

Ein Schema zur vereinfachten verteilten Software-Entwicklung auf Basis von Rational-ClearCase

Vortrag zur CMConf 2009, München

Horst Eckardt
Siemens AG
Corporate Technology
München



Software &
Engineering
Development
Techniques

Ein Schema zur vereinfachten verteilten Software-Entwicklung auf Basis von ClearCase

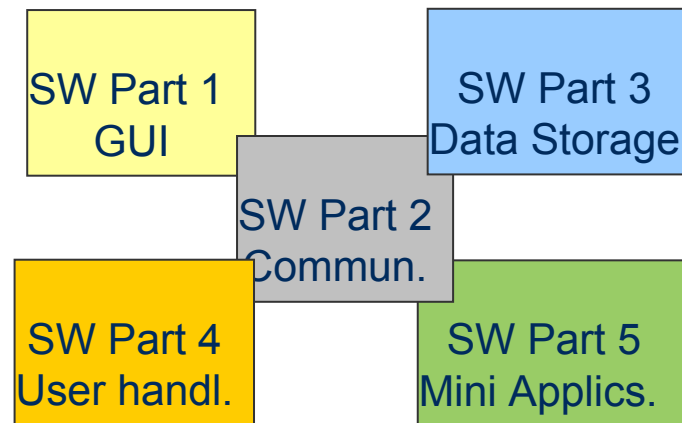
• Übersicht

- Probleme der verteilten Software-Entwicklung
- Konsequenzen für Entwicklungs- und Integrationsprozess
- Tool-Abhängigkeiten
- Parallele Entwicklung und Varianten-Handling
- Lösungen mit ClearCase
 - allgemeine Lösung
 - optimierte Lösung
- Zusammenfassung

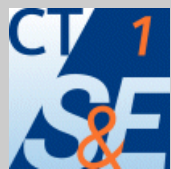
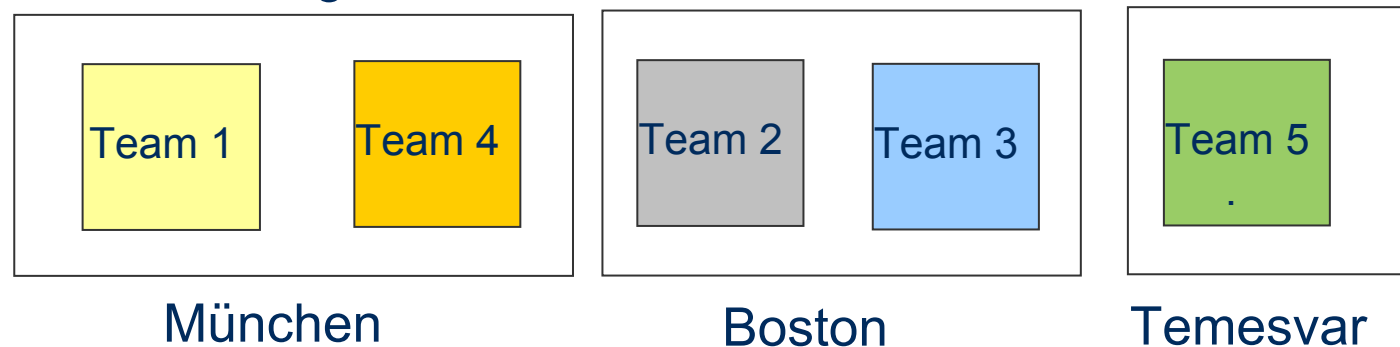


Problematik verteilter Software-Entwicklung

Software mit Komponenten-Architektur



Aufteilung in Entwickler-Teams und Standorte



Zu lösende Entwicklungsprobleme

- **Wer ist verantwortlich für SW-Parts?**
- **Wer definiert die Schnittstellen?**
- **Wie werden Änderungen in den Schnittstellen mitgeteilt?**
 - Eigener Prozess
 - Durch Programmiersprachen-Mittel
- **Wie wird entwickelter Code an andere Standorte verteilt?**
 - Austausch per Email
 - Austausch per CM-Tool
 - Zentrales CM-Tool
- **Wie wird die Entwicklungsbasis definiert?**
 - Baselines, wie definiert?

