Software nach Gernot Starke: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit, Übertragbarkeit. Szenarien: Anwender, Änderungen, Stress- und Grenzszenarien  Grobbetrachtung der verfügbaren Tools |
| 01.03.2012 | Evaluierungskriterien nach Module einer Deployment Pipeline: Versionskontrolle, Artefakte-Repository, Acceptance Stage, Self-Service für Tester, Kapazitätstests, Auslieferung in die Produktivumgebung, Report |
| 02.02.2012 | Evaluierungskriterien nach Paul M. Duvall (DZone Refcardz): Generelle Unterteilung nach Visibility, Feedback, Continuous Deploy. Einteilung nach Pattern: Konfigurationsmanagement, Continuous Integration, Testing, Deployment Pipeline, Build- und Deployment Scripting, Deployment und Application-Release, Infrastruktur und Environment, Daten, Inkrementelle Entwicklung, Zusammenarbeit |
| 05.03.2012 | Installation VM für Jenkins und SVN: je eine System mit 128 MB RAM und 8 GB HDD, OpenSuse 11 (wurden am 10.04. wieder verworfen) |
| 06.03.2012 | Konfiguration SVN System: Firewall, Putty und SSH Zugriff |
| 04.04.2012 | Versuch der Installation von Thoughtworks GO auf Amazon EC2. Preiswerte Microinstanzen der EC2 bieten leider nicht genügend Kapazitäten. Der Go-Server benötigt mindestens 1 GB RAM, zu Verfügung stehen hier aber nur 660 MB. |
| 10.04.2012 | Einrichtung und Installation von drei VMs für die Installation von GO. Einrichtung des GO-Server, Installation eines SVN-Server, der GO-Client fehlt aber noch. Der GO-Server und die Web-Konsole lassen sich von außen ansprechen. |
| 13.04.2012 | Ziele: Aufbau einer Testdeployment Pipeline mit GO  Konkret erreicht: Konfiguration des SVN-Systems mit Nutzerrollen abgeschlossen, Test-Projekt in Eclipse mit einfacher JSP aufgesetzt, Einarbeitung in ANT, erstes Build-Script in ANT. |
| 23.04.2012 | Go Build-Stage vorbereitet, Installation von Ant auf dem Agent  Go sucht im SVN nach neuer Revision und lädt den aktuellen Stand, die aktuelle Revision wird als „Material“ an den Go-Agent übergeben, dieser führt das Build-Skript aus und übergibt das gebaute War-File an den Go-Server, welches über Artefakte aufgerufen werden kann. |
| 24.04.2012 | Go Test-Stage vorbereitet, Tomcat-Server für Deployment installiert, Recherche: Deployment mit Ant auf Tomcat-Server, Ant-Skript für Deployment, notwendige Bibliothken auf Server installiert |
| 26.04.2012 | Wie kann über Ant auf einen Tomcat-Server deployt werden?   * http PUT Upload auf <http://ip:8080/manager/deploy> |
| 27.04.2012 | Probleme mit dem Betrieb von Go auf virtuellen Systemen aufgetaucht: mögliche Ursache ist das Preset von PermGem im Server: Link zum Performance-Tuning in Go: <http://www.thoughtworks-studios.com/docs/go/12.2/help/performance_tuning.html> |
| 30.04.2012 | Problem beim Deployment auf das war-File zuzugreifen -> FileNotFoundException  Zu exportierendes WAR-File: <http://192.168.56.101:8153/go/files/Test_Pipe/24/Compile_Stage/1/CreateWarFile/warFile/HelloWorld.war>  Aber Go versucht folgendes Artifakt zu laden: <https://192.168.56.101:8154/go/remoting/files/Test_Pipe/24/Compile_Stage/1/CreateWarFile/warFile/HelloWorld.war>  Versuche das Deployment über ANT zu realisieren, hierfür können Umgebungsvariablen von GO genutzt werden.  Fehlermeldung:  /home/goclient/pipelines/Test\_Pipe/build.xml:55: java.io.IOException: Server returned HTTP response code: 403 for URL: <http://192.168.56.102:8080/manager/deploy?path=%2FHelloWorld&war=deployWar%2FHelloWorld.war>  Anpassung des ANT-Skripts mit vorherigen undeploy der WEB-Anwendung  Fehlermeldung2:  BUILD FAILED  /home/goclient/pipelines/Test\_Pipe/build.xml:60: FAIL - Failed to deploy application at context path /HelloWorld  Hilfe zum Deployen von Web-Anwendungen mit Ant: <http://www.avajava.com/tutorials/lessons/how-do-i-deploy-to-tomcat-using-ant.html>  Versuch mit einem einfachen Ant-Deploy-Task die Anwendung zu deployen.  Hier gibt es scheinbar noch einen Fehler mit dem Deploy-Path. |
