



Konfigurations- und Änderungsmanagement Konzepte und Überblick

Hans-Joachim Erchinger
HJErchinger@serena.com

- Zum Anwärmen
[Beispiele für Konfigurationsmanagement](#)

- Es wird ernst

[Definitionen](#) für

- [Konfigurationsmanagement](#)
- [Teilgebiete des Konfigurationsmanagements](#)
 - [KMO](#) – Konfigurationsmanagementorganisation und –planung
 - [KI](#) – Konfigurationsidentifizierung
 - [KÜ](#) – Konfigurationsüberwachung
 - [KB](#) – Konfigurationsbuchführung
 - [KA](#) – Konfigurationsaudit

- Es geht zur Sache

Grundlagen und Konzepte

- Versionsmanagement
- Workflows und Reifegrade
- Objekttypen
- Strukturierung (Subsysteme, Module)
- Projekt-Sichten und Projekt-Hierarchien (Re-Use)
- Baselines: Stände von Konfigurationen
- Änderungsmanagement
- Auftrags-basiertes Arbeiten
- Auftrags-basiertes Baselining



Serena Software

Hans-Joachim Erchinger
HJErchinger@serena.com

- **Qualifiziert**

1980 in Kalifornien gegründet

Produktpalette: Requirementsmanagement, Changemanagement, Configuration Management, Buildmanagement, Deploymentmanagement, Product and Portfolio Management

- **Investitionssicherheit**

Mitarbeiterzahl gesamt > 900 Mitarbeiter

Standorte in USA, EMEA und ASPA

Umsatz FY 2007 \$255 Mio.

Ertrag (EBIT) FY 2007 \$103 Mio.

- **Zukunftsorientiert**

ca. 280 Mitarbeiter in Forschung und Entwicklung innovativer Software-Lösungen

- **Lösungsorientiert**

ca. 480 Mitarbeiter in Kundenberatung, Consulting und Customer Support (7x24)

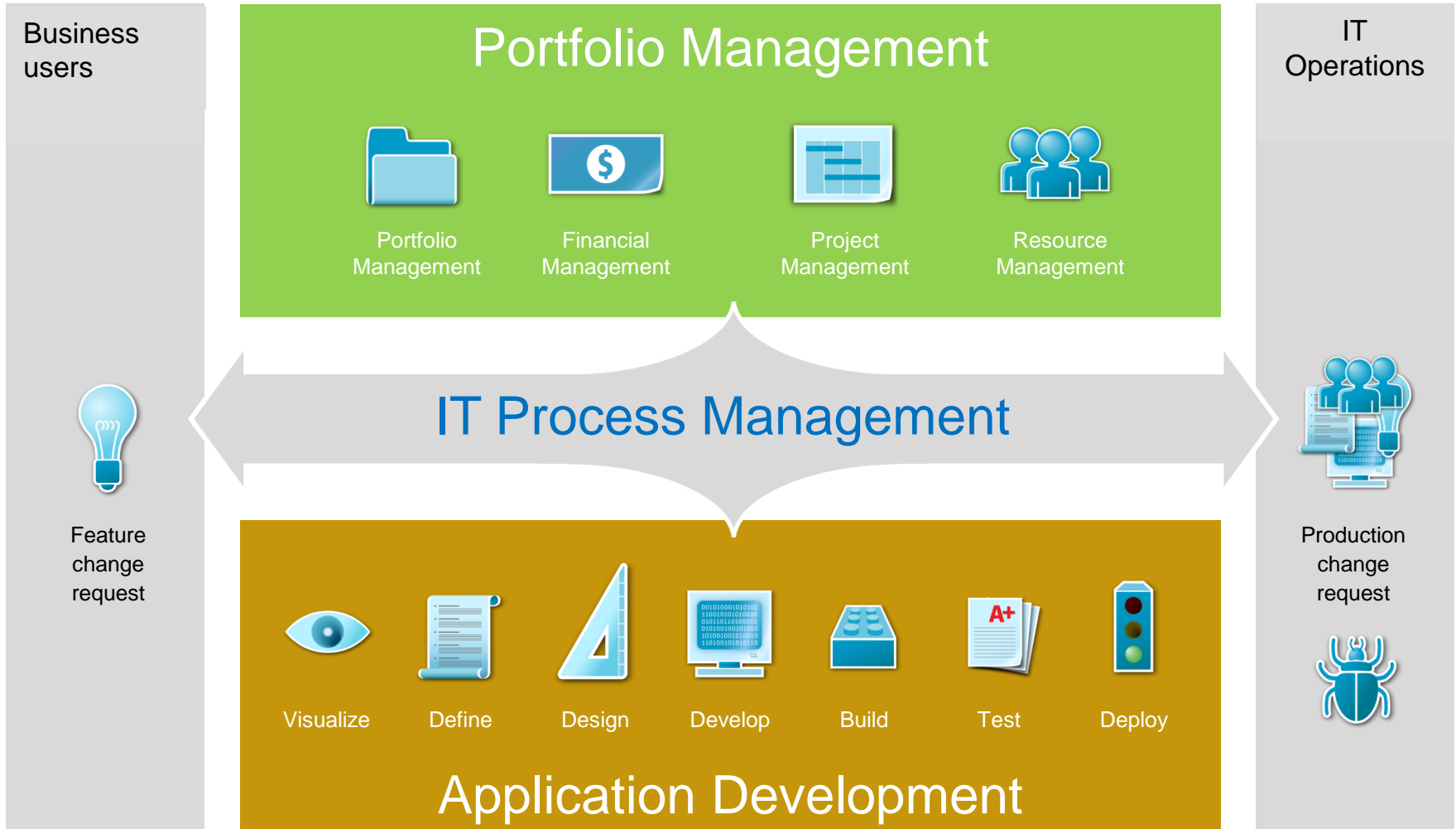
- **Kunden**

weltweit >15.000 vom KMU bis zum Blue Chip Unternehmen

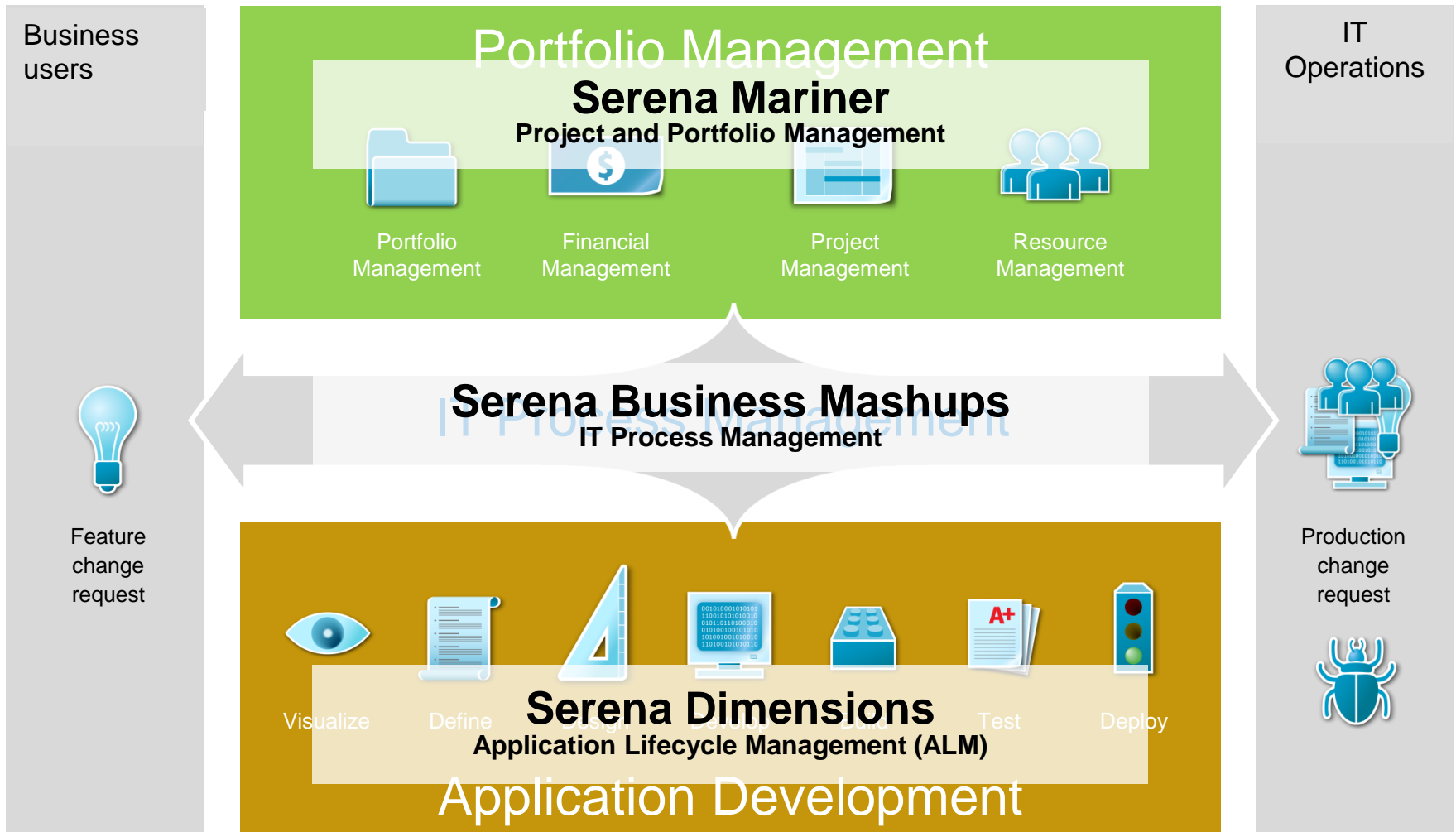
Wir ermöglichen unseren Kunden unternehmensweite Anforderungen und Prozesse zu visualisieren, zu steuern sowie deren Umsetzung und Anwendung zu automatisieren.

