

Branching

Erfahrungen aus der Praxis

Dipl.-Ing. Univ. Edwin Wagner
edwin-wagner@t-online.de

Agenda

- Branching – wann und warum wird es benötigt
- Branching – Hilfe und Risiko zugleich
- Anforderungen / Regeln fürs Branching
- Branching für Integrations-Steps
- Branching für agile Entwicklung
- Fazit

Branching – **wann** und **warum** wird es benötigt

- Korrekturen für ausgelieferte SW Versionen (Nachlieferungen)
- Parallele Entwicklung von SW Versionen
- Prototyping um die bestehende SW nicht zu verändern
- Gemeinsam genutzte SW Komponenten in mehreren Projekten
- Kundenanforderung – nur spezielle Korrekturen / Funktionen auszuliefern
- Unterstützung für die Entwickler für eine eigene „geschützte“ Entwicklungs- und / oder Test-Umgebung
- Verteilte SW Entwicklung an mehreren, weltweit verteilten Standorten (verteilte Entwicklungsumgebung)
- Risikobereitschaft des Entwicklungsleiters (risikobehaftete Änderungen werden zuerst in einem Branch eingebracht)
- ...

Branching – Hilfe und Risiko zugleich

- Erlaubt viele Aktivitäten zu parallelisieren
- Kann aber dazu führen
 - Dass man den Überblick verliert
 - was muss wo implementiert / gemerged werden
 - wo wurden die Korrekturen schon implementiert
 - Erhöht den Integrationsaufwand – alles muss gemerged werden
- Branching und Merging sind für jede neue SW Version, nicht nur für ein Produkt, immer wieder von Neuem zu überprüfen und an die aktuellsten Projekt-Anforderungen anzupassen
- Branching und Merging sind immer gemeinsam zu betrachten !
- Nur der optimale Mix führt zum Erfolg

Anforderungen / Regeln fürs Branching

1. Muss parallele SW Entwicklung an einem SW “Element” erlauben
2. Muss parallele SW Entwicklung an mehreren, weltweit verteilten Standorten unterstützen
3. Muss die SW Entwicklung von gemeinsam genutzten Komponenten in mehreren Produkten unterstützen (common components)
4. Der Branching “Baum” muss übersichtlich und kompakt bleiben
5. Alle verteilten „Seiten“ sollten den gleichen Versions-Branch verwenden – erleichtert die Integration und „Continuous Integration“
6. SW Freigaben sind nur von “offiziellen” Versions-Branches erlaubt
7. Zulieferungen für die Integration müssen jederzeit möglich sein (Continuous Integration)
8. Für die Integration muss ein “code-freeze” möglich sein, ohne die Entwickler in ihrer weiteren Tätigkeit zu blockieren
9. die Config Spec für die Entwickler soll sehr einfach und leicht verständlich sein (Strategien nicht bei jeder Version ändern !)
10. Das Branching Konzept soll für alle Versionen und für alle Standorte gleich sein
11. **Vermeiden von unnötigem Branches !!!
NUR die Elemente branchen die es wirklich benötigen.**

Branching für Integrations-Schritte

(IS)

■ Eine

- neue SW Version 7.0 für das
- Produkt BMX soll
- in 6 Monaten und
- in 3 Integrations-Schritten erstellt und integriert werden
 - inkompatible Interface-Änderungen sind einzubringen
 - die Tests sollen so früh wie möglich beginnen
 - ein Vorhab-Produkt soll z.V. stehen

■ Entwickelt wird an 3 verteilten Standorten

- München - muc
- Wien - vie
- Bangalore - blr

■ /main

Der /main Branch wird nur für Baselines verwendet und ist der Startpunkt für neue Versionen. Die Elemente auf dem /main Branch sind Kopien von SW Freigaben von einem Versions-Branch.

■ Version

Diesen Branch Typ gibt es für jede neue SW Version und er startet immer von einer Baseline von /main. Dieser Branch Typ ist **NICHT seitenabhängig**.

■ Seitenabhängige Version

Der seitenabhängige Versions-Branch startet immer von einer Baseline von /main. Wird manchmal für Interfaces verwendet.

