SoftwareEntwicklung ▸ Beratung ▸ Schulung

Branching Strategies

Usage of Branching Strategies within Software Development

Web Site:
www.soebes.com
Blog:
blog.soebes.com
Email:
info@soebes.com

Dipl.Ing.(FH) Karl Heinz Marbaise

Agenda

1. Arguments against Branching
2. Without Branches
3. When do we really need Branches?
4. Why Branching Concepts?
5. Branching Concepts
6. Recipes for successful Branching/Merging
7. Pitfalls of Branching

1. Arguments against Branching

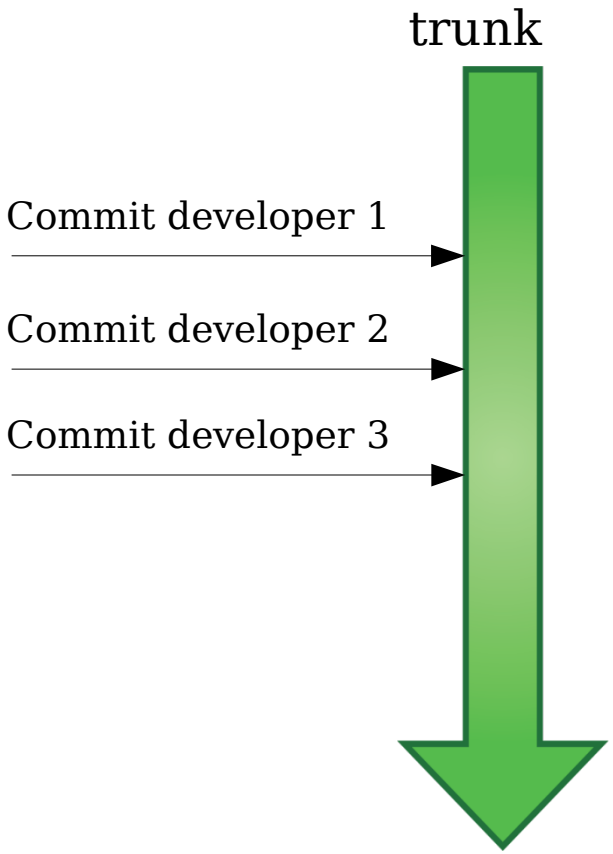
- We have a small project, so we don't need that.
- To complicated.
- There is no benefit on using Branching.
- We have already a Branching strategy ;-)
- What about conflicts?
- ....

2. Without Branches

- Let us make an experiment:
  - We assume to have a Version Control Tool (VCT) which does **not** support the concept of branching.
  - So we have **no branches at all.**

2. Without Branches

- This means we have only a single line of development.
- Let us think in Subversion terms „trunk“ for simplicity.
- Every developer has to commit on the same line.

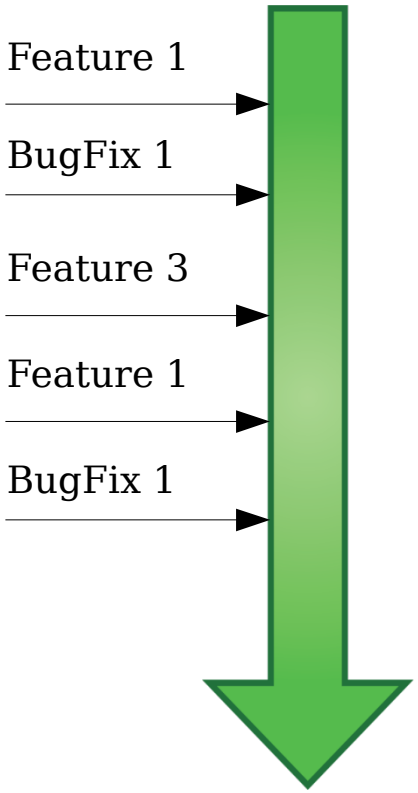


2. Without Branches Consequences

- What are the Consequences of the given scenario?
  - The environment for every developer is changing by every commit.
  - Code is not stabilizing
  - Release time points are problematic.
    - Mixture of code ready-for-production, bug fixing and feature implementation.