So können notwendige Änderungen in nachhaltige Geschäftsvorteile umgewandelt werden.

The big picture: Serena ALM



The big picture: Serena ALM



Konfigurations- und Änderungsmanagement

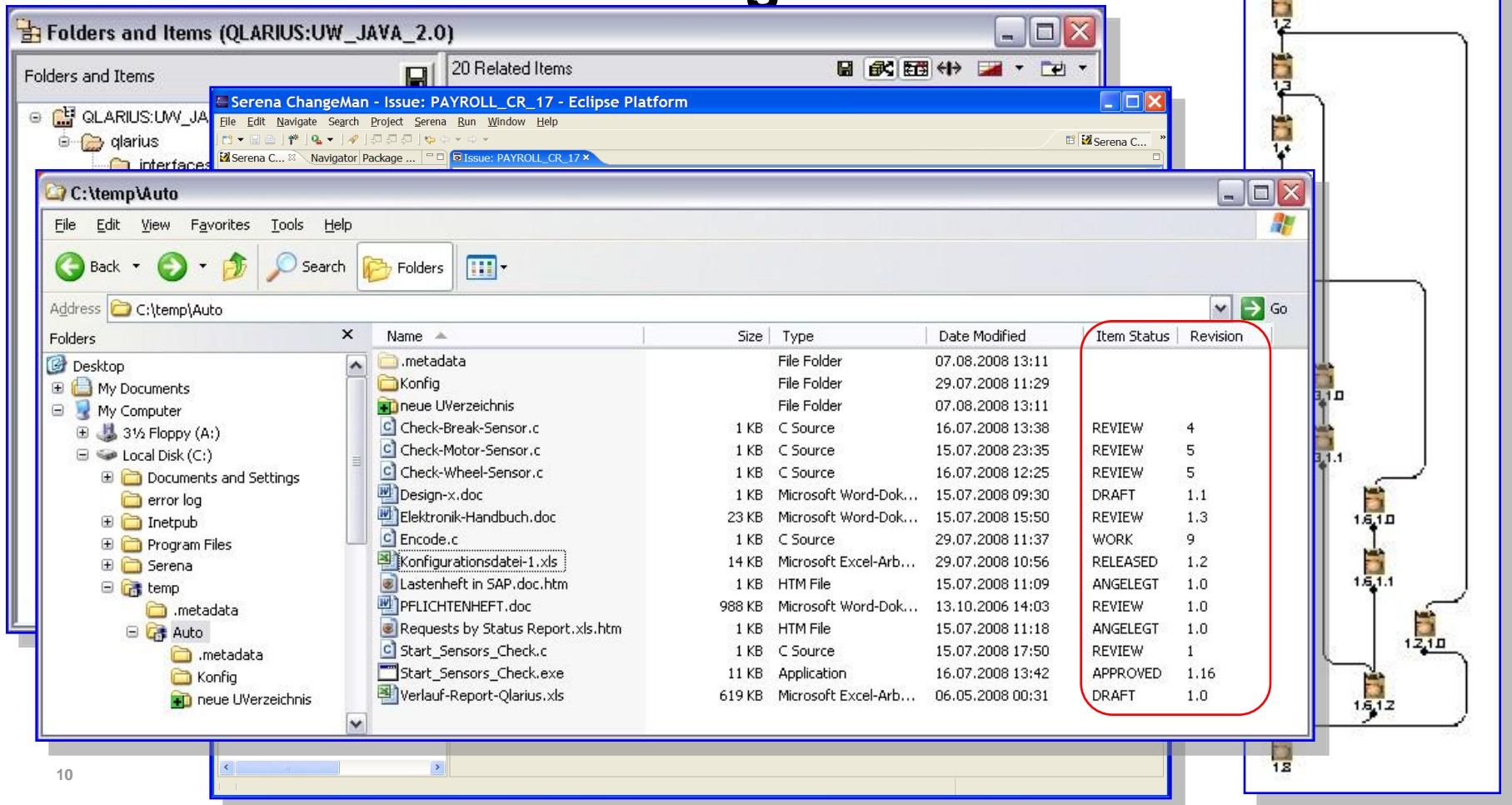
Konzepte und Überblick

Beispiele

Hans-Joachim Erchinger

HJErchinger@serena.com

Reine Versionierung in der Software-Entwicklung



The screenshot displays the Serena ChangeMan interface. On the left, a file explorer shows the directory structure for 'C:\temp\Auto'. The main area shows a table of files and folders with their metadata and version control information.

Name	Size	Type	Date Modified	Item Status	Revision
.metadata		File Folder	07.08.2008 13:11		
Konfig		File Folder	29.07.2008 11:29		
neue UVerzeichnis		File Folder	07.08.2008 13:11		
Check-Break-Sensor.c	1 KB	C Source	16.07.2008 13:38	REVIEW	4
Check-Motor-Sensor.c	1 KB	C Source	15.07.2008 23:35	REVIEW	5
Check-Wheel-Sensor.c	1 KB	C Source	16.07.2008 12:25	REVIEW	5
Design-x.doc	1 KB	Microsoft Word-Dok...	15.07.2008 09:30	DRAFT	1.1
Elektronik-Handbuch.doc	23 KB	Microsoft Word-Dok...	15.07.2008 15:50	REVIEW	1.3
Encode.c	1 KB	C Source	29.07.2008 11:37	WORK	9
Konfigurationsdatei-1.xls	14 KB	Microsoft Excel-Arb...	29.07.2008 10:56	RELEASED	1.2
Lastenheft in SAP.doc.htm	1 KB	HTM File	15.07.2008 11:09	ANGELEGT	1.0
PFLICHTENHEFT.doc	988 KB	Microsoft Word-Dok...	13.10.2006 14:03	REVIEW	1.0
Requests by Status Report.xls.htm	1 KB	HTM File	15.07.2008 11:18	ANGELEGT	1.0
Start_Sensors_Check.c	1 KB	C Source	15.07.2008 17:50	REVIEW	1
Start_Sensors_Check.exe	11 KB	Application	16.07.2008 13:42	APPROVED	1.16
Verlauf-Report-Qlarius.xls	619 KB	Microsoft Excel-Arb...	06.05.2008 00:31	DRAFT	1.0

On the right side of the interface, a vertical flow diagram shows the version history for a specific item, with nodes labeled with version numbers such as 1.0, 1.1, 1.2, 1.3, 1.5.1.0, 1.5.1.1, 1.2.1.0, and 1.5.1.2.



Konfigurationsmanagement im Catering



Konfigurationsmanagement gibt es tatsächlich auch in ganz anderen Bereichen, z.B. im Catering:

Um die Nachweispflicht gegenüber dem Gesundheitsamt zu gewährleisten (z.B. dass der aus der Großküche an die Hochzeitsgesellschaft gelieferte Kartoffelsalat qualitativ und hygienisch einwandfrei war), bewahrt der Cateringbetrieb Proben auf.

Diese Proben stellen Stände dar, das Repository ist in diesem Fall ein Kühl- oder Gefrierschrank.

Verwalten von Konfigurationen und Auslieferungen – ein (scheinbar?) abstruses Beispiel

In meinem Schrank hängen und liegen

- Anzüge
- Hemden
- Krawatten
- Socken
- Schuhe



Verwalten von Konfigurationen und Auslieferungen – ein (scheinbar?) abstruses Beispiel

Die Kombination aus diesen bildet eine *Konfiguration*, wann ich welche Zusammenstellung getragen habe, ist letztlich ein „Stand“, also eine „*Version*“ oder „*Baseline*“.

Habe ich diese Konfiguration bei einem Kunden getragen, ist das eine „*Auslieferung*“ oder ein „*Release*“.

Wollte ich Wiederholungen vermeiden,
könnte ich diese Versionen unter KM stellen 😊



Konfigurations- und Änderungsmanagement

Konzepte und Überblick

Definitionen

Hans-Joachim Erchinger

HJErchinger@serena.com

Was ist Konfigurationsmanagement?

Konfigurationsmanagement ist eine Menge von *Verfahren*, Methoden und Werkzeugen zur *Verwaltung aller Arbeitsergebnisse* der Entwicklung und ihrer *Änderungsgeschichte*.

<http://www.ias.uni-stuttgart.de/st2/infoquellen/ans1.html>

Universität Stuttgart, Institut für Automatisierungs- und Softwaretechnik

Konfigurationsmanagement ist die Sicherstellung der vollständigen *Reproduzierbarkeit* eines bestimmten Produktes durch die eindeutige *Identifizierung* aller *Teile* und *Umstände*, die zu dem Produkt geführt haben.

