



➤ “ALM gestern und heute – Infrastrukturen mit Open Source modernisieren”

CM Conf 2009

Rainer Heinold  
Technical Director Europe

October, 29th 2009

# ↘ Wer ist Elmar Mock?!?

## ↘ Wer ist Elmar Mock?!?

- Elmar Mock ist der Erfinder der Swatch. Er half damit vor 30 Jahren der Schweizer Uhrenindustrie aus einer ihrer größten Krisen
- Er glaubt an die positiven Aspekte von Krisen, weil sie Unternehmen zwingt, ihre Arbeitsweisen zu überdenken.
- „Eine Gruppe von Menschen in einem geschlossenen Raum atmet zuerst flacher, wenn der Sauerstoff knapp wird – aber irgendwann schlägt einer ein Loch in die Wand.“

↘ **Die 4 Stufen von ALM Lösungen – der inoffizielle Versuch einer Charakterisierung**

## ➤ Geschichte der ALM Lösungen

- Die größte Schwierigkeit ist die Unschärfe des Begriffs
  - Abhängig von der Person oder der Organisation
  - Oft wird ALM als Synonym für SCM (Software Configuration Management) benutzt
- Eine ALM Lösung im Rahmen dieser Präsentation
  - ist eine Menge von Tools, die zusammen einem verteilten Team erlauben zusammenzuarbeiten
  - hat eine gemeinsame Datenbasis
  - deckt den Softwareentwicklungs Prozess ab
    - Portfolio Management und Deployment sind ausgenommen

# ALM 1.0 - 1980-1995 (die frühen Jahre)

- Mit zunehmender Leistungsfähigkeit der Systeme stieg auch die Komplexität der Software
- Größere Teams und längere Projektlaufzeiten
- Client/Server war die dominante Systemarchitektur
- IT war im wesentlichen LAN basiert
- Sprachen der dritten Generation
  
- Projektteams arbeiteten meist an einem Standort mit einem dedizierten Projektserver; kaum direkter Kundenkontakt
- Wasserfall Modell
- Email war innerhalb der Entwicklerteams kaum verbreitet
  
- ALM Tools waren meist Client/Server Lösungen, mit Fat clients auf (proprietären) LAN Protokollen
- Geringer Integrationsgrad

- Viele spezialisierte Anbieter
  - Die meisten Designs stammen aus den 80er Jahren
- Schwach integriert, zumeist Datenaustausch
- Kein gemeinsames Daten oder Rollenmodell
- Toolketten umfassten Defect Tracking, Versionskontrolle und Makefile basierte Builds
- Dokumente und andere relevante Information wurde getrennt verwaltet

- Verteilt arbeitende Teams entwickeln sich zum De facto Standard
- Internet und WAN verändern die Infrastruktur
- Open Source Lösungen wurden „Salonfähig“
  - Apache
  - Linux
- Projektrelevante Daten sind auf viele Server verteilt
  - Geteilte Verantwortlichkeit
- Iterative und Agile Methoden ersetzen den Wasserfall mehr und mehr
- Email entwickelt sich zu einem der wichtigsten Kommunikationskanal

- Es entwickelten sich 2 entgegengesetzte Ansätze
- Integrierte Client/Server basierte Suiten
  - Hersteller beginnen Einzelprodukte zu integrieren
  - Spezifische, meist schwergewichtige Prozesse
  - Daten werden repliziert
  - Kommunikation findet meist abseits der Projektdaten statt
- Web-basierte Kollaborations Umgebungen
  - Hochintegrierte Tools mit geringerem Funktionsumfang
  - Leichtere, aber strikt umgesetzte Prozesse
  - Zusammenarbeit geschieht in Echtzeit
  - Kommunikation ist eng mit dem Rest des Projektes verbunden

- Verteilt arbeitende Teams sind der De-facto Standard
- Internet und WAN sind elementarer Bestandteil der IT Landschaft
- Open Source Lösungen sind auch in Unternehmen akzeptiert
- Projektdaten sind verteilter denn je
- Iterative und Agile Methoden lösen den Wasserfall in weiten Bereichen ab
- Email, Wiki u.ä. werden auch in Entwicklerteams täglich genutzt
- Hersteller beginnen die ALM Lösungen als SaaS Modelle anzubieten
  - Kosten vs. Investition
  - SLA anstelle von Software

- Eng verzahnte Toolketten
  - I.d.R. ein Mix aus proprietären on Open Source Komponenten
  - Nicht immer „Best-of-Breed“ – die Integration ist wichtiger und bringt mehr Vorteile als spezielle Features
- Übergreifendes Rollenbasiertes Zugriffsmodell
- Integrierte Kommunikation (email etc.)
- Benutzt das Internet zur Kommunikation
- Unterstützung moderner agiler Methoden
- Gemeinsame Prozessschicht über alle Komponenten

- Projekte sind fast immer verteilt
- Teams wechseln in ihrer Zusammensetzung häufiger (Virtuelle Teams)
- Green IT- durch Virtualisierung wandern Entwicklung und Betrieb in Cloud Umgebungen
  - Governance und Kontrolle von Entwicklungsrechnern gewinnt an Bedeutung
- Die Art der Kommunikation in Entwicklerteams wird durch neue Tools unterstützt

- Governance der Entwicklungstools UND der Infrastruktur in einem gemeinsamen Rollenmodell
- Kostentransparenz
  - Projektspezifische Abrechnungen
- Kontrollierte „Selbstbedienung“ für Anwender
- Unterstützung von private und public Clouds
  - Infrastruktur wandelt sich von einem Investment zu einem Kostenmodell
- Offene Architekturen für einfache Erweiterung
- Eingebauter Support für Agile Methoden
  - Beispiele: Burndown Charts, Drag-and-Drop von Aufgaben zwischen Entwicklungsschritten etc.