| 03.05.2012 | Weiterer Link zur Erklärung des Deploy-Tasks: <http://raibledesigns.com/wiki/Wiki.jsp?page=TomcatAntTasks>  Bug-Fixing und Anpassung des Build-Skripts. Eine Deployment der Anwendung kann nun manuell gestartet werden. Im Ant-Skript muss nun noch festgehalten werden, dass auf die „Test-Umgebung“ deployt wird. |
| 04.05.2012 | Ant-Skript für Deployment auf Test-Server abgeschlossen. Für ein Deployment wird zuerst der Status beim Tomcat eingeholt und in Properties geschrieben. Die Pipeline führt die Aufgaben Stop, Undeploy, Deploy und Start aus.  Nächster Schritt ist die Bereitstellung eines „Acceptance“-Testes. Dieser wird über JMeter ausgeführt. JMeter wird als ANT-Task aufgerufen und der Testplan im SVN verwaltet.  Offen: JUnit Test in der Build-Stage |
| 07.05.2012 | Ziele: JUnit-Test beim Bauen durchführen, Build-Bewertung evtl. Scoring des Build durchführen. (Monitoring für Web-Anwendungen?)  Lizenz-Key für die Evaluierung von Go, vom Go-Support:  SRaSNiYRBwYWS7GsoHXedu6d6WLf76yJyFsv2iuq1dLuRfL9AxaaNn1uH8W2  pS1SK/3BXmxPmZ7SyJr+SAxEN1lib0PhNDr19NC0Gf908o9XBI2BWXX1xB1D  eS7uyXi9yTdQ5lAfVlje7deMiImA3vMg+EJv1rxNcLw28IJrbijYeztB8B0d  jefRF0Tu1L2o8w7FF4WbdzQV3Fj4Y9uPDTw0T995tZSV4FRACs6zTcrZn3R4  nVZSlfHSpbvbeVE1T3aYfifJLGaAY6qng1UE6F8+xd+S9Efu2kPGGzq81j9v  iSAHRMd3aVzMLL+QyGgQJKJDsIX0mpZ03vjFZ9oMhQ==  Was wird für einen JUnit-Test benötigt?   * Test-Klassen müssen kompiliert werden * Junit.jar muss dem Compiler über dem Classpath bekannt sein * Bin-Folder muss beim Test im Classpath gesetzt sein |
| 08.05.2012 |  |
| 09.05.2012 | JMeter-Plug-In für Ant, welches Results loggt und diese in einer index.html, ähnlich der JUnit Testreports erzeugen kann: <http://www.programmerplanet.org/pages/projects/jmeter-ant-task.php> |
| 10.05.2012 | JMeter Integration |
| 11.05.2012 | Visualisierung Deployment Pipeline |
| 14.05.2012 | JWebUnit ermöglicht ein ähnliches Testverfahren wie JMeter. Hier wird jedoch das JUnit Test-Framework genutzt und um die JWebUnit Bibliotheken erweitert. Hierdurch können Akzeptanztest wie JUnit-Test auch programmiert werden.  <http://jwebunit.sourceforge.net/quickstart.html#Creating_a_TestCase> |
| 15.05.2012 | Integration JWebUnit:   * NullPointerException nach Umbau im Build-Script, verändert wurden PATH-Elemente * Alle Pfade wurden repariert: Pfade können vorher in Ant definiert werden: <path id=“abc“> <….> </path> und später innerhalb von <classpath> mit <path refid=“…“ /> genutzt werden. * Integration des JWebUnit Tests als neuen Job in der Test\_Stage Pipeline. Diese ist soweit umgesetzt, als das ein einfacher Test durchlaufen wird und das Ergebnis in der Job-Übersicht dargestellt wird. Vorteile von JWebUnit gegenüber JMeter bei der Integration von Go: Programmatischer Test welcher in wie eine JUnit-Test bearbeitet werden kann, automatisierte Komponenten und Akzeptanz-Test möglich, erzeugen von JUnit-Reports und damit bessere Einbindung in GO möglich. Nachteile: Keine Last-Tests, da nur ein User simuliert werden kann. |
| 16.05.2012 | Screen-Capture Thoughtworks GO  Links für weitere Tools der Qualitätssicherung:  <http://java-source.net/open-source/code-analyzers/jcsc>  <http://findbugs.sourceforge.net/manual/anttask.html#d0e1192>  <http://www.clarkware.com/software/JDepend.html#ant>  <http://pmd.sourceforge.net/pmd-5.0.0/ant-task.html>  <http://java-source.net/open-source/code-analyzers/jcsc> |
| 21.05.2012 | Tex-Dokument für Evaluation GO aufgesetzt |
| 22.05.2012 | Dokumentation Delivery Pipeline mit GO |
| 23.05.2012 | Dokumentation Delivery Pipeline mit GO |
| 24.05.2012 | Dokumentation Delivery Pipeline mit GO |
| 25.05.2012 | Dokumentation Delivery Pipeline mit GO |
