



Konfiguration

Kern komplexer Projekte und ein Produktziel

CMConf 2009

*Günter Dörfler, Nürnberg
gd.cmm@t-online.de*



Übersicht

- Scope und Ziel
- Zweck von CM, Begriffe
- Eigenschaften eines Produktes/Projekt
- Zusammenfassung

- Anhang



Scope

- Software Projekte und Produkte
- Versionieren und Konfigurieren
- Datenmanagement und Verteilung
- Change Management
- Build & Integration

≠ CM Werkzeuge

≠ Konfiguration von Kundenanlagen und Netzwerken



Ziel

- Faktoren mit Einfluss auf CM kennenlernen
 - Zusammenhang Projekt – Produkt
 - System-Architektur
 - Art des Produktes
 - Ziele des fernen Partners

- Anregungen zur CM-Instanziierung
 - Repositories
 - Versionsbildung
 - Feature- und Change Management
 - Verteilung und Integration
 - Kriterien für Annahme/Ablehnung von Ergebnissen

- Diskussion
 - Rollenverteilung
 - Automatisierung
 - Datennormierung



Zweck eines CM

- Vielfalt der Ergebnisse
 - Identifizierbarkeit, Eindeutigkeit sicherstellen
 - Datenmengen systematisch verwalten (automatisieren)
 - Reproduktion und Rückverfolgung ermöglichen
 - Evolution und Änderungen erfassen und sichern
 - Wiederverwendung auf verschiedenen Ebenen
- Kundenaspekte
 - Schnelle Versorgung mit gezielten Korrekturen
 - Produktfamilien durch hohen Grad an Wiederverwendung kostengünstig und in passenden Zeitspannen pflegen
- Besondere Eigenschaft: **CM lebt viel länger als ein Projekt**

Configuration Management ist eine zum Projektmanagement gehörende Disziplin, welche sich auf viele beteiligte Abteilungen auswirkt, wie Entwicklung, Qualitätssicherung, Integration und Test sowie Wartung und Service.



Begriffe

- Repository:
Datenbasen eines Projektes, auch außerhalb von „CM-Tools“
- Configurable Items (CI):
 - Element, Modul, Subsystem, Package, Component, System
(auch 3rd Party und OEM Zulieferungen)
 - Das Produkt (aus Kundensicht, inkl. Dokumentation)
 - Dokumentation (interne und für Kunden)
 - Test Tools, Testdaten, Testergebnisse (Regression)
 - Arbeitsumgebung: Werkzeuge
- Change Management: Gezieltes/kontrolliertes Ändern
- Build Management (inkl. Regeln und Q-Datenerfassung)
- Release Management
- Maintenance: Ablauf, wie das Produkt für den Kunden gepflegt wird



Projektszenario

- Einbettung:
 - Projekt innerhalb einer Produktfamilie
 - Teilprojekt, mit Teilfunktionen
 - Autarkes Projekt mit vielen verschiedenartigen Kunden (unterschiedliche Interessen zu verschiedenen Zeiten)
- Produktarten:
 - Produktfamilie: Versionsvielfalt, lange Wartungsperioden, Kompatibilität
 - Singuläre Produkte: Einzellösungen, Wartung nur durch aktuellere Version
- Ressourcenaspekt
 - Know-How Schwerpunkte, aber nicht zersiedelt (günstig)
 - Völlige Verteilung, kein Know-How Schwerpunkt (extrem hoher Kommunikations- und Synchronisationsaufwand)
 - homogene Technik, etablierte, aber unterschiedliche Arbeitsweisen?



MultiSite

- Beginnt oft schon im Nachbarbüro ...
... „das ist mein Produkt, meine Idee“
- Kann sich der Partner mit dem Auftrag identifizieren, ist er motiviert?
- eigenes Budget, Bericht an zwei Chefs?
Standortsicherung
- (Teil-)produkt aus anderem Kontext
(eigener Markt, eigene Roadmap)?
- Fremdfirmen
- Kulturkreise
- Zeitzonen