↘ **Best Practices**  
**Was Unternehmen von OpenSource lernen können ...**

# Modernisieren und zentralisieren Sie ihr Versionskontrollsystem

- Es ist die Basis für alle anderen Prozesse!
- Verteilte Daten sind schwerer zu kontrollieren, isolierte Daten können nur schwer mit anderen verknüpft werden
- Eine zentralisierte Datenhaltung erlaubt einfachere aber umgesetzte Prozessregeln
- Eine Datenmigration ist machbar
  - Es geht vor allem um die Erwartungshaltung, nicht um technische Hürden

# Ändern Sie die Kommunikation

- Falls noch nicht geschehen
  - Führen Sie ein Mailinglisten Tool ein
  - Stellen Sie soweit möglich auf Webinterfaces um
  - Keine Angst vor Wiki oder Diskussionsforen
- ... vergessen Sie die Verknüpfung zum VC System nicht!
- Open Source Lösungen bieten meistens gebrauchsfertige Lösungen für die meisten Werkzeuge an
  - Hook Skripte
  - Integrationen aller Art

# ... verknüpfen Sie die Kommunikation mit dem Rest des Projektes

- Einheitliches Set von Mailinglisten für jedes Projekt
  - Ersetzt die direkte Kommunikation zwischen den Entwicklern
  - Dadurch entsteht ein Archiv, welches das „Warum“ beantwortet
- Zusätzliche Komponenten helfen
  - Wiki für Entwicklerdokumentation etc.
  - Instant Messaging für schnelle Fragen
- Jede Information sollte ihre URL bekommen!
  - Erlaubt einfache Referenzierbarkeit und Nachvollziehbarkeit

# Entdecken Sie „Allgemeingut“

- Allgemeingut bereitstellen und nutzen
- Benutzen Sie allgemein verfügbare (meistens OpenSource) Komponenten als Teil des Software Stacks
  - Alles was keinen spezifischen Mehrwert erbringt
  - Eine OpenSource ähnliche Infrastruktur und die Vertrautheit mit dem OpenSource way erlauben eine fundierte Entscheidung
- Stellen Sie Komponenten aus Ihrem Software Stack als OpenSource bereit
  - In Frage kommt alles, was Sie derzeit selbst entwickeln, aber keinen echten Wettbewerbsvorteil liefert
  - Eine OpenSource Infrastruktur erlaubt eine kontrollierte Freigabe, oft auf derselben Umgebung
  - Hilft beim Setzen von Standards

- Anstatt über die Replikation von Daten nachzudenken, bringt oft eine höhere Datenkapazität für moderne Tools mehr
- Open Source Tools verlassen sich auf das Internet für eine Zusammenarbeit in Echtzeit
- Die Kommunikation wurde optimiert, um die Einstiegsschwelle so niedrig wie möglich zu halten.
- Zentralisierung erleichtert die Kontrolle über ihr geistiges Eigentum
  - Technisch
  - Logisch

# Schauen Sie sich SaaS Modelle an

- Es gibt viele Anbieter von OpenSource-basierten ALM Lösungen als SaaS
- Die Angebote unterscheiden sich in
  - Anpassbarkeit
  - Funktionalität
  - Sicherheit
  - Verfügbarkeit (SLA)
- Reduziert die Investmentkosten
  - Kein Betrieb eigener Server
  - Einfaches Setup
  - Subskription vs. Kauf

# Weniger ist mehr – Prozesskontrolle

- Open Source Communities verwenden einfache, aber streng umgesetzte Prozesse
- User haben unterschiedliche Dienstgrade
  - User
  - Contributor
  - Committer
- Users „verdienen“ sich die Beförderung durch ihr Wissen
- Offene Informationspolitik ist das A und O!

## ↳ Zusammenfassung

- ALM umfasst heute alle Aspekte der Entwicklung, einschließlich der Ressourcen
- ALM ist reif für SaaS
- OpenSource ist Frei(heit)
  - Nicht Frei im Sinne von Gratis
  - Es gibt einem die Freiheit, jederzeit Komponenten in der Toolkette sowie im Software Stack austauschen zu können
- Die Vertrautheit mit OpenSource Konzepten und Infrastrukturen erlaubt eine Nutzung in beide Richtungen
  - OpenSource als Teil des Lösungsstacks
  - OpenSource als Option, sich auf die wirklich wichtigen Teile Ihrer Applikationen zu fokussieren

# Fragen?