| 30.05.2012 | * VM für Deployinator eingerichtet (512MB RAM, 8GB HDD, 32-BIT Ubuntu-Server) * Ruby, Rack, Sinatra installiert   Offen: Konfiguration des Netzwerkinterfaces für Zugriff mit WinSCP um Deployinator-Projekt auf das VM-System zu schieben. |
| 31.05.2012 | * Derzeit ist ein Start von Deployinator nicht möglich, da scheinbar Abhängigkeiten nicht korrekt aufgelöst werden können, obwohl die angegebenen Pakete installiert wurde. Möglicherweise ist bei der Installation von Rails etwas schief gelaufen. * Wiederholte Abstürze des Grundsystems (Blue Screen) * Alternative: Installation auf EC2, Neuinstallation von Ruby, Rack, Sinatra … |
| 01.06.2012 | * Versuch der Installation bisher gescheitert, da einige Abhängigkeiten von Ruby nicht richtig aufgelöst werden können. |
| 04.06.2012 | * Abhängige Pakete konnten Identifiziert werden und die Installation läuft jetzt, allerdings besteht noch ein Problem mit der Authentifizierung bei der Anwendung, da ein http-Username und eine http-Groups erwartet wird. Aus der Beschreibung geht jedoch hervor, dass bei etsy ein Single-Sign-On Verfahren für die Anwendung verwendet wird. |
| 05.06.2012 | * Die Intallation von Deployinator lässt sich nun auf Ubunut12-amd64-server reproduzieren. * Die Authentifizierung konnte durch setzten von Umgebungsvariablen in der config.ru umgangen werden. Für den Testzweck der Anwendung reicht dies jedoch aus. |
| 06.06.2012 | * Dokumentation Installation Deployinator |
| 07.06.2012 | * Dokumentation Go / Deployinator |
| 12.06.2012 | * Stack-Konfiguration: neuer Stack erzeugt, Environments erzeugt die auf der Oberfläche ausgegeben werden. * Fragwürdig bleibt, wie Deployinator eine Deployment-Pipeline umsetzt. Über Environments können Buttons auf der Oberfläche erzeugt werden, die mit „deploy …“ bezeichnet sind. Dahinter verbirgt sich jeweils eine Methode, die, wie Environments auch, in „../stacks/my\_stack.rb“ definiert werden. Eine Umgebung hat demnach folgende Eigenschaften / Methoden: Name, Aktion (Methode), aktuelle Version, aktuelles Build und nächstes Build. * Aus dem Vortrag von Eric Kastner zu Deployintor kann ein Hinweis auf weitere Werkzeuge entnommen werden. Ob diese für eine Deplyoment-Pipeline Optional oder Verpflichtend sind, kann derzeit aber nicht gesagt werden. * **Tools, die Etsy biem Deployment-Prozess unterstützen (können / müssen?): Capistrano, Puppet, Fabric, Chef, Jenkins?** * Anfangsverdacht zur Funktionsweise / Ablauf von Deployinator:   + Kompilieren, Packen, Komponententests einer Anwendung mit CI-Server (z.B. Jenkins)   + Build-File wird in Projekt-SVN oder GIT hochgeladen   + Deployinator als Self-Service-Portal   + Das aktuelle Build soll deployt werden   + SVN-Check-Out des aktuellen Build-File nach Deployinator (oder GIT)   + Falls nötig wird Infrastruktur mit Puppet aufgebaut.   + Deployment wird dann via Capistrano ausgeführt (File kopieren, Server stoppen, starten, etc.) |
| 13.06.2012 | * Was macht Deployinator eigentlich?   + Web-Oberfläche für Einknopf-Deployment   + Genutzt werden Tools wie rsync um Code vom Deploy-System auf so genannte Web-Boxen zu spielen * Grundlegend scheint es zwei Umgebungen zu geben: Staging für Tests (wird Princess genannt) und Production (das Produktiv-System) * Wenn eine Anwendung den Zustand erreicht hat, dass diese auf die Staging-Umgebung deployt werden kann, startet z.B. ein Entwickler den Deployment-Vorgang. Besteht die Anwendung die Test-Anforderungen (nicht Teil von Deployinator) kann die Anwendung in das Produktivsystem deployt werden. * Die Entscheidung, ob deployt wird, ist bei Deployinator ein händischer Schritt. Jedes Deployment muss über das Self-Service-Portal angestoßen werden.   Ein interessantes Projekt ist das Jenkins-Deployinator-Plugin. <https://github.com/etsy/jenkins-deployinator> |
| 14.06.2012 | * Recherche zu GitHub und diversen stack-files von Deployinator-Forks auf Github |
