

Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs

Harald Radi
Catalysts GmbH
harald.radi@catalysts.cc

Hallo und willkommen zur 13. Wissensspritze, der ersten Wissensspritze nach unserer Sommerpause und zugleich der ersten Wissensspritze die nicht von Christoph Steindl gehostet wird.

Ich bin Harald Radi und der Titel der heutigen Wissensspritze, mit der eine ganze Serie zu diesem Thema beginnt, ist „Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs“.

Harald Radi hielt diesen Vortrag am 28.8.2009.

Die Aufnahme des Vortrags steht unter <http://wissensspritze.catalysts.cc> zur Verfügung.

Mainstream im Projektgeschäft Catalysts

Bisher

- Reine Desktop Anwendungen
- Thick Clients
- Formularbasierte Webanwendungen

Derzeit

- Thin Clients
- AJAX / RIA
- SOA

© www.catalysts.cc, 2009
Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs
2

In der letzten Wissensspritze ging es darum, welche Lehren aus den letzten 10 Jahren Projektgeschäft gezogen werden können. Heute betrachten wir zunächst, wie sich das Umfeld diverser Projekte in diesem Zeitraum verändert hat.

In der Vergangenheit Projekte -> sehr einfach gestrickt technologisch überschaubar.

Kunde forderte oft nur reine Desktop Anwendungen die für sich eigenständig funktionierten.

Reichte nicht aus -> zusätzlich eine Datenbank Anwendungen konnten Daten austauschen

Webanwendungen -> Aneinanderreihung mehrere Formulare der Komfort lies zu wünschen übrig und erinnerte nicht an herkömmliche Applikationen.

Neue Technologie -> alles andere unberührt lassen!!

Heute Hardware leistungsfähiger Trend zur Konsolidierung -> Hardware virtualisieren, Software auf Server bringen

Ergebnis: Thin Clients -> nur noch Präsentationsschicht eigentlicher Code am Server

Technologie: AJAX (async. Javascript and xml), RIA Frameworks (Flex, Silverlight, GWT, ext JS, dojo, ...)

Paradigma kommt zum Einsatz SOA (service oriented architecture) -> **Technologiemix**

Diskussion um die Tools

Catalysts



VS.



© www.catalysts.cc, 2009 Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs 3

Doch nicht nur das Projektumfeld hat sich über die Zeit verändert. Es werden auch mit jeder neuen Technologie diverse Entwicklungswerkzeuge angeboten, die den Einsatz derselben vereinfachen sollen.

Doch überall wo es Auswahl gibt, gibt es auch Diskussionen über Vor- und Nachteile.

Jedes Werkzeug Stärken und Schwächen -> Jeder Entwickler andere Vorzüge


Im Team sollte Wahl getroffen werden -> Entscheidung oft nur ein Kompromiss

Problem existiert seit es Werkzeuge gibt -> emacs vs. vi


Emacs vs. Vi "The endless geek 'holy war'"


Diskussion um die Tools Catalysts

Java:




Flex:







PHP:



.NET:





© www.catalysts.cc, 2009 Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs 4

Heute gibt es aber nicht mehr nur diese beiden Editoren, praktisch jede Technologie kommt mit Ihrer eigenen bevorzugten Entwicklungsumgebung.

Spezialfall -> Eclipse IDEA

Eclipse basiert -> Flex Builder Zend Studio


Proprietär -> MS Visual Studio

Freie Alternativen -> Sharp Develop

Technologie schränkt Auswahl ein z.b. .NET -> Visual Studio iPhone SDK -> Mac OS X

Nur zur Veranschaulichung -> natürlich keine vollständige Auflistung

Diskussion um die Tools



- Es gibt nicht „die eine“ IDE für alle Zwecke
- Damit die unterschiedlichen Entwicklungsumgebungen miteinander harmonisieren werden IDE-unabhängige Build-Werkzeuge benötigt

© www.catalysts.cc, 2009

Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs

5

Die eierlegende Wollmilchsau unter den IDEs gibt es leider nicht – oder auch gottseidank, sonst gäbe es nie wieder neue Features in IDEs, die sich ja ständig versuchen voneinander abzuheben.