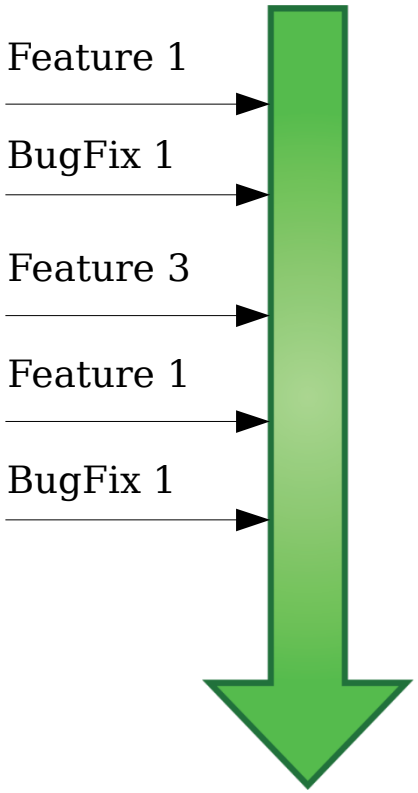
2. Without Branches Consequences

- The environment for every developer is changed by every commit.
  - This will leading to a commit policy:
    - Only commit if change has been ready
  - Prevent from commits!



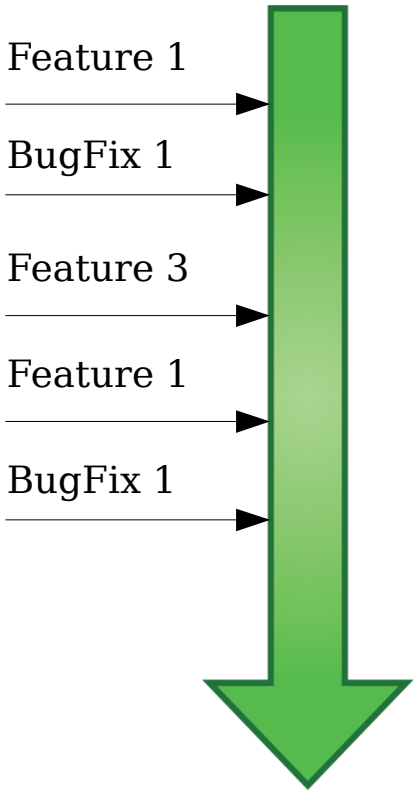
2. Without Branches Consequences

- Complex relationship between code changes and requirement/change management.
- This will make undoing of any changes more or less impossible.
- Defect analysis can be very complex.