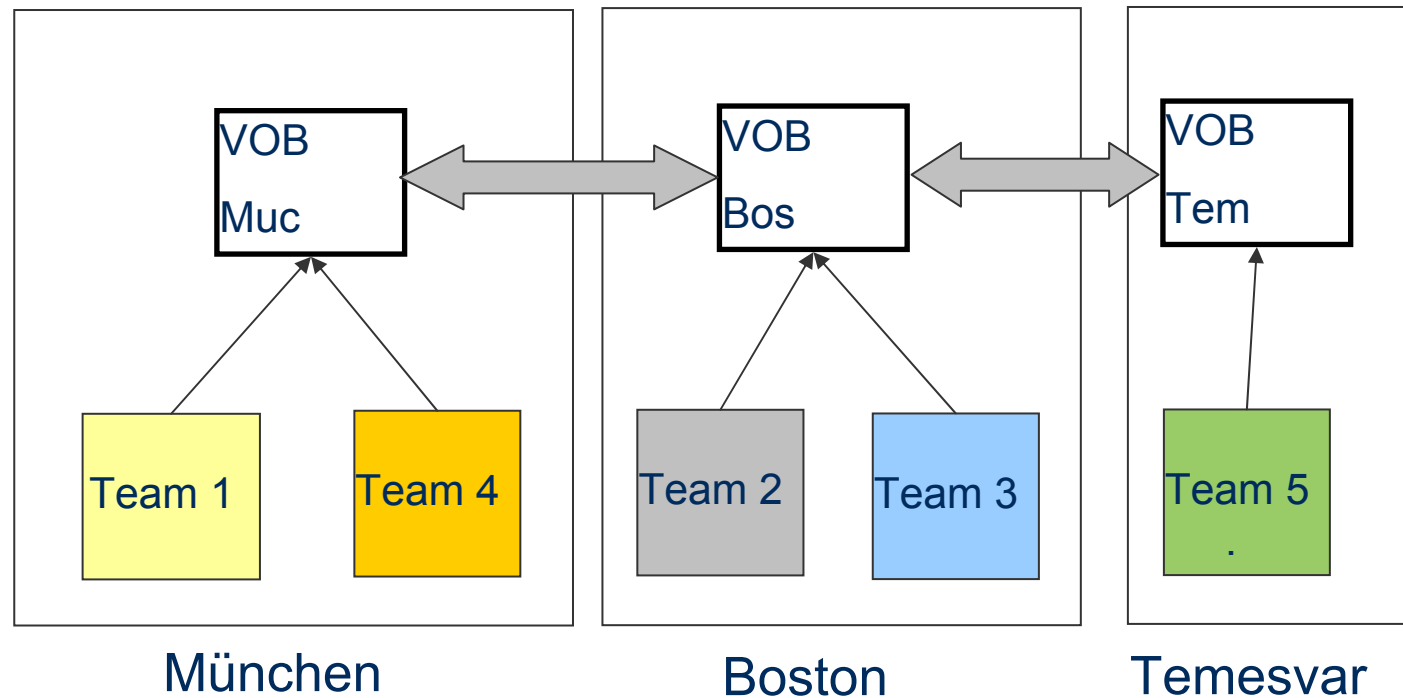


Integrationsproblem

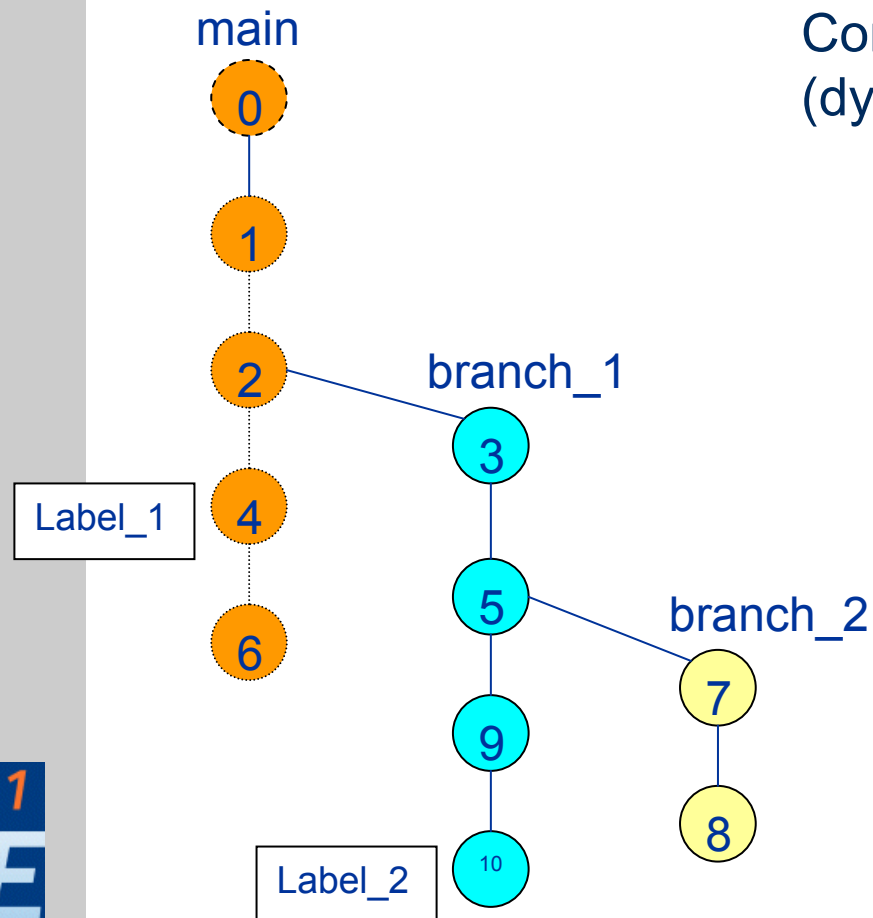
- **Entwicklungsergebnisse müssen zusammengeführt werden**
- **Varianten müssen unterscheidbar bleiben**
- **Releases müssen erstellt werden**
- **Releases und Baselines müssen an Entwicklerstandorte zurückpropagiert werden**
- **Änderungsmanagement muss integriert sein**



Eine Tool-Lösung des Problems der verteilten Entwicklung: Rational ClearCase



Arbeitsweise von ClearCase



Config Spec
(dynamische Selektion der Versionen)