Diese Aussage fasst den Kern des Konfigurationsmanagement allgemeingültig zusammen. Zum einen muss genau bekannt sein, aus welchen Bestandteilen ein Produkt gefertigt wurde. Zum anderen müssen auch die genauen Umstände und die Umgebung bekannt sein.

<http://www.infos.informatik.uni-stuttgart.de/zeitung/01-2.html#epos>

Konzeptionelle Teilgebiete

- KMO – Konfigurationsmanagementorganisation und –planung
- KI – Konfigurationsidentifizierung
- KÜ – Konfigurationsüberwachung
- KB – Konfigurationsbuchführung
- KA – Konfigurationsaudit

<http://de.wikipedia.org/wiki/Konfigurationsmanagement>

Konzeptionelle Teilgebiete

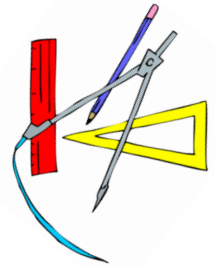
KMO – Konfigurationsmanagementorganisation und –planung

Im Rahmen der KMO erfolgt das Treffen organisatorischer und technischer Festlegungen zu KI, KB, KÜ und KA.

Dies kann sowohl *projekt-* als auch *produktspezifisch* erfolgen.

Wesentliche Entscheidungen betreffen

- die Auswahl der einzusetzenden *Werkzeuge*,
- die *Zuweisung* von zu erfüllenden *Aufgaben* an Aufgabenträger
- sowie deren *Informationsrechte* und *–pflichten*
- und die Festlegung von *Prozessdefinitionen*.



Zudem fällt die Auditierung des Gesamtsystems zur Überprüfung der Einhaltung und Wirksamkeit der getroffenen Entscheidungen der KMO zu.

<http://de.wikipedia.org/wiki/Konfigurationsmanagement>

Konzeptionelle Teilgebiete

KI – Konfigurationsidentifizierung

KI ist die Voraussetzung für die Durchführung von KB, KA, und KÜ.

Sie umfasst

- die Auswahl von *Konfigurationseinheiten*,
- deren Formierung zu einer *Produktstruktur*,
- Dokumentation sowie Nummernbildung zum Zweck der eindeutigen *Identifizierung*.

Wesentliche Fragestellungen beziehen sich auf die *Granularität* einer Konfigurationseinheit, die Festlegung von *Bezugskonfigurationen* (*Baseline*) sowie die Auswahl einer Nummernsystematik.

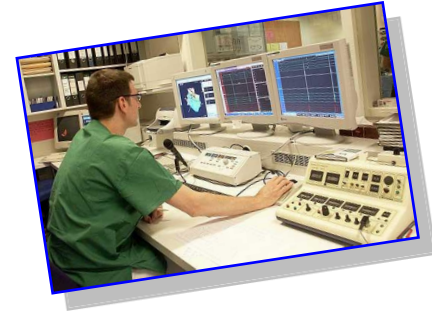


<http://de.wikipedia.org/wiki/Konfigurationsmanagement>

Konzeptionelle Teilgebiete

KÜ- Konfigurationsüberwachung

- KÜ adressiert Probleme, die sich aus der *Veränderung* einer Konfiguration ergeben.
- Zentrale Aktivität stellt daher das *Änderungsmanagement* (engl. *change management*) dar.
- Ziel dessen ist es, alle *Änderungen* an Konfigurationseinheiten sowie deren zugeordneten Dokumenten zu *identifizieren, beschreiben, klassifizieren, bewerten, genehmigen* und *einzuführen*.
- Die mit entsprechenden Kompetenzen und Befugnissen ausgestattete Instanz wird als Konfigurationsausschuss (engl. *Change Control Board (CCB)*) bezeichnet.



<http://de.wikipedia.org/wiki/Konfigurationsmanagement>

Konzeptionelle Teilgebiete

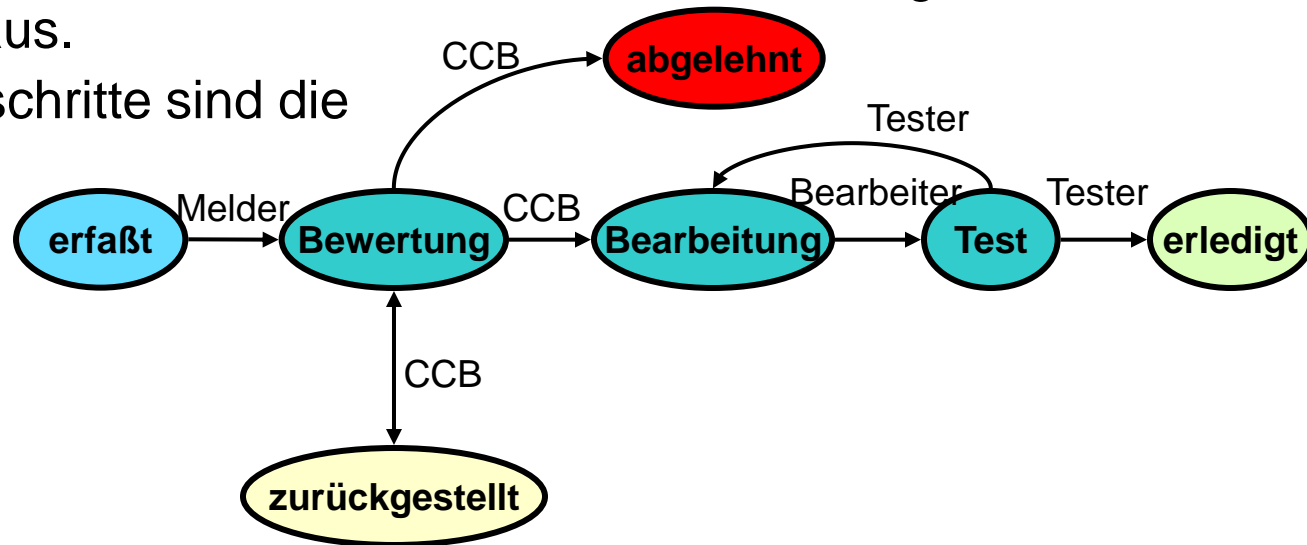
KÜ- Konfigurationsüberwachung

- Sinnvolle Konfigurationsüberwachung setzt einen *festgeschriebenen, formalen Prozess* für die Behandlung von Änderungen voraus.

- Übliche Prozessschritte sind die

- Beantragung,
- Bewertung,
- Entscheidung,
- Beauftragung,
- Review
- Freigabe

der umgesetzten Änderungen.



<http://de.wikipedia.org/wiki/Konfigurationsmanagement>

Konzeptionelle Teilgebiete

KÜ- Konfigurationsüberwachung



- Referenzobjekt für das Änderungsmanagement ist eine *Bezugskonfiguration*.
- Zu einem bestimmten Zeitpunkt stellt diese *zusammen mit allen bis dahin freigegebenen Änderungen die gültige Konfiguration* dar.

<http://de.wikipedia.org/wiki/Konfigurationsmanagement>

Konzeptionelle Teilgebiete

KB – Konfigurationsbuchführung

- Der Prozess der Konfigurationsbuchführung hat die *rückverfolgbare Dokumentation* der Konfigurationen und Konfigurationseinheiten zum Ziel.
- Sie sollte von daher bereits mit der ersten Erstellung von Konfigurationsdaten einsetzen.
- Gegenstand der KB sind alle Daten zur Konfigurationsidentifizierung und -überwachung.



<http://de.wikipedia.org/wiki/Konfigurationsmanagement>



Konfigurations- und Änderungsmanagement Konzepte und Überblick

Grundlagen & Konzepte

Hans-Joachim Erchinger

HJErchinger@serena.com

Definitionen: Versionierung

Versionierung

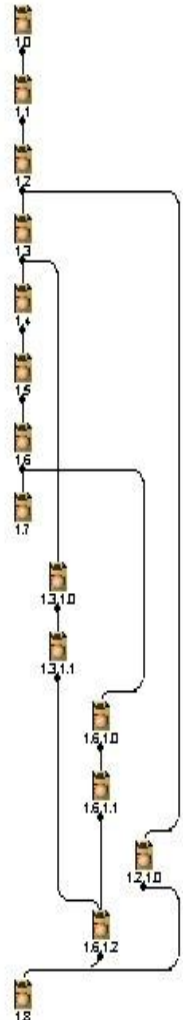
Als Versionierung bezeichnet man in der Software-Entwicklung die *Archivierung alternativer Dateiversionen*.

In diesem Zusammenhang kann es sich um eine *alte* Dateiversion, aber auch um eine *alternative* nahezu gleichwertige Dateiversion handeln, die beispielsweise Bestandteil eines neuen Entwicklungszweigs ist.

Sinn und Zweck ist die Möglichkeit, *vorige Versionsstände wiederherzustellen und Änderungen zurücknehmen* zu können.

Zum *Versionsstand* sagt man auch *Build*, zu dessen Veröffentlichung *Release*. Der Versionsstand wird in einer *Versionsnummer* angegeben.

<http://de.wikipedia.org/wiki/Versionierung>



Definitionen: Versionsverwaltung

Versionsverwaltung

Unter einer *Versionsverwaltung* versteht man ein System, welches typischerweise in der Softwareentwicklung zur *Versionierung* und um den *gemeinsamen Zugriff auf Quelltexte* zu kontrollieren, eingesetzt wird.