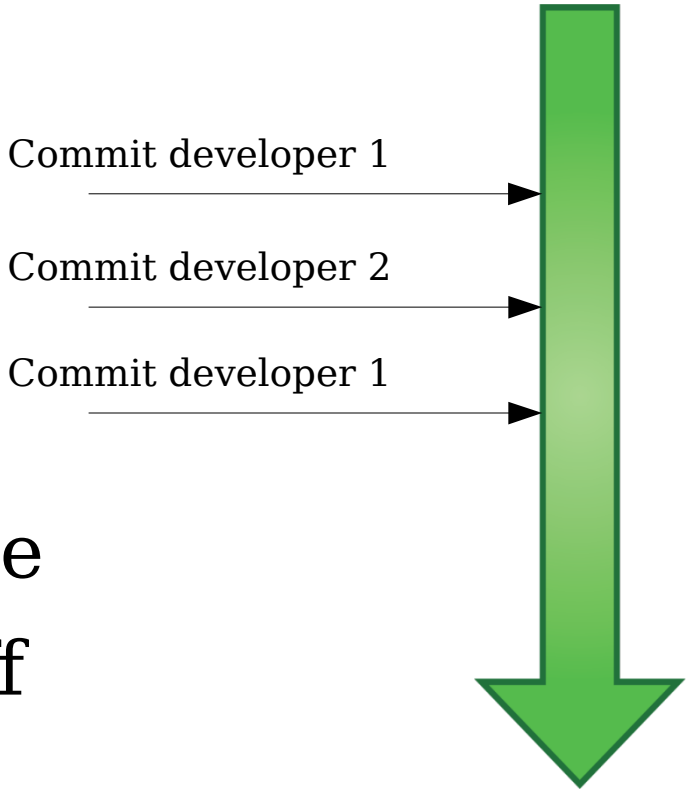
2. Without Branches Consequences

- Developers can not work on a Feature/Bug parallel.
- They will not synchronize with the repository, cause this can break their current work.
  - They have to solve conflicts instead of working on Features or Bugs.



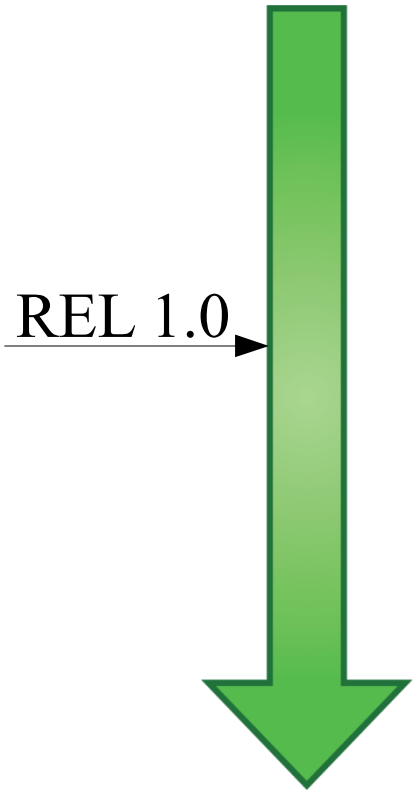
2. Without Branches Consequences

- Release Preparation:
  - Stop development!
  - Only Bug Fixing for Release is allowed
  - No parallel development possible
  - Code fixes to turn off non ready code.



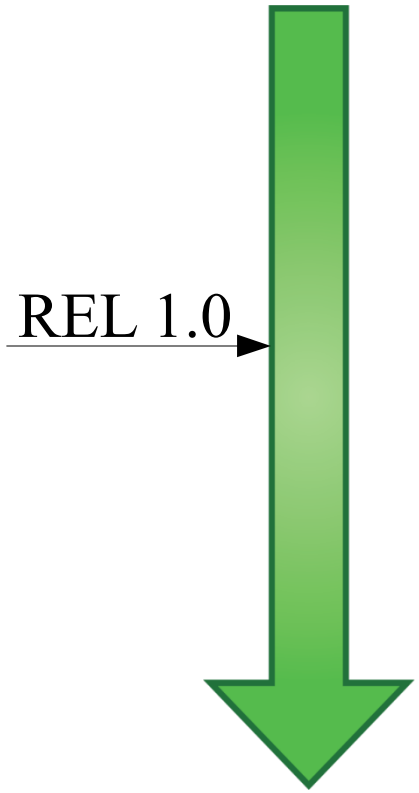
2. When do we really need Branches?

- We made an release (REL 1.0) and the customer is calling that a bug has been found.
  - We have to fix that bug as soon as possible and deliver a new release to the customer.