Committer in opensource projekten -> einigung nicht möglich, zu viele beteiligte personen, unterschiedliche kulturen

Großteils Freizeitbeschäftigung -> vorhandene werkzeuge werden genutzt (lizenzen)

Großteils Freizeitbeschäftigung -> keine argumente, kein druckmittel take it or leave it

Unterschiedliche Betriebssysteme -> Windows, Linux, Mac, ... Chrome OS nicht alle tools auf allen plattformen

Gerade im Opensource Umfeld gängige Praxis -> IDE Unabhängige Build-Werkzeuge verwenden (make, maven, ant, nant, msbuild, etc.)

IDEs stoßen externen build prozess an -> meist gut unterstützt oft plugin unterstützung für build werkzeuge

Was man braucht

Catalysts

- Software Configuration Management (SCM)
 - Versionsverwaltung
 - Variantenverwaltung
 - Abhängigkeitenverwaltung
- Build Tool
- SDKs bzw. Kommandozeilentools der benötigten Compiler
- Test Framework
- Continuous Integration Server

© www.catalysts.cc, 2009
Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs
6

Ein Build-Werkzeug alleine kann aber noch nicht der Schlüssel zum Erfolg sein, für gemeinschaftliche Softwareentwicklung im Team braucht man noch einiges mehr.

Zunächst Software Configuration Management -> das ist ein Sammelbegriff für Versionsverwaltung, Variantenverwaltung, Abhängigkeitenverwaltung

Mehr dazu auf der nächsten Folie

Das eben erwähnte Build Tool -> einige davon, z.B. maven und ant setzen ein installiertes Java Runtime Environment voraus, nant -> .NET

Build tool selbst ist noch kein Compiler -> lediglich ein batch interpreter, der die jeweiligen Compiler ausführt

Daher müssen auch sämtliche SDKs und Kommandozeilentools der benötigten Compiler installiert sein - und zwar auf allen Entwicklungsrechnern - unabhängig von den eingesetzten SDKs

Da nicht alle Entwickler am gesamten Code arbeiten -> unit test frameworks für alle codeteile UND testdurchlauf als teil vom build -> dadurch werden entwickler auch über fehler benachrichtigt, die sie in fremdem Code verursachen (z.b. unabsichtlich durch Änderung von Abhängigkeiten)

Damit die Integrität des Codes auch regelmäßig von „unabhängiger Stelle“ überprüft wird ist auch ein Continuous Integration Server unerlässlich.

Builded in regelmäßigen Abständen (z.b. nach jedem Commit in die Versionsverwaltung) -> Integritätsfehler, fehlgeschlagene unit tests, etc. sofort für jeden sichtbar

„objektive“ builds (keine lokalen spezialkonfigurationen, keine systemeigenheiten, keine „verschmutzung“ durch entwicklungs Umgebung, etc.) ->

Erfolgreiche builds als referenzbuilds für späteres deployment bzw. generieren von Installationspaketen

SCMCatalysts

- Versionsverwaltung
 - ClearCase, CVS, Subversion, Perforce, SourceSafe, StarTeam
[\[http://en.wikipedia.org/wiki/Comparison_of_revision_control_software\]](http://en.wikipedia.org/wiki/Comparison_of_revision_control_software)

- Variantenverwaltung
 - Debug, Release
 - Java 1.5, Java 1.6
 - .NET 2.0, .NET 3.5, .NET CF 2.0, .NET CF 3.5

- Abhängigkeitenverwaltung
 - Produkt 1.0 => Framework 1.0
 - Produkt 2.0 => Framework 1.1, Library 2.0

© www.catalysts.cc, 2009Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs7

Versionsverwaltung: gemeinsamer sourcecode, erkennen von änderungskonflikten


Variantenverwaltung: varianten := builds des selben quelltextes mit unterschiedlichen build konfigurationen

Abhängigkeitenverwaltung: wie der name schon sagt -> Verwaltet Abhängigkeiten eines Projektes zu anderen Projekten und benötigten Bibliotheken

ermöglicht transitives Auflösen dieser Abhängigkeiten

ermöglicht Auflösen von Abhängigkeitskonflikten (Lib A -> Lib C 1.0, Lib B -> Lib C 2.0, Proj -> Lib A & Lib B)

Bedingungen



- Alle IDEs unterstützen die gewählte Versionsverwaltungssoftware
- Das Build Tool und alle zum Kompilieren und Testen benötigten SDK Komponenten sind auf allen Entwicklungsrechnern vorhanden
- Für Projektsourcen, SDKs und das Build Tool gibt es eine definierte Verzeichnisstruktur (ohne Leerzeichen und Umlaute)

© www.catalysts.cc, 2009
Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs
8

Für eine reibungslose Integration des externen Build Prozesses gibt es trotzdem noch einige Bedingungen.