Hierzu werden alle laufenden Änderungen erfasst und alle *Versionsstände* der Dateien in einem Archiv mit *Zeitstempel* und *Benutzerkennung* gesichert.

Die Versionsverwaltung ist eine Form des Variantenmanagements. Die übergreifende Disziplin ist das Software Configuration Management (kurz: SCM).

(...)

<http://de.wikipedia.org/wiki/Versionsverwaltung>

In der Versionsverwaltung finden wir typischerweise diese Operationen:

Check-Out reserviertes Ausleihen eines Objektes (eine Revision einer Datei) aus dem Versionsverwaltungssystem

Check-In Rückgabe eines vorher reserviert ausgeliehenen Objektes (eine Dateirevision)

Get Nicht-reserviertes Ausleihen eines Objektes (Dateiversion)

Häufig für Build-Läufe angewendet

Darüber hinaus auch:

Löschen (?!), umbenennen (Historie!), sperren, ...

Merkmale der Versionsverwaltung

Merkmale von Versionsverwaltungssystemen:

- Das VS ist der „Master“:
der Benutzer kann sich immer nur eine *Kopie* eines Objekts *ausleihen*
- *Variablensubstitution*:
Bestimmte Angaben innerhalb der Datei können durch aktuelle Werte des VS ersetzt werden (z.B. Datum des letzten Check-In, aktuelle Revisionsnummer, ...)
- *Historisierung*: → KB
Das VS führt automatisch alle Verwaltungsinformationen mit (wer, wann, was, warum)
- *Berechtigungsprüfung*: → KMO
Darf der Benutzer die angeforderte Operation überhaupt ausführen?

Merkmale der Versionsverwaltung

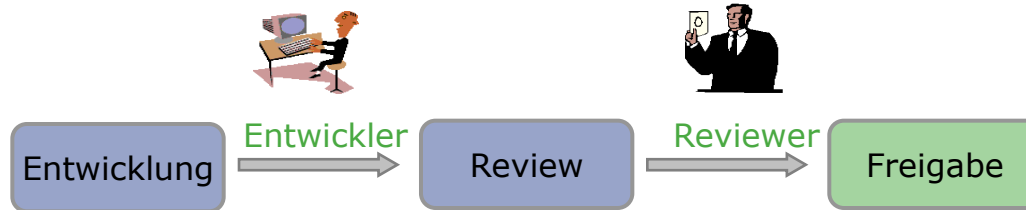
- Die bisher ausgeführten Merkmale kennzeichnen einfache Versionsverwaltungssysteme, wie z.B. RCS, VCS, ...
- Diese Merkmale definieren *kein Konfigurationsmanagement*
- Solche Systeme bieten *keine* Möglichkeiten, *Beziehungen* zwischen Objekten zu verwalten
z.B. Beziehung zw. Source und zugehöriger Dokumentation, ...
- *Berechtigungsprüfungen* basieren häufig auf Dateisystem-Gruppen und –Rechten
- Diese Konzepte bieten *keine Gruppierung* von Objekten zu Modulen oder (Sub)Systemen
- Es gibt keine *Gültigkeits-Markierung*.
Ist die Tip-Revision immer die gültige?

Workflows und Reifegrade

- Bloße Revisionsverwaltung kann *nicht* die *Gültigkeit* einer Revision kennzeichnen.
- Ebenso kann auch *keine Zuständigkeit* z.B. bei Review-Prozessen oder Build-Reife dargestellt werden.
- Dazu benötigt man *Workflow-* oder *Promotionsmodelle*.
- *Workflows* bestehen üblicherweise aus
 - *Zuständen*
 - *Übergängen*
 - *Rollen, welche einen Übergang ausführen dürfen*



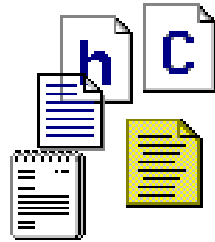
- Nicht alle Objekte sind gleich zu behandeln (z.B.):
 - Sourcen oder Dokumente durchlaufen ggf. ein Review



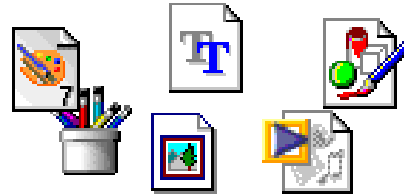
- Notizen werden lediglich abgelegt, durchlaufen kein Review und werden nicht freigegeben
 - gespeichert
- Daher muss eine *Typisierung* möglich sein, mit der
 - *unterschiedliche Workflows* verknüpft sind
 - und damit *unterschiedliche Behandlung* möglich wird
- Diese Typisierung muss abstrakt sein, orientiert sich also *eben nicht an Extensions*

Source-Dateien

C, C++,
Java, VB,
COBOL, PL/I,
...



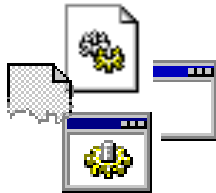
Grafikdateien



GIF, JPG,
BMP, TIF,
...

Executable's

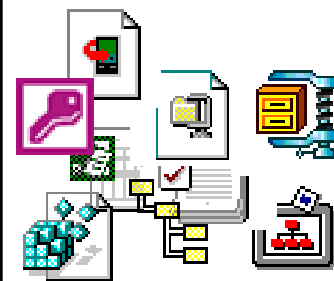
Exe, Dll,
Obj, ...



Produkt

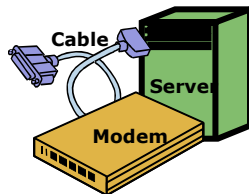


Konfig-Dateien, T-Scripte



PRC, XLS,
DB-Schema,
REG, INI,
APP, tds,
...

Hardware



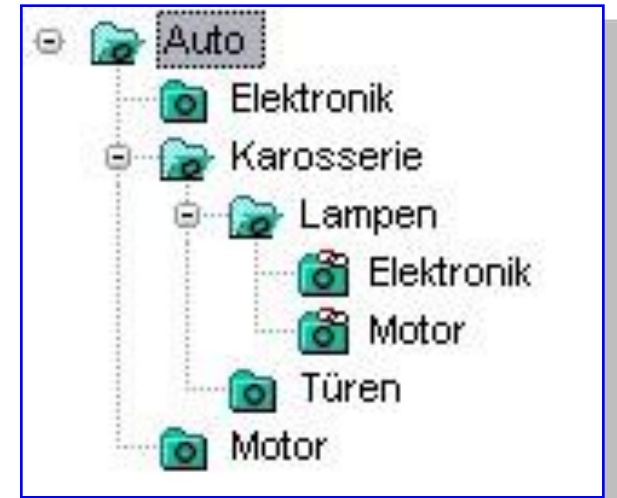
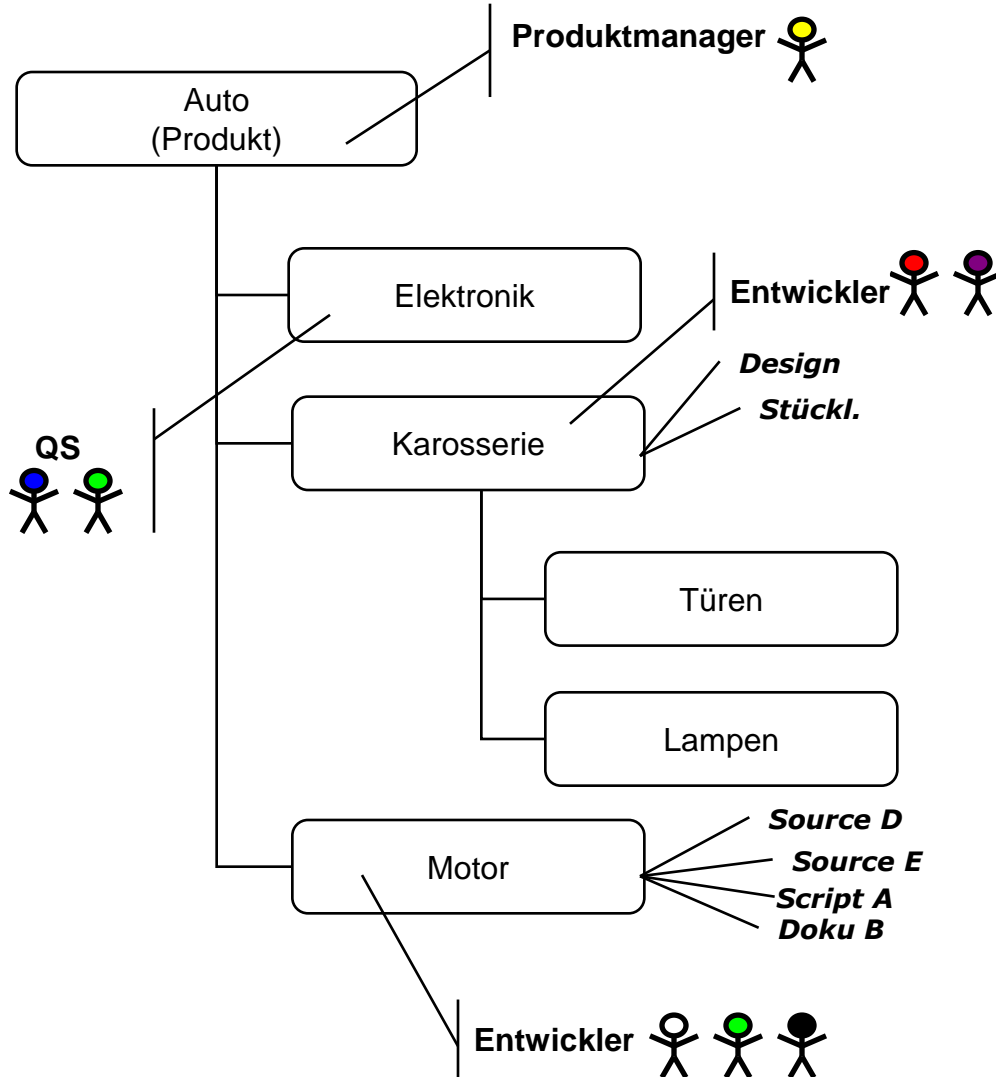
Dokumente



TXT, DOC, RTF, PDF, XLS, ...