3. When do we really need Branches?

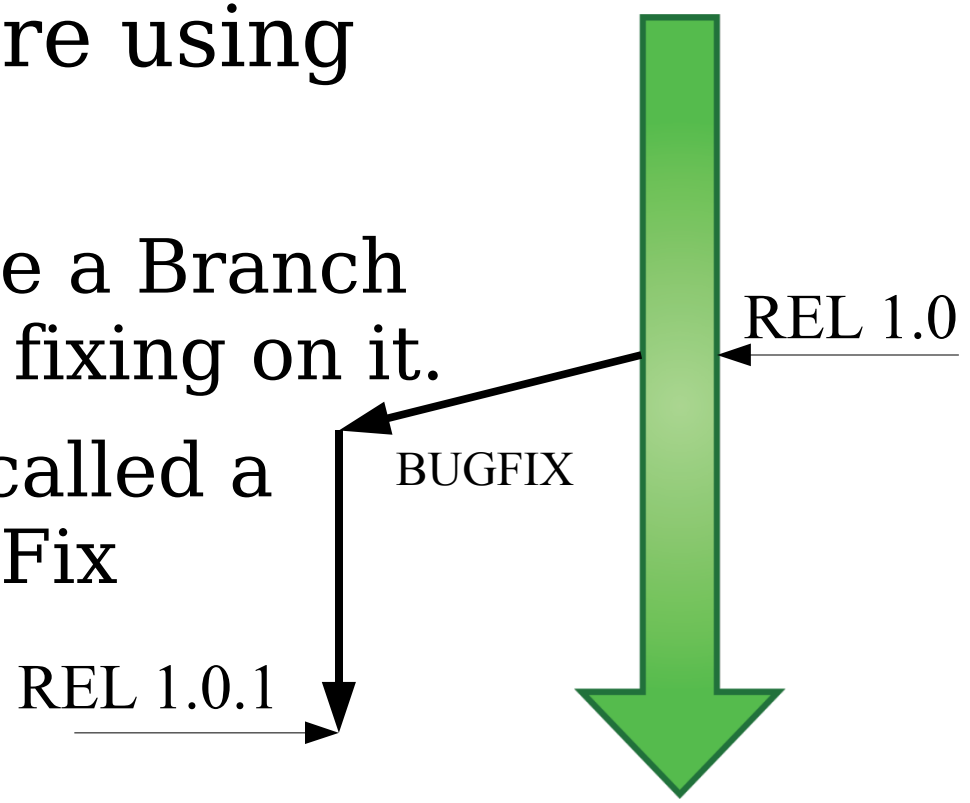
- If we have a VCT which does not support the concept of branches (independent how it is called) we are **really in trouble**.



Note:
RCS already supports branching (1985!)

3. When do we really need Branches?

- In real life we are using Subversion ;-)
  - We would create a Branch and do the Bug fixing on it.
  - This Branch is called a Hot-Fix or Bug-Fix Branch.



4. Branching Concepts Why?

- Why using Branching Concepts?
  - Change/Defect Management
  - Project/Release planning
    - Test, Integration Test, Q&A
  - Versions for:
    - Cross compiling, Operation System, GUI / Hardware.

4. Branching Concepts Why?

- Why using Branching Concepts?
  - Get a better relationship between Change/Defect Management and the changes made to the software.
  - Better informations for you and of course for the customers.

4. Branching Concepts Why?

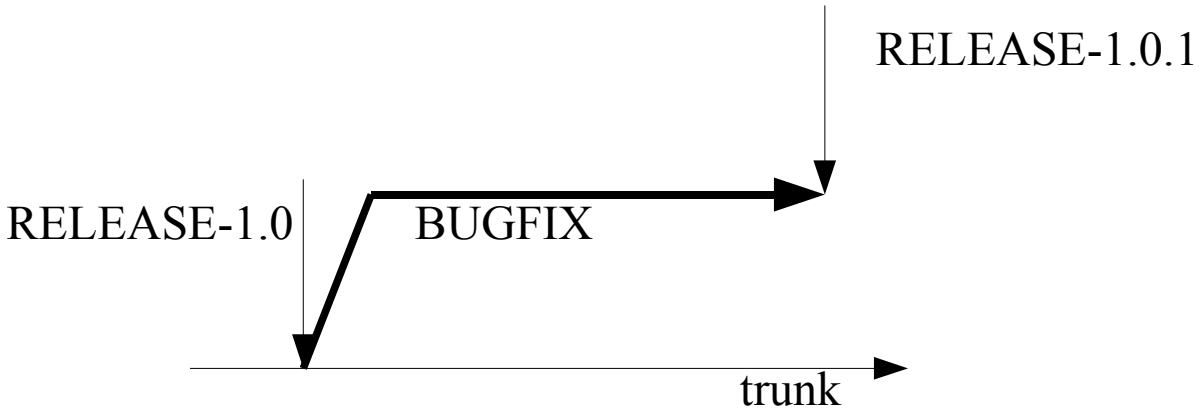
- Why using Branching Concepts?
  - Project/Release planning
    - Better identification of thinks which are comming into a release/milestone etc.
    - You can control what exactly is going into a particular release/milestone.
    - You can control the time when it's integrated into the release/milestone.