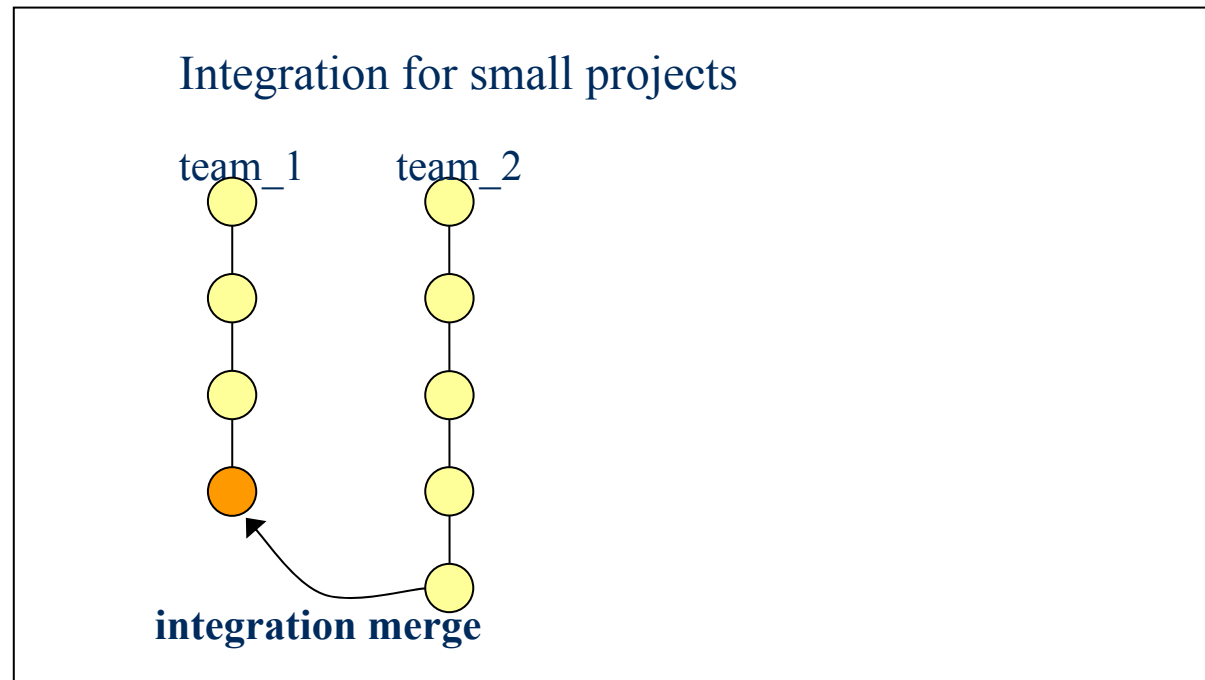
```
element * Label_1
element * ../branch_1/LATEST
element * /main/LATEST
```

Versuibsbaum pro Element
(Datei oder Verzeichnis)



Software &
Engineering
Development
Techniques

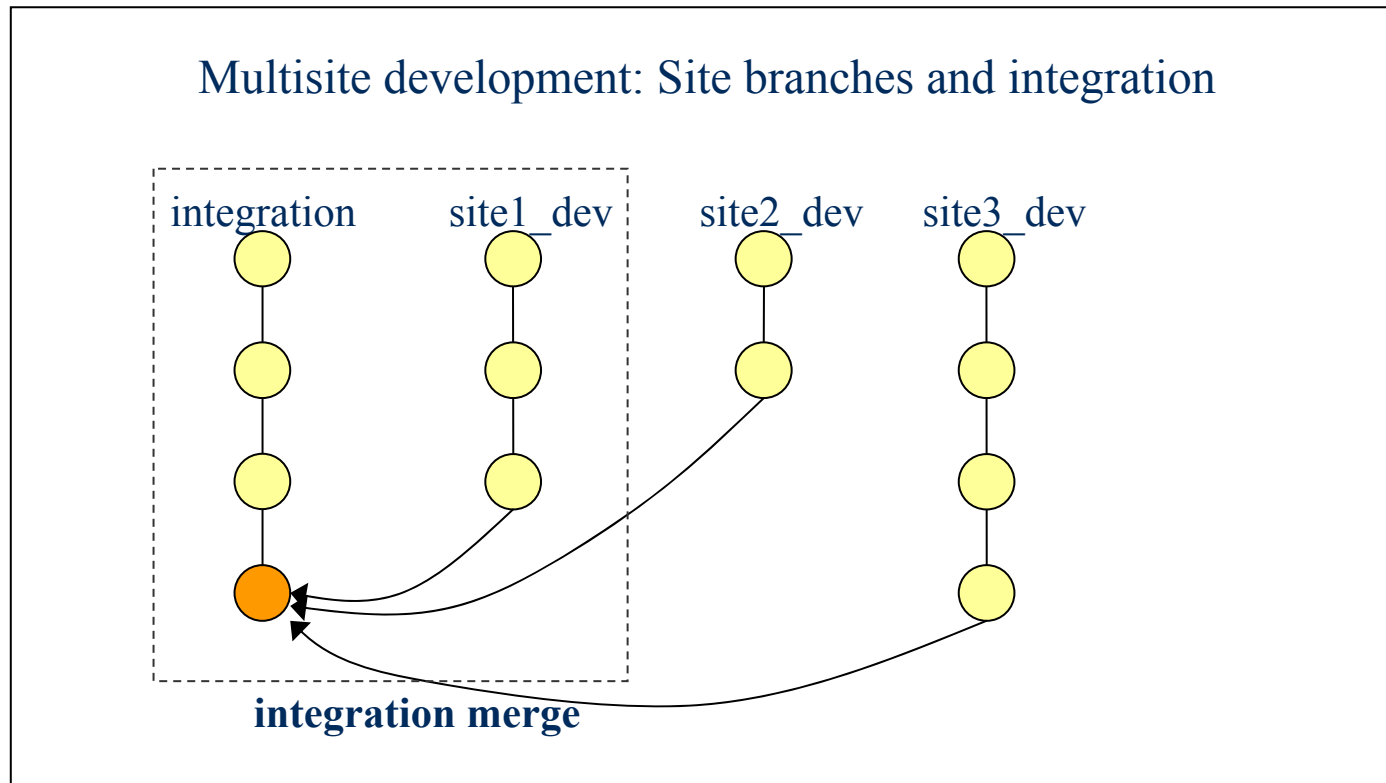
Organisation der Entwicklung in Entwicklerteams