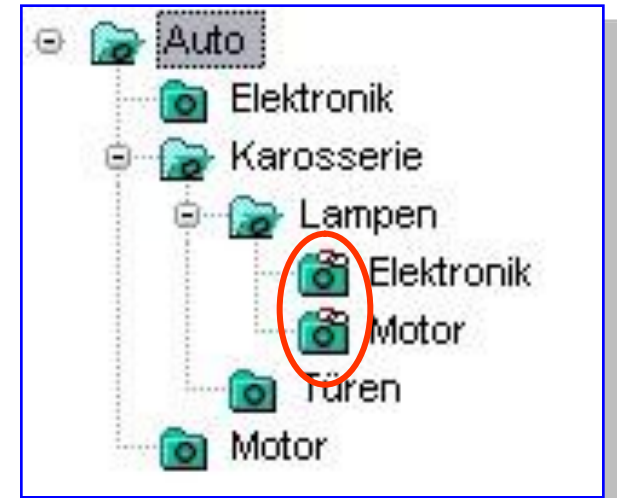
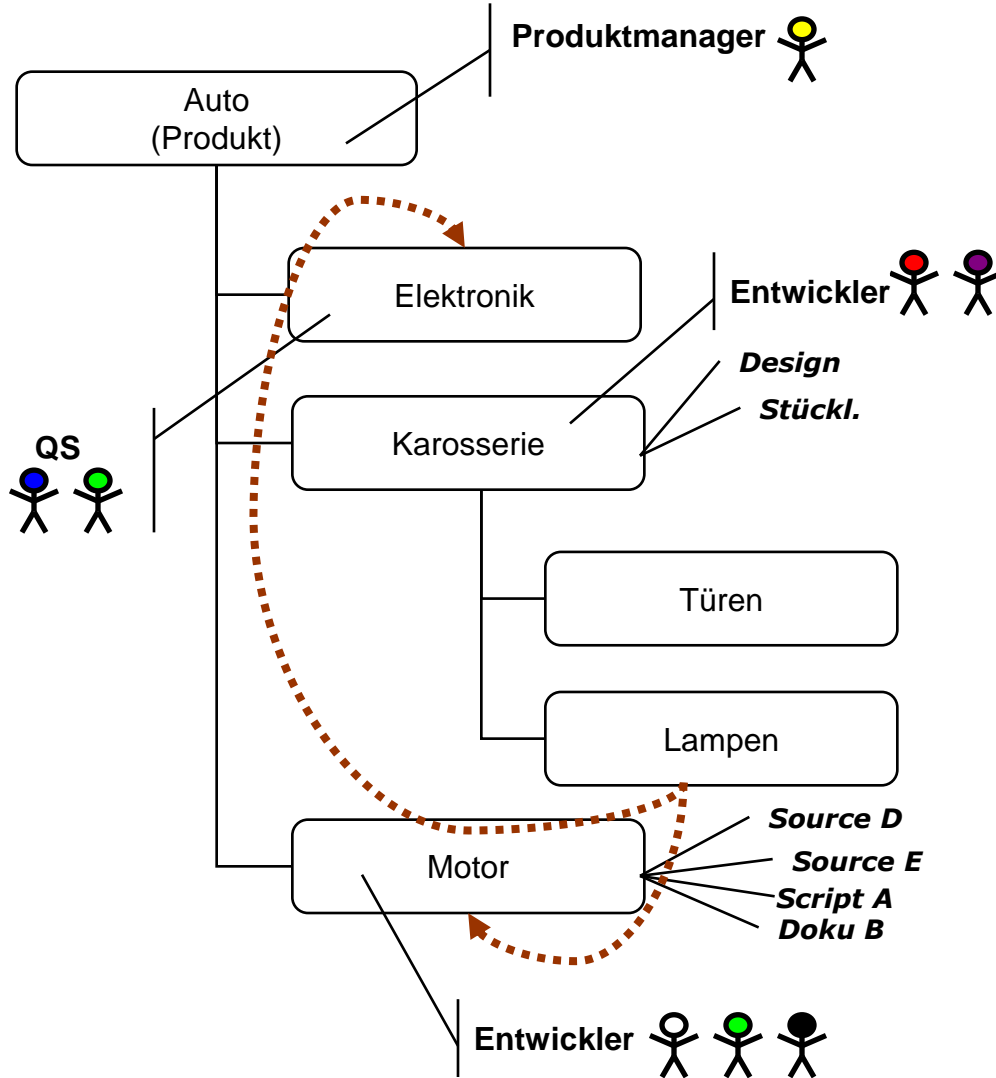
- Solange man nur Ordnerstrukturen zur Verfügung hat, werden häufig aus Gründen des Compilierens, des Ausliefernens oder der Zugriffsrechte die eigentlichen *Nutzobjekte* (Sourcen, Executables, Load-Modules, Deliverables) von den sie beschreibenden *Dokumenten getrennt*.
- Damit ist aber die *Zusammengehörigkeit nicht mehr erkennbar* oder es müssen Hilfskonstrukte (wie z.B. Namensgleichheiten o.Ä.) verwendet werden.
- Führt man hingegen eine abstrakte Strukturierung ein, kann innerhalb eines Projektes z.B. eine Hierarchie vom *System* über *Subsysteme* bis hin zu *Modulen* dargestellt werden.

Strukturierung (Subsysteme, Module)



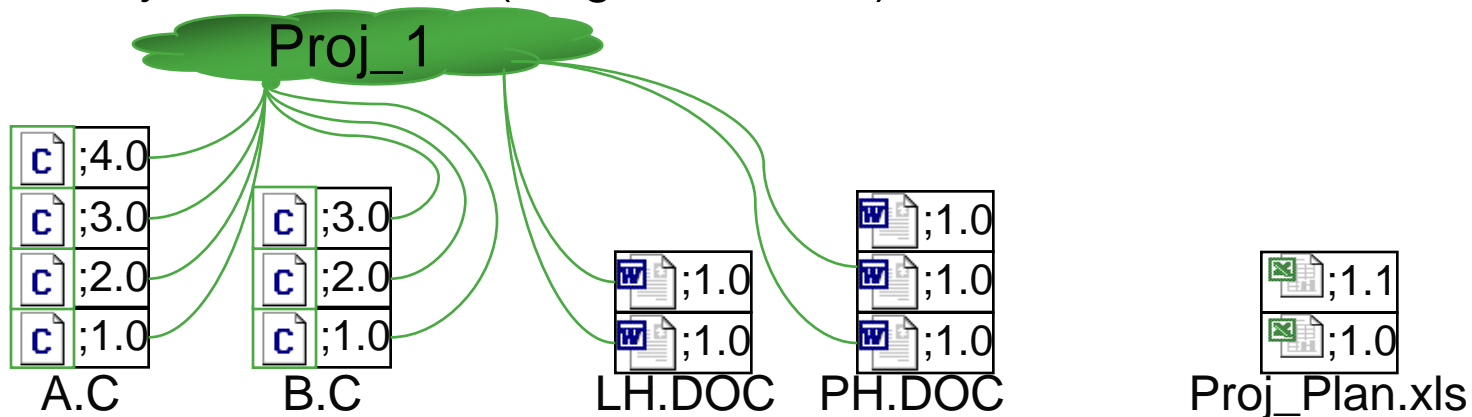
- Bietet das Konfigurationsmanagement-System darüber hinaus auch noch die Möglichkeit, zwischen solchen Modulen Referenzbeziehungen (z.B. *used-by*) zu knüpfen, kann auf dieser Ebene sogar *Re-Use* ermöglicht und dargestellt werden.
- Von dieser Möglichkeit wird häufig Gebrauch gemacht, wenn z.B. *Software-Bibliotheken* aufgebaut und verschiedenen Projekten zur Verfügung gestellt werden.
- Wir sehen im nächsten Bild eine solche Struktur incl. *Re-Use* – allerdings noch ohne Inhalte.