Topologie

■ Repository für Projektdaten

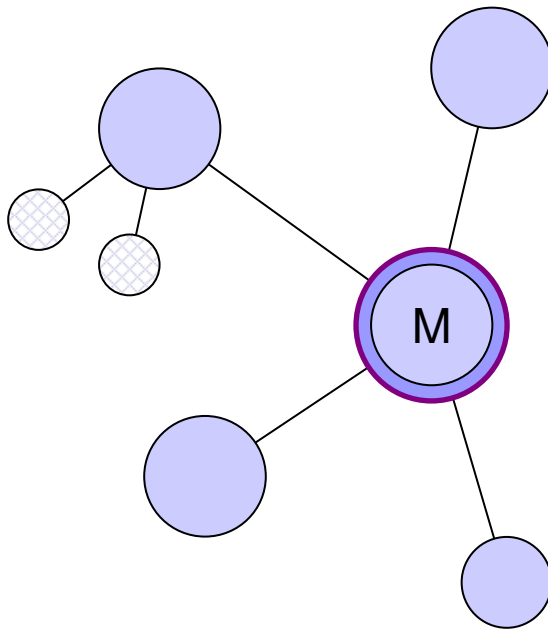
- STERN: Zentraler Master und Replikate, weitestgehend mit homogener Umgebung und sehr ähnlicher Struktur; Update ist spätestens nach zwei Transaktionen abgeschlossen
- SATELLIT: Lokales Mikro-CM mit eigenen Regeln und definierter Übergabe (Export/Import und ggf. Mappen von Strukturen). Kommt auch vor bei der Verbindung zweier Sterne aus unterschiedlichen Projekten oder bei externen Partnern.
- PEER: kommt gerade für kleinere Projekte in Frage. Es gibt viele Pfade für den Update, alle auf dem gleichen zu Stand halten ist aber schwieriger
- CHAIN: ist praktisch nicht mehr im Einsatz

Aber: je mehr Kopien der Daten existieren, desto komplexer wird das Datenmanagement (besonders bei konkurrierender Bearbeitung) für Integration, Merge, Updates, ChangeControl, Ownership

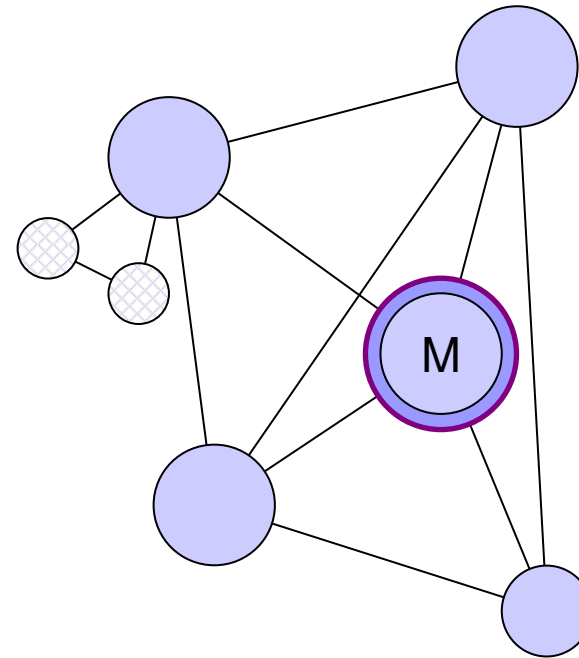
■ Change/Feature Datenbank

- Empfehlung: Nur eine zentrale Datenbank je Projekt, ggf. mit entsprechenden VIEWS auf die Daten bzgl. Teilprojekten oder Teilaufgaben (Standort). Die geringen Datenmengen bei Transaktionen erlauben den direkten Zugriff weltweit.