Entwicklung auf Branches



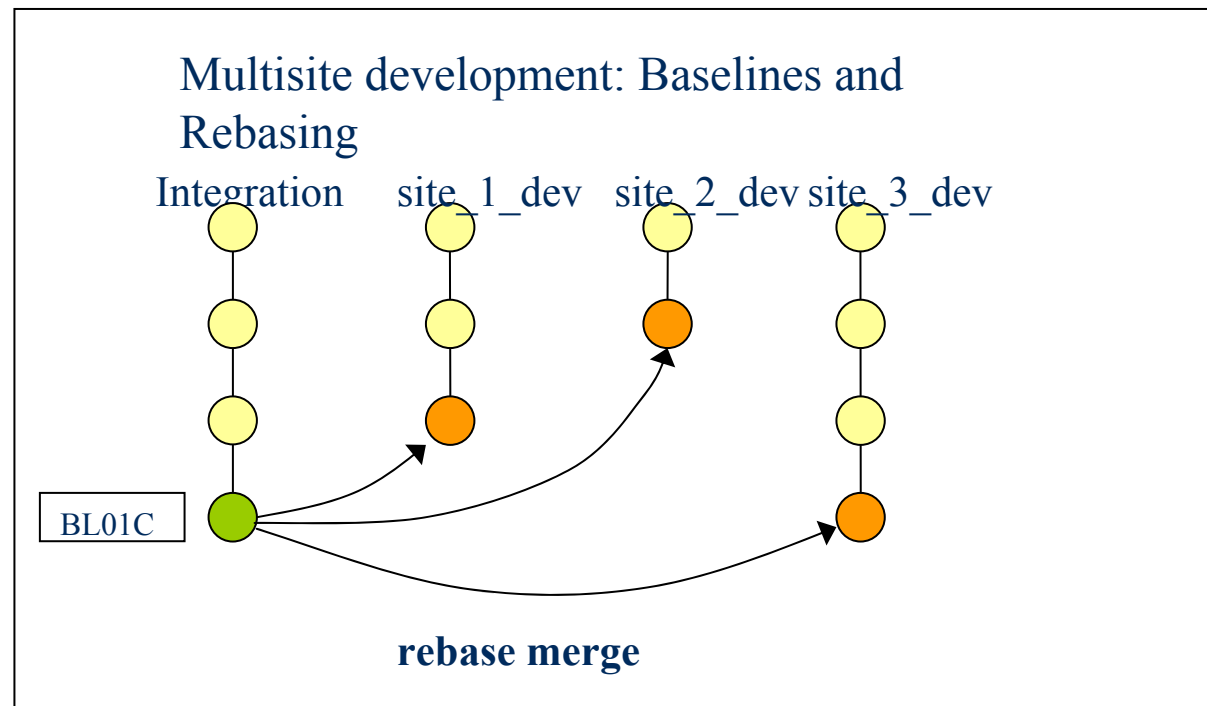
Organisation der Entwicklung über Standortgrenzen, mit Integration



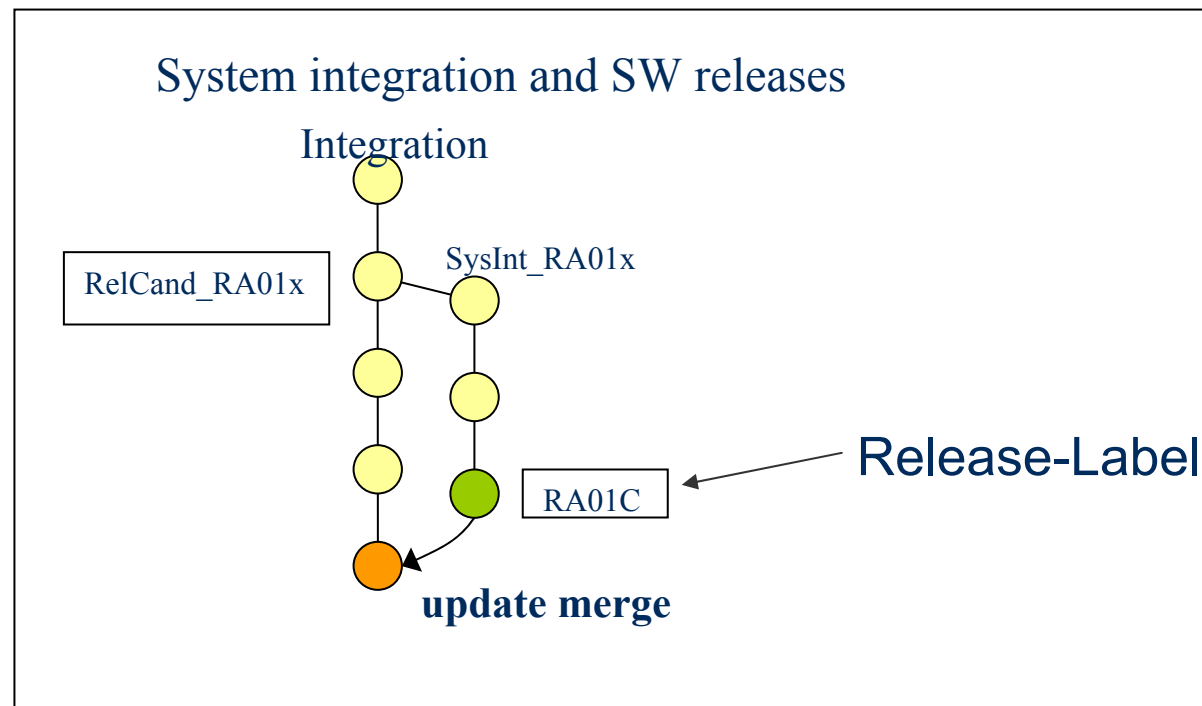
ClearCase: jeder Standort ist Eigentümer seines Entwicklungs-Branches