■ Integrations-Schritt

Der Integrations-Schritt-Branch startet immer von einem Versions-Branch und dient zur Implementierung aller Funktionen für einen Integrations-Schritt. Dieser Branch Typ ist seitenabhängig.

■ Maintenance-Release

Der Maintenance-Release-Branch startet immer von einem Versions-Branch und dient zur Fehlerbehebung (keine Implementierung neuer Funktionen). Dieser Branch Typ ist seitenabhängig.

■ User (Feature)

Der User- / Feature-Branch erlaubt den Entwicklern unabhängiges Entwickeln und Testen. Dieser Branch Typ ist seitenabhängig.

■ Bugfix

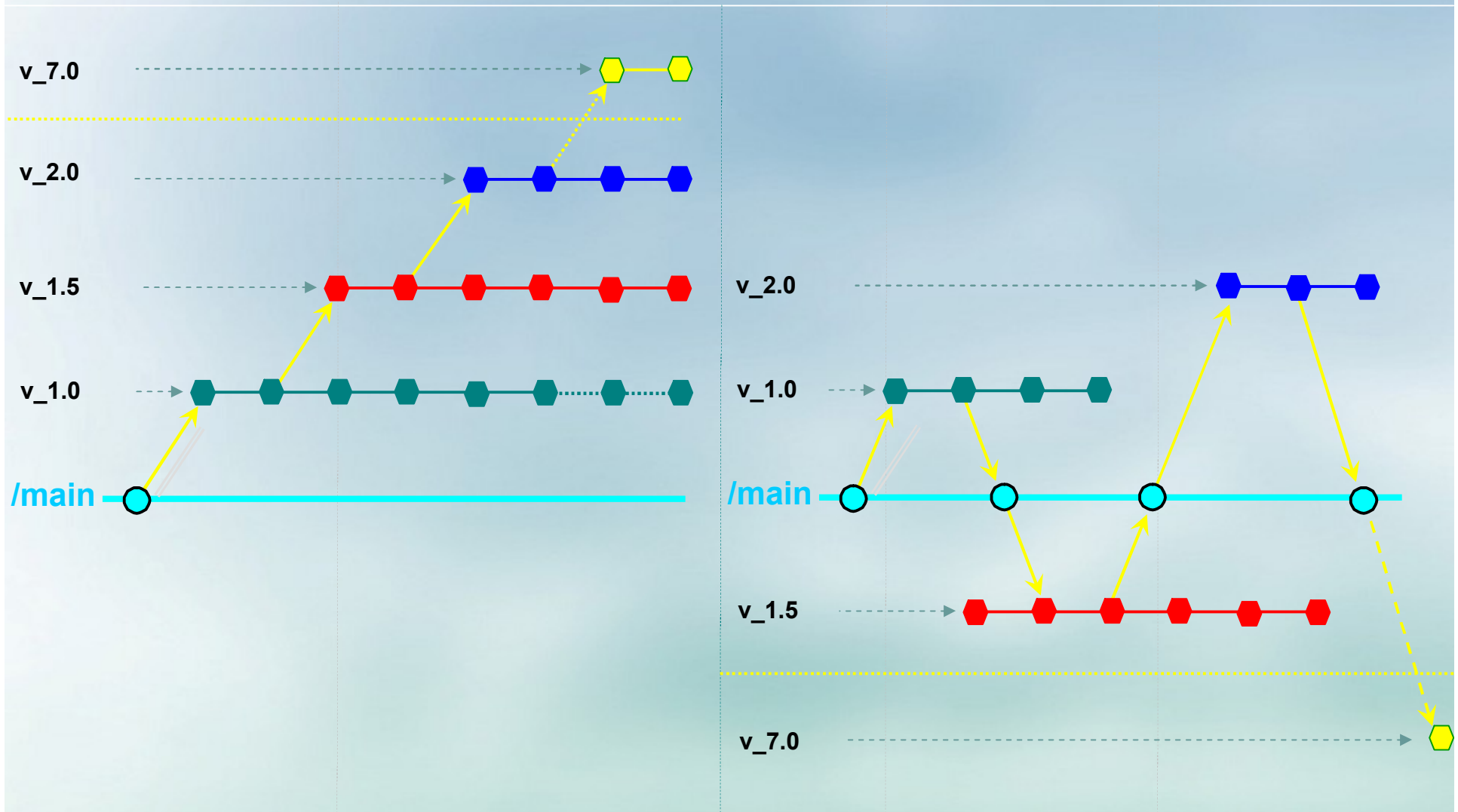
Der Bugfix-Branch startet immer von einem Release / Freigabe Label. Für jede SW Freigabe wird ein Bugfix-Branch erstellt auf dem Die Fehler für diese Freigabe korrigiert werden. Dieser Branch Typ ist seitenabhängig.