Alle IDEs unterstützen ausgewählte versionsverwaltung -> toolgestützte Bulk-Modifikationen des Quelltextes und toolgestützte Refactorings arten sonst in einem Massaker im Dateisystem mit nicht mehr nachvollziehbaren Modifikationen aus.

Problem oft schon -> subversion versionsupgrade, unterschiedliche clients -> repository kaputt

Um lokal einen vollständigen Build erzeugen zu können müssen alle SDK Komponenten und Compiler vorhanden sein -> iPhone SDK setzt Mac OS voraus und wird daher niemals auf einem Windows Rechner vorhanden sein.

Lösung????? Nicht funktionierende builds lokal deaktivieren


abhilfe für continuous integration server: mac os build agents

Optional (best practice):

Pfade für tools und sourcen auf allen rechnern gleich:

- Vermeidet Überraschungen beim builden (path not found, etc.), ermöglicht arbeiten mit relativen pfeaden
- Erleichtert arbeiten auf fremden rechnern
- keine umlaute od. leerzeichen in den pfeaden -> trotz jahr 2009 immer noch probleme (z.b. flex compiler)

Vorteile



- Alle Entwickler verwenden das selbe Build Script, der Continuous Integration Server ebenfalls
- Alle Entwickler können einen vollständigen Build-Durchlauf ausführen ohne die entsprechenden IDEs installiert zu haben, der Continuous Integration Server ebenfalls
- Die Abhängigkeitenverwaltung garantiert, dass „frische“ Builds des Continuous Integration Servers sofort jedem zur Verfügung stehen, das lästige manuelle Kompilieren von abhängigen Modulen entfällt
- Haben Änderungen am Code einer Sprache (z.B. Java) auch Änderungen am Code einer anderen Sprache (z.B. C#) zur Folge, die dort einen Fehler auslösen (Autom. Code Generierung), dann sieht das auch der verursachende Entwickler


© www.catalysts.cc, 2009

Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs

9

Der Aufwand eine gemeinsame Build Umgebung einzurichten rentiert sich nur dann, wenn man sich dadurch auch gewisse Vorteile erkaufte.

Freiräume



- Definierte Builds sind ideal für die Auslieferung von Software, aber während der Entwicklung evtl. auch störend

- Entwickler müssen den Build individuell beeinflussen können
 - Erzeugen von Debuginformation
 - Verwenden neuerer Versionen von Bibliotheken
 - Selektives Auslassen einzelner Komponenten (z.B. Installer)

© www.catalysts.cc, 2009Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs10

Definierte Builds sind grundsätzlich eine gute Sache, Software“entwicklung“ auf der anderen Seite ist natürlich ein Prozess der sehr häufig mit Änderungen verbunden ist (versionsupgrade, technologiewechsel, neue features, etc.).

Definierter build nicht in allen situationen passend -> lokale konfiguration nötig







- erzeugen von debug informationen, statisches linken (bei flex) -> lokal notwendig, für deployment ungeeignet

- verwenden neuerer Versionen von bibliotheken während der entwicklung

- selektives auslassen einzelner komponenten (iphone sdk, installer lokal, etc.)

Unser Tool Stack

Catalysts

- Versionsverwaltung 
- Variantenverwaltung
- Abhängigkeitsverwaltung 
- Build Tool 
- Test Frameworks  
- Continuous Integration Server 

© www.catalysts.cc, 2009 Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs 11

Die Diskussion um die Wahl der geeigneten Tools gibt es nicht nur bei Entwicklungsumgebungen, sondern bei allen am Build Prozess beteiligten Komponenten. Nachdem schon viele unterschiedliche Tools ausprobiert haben, und mit einigen davon auch längere Zeit produktiv arbeiteten, fiel unsere Wahl auf folgende Komponenten.


.....