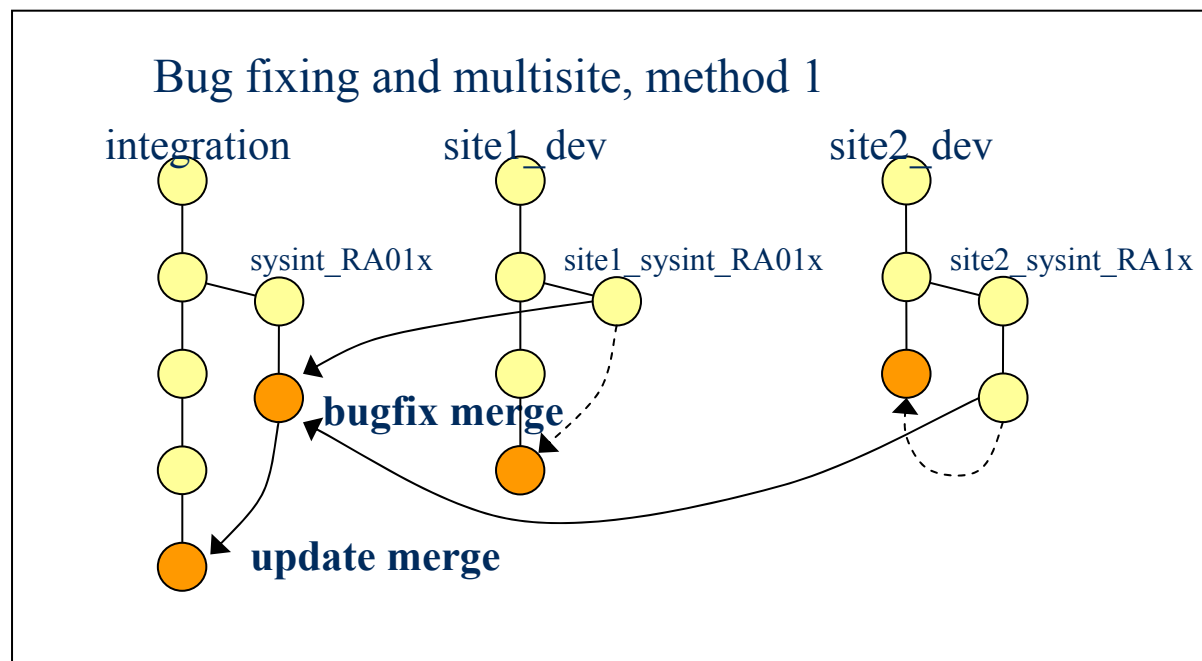
Entwicklungs-Workflow, Rebasing



Systemintegrations-Branch für SW-Release-Vorbereitung



Systemintegration, Fehlerkorrektur, Methode 1

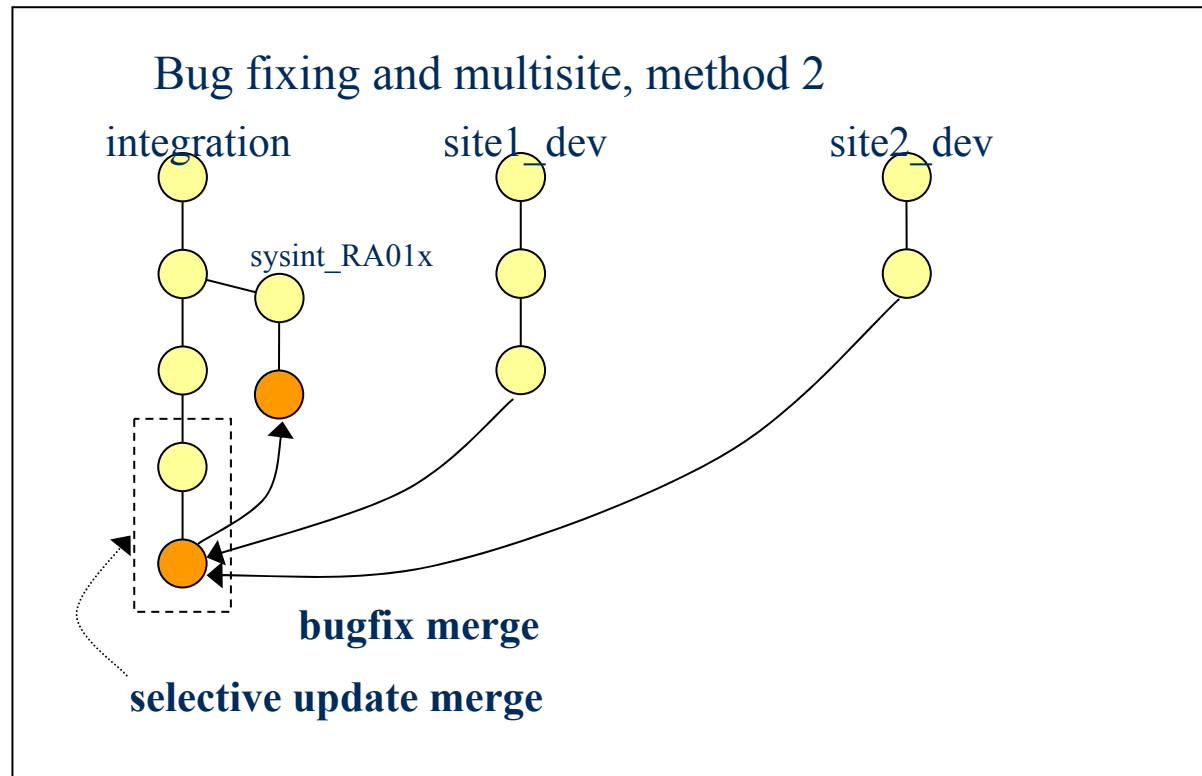


Lokale Integrations-Branches



Software & Engineering Development Techniques