Namenskonventionen

(IS)

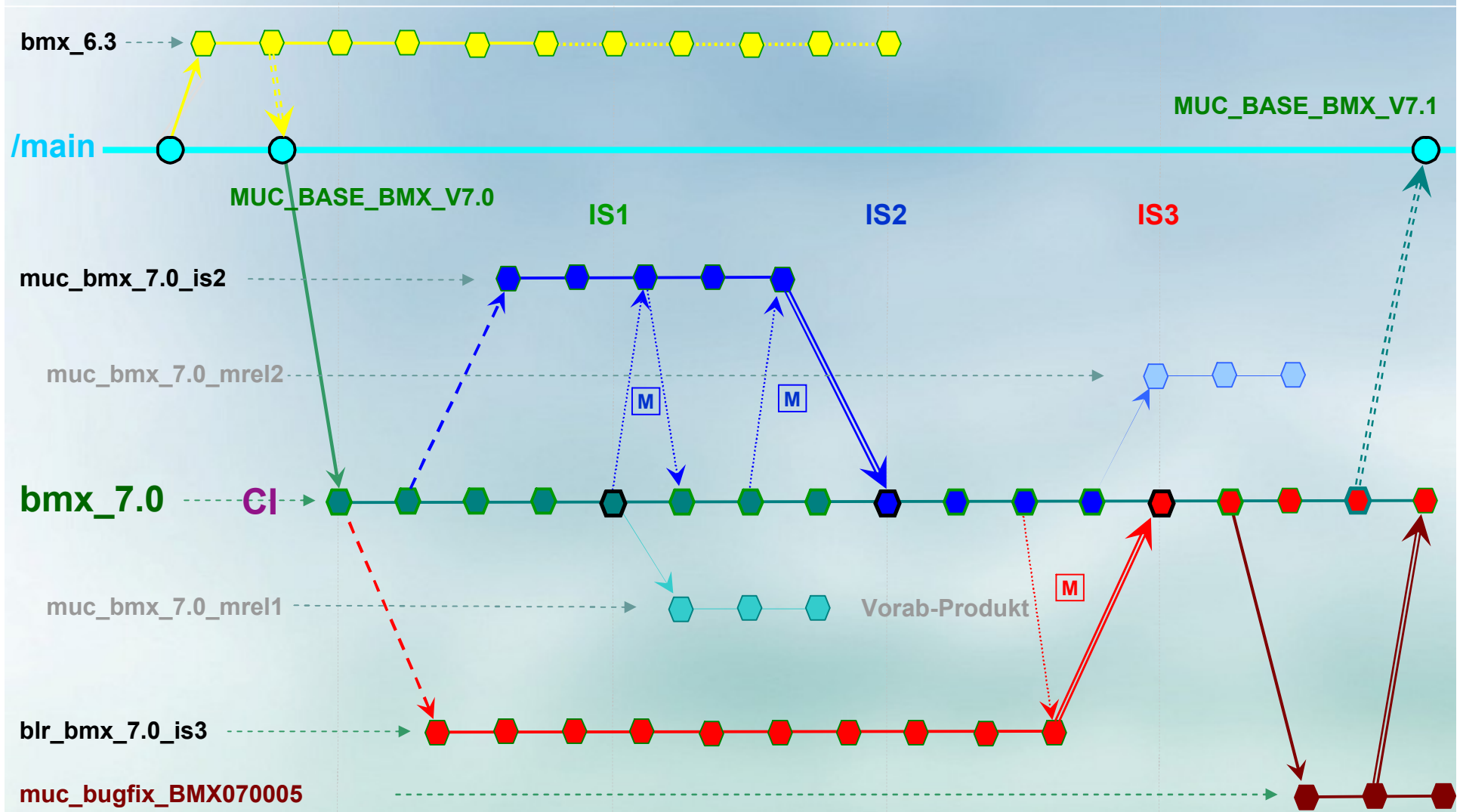
- /main
- Version - bmx_7.0
- Seite Version - muc_bmx_7.0
- Integrations-Schritt - muc_bmx_7.0_is2
vie_bmx_7.0_is2
blr_bmx_7.0_is3
- Maintenance Release - muc_bmx_7.0_mrel2
- User / (Feature) - muc_scotty_charging
- Bugfix - muc_bugfix_BMX070005