Bis auf Teamcity alle Opensource Projekte

Teamcity für kleine Teams auch gratis

Grund: weitverbreitet, gut unterstützt durch andere Tools, Pflegeleicht in der Wartung, geringer Initialaufwand beim Einrichten

Unsere Einsatzbereiche



- Serverentwicklung in Java
 - Server API definiert durch Java Interfaces
- Abbilden des Server APIs in unterschiedliche Client-Technologien mittels autom. Code Generation (<http://asdoclet.fluffnstuff.org/>)
 - Java Server API => Actionscript API, C# Client API, Java Client API, Objective-C API
- Cliententwicklung in Actionscript, C#, Java und Objective-C
- Unterschiedliche Variante desselben Client Codes für unterschiedliche Plattformen (z.B. .NET und .NET CF für Windows Mobile Client)
- Gemeinsame Installer für unterschiedliche Komponenten

© www.catalysts.cc, 2009

Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs

12

Basierend auf diesem Tool Stack entwickelten wir die Catalysts Platform und setzen damit die meisten unserer Projekte um.

Zusammenfassung Catalysts

1. Die Vielfalt von Technologien lässt eine freie Wahl der IDE nicht mehr zu.


2. Um die Integrität in heterogenen Projekten zu gewährleisten sind Software Configuration Management und ein eigenständiges Build Tool nötig.

3. Der Entwickler muss den Build Prozess trotzdem individuell beeinflussen können.

© www.catalysts.cc, 2009Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs13

Hiermit sind wir schon so gut wie am Ende mit der heutigen Wissensspritze, abschließend noch eine kurze Zusammenfassung des heutigen Themas ...

Ausblick



- 14. Wissensspritze:** Continuous Build über mehrere Programmiersprachen
- 15. Wissensspritze:** Automatische Prüfungen im Build-Prozess
- 16. Wissensspritze:** Dependency Management mit Ivy

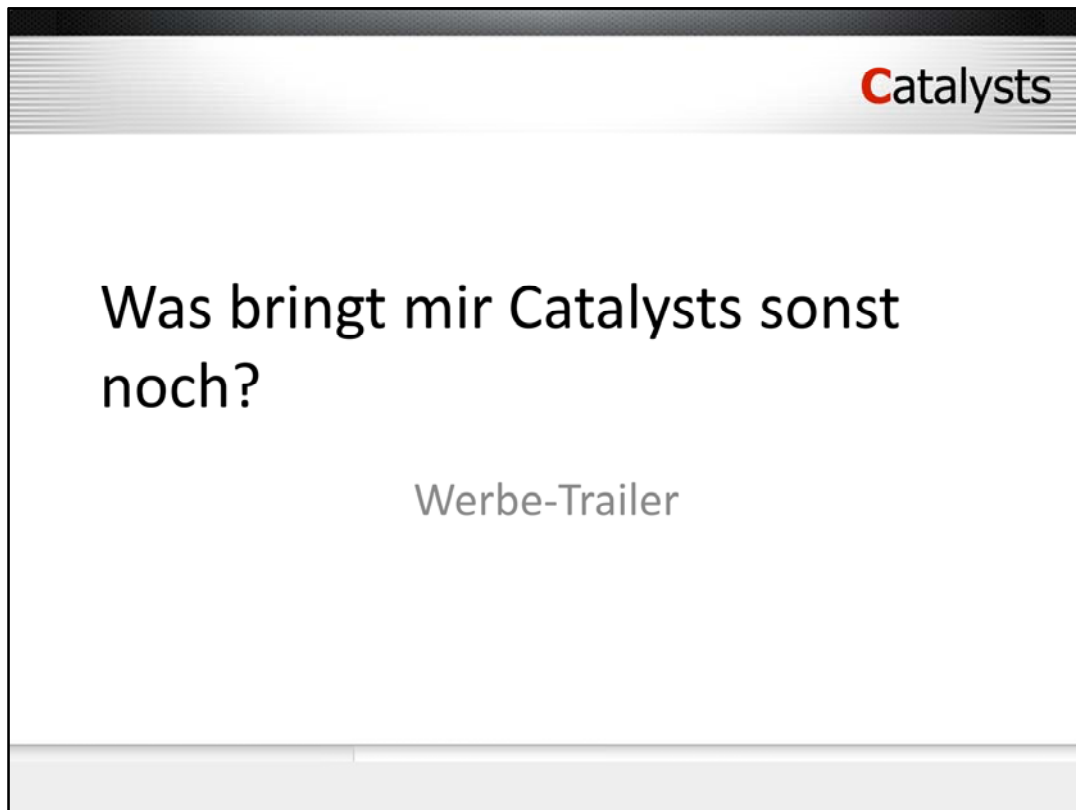
© www.catalysts.cc, 2009

Gemeinsame Softwareentwicklung im Team trotz unterschiedlicher IDEs

14

Wie schon im Eingang erwähnt bot die heutige Wissensspritze nur eine Übersicht über dieses Thema. Die nun folgenden werden sich intensiver mit den bereits angeschnittenen Teilbereichen befassen.