4. Branching Concepts Why?

- Why using Branching Concepts?
  - Cross compiling, Operation System, GUI / Hardware.
    - You can create Branches for particular operation systems
    - Hardware
    - GUI
    - etc.

5. Branching Concepts Bug-Fix Branching

- We create the Bug-Fix Branch based on the Release 1.0



5. Branching Concepts Bug-Fix Branching

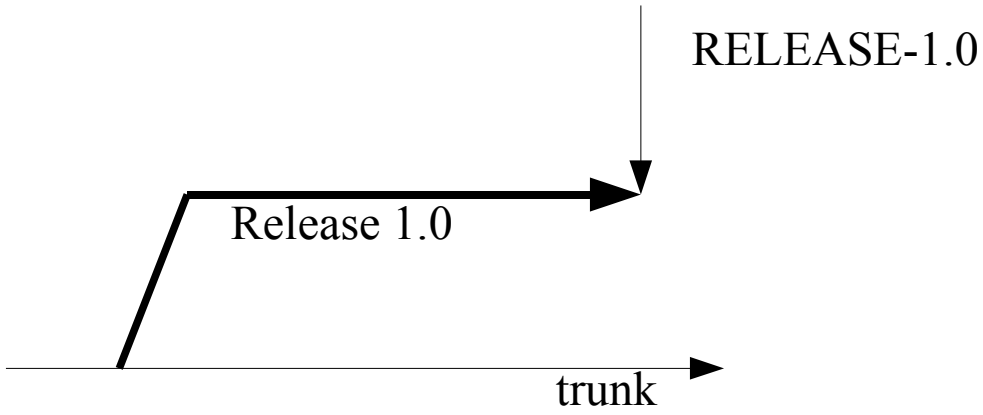
- Advantages:
  - Development (trunk) and bug fixing line are separated.
  - Separated deployment/delivery from development. So the fixed released can be delivered very fast to the customer.

5. Branching Concepts Bug-Fix Branching

- Disadvantages:
  - The trunk is of course unstable yet.

5. Branching Concepts Release Branching

- We define a point in time to start with the Release Branch as a preparation for a particular release (Feature Freeze).



5. Branching Concepts Release Branching

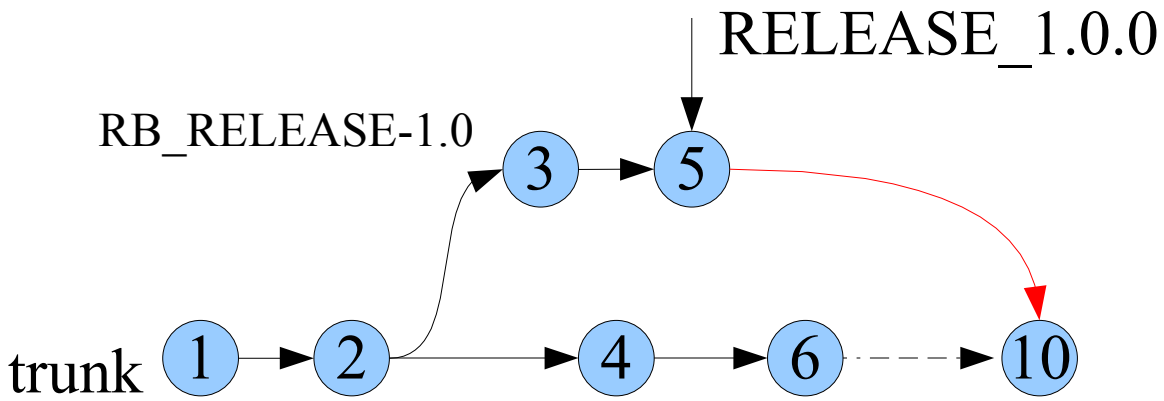
- Advantages:
  - Development (trunk) and release line are separated. There is no need to stop development on trunk.
  - Changes on the release line only affecting the release not the current development and vice versa.
  - Separated deployment/delivery from development.