Strukturierung (Subsysteme, Module)



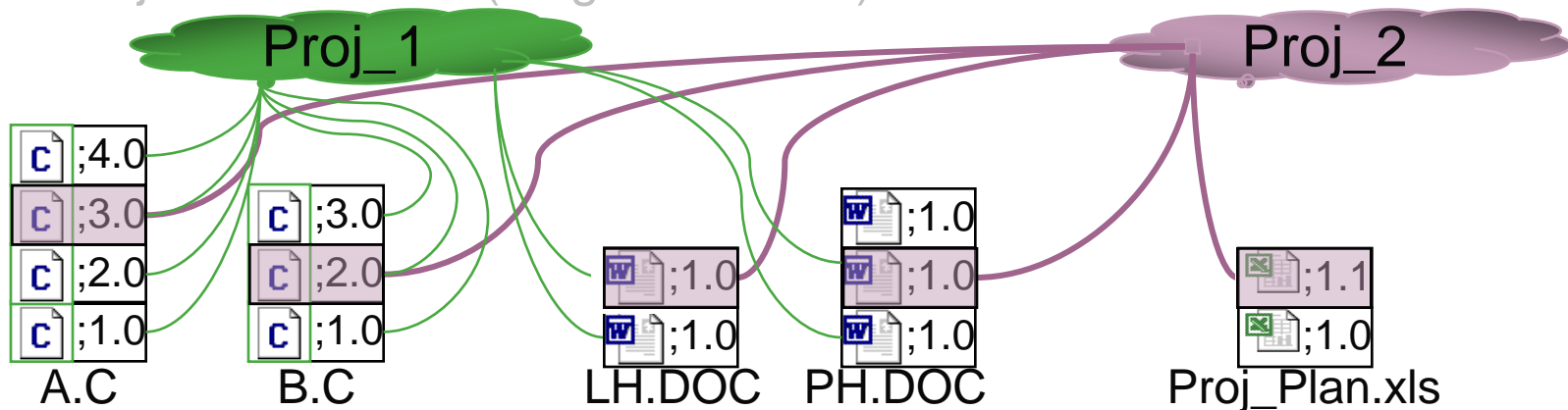
Projektsichten und -hierarchien

- In Projekten werden *Teilmengen aller Objekte* (K-Elemente) bearbeitet.
- Dies können sowohl *bestimmte K-Elemente* sein (z.B. nur die GUI-Sourcen, nur die Installationsdokumente, ...)
- oder auch nur *bestimmte Revisionen* bestimmter K-Elemente (z.B. beim Bugfix: nur die Sourcen, die in einer Auslieferung enthalten waren und die darauf gemachten Änderungen)
- Projekte sind also (eingeschränkte) *Sichten* auf die K-Elemente



Projektsichten und -hierarchien

- In Projekten werden *Teilmengen aller Objekte* (K-Elemente) bearbeitet.
- Dies können sowohl *bestimmte K-Elemente* sein (z.B. nur die GUI-Sourcen, nur die Installationsdokumente, ...)
- oder auch nur *bestimmte Revisionen* bestimmter K-Elemente (z.B. beim Bugfix: nur die Sourcen, die in einer Auslieferung enthalten waren und die darauf gemachten Änderungen)
- Projekte sind also (eingeschränkte) *Sichten* auf die K-Elemente



Projektsichten und -hierarchien

Die K-Elemente sollen dabei tunlichst

- *nicht kopiert* werden,
- *nur referenziert*

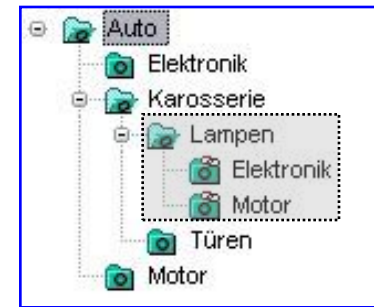
- Projekte können auch *Hierarchien* von Projekten und Sub-Projekten sein
- Besonders interessant: Zugelieferte Software oder Bibliotheken werden in einem bestimmten Stand (*Baseline*) vom Projekt referenziert
 - nicht die neueste Version, sondern eine bestimmte!

- Eine essenzielle Anforderung an Konfigurationsmanagementsysteme ist, *Versionen* oder *Stände* (von Konfigurationen oder Teilsystemen) bilden zu können, sog. *Baselines*
- Baselines sind Stände eines *gesamten Pakets* von Information (im Gegensatz zu Revisionen = Stände von einzelnen Dateien) und müssen
 - die relevanten Objekte *zusammenbündeln*
 - und diese Objekte *einfrieren*
- Baselines ermöglichen *schnelle exakte Rekonstruierbarkeit* ganzer Stände und damit *Revisionssicherheit*



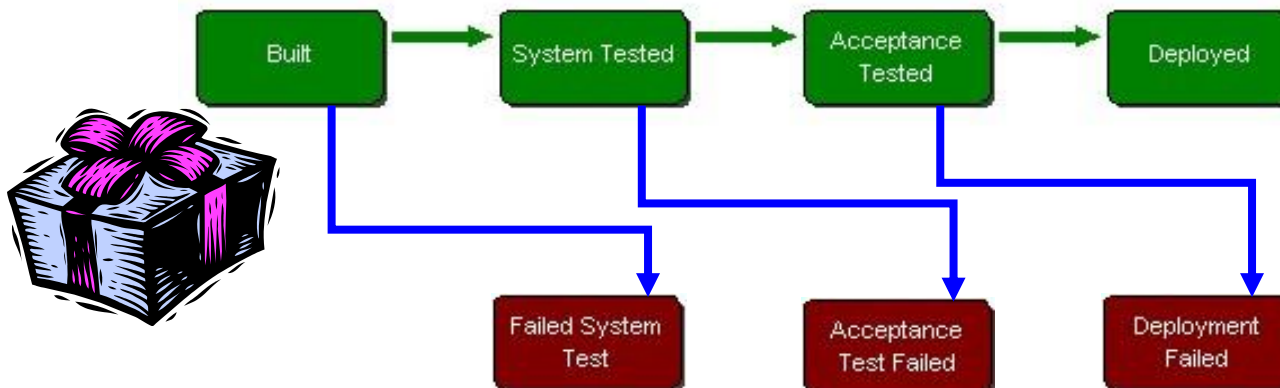
Weitere Anforderungen an Baselines (1):

- Baselines müssen *auf jeder Ebene* gebildet werden können:
 - System
 - Subsystem
 - Modul
- Baselines müssen *automatisiert* gebildet werden können
- Baselines müssen aus *Projektsicht* (Mainstream-Entwicklung, Bugfix, Wartung) gebildet werden können
- Nur *bestimmte Typen* von Objekten sollen in eine Baseline kommen (*konfigurierbar!*)
- Nur *reife Objekte* (Workflow) dürfen in eine Baseline kommen – was „reif“ bedeutet, muss konfigurierbar sein
- Die Bildung einer Baseline muss jederzeit *nachvollziehbar* sein



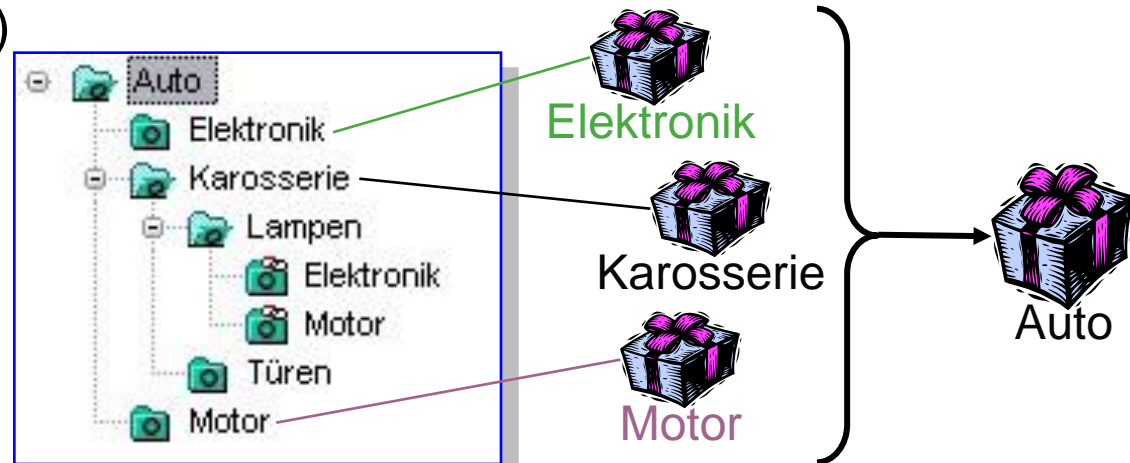
Weitere Anforderungen an Baselines (2):

- Baselines führen dann das Leben der Einzelobjekte weiter und *durchlaufen als Ganzes* die verschiedenen *Test- und Integrationsschritte (-ebenen)*
- Daher braucht ein Konfigurationsmanagementsystem *Workflows auch (und gerade) auf Baseline-Ebene*



Weitere Anforderungen an Baselines (3):

- Es muss möglich sein, *Baselines zusammenzuführen* (Merge auf Paket-Ebene)



- Um Bugfixes schnell initiieren zu können, muss *aus einer Baseline* jederzeit ein *neues Projekt instanziiert* werden können. Damit ist es möglich, genau den früher fixierten Stand wieder aufleben zu lassen („*Reinkarnation*“)

Labeling bietet nicht alle diese Merkmale!