| 15.06.2012 | * Git-Hub Fork von Deployinator * Installation Eclipse für Deployinator-Konfiguration * Installation von Git-Plugin für Deployinator * Installation von Git auf Deployinator-VM * Deployinator-Clone auf VM-System * Deployinator-Clone auf lokalem System (Laptop) * Shell-Script um Deployinator zu aktualisieren und anschließen zu starten. * Konfiguration kann jetzt über Eclipse auf lokalem System vorgenommen werden. Änderungen werden nach Git commited. Bei jedem Start von Deployinator über eigen Startskript werden die Code-Dateien aktualisiert, danach wird rackup aufgerufen und Deployinator läuft. |

## Deployment Pipeline

|  |  |
| --- | --- |
| Datum | Beschreibung |
| 28.02.2012 | Recherche Aufbau Deployment Pipeline |
| April 2012 | Die Umsetzung einer Deployment-Pipline für ein Test-Projekt wird nicht weiter fortgesetzt, da es derzeit kein Projekt gibt, für das ein solcher Prozess implementiert werden kann. |

## Entwicklungsstand adesso

|  |  |
| --- | --- |
| Datum | Beschreibung |
| 27.06.2012 | Beginn des Entwurfs von Fragen und Gedanken, was durch die Fragen beantwortet werden soll. |
| 28.06.2012 | Ausformulierung des Fragebogen |
| 29.06.2012 | Ausformulierung des Fragebogen |
| 04.07.2012 | Ausformulierung des Fragebogen |
| 05.07.2012 | Ausformulierung des Fragebogen |
| 11.07.2012 | Ausformulierung des Fragebogen  Tools zur statischen Code-Analyse:  <http://www.excelsior-usa.com/jet.html> |
| 12.07.2012 | Gestaltung eines Fragebogens mit Hilfe der Formularfunktion in Google-Docs. Dieser kann dann auf einfache Weise ausgefüllt werden. Neben einfachen Textboxen zur Beantwortung der Fragen können auf Auswahlfelder präsentiert werden. |
| 13.07.2012 | Einige Fragen, die zuvor mit Textboxen beantwortet werden sollten, wurden auf Statements umgestellt, den der Befragte zustimmen oder ablehnen kann. Auf diese Weise kann ich näher die Richtung der Antwort bestimmen und die Gefahr ist geringer unpräzise Antworten zu erhalten, die sich später schlecht verwenden lassen. Freie Textfelder bleiben aber wichtig, damit der Befragte die Möglichkeit hat, eigene Ideen und Anregungen an mich weiter zu geben. Die wesentlichen Gedanken des Fragebogen stammen aus dem Enterprise Continuous Delivery Maturity Model von Eric Minick und Jeffrey Frederick von 2011. Beim Durchsehen ist mir aufgefallen, dass dieses Modell auch das Enterprise Continuous Integration Maturity Model von 2009 ist. An einigen Stellen wurde der Text etwas ergänzt und die Zielformulierung innerhalb der Stufen an wenigen Stellen modifiziert. |
| 31.07.2012 | Fragen und Antworten in einem Dokument zusammengefasst, dieses kann als Anhang verwendet werden und zu einem Kapitel zusammengefasst werden. |

## Master-Thesis

|  |  |
| --- | --- |
| Datum | Beschreibung |
| 19.04.2012 | Aufbau Master-Dokument |
| 20.04.2012 | Gliederung, Kommentare, Verbindungen zwischen den Abschnitt herstellen |
| 08.06.2012 | Neuer Titel: „Evaluierung von ausgewählten Werkzeugen zur Realisierung von Continuous Delivery im Vergleich zur aktuellen Auslieferungspraxis von Software der adesso AG“ |
| 12.07.2012 | Was bedeutet Auslieferung von Software:  <http://www.jurpc.de/aufsatz/20010129.htm>  Software Distribution und Auslieferung:  <http://www.safenet-inc.de/software-monetization/electronic-software-delivery/>  Softwaremanagement:  <http://edoc.hu-berlin.de/cmsj/34/fabel-barbara-35/PDF/fabel.pdf> |
| 31.07.2012 | DB-Schema-Migration  <http://code.google.com/p/flyway/>  <http://liquibase.org/>  <http://code.google.com/p/solidbase/>  Alternative Mongo-DB: <http://hypoport.files.wordpress.com/2012/04/mongodb-fuer-java-entwickler.pdf>  Test-Tools:  <http://jbehave.org/> |
| 01.08.2012 | Kapitel: Entwicklungsstand adesso |
| 02.08.2012 | Kapitel: Entwicklungsstand adesso |