5. Branching Concepts Release Branching

- Disadvantages:
  - The trunk is unstable
    - Mixture of Bug-Fixes / Features / Enhancements etc.
  - No good relationship between Change/Defect Management on the release line.
  - Undoing changes is not very simple.

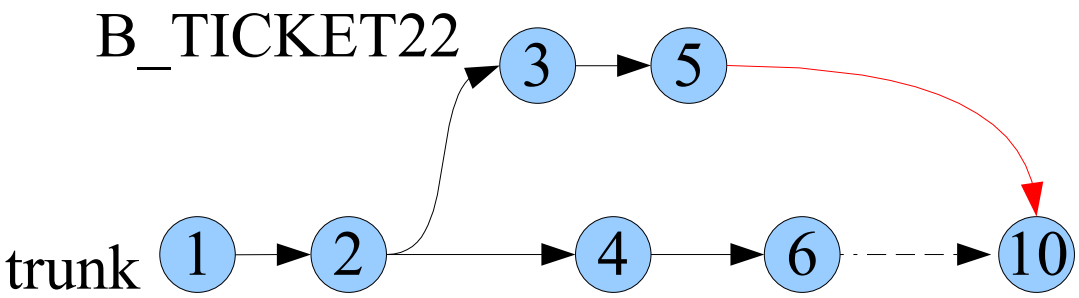
5. Branching Concepts Release Branching

- Don't miss to merge the changes back into the development line ;-)



5. Branching Concepts Issue Branching I

- Create a Branch for issues for example Bugs, Changes etc.



5. Branching Concepts Issue Branching I

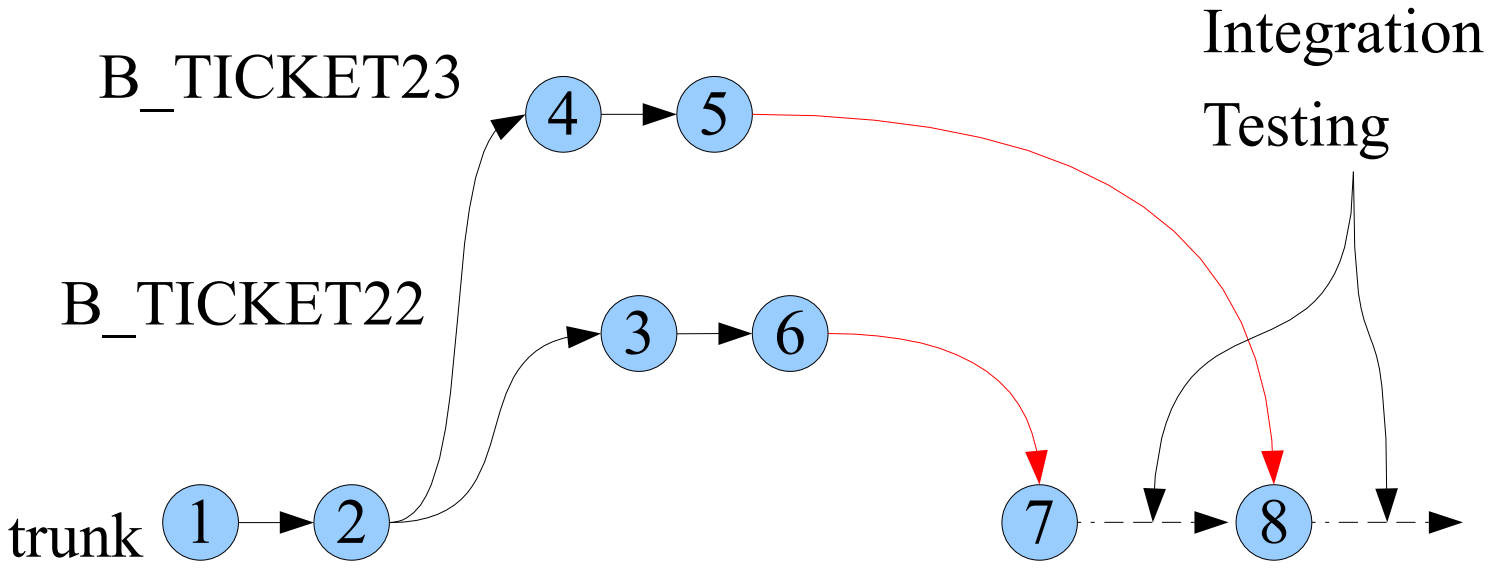
- Advantages:
  - Exact association between Code changes and Change/Defect Management for the created branches
  - Undoing of changes will be simplified on the “trunk”.