Kompakter „Baum“ mit Hilfe von Baselines (IS)



Branching BMX 7.0

(IS)



■ Eine

- neue SW Version 7.0 für das
- Produkt BMX soll
- ab Version 7.0 agile Entwickelt werden und
- die Ergebnisse von Sprint#1 und Sprint#3 sollen als potenzielle Auslieferungs-Kandidaten z.V. gestellt werden.

■ Entwickelt wird an 3 verteilten Standorten

- München - muc
- Wien - vie
- Bangalore - blr

■ /main

Der /main Branch wird nur für Baselines verwendet und ist der Startpunkt für neue Versionen. Die Elemente auf dem /main Branch sind Kopien von SW Freigaben von einem Versions-Branch.

■ Version

Diesen Branch Typ gibt es für jede neue SW Version und er startet immer von einer Baseline von /main. Dieser Branch Typ ist **NICHT seitenabhängig**.

Bei agiler Entwicklung kann ein versionsunabhängiger Branch die bessere Wahl sein (keine zw. Releases)

■ Seitenabhängige Version

Der seitenabhängige Versions-Branch startet immer von einer Baseline von /main. Wird manchmal für Interfaces verwendet.

■ Integrations-Schritt

Der Integrations-Schritt-Branch startet immer von einem Versions-Branch und dient zur Implementierung aller Funktionen für einen Integrations-Schritt. Dieser Branch Typ ist seitenabhängig.

■ Maintenance-Release

Der Maintenance-Release-Branch startet immer von einem Versions-Branch und dient zur Fehlerbehebung (keine Implementierung neuer Funktionen). Dieser Branch Typ ist seitenabhängig.

■ User (Feature)

Der User- / Feature-Branch erlaubt den Entwicklern unabhängiges Entwickeln und Testen. Dieser Branch Typ ist seitenabhängig.

■ Bugfix

Der Bugfix-Branch startet immer von einem Release / Freigabe Label. Für jede SW Freigabe wird ein Bugfix-Branch erstellt auf dem Die Fehler für diese Freigabe korrigiert werden. Dieser Branch Typ ist seitenabhängig.