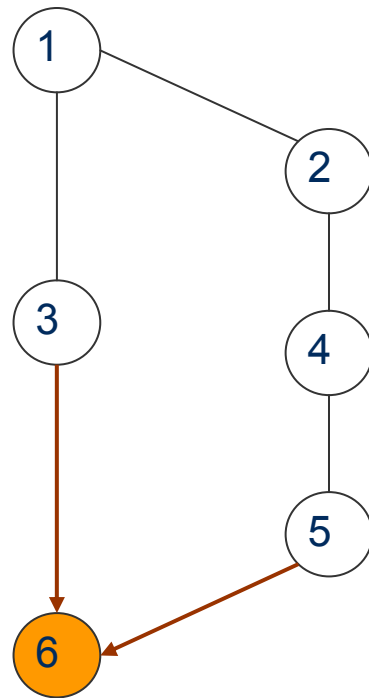
Systemintegration, Fehlerkorrektur, Methode 2



Keine lokalen Integrations-Branches
 Selektiver Update-Merge



2-Wege- und 3-Wege-Merge

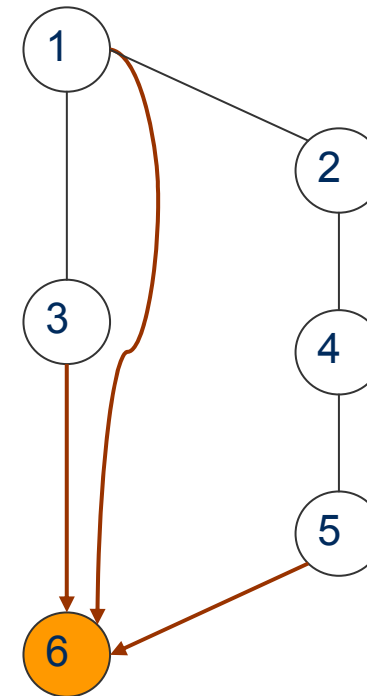


2 Wege (kein gemeins. Ursprungselement)

a=3

a=4

a=?