Topologie



Stern mit Satelliten



Peer-to-Peer



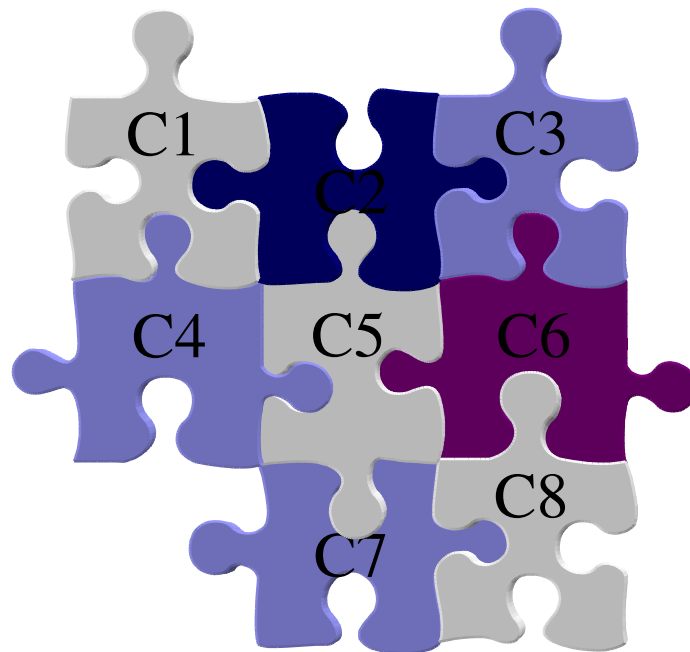
Datenreplikation

- Grad der Datenspiegelung
 - Voll synchronisiert (gespiegelt): alle Standorte können jede Aufgabe übernehmen, großer Ressourcenbedarf
 - Selektive Synchronisation: für Teilaufgaben oder zum Schutz von Produktteilen
 - Lose gekoppelte Systeme: für Zulieferungen eigenständiger Produktteile und weniger häufigem Datenaustausch, wenige und sehr stabile Schnittstellen
 - Third Party / OEM Produkte werden i.d.R. über vereinbarte Importmechanismen in das Projekt eingebracht.

Oft beschränken Lizenzbedingungen und Sicherheitsaspekte den Datenaustausch zusätzlich.

Architektur & Konfiguration

Das Produkt – ein Puzzle



- Der Systemarchitekt ist mit einer der ersten Gestalter
- Die Komplexität wird u.a. auch durch die Architektur bestimmt
- Konfiguration: Kann man Teile isolieren und durch ein andere ersetzen, ohne das System zu (zer-)stören?
- Kann man ein anderes Teil (eine andere Zulieferung) einsetzen?
- Welche Teile bestimmen die Funktionalität für Varianten, wie ist die Wartung geplant?



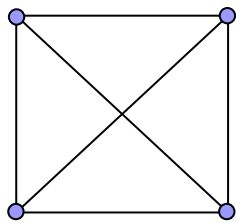
Zerlegung des Produkts

- **Abhängigkeits-Hierarchie**
 - Autarke Einheiten: alle an einem Ort?
 - Abhängigkeit zu volatiler Einheit an fremden Ort?
 - Keine zyklischen Abhängigkeiten!
 - Integration (Reihenfolge) ist definiert in der Architektur und entsprechend der technischen und funktionalen Abhängigkeit
 - Verwendung lokaler Basisklassen und Vererbungshierarchien

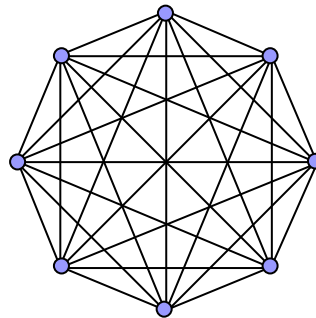
- **Ziel:**
 - Stabile Basis der jeweils anderen Sites wird lokal nicht verändert – nur genutzt
 - Interfaces werden immer bilateral synchron aktualisiert
 - Elementare CIs werden jeweils nur an einem Ort gepflegt (Verantwortlichkeit – Ownership, ggf. zerlegen)
 - Was soll in binärer Form (Library) den anderen Standorten zur Verfügung stehen?

Reduktion der Komplexität

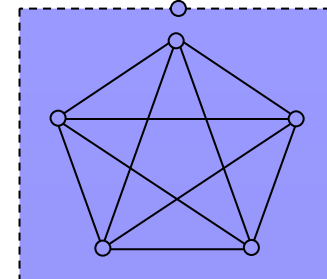
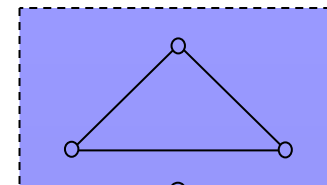
- Bildung von Subsystemen -