- insbesondere nicht die Anforderungen an
 - Auswahl über *Reifegrad* der K-Elemente
 - *Typ-Gebundenheit* der K-Elemente
 - *Workflow* auf Paket-Ebeneund häufig auch nicht die Anforderungen an
 - *Freezing* der Einzelobjekte
 - *Automatisierung* bei der Bildung

**Konfigurations- und
Änderungsmanagement
Konzepte und Überblick**

Änderungsmanagement

Hans-Joachim Erchinger
HJErchinger@serena.com

Änderungsmanagement

- *Change Control* gibt Antworten auf diese Fragen:
 - auf *Versionierungsebene*:
 - wer darf ein K-Element ändern
 - wer darf neue K-Elemente anlegen
 - auf *Projektebene*:
 - wer darf Projekte instanziiieren
 - auf *Baselineebene*:
 - wer darf Baselines bilden
 - wer darf Baselines zusammenführen („*mergen*“)
 - wer darf Baselines freigeben
 - auf *Planungs- und Organisationsebene*:
 - wer darf KM-Pläne ändern
 - wer darf Berechtigungen definieren und ändern
 - ...

Änderungsmanagement

Change Control, at the version control level, deals with controlling who may make changes to a file, and how it's version tree may evolve.

A versions control tool may do a good job at Change Control for a given file, but the problem is that there is a bigger picture that needs to be tied together.

And data is generally stored at the file level making it harder to evaluate the bigger picture.

Jim Hickey: Version Control vs. Configuration Management

<http://www.cmcrossroads.com/content/view/6786/120/>

Änderungsmanagement

Die *Change History* muss automatisiert vom KM-System erzeugt werden und darf *nicht manuell änderbar* sein:

- wer hat
 - *wann*
 - *was*
 - *woran* (an welchen Objekten, Sourcen, Dokumenten, ...) *geändert?*
- Hierzu gehören:
 - *Time-stamping*
 - *User-stamping*
 - *Change Records auf jeder Ebene:*
K-Elemente, Baselines, Projekte

Änderungsmanagement

Change Management steuert hingegen den gesamten Änderungsprozess:

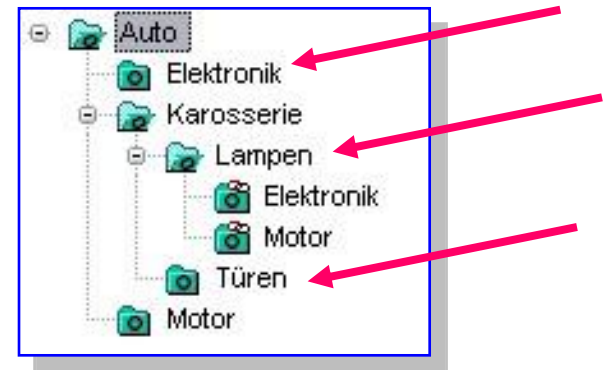
- *Wer soll* eine Änderung durchführen?
- *Wer ist in welcher Rolle* involviert
 - Bewerter = CCB,
 - Ausführer = Entwickler / Dokumentierer,
 - Tester
- *Warum*: Was ist das Ziel der Änderung?
- *Woran*: Welche Objekte *sollen* geändert werden?



Änderungsmanagement

Es muss jederzeit erkennbar sein,

- *welche* Objekte
- aufgrund welcher *Anforderungen* (Aufträge)
- *von wem*
 - zu ändern sind
 - geändert wurden
- auf welches Modul, Subsystem, ... sich die Änderung bezieht
- und in welchem *Stadium* sich die Änderung aktuell befindet (Workflow)



Änderungsmanagement

Change Management lebt also vom Workflow!



Dimensions Change Request

CR SUMMARY

Project Name: QARUS CR Title: TEST
 CR Number: 14.0001.00.01 Create Date: 14.07.2008
 Requester: Pils/May Date: 2007

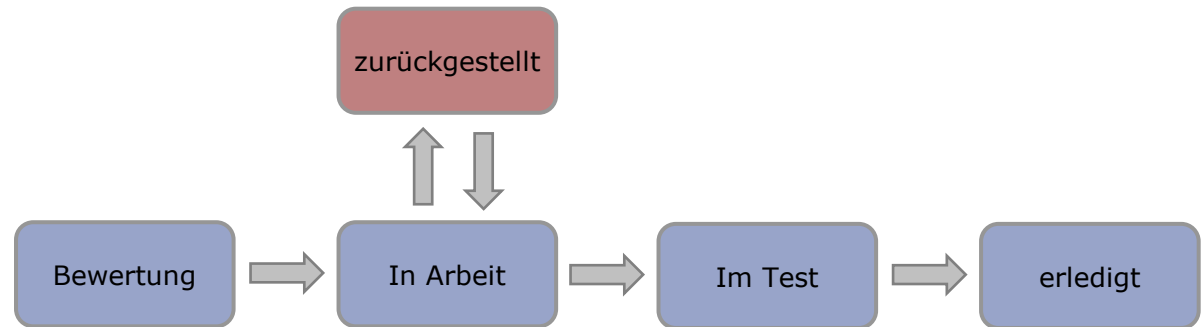
CHANGE DETAILS

Change Description: The Bank's backfile system will automatically capture application data with complete relational record files.
 Severity: 1

Change Effort		
Planned Work Effort	Estimated Development Effort	Actual Development Effort
14.07.2008 09:00	0	0

NOTES

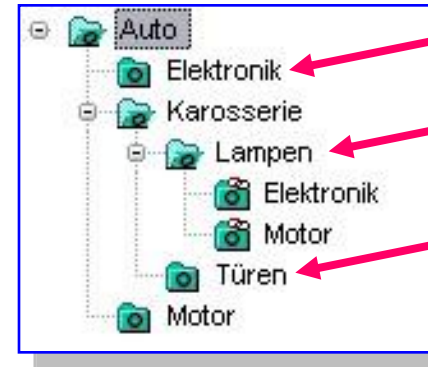
Test Passed
 Solution
 Reported Problem
 Reported Error Message
 Other Comments
 Action Description



Änderungsmanagement

Change Management

- Änderungen sollen *nur möglich* sein, wenn *Aufträge* dazu vorhanden sind
- Diese Restriktion darf aber nicht für alle Objekte gelten, muss vielmehr auf Basis der *Objekt-Typen* definierbar sein
- Die *Aktoren* (Bearbeiter) von Änderungen sind für verschiedene *Bereiche* (Subsysteme, Module) unterschiedlich, je nach Teilprojekt-Zugehörigkeit und Befähigungen (*Skills*)



Auftrags-basiertes Arbeiten

- Beim Auftrags-basierten Arbeiten steht nicht mehr die (zufällig wirkende) Initiative eines Einzelnen am Anfang,
- vielmehr ist es die Änderungsanforderung, die Aktionen an KM-Objekten auslöst.
- Es ist also nun der *Änderungsauftrag*, der die Änderungen an den betroffenen K-Elementen *initiiert*.

Auftrags-basiertes Arbeiten

Dennoch darf hier nicht schwarz-weiß gedacht werden:

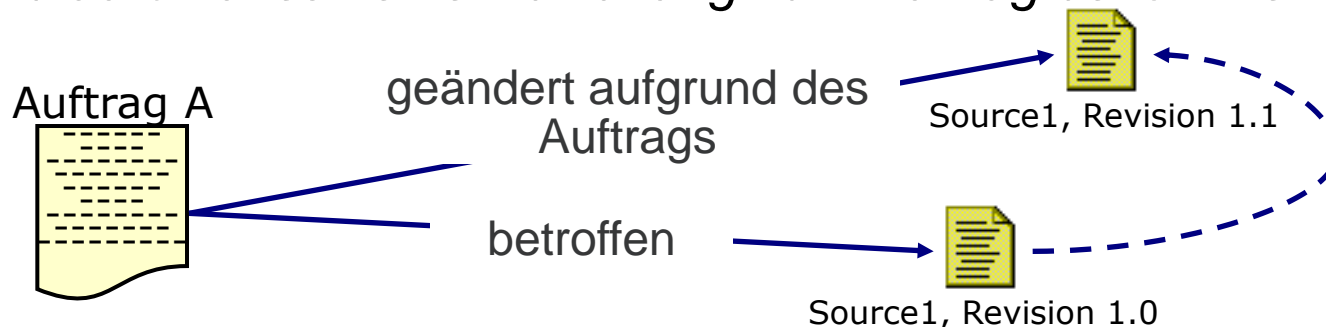
- Nicht alle Änderungen müssen auftrags-basiert durchgeführt werden,
- Änderungen z.B. an Notizen müssen *auftrags-los möglich* sein
- Auftrags-basiertes Arbeiten muss also *Typ-spezifisch* möglich sein

- Es muss möglich sein,
 - sowohl *zwanglos Auftrags-basiert* zu arbeiten
 - als auch *zwingend* Änderungen nur noch *Auftrags-basiert* zuzulassen
 - ggf. sogar beide Formen für unterschiedlich kritische Objekttypen *parallel*

Auftrags-basiertes Arbeiten

Dazu ist es dann auch nötig,

- *Bezüge* zwischen dem *Auftrag* und den *betroffenen K-Elementen* herzustellen
- *unabhängig* davon, *wer* diese Bezüge herstellt
 - der Auftraggeber = Projektleiter (*direktiv*)
 - oder der Ausführende = Entwickler (*dokumentierend*)
- dass die aufgrund eines Auftrags *entstandenen* neuen *Revisionen* von K-Elementen als solche *erkennbar* sind
- und *automatisch* eine *Beziehung* zum *Auftrag* bekommen



Auftrags-basiertes Arbeiten

Haben wir das Auftrags-basierte Arbeiten erreicht,
können wir davon „ernten“:

In einer der Definitionsfolien (zum Thema Konfigurationsüberwachung) haben wir
schon gesehen:

Referenzobjekt für das Änderungsmanagement ist eine
Bezugskonfiguration.