Vielen Dank und bis zum nächsten Mal!



Effizient-Katalysator fürs Team

Catalysts

Nur perfekt funktionierende Teams arbeiten
hocheffizient.



hilft Teams bei der Planung und Organisation.

Registrieren Sie sich gratis unter
<http://www.taskmind.net>

Gesundheitscheck für Projekte



- „Wenn ich vorher gewusst hätte, dass all die Probleme auftreten, dann wären wir das Projekt anders angegangen...“
- Sind Ihnen zu Projektbeginn alle technischen und organisatorischen Risiken bewusst?
- Oder gibt's auch bei Ihnen immer wieder viel Unvorhergesehenes?
- Fordern Sie heute noch ein Experten-Team von Catalysts für eine Vorsorgeuntersuchung für Ihr Projekt an. 400 Euro, die sich auszahlen!

Wir setzen Ihre Ideen um

Catalysts

- Sie wissen was – wir wissen wie
- Mit gewohnter Catalysts-Qualität
- Zu vernünftigen Preisen
- Schnell
 - erster Prototyp nach wenigen Tagen
 - Wochenweise mehr Funktionalität
- Probemonat – Ausprobieren und nichts riskieren!

Schneller am Ziel mit Catalysts - 1A-Qualität



- Wir entwickeln **Software nach Ihren Bedürfnissen**
 - für den Büroarbeitsplatz,
 - für unterwegs am Notebook,
 - für Ihr Handy.
- Wir entwickeln **Software auf agile Art.**
- Dadurch gibt's
 - frühzeitige und regelmäßige Auslieferung,
 - rasches Feedback und
 - ausschließlich wertvolle Funktionen im Produkt, keine Schnörkel.

Karin S. über Catalysts



Karin S. (Geschäftsführerin eines kleinen Dienstleistungsunternehmens, Nicht-IT)

“Ich leite ein kleines Dienstleistungsunternehmen (16 Mitarbeiter). Wir brauchen zuverlässige Simulationssoftware nach Maß, um unsere Aufträge schneller abwickeln zu können. Wir haben selbst keine Software-Entwickler. Ich kenne mich mit Software nicht aus, verwende sie nur. Über unseren bisherigen Softwarelieferanten ärgere ich mich, weil er für jede kleine Änderung Länge mal Breite verrechnet.”

Erfahrungen von Karin S. mit Catalysts:

- [Günstiger](#) als andere Firmen in OÖ
- „Ausprobieren und nichts riskieren!“ ([Probemonat](#))
- [Wertvolles zuerst, Unwichtiges später, Schnörkel gar nicht](#)
- [Papier-Prototyp nach einer Woche, erste Version nach einem Monat](#)
- [Monatliche Auslieferung](#) und monatliche Kurz-[Retrospektiven](#)
- [Änderungen gratis](#)
- [Von Profis geführt, kritische Denker](#)

Hans M. über Catalysts



Hans M. (Abteilungsleiter IT in einer größeren Firma)

“Ich leite die Softwareentwicklung (25 Entwickler) in einer größeren Firma. Meine Leute sind mit der Wartung der alten Programme schon ausgelastet. Sie kommen bei den neuen Technologien allerdings nicht mehr mit. Aus den Fachabteilungen kommen immer mehr Anforderungen, die wir nicht erfüllen können. Wir suchen ständig nach guten Software-Entwicklern, finden die aber nicht.”

Erfahrungen von Hans M. mit Catalysts:

- [Misch-Stundensatz](#) unter unseren internen Stundensätzen
- [Kreativere und bessere Lösungen](#)
- [Scrum, XP, TDD, laufende Integration, Personas, User Stories](#)
- [Stabile Technologie-Plattform](#)
- Modulare und zyklenfreie Architektur, [ausführlich getestet](#)
- [Unternehmensberatung inbegriffen](#)
- Exzellentes Team