3 Wege

a=3

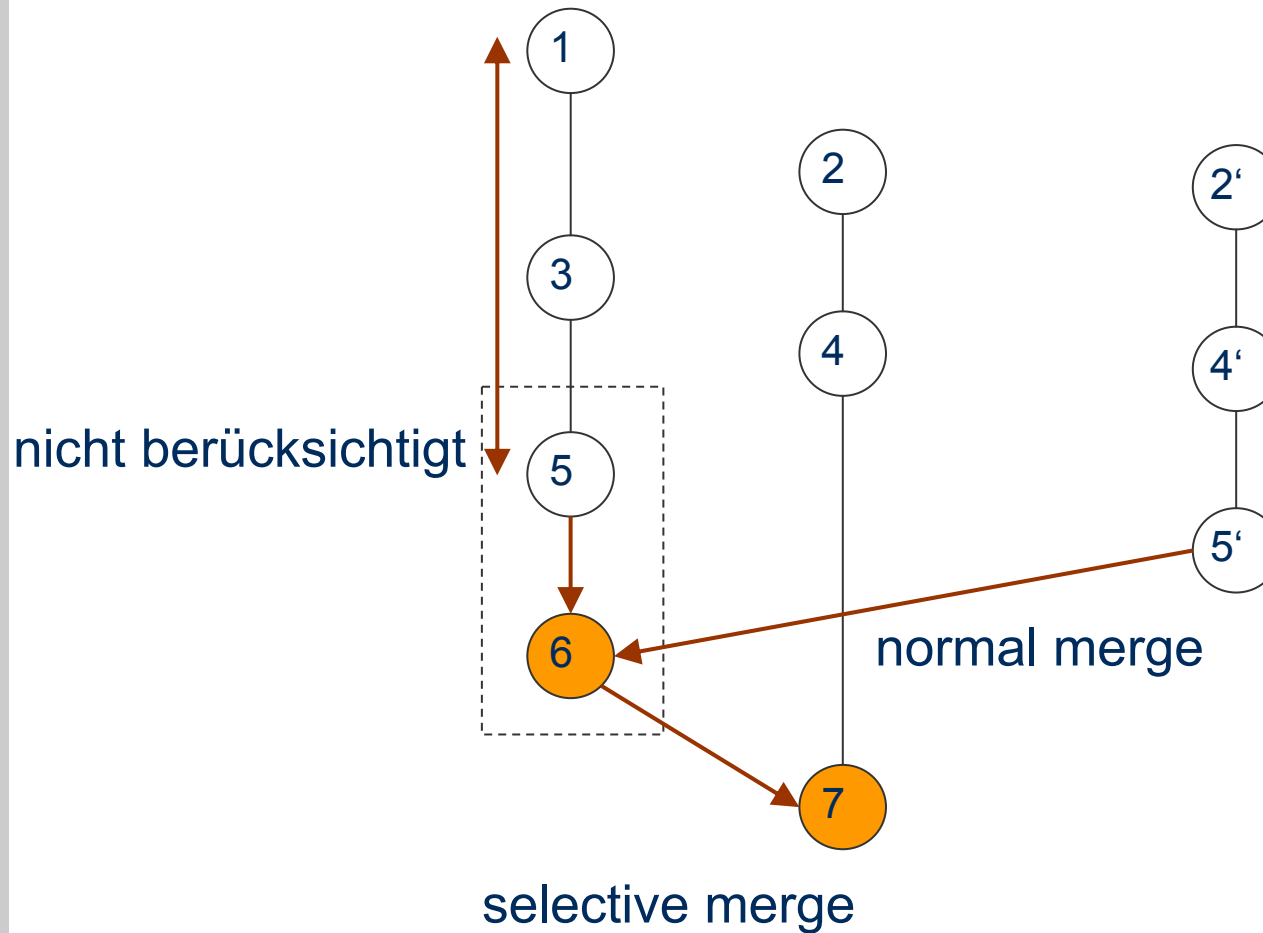
a=3

a=3

a=4



Selective Update Merge

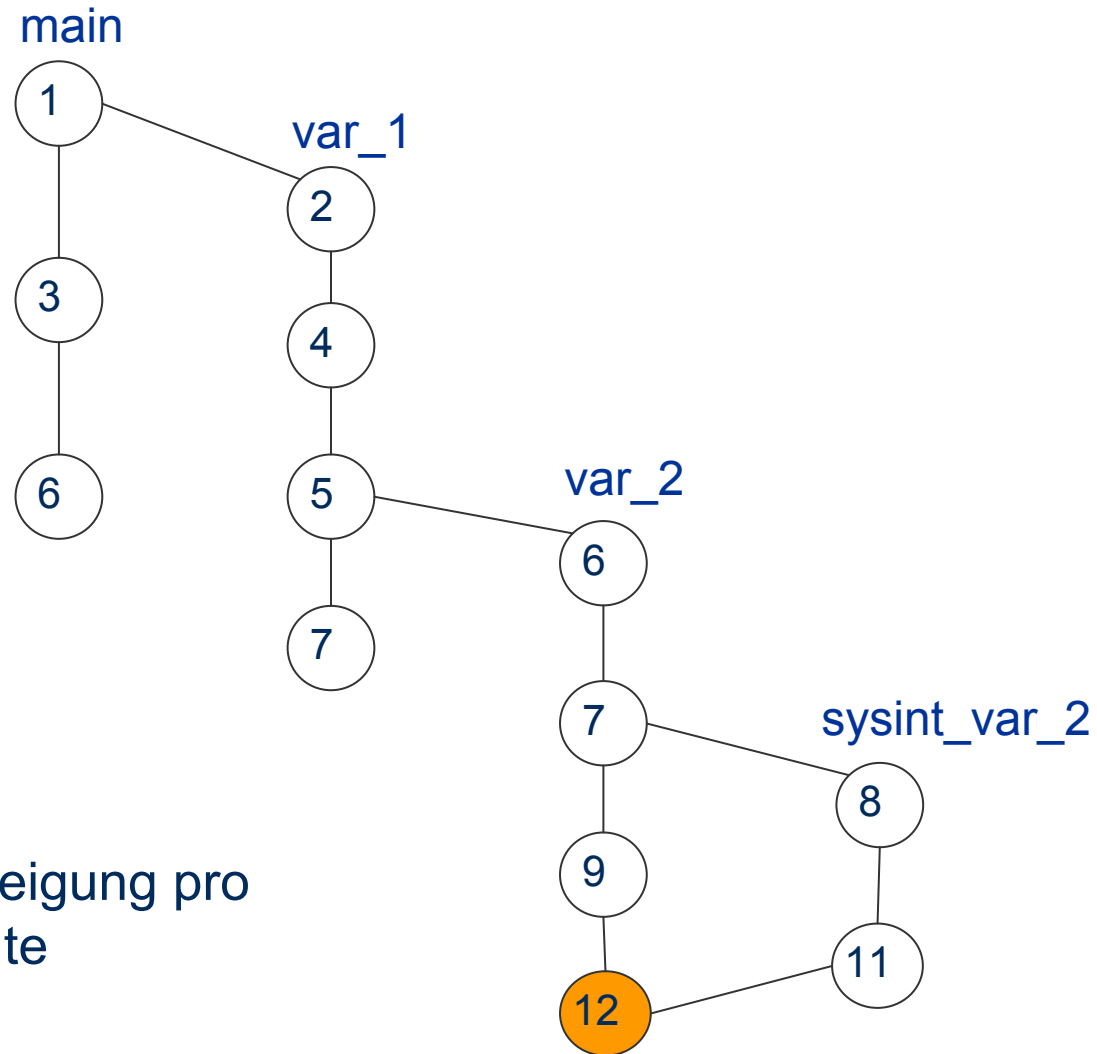


Vergleich der Integrationsmethoden

	Methode 1		Methode 2	
Branching-Struktur an Entwicklungs-Standorten	kompliziert	-	einfach	+
Bugfix-Branch an jedem Standort erforderlich	ja	-	nein	+
Config-Spec für Entwickler in Integrationsphase	verändert	-	unverändert	+
Hoher Kommunikations- und Koordinierungsaufwand	ja	-	nein	+
Hohe Anforderungen an Integrator-Rolle am Hauptstandort	nein	+	ja	-
Standard-Merge-Mechanismus benutzbar	ja	+	nein	-
Selektiver Merge erforderlich	nein	+	ja	-



Varianten-Handling

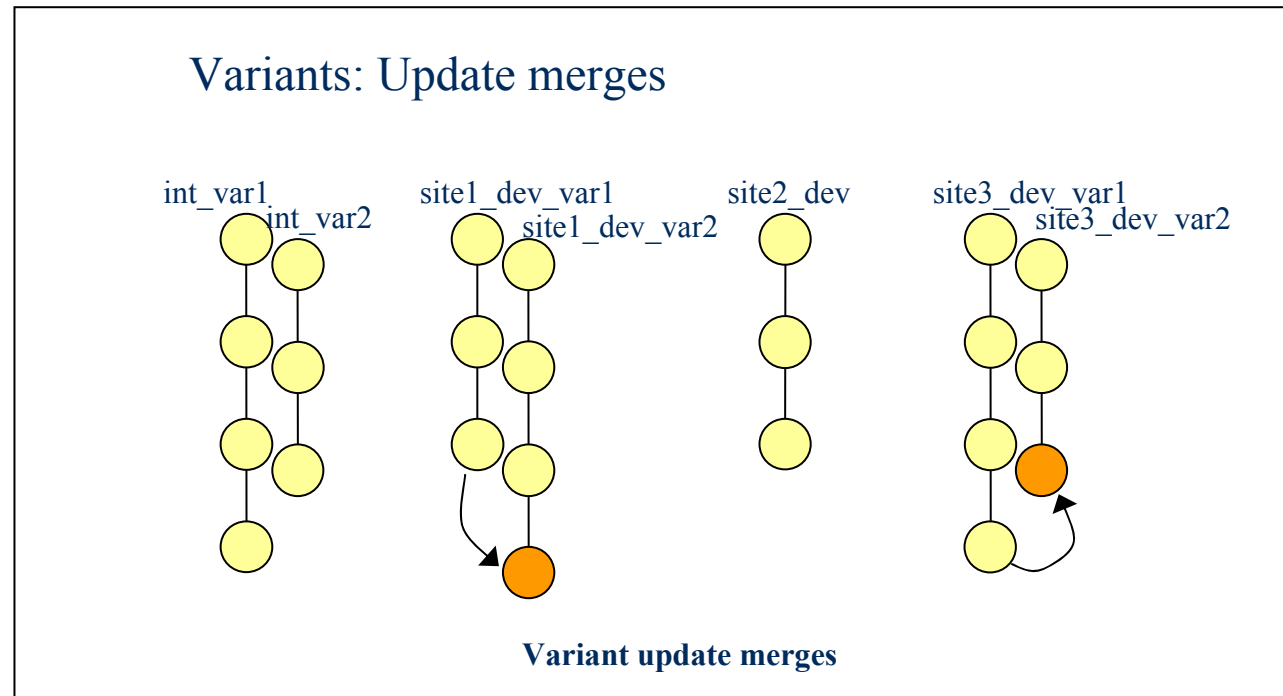


Verzweigung pro
Variante



Software &
Engineering
Development
Techniques

Merge bei Varianten-Handling



Höhere Komplexität als Update Merges
Händisch durchzuführen



Zusammenfassung

- **Entwicklung großer Softwaresysteme erfordert spezifischen Entwicklungsprozess**
- **Standort-übergreifende Entwicklung mit ClearCase-Multisite ist Standard**
- **Entwicklergruppen entwickeln typischerweise auf Zweigen (Branches)**
- **Integrationsprozess erfordert Merging von Entwicklungszweigen**
- **Standardmethode: detailreicher Prozess an allen Standorten erforderlich**
- **Optimierte Methode: es genügt Standardprozess an allen Standorten, mit Ausnahme der System-Integration**
- **Varianten-Entwicklung kann in optimierte Methode integriert werden**
- **Optimierter Prozess kann durch Skripting automatisiert werden**