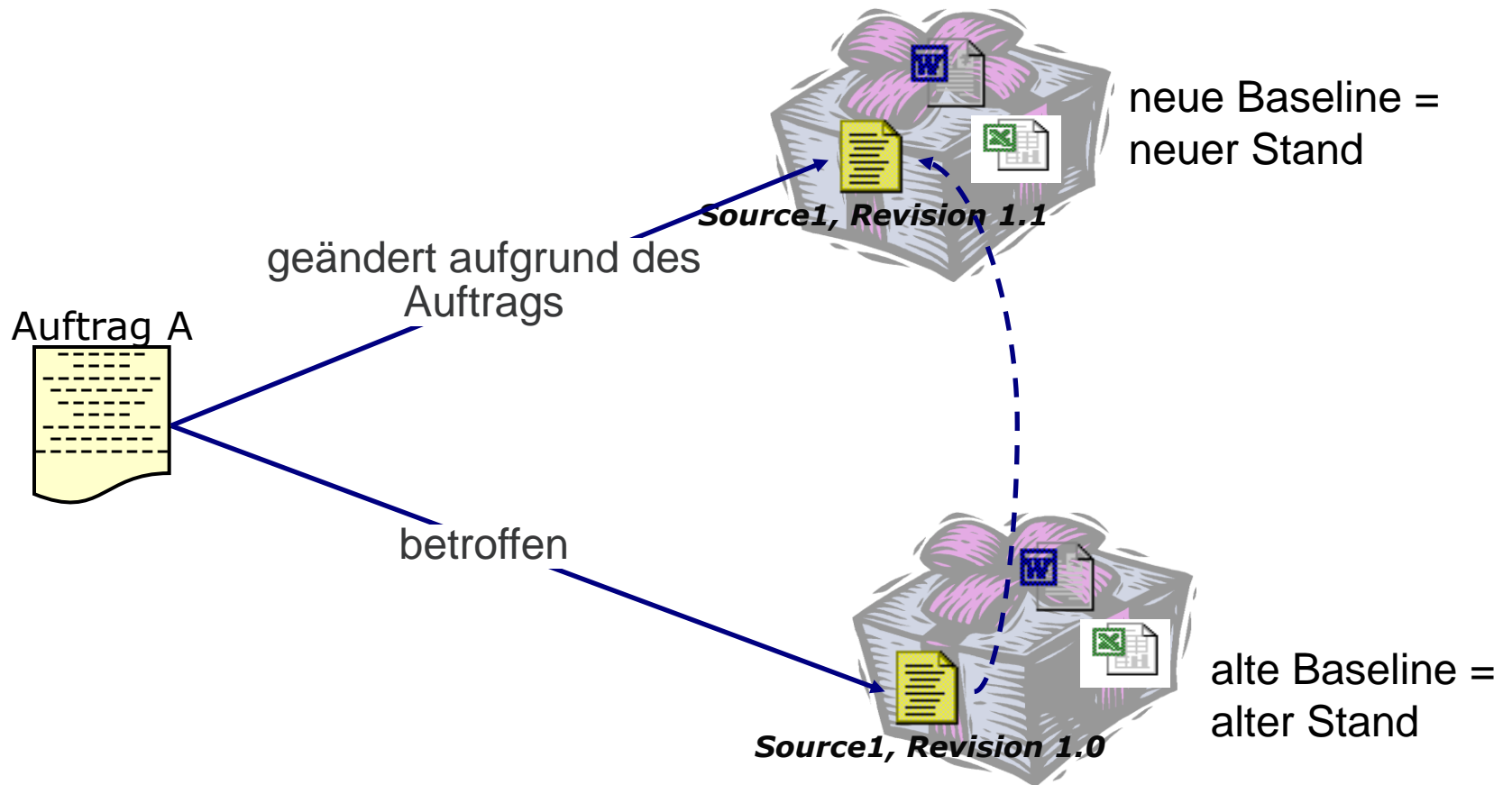
Zu einem bestimmten Zeitpunkt stellt diese *zusammen mit
allen bis dahin freigegebenen Änderungen die gültige
Konfiguration* dar..



<http://de.wikipedia.org/wiki/Konfigurationsmanagement>

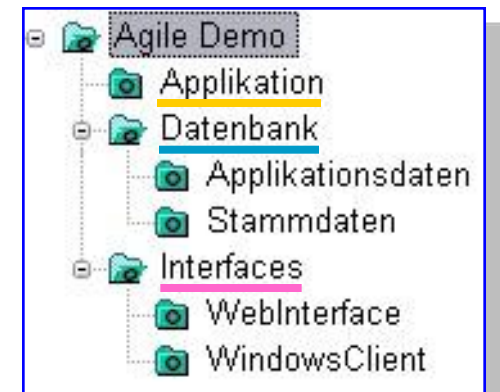
Auftrags-basiertes Arbeiten

Wir finden also Änderungsmanagement auf unterer wie auch auf höherer Ebene:



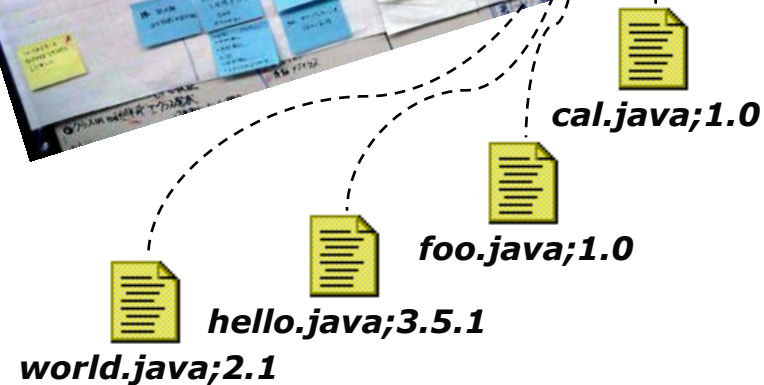
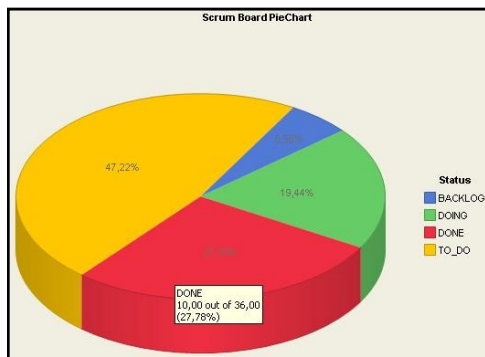
Tatsächlich ist damit auch bereits die Grundlage für agiles Arbeiten gelegt:

- es gibt einen Pool (*Scrum-Board*) von Aufträgen (*User-Stories = Post-its*)
- jeder Entwickler nimmt sich einen oder mehrere dieser Aufträge und implementiert ihn
- dabei verknüpft er die geänderten Sourcen mit der User-Story (oder Sprint)
- am Ende eines Sprints ist klar erkennbar, was realisiert wurde und was noch offen ist
- User-Stories können bestimmten Bereich zugeordnet sein
- es ist ein Workflow vorhanden :
ToDo -> Doing -> Done



Am Ende eines Sprints ist dann sofort klar erkennbar:

- welche User-Stories bereits erledigt sind (*done*)
- über die Beziehung der geänderten Sourcen zu ihren User-Stories wird klar, was in den Build geht
- die noch offenen User-Stories (*ToDo*) kommen in den Sprint-Backlog



Schlusszitat



Das KM hat die nicht ganz einfache Aufgabe im heutigen dynamischen Umfeld die Lebensfähigkeit von Produkten, ja gar von ganzen Unternehmungen zu meistern.

KM bekommt die vielschichtige Dynamik der Softwareentwicklung in den Griff.

Im Weiteren wirkt KM als multipler Integrationsfaktor.

KM bildet dabei als Dach über Entwicklung, Integration, Implementierung und Betrieb während des gesamten Lebenszyklus eines Produktes.

Unternehmen, die Konfigurationsmanagement so verstehen und umsetzen, sichern sich die Leistungsfähigkeit und bleiben wettbewerbsfähig.

Lebensfähigkeit bedeutet in diesem Zusammenhang nicht nur zu überleben, sondern auch wachsen und vermehren.

*MANFRED SAYNISCH, DIETMAR LANGE, 3. Fachtagung „Konfigurationsmanagement“, Änderungsmanagement mit System – Schlüsselfaktor Konfigurationsmanagement, 1999, GPM Deutsche Gesellschaft für Projektmanagement Nürnberg.
Referat Dipl.-Ing. Manfred Saynisch „KM als Schlüsselfaktor – Virtuelle Produktentwicklung“*



Sie möchten mehr wissen?

Kommen Sie zu uns an den Stand!

Reden Sie mit uns!

Danke

Konfigurations- und Änderungsmanagement Konzepte und Überblick

Hans-Joachim Erchinger

HJErchinger@serena.com

*und alle, die sehen
wollen, mit welchem
KM-Tool man auch
Kleider-Management
(KM) machen kann ...*