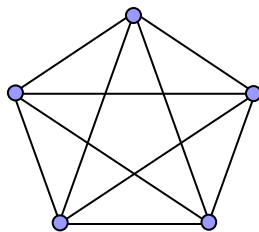
$$n = 4$$
$$b = 6$$



$$n = 8$$
$$b = 28$$



$$n = 8$$
$$b = 14$$

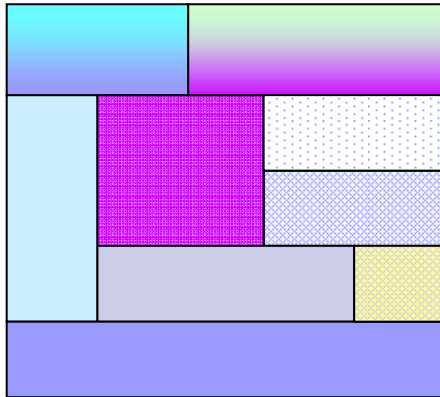


$$n = 5$$
$$b = 10$$

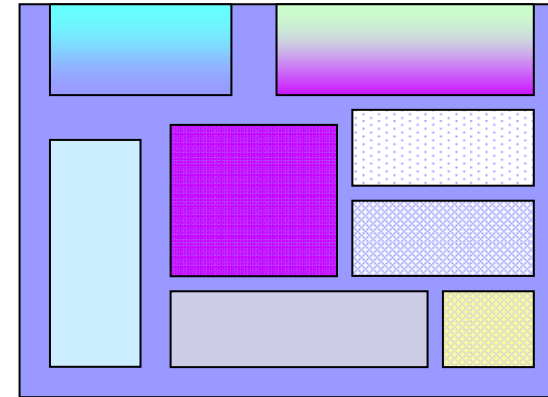
$$b = n \cdot (n-1) / 2$$

Reduktion der Komplexität

- Systemlayout in Schichten -



- Nur vertikale Interfaces
- Die Integration benötigt jeweils zwei Nachbarn
- Die Interfaces sind eindeutig und transparent
- Entwicklung ohne Gesamtsystem leichter möglich



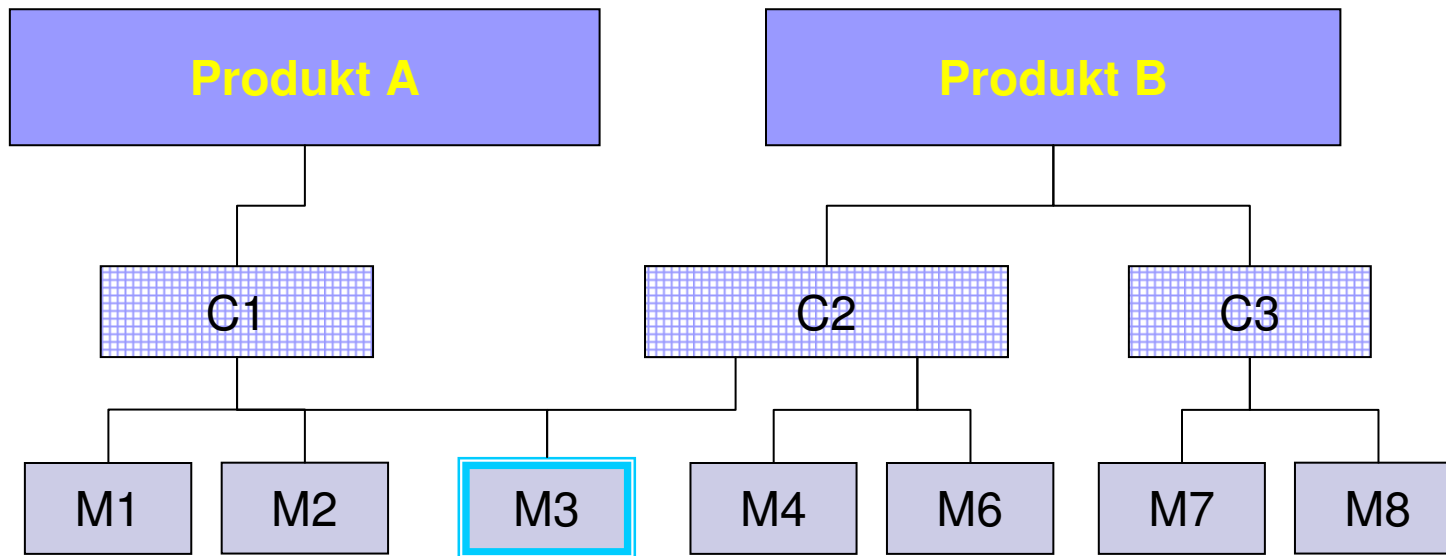
- Viele/alle Komponenten haben Schnittstellen zum Basissystem
- Integration benötigt das Basissystem
- Mehrfach verwendete Interfaces – indirekte Abhängigkeiten
- Gesamtsystem größer



Produkthierarchie

- Ablageordnung # Produkthierarchie
 - Ablage im Dateisystem nur auf untersten Ebenen brauchbar
- Höhere Ebenen nutzen die Basiselemente mehrfach, d.h. es gibt polyhierarchische Abhängigkeiten (Wiederverwendung)

Produktthierarchie



Konsequenz der polyhierarchischen Beziehungen:

- Ablage der Basiselemente Mx möglichst flach
- Beziehungen/Abhängigkeiten separat darstellen
- Produkt-Eigenschaften werden aus übergeordneten Komponenten geerbt
- Jede Ebene kann parallel, separat, unabhängig geändert, gebaut und getestet werden



Integrationsplan

- Festlegung der einzelnen Integrationsstufen mit der jeweiligen Verifikation (Tests)
- Methode: Welcher Teil bleibt als stabile Basis, was wird jeweils ausgetauscht.
- Die elementaren Austausch-/ Integrationssschritte werden auch später bei Korrekturen / Patches verwendet (minimaler Eingriff).