5. Branching Concepts Issue Branching I

- Disadvantages:
  - Unstable trunk
  - No good relationship to Change/Defect Management for the trunk.

5. Branching Concepts Issue Branching II

- Changes/Bug Fixes will be made by Branches **only**.



5. Branching Concepts Issue Branching II

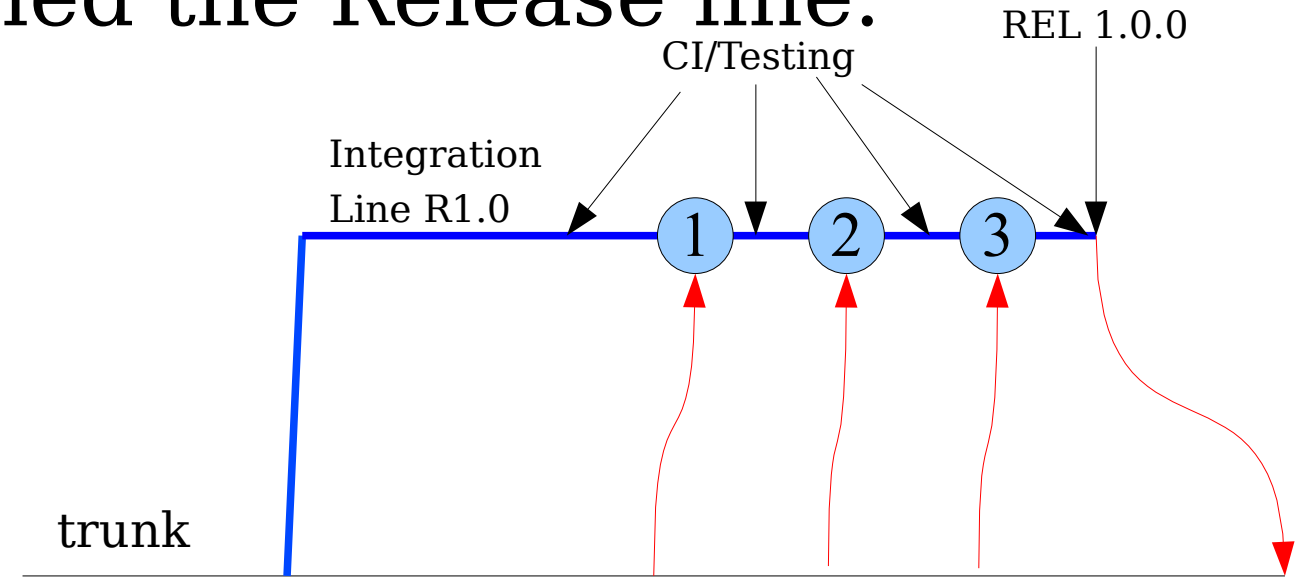
- Advantages
  - Very good relationship between Change/Defect Management either on the branches and what has been integrated into the “trunk”.
  - Analysing is simplified cause the problem can only occur at the integration point.
  - Simplifies undoing of changes.