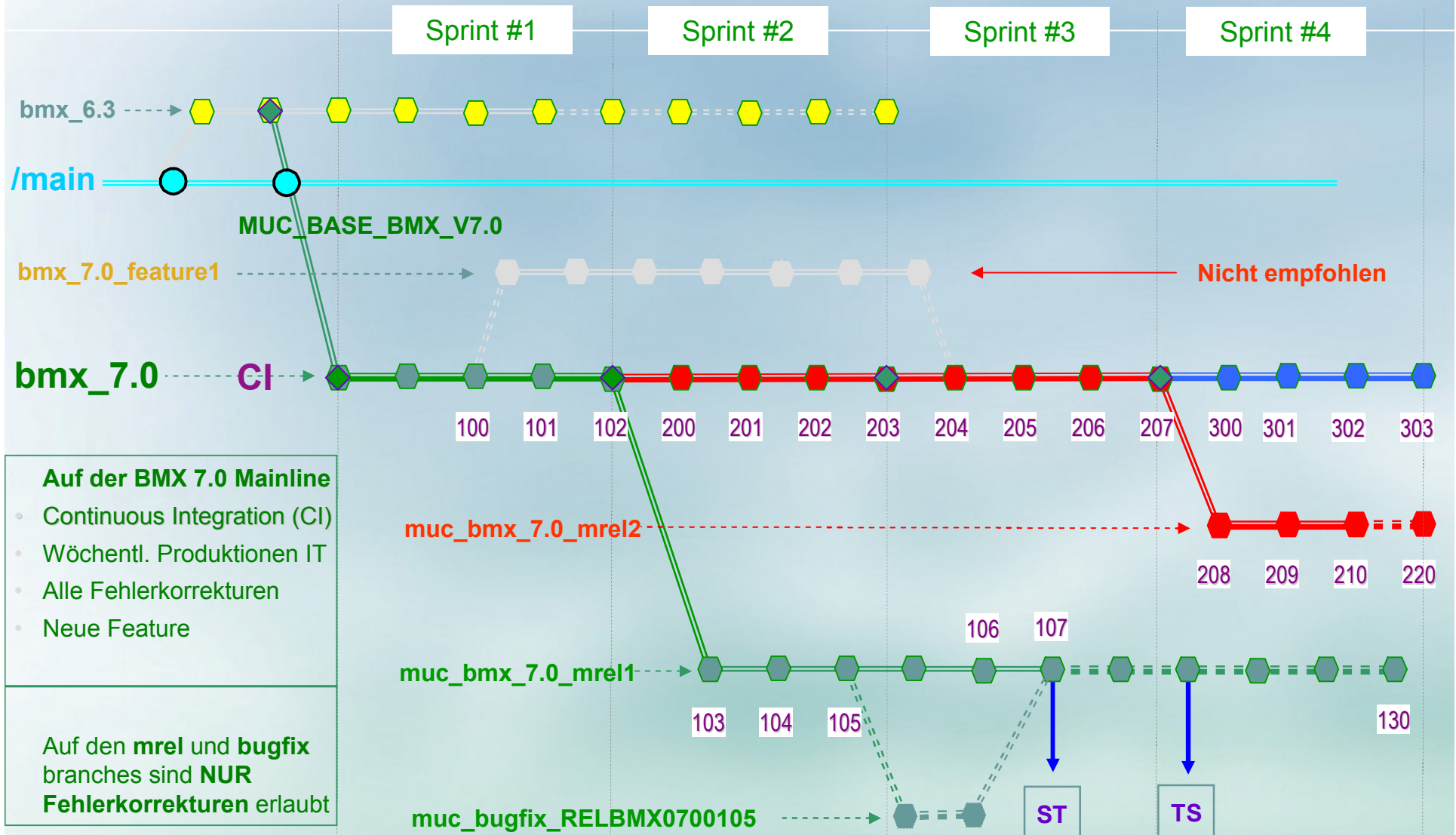
Namenskonventionen

(AE)

- /main
- Versions-Branch - **bmx_7.0** - **bmx_7**
- Site Versions-Branch - **muc_bmx_7.0** - **muc_bmx_7**
- Integrations-Schritt - muc_bmx_7.0_is2
vie_bmx_7.0_is2
blr_bmx_7.0_is3
- Maintenance Release - **muc_bmx_7.0_mrel2**
- User-Branch (Feature) - muc_scotty_charging
- Bugfix-Branch - **muc_bugfix_BMX0700105**

Branching BMX 7.0

(AE)



Fazit

- Ohne Branching kommt die SW Entwicklung nicht aus
- Aber das Branching sollte mit Maß eingesetzt werden (nur wo es unbedingt notwendig ist)
- Branching und Merging sind für jede neue SW Version, nicht nur für ein Produkt, immer wieder von Neuem zu überprüfen und an die aktuellsten Projekt-Anforderungen anzupassen
- Vor der Erstellung des Konzeptes sollten die Anforderungen / Regeln für das konkrete Projekt neu definiert werden
- Branching und Merging sind immer gemeinsam zu betrachten !
- Nur der optimale Mix und die Berücksichtigung der Anforderungen führt zum Erfolg
- Das Branching Konzept muss **vor** dem Start der Implementierung feststehen und den Beteiligten **zeitnah** kommuniziert werden
- Das Instrument Continuous Integration (CI) sollte so früh wie möglich aufgesetzt werden.
- Bei agiler Entwicklung ist CI von Anfang an ein MUSS



D a n k e