Build Prozess

- Build Regeln standortunabhängig
- Build an Komponenten orientiert
- Build Regeln erben von übergeordneter Komponente des aktuellen Produktes
- Komponenten tragen Buildinformation (Auswertung im Fehlerfall und zur automat. Sicherstellung der Kompatibilität)



Change Management

■ Die Änderung

- Wer fordert eine Änderung (intern/Kunde)?
- Was ist zu ändern (mit/ohne Interface)?
- Wer betreut die betreffende Komponente?
- Wann muss die Änderung verfügbar sein?
- Wie ist zu Ändern (gibt es Alternativen)?
- Kriterien zur Abnahme?
- Feedback und Lieferung an den Anforderer

■ Zentrale Datenbasis

- Regelmäßige, projektweite CCBs
- Eindeutige Kennzeichnung der Änderungen



Zusammenfassung

- MultiSite Projekt -

- Starten eines neuen Projektes
 - Alle Beteiligten haben eine ähnlich Situation
 - Einbeziehen der Rollen auf gleicher Ebene von Anfang an
 - Schulung/Training frühzeitig und gleichzeitig
 - Ansatz mit homogener Umgebung

- Hinzunehmen eines Standortes zum Projekt
 - Das bestehende Projekt nicht ändern
 - Eine nahtlose Integration über Schnittstellen und Vereinbarungen anstreben
 - Die Ressourcen des neuen Partners nicht auf den Kopf stellen, sondern sinnvoll nutzen



Zusammenfassung

- Projektstart mit CM -

- Ergebnisstruktur und Scope festlegen
 - Namensraum in der Produktlinie und für das Projekt
 - Produkthierarchie und Abhängigkeiten analysieren
 - Namensabbildung für existierende und fremde Teile
 - Attribute und Statuswerte projektweit definieren
 - Datenbasen vorbelegen (Nutzung durch Auswahl)

- Datenbasis/Repositories initialisieren
 - Technische Umgebung konfigurieren und in Betrieb nehmen
 - Sources, Buildrules, Tools
 - Zugänge und Zugriffsrechte gewähren
 - Initiale Spiegelung vornehmen



Zusammenfassung

- Projektstart mit CM -

■ Partner und Schnittstellen

- Zeitliche Raster für Ergebnisereignisse definieren
- Kriterien für die Akzeptanz für jede Ergebnisart vereinbaren
- Aufgabenverteilung für Registrieren und Weitergabe von Cis klären (veröffentlichen)
- Buildabläufe und Hierarchie klären
- Integrationsstufen festlegen
- Change Management: CCB und Abläufe einrichten
- Formate für Ergebnisse und Reports festlegen
CM liefert i.d.R. Rohdaten, nicht in Präsentationsform (z.B. CSV)

■ Probelaufe der Prozesse



Zusammenfassung

- Was CM nicht ist ...
 - Ein CM Tool
 - Versionsverwaltung
 - Wenn man make verwendet
 - Versionsnummern an Dateien/Verzeichnissen
 - Falls etwas im Langzeitarchiv liegt



Anhang

CM Plan für das Projekt/Produkt

- CM Organization
 - Responsibilities
 - Policies
 - Technical Resources
 - Data Exchange
- Configuration Identification
 - Data Structure
 - Naming Convention
 - Attributes
 - Versioning, Branching
- Configuration Control
 - Change Management
 - CCB
 - Correction, Patch Management
- Status Accounting
 - Reporting
 - Auditing
- Build Process
 - Build environment
 - Build control (make)
 - Target and Variant control
- Release Procedure
 - Internal releasing to integration and test
 - External releases (superordinated project, customer, service)
 - Publishing (Internet)
- Archiving
 - Development (sources, tools, environment)
 - Product (final packages as delivered to customers)



Anhang

Suche zu CM

Links im Internet zu CM Themen gibt es genug, oft in unterschiedlichen Zusammenhängen

■ CM Tools

- CVS, SVN, GIT (Linux), Bugzilla, Subversive, PVCS
- IBM Rational ClearCase, Telelogic Change, Synchronicity
- Produktdatenmanagement

■ Standardisierung, Prozesse, Training

- CMM & CMMI: systematische Prozessverbesserung
- CMII: <http://www.cmii.de/>
- ICM: <http://www.icmhq.com/>
- SEI: <http://www.sei.cmu.edu>
- ISO 10007:2003 Leitfaden zum Anwenden von Konfigurationsmanagement