5. Branching Concepts Issue Branching II

- Disadvantages
  - You have to be careful with code refactoring, cause changes in folder structure can be a nightmare during a merge.

5. Branching Concepts Integration/Dev Line

- Integration / Development Line
- The Integration line is sometimes called the Release line.



5. Branching Concepts Integration/Dev Line

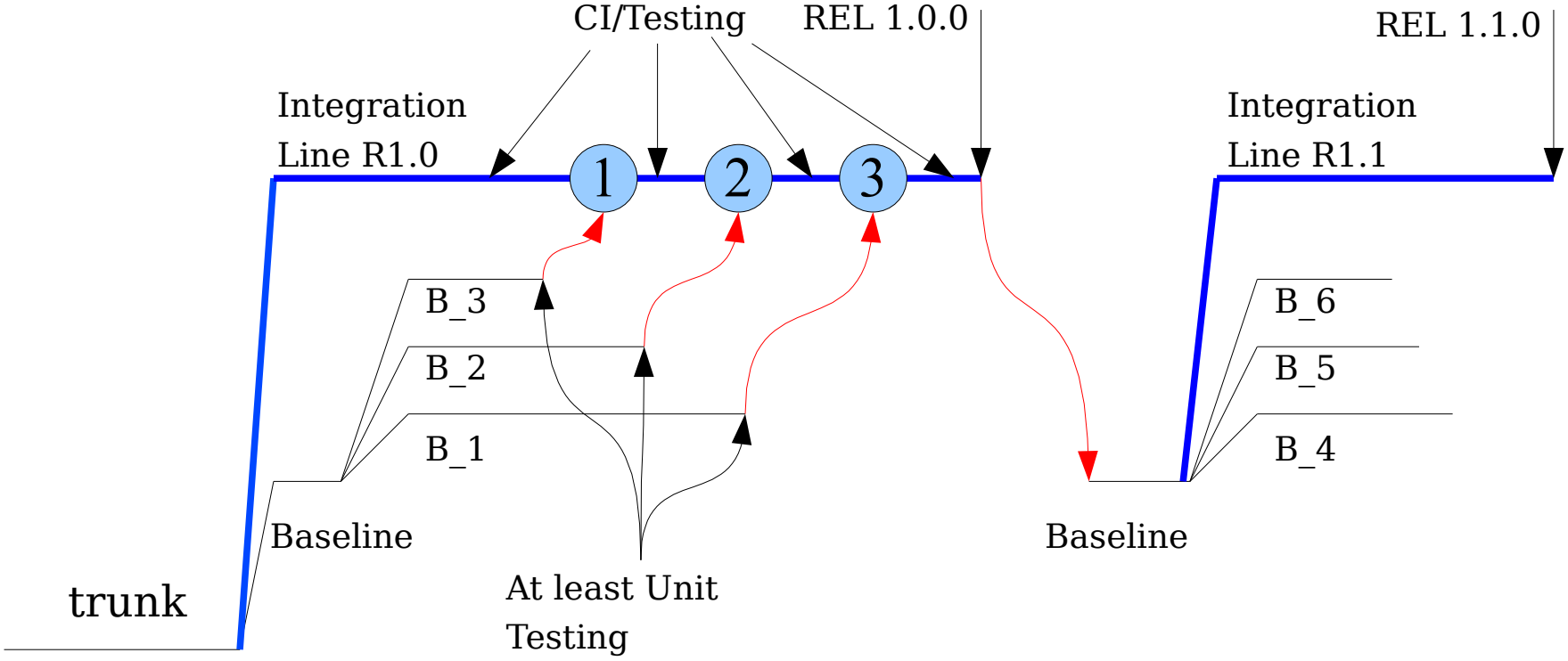
- Advantages
  - Separated development/deployment.
  - Parallel development and release line.

5. Branching Concepts Integration/Dev Line

- Disadvantages
  - Code stabilizing is not very good, cause you are integrating from an unstable code line „trunk“
  - Loosing relationship to Change/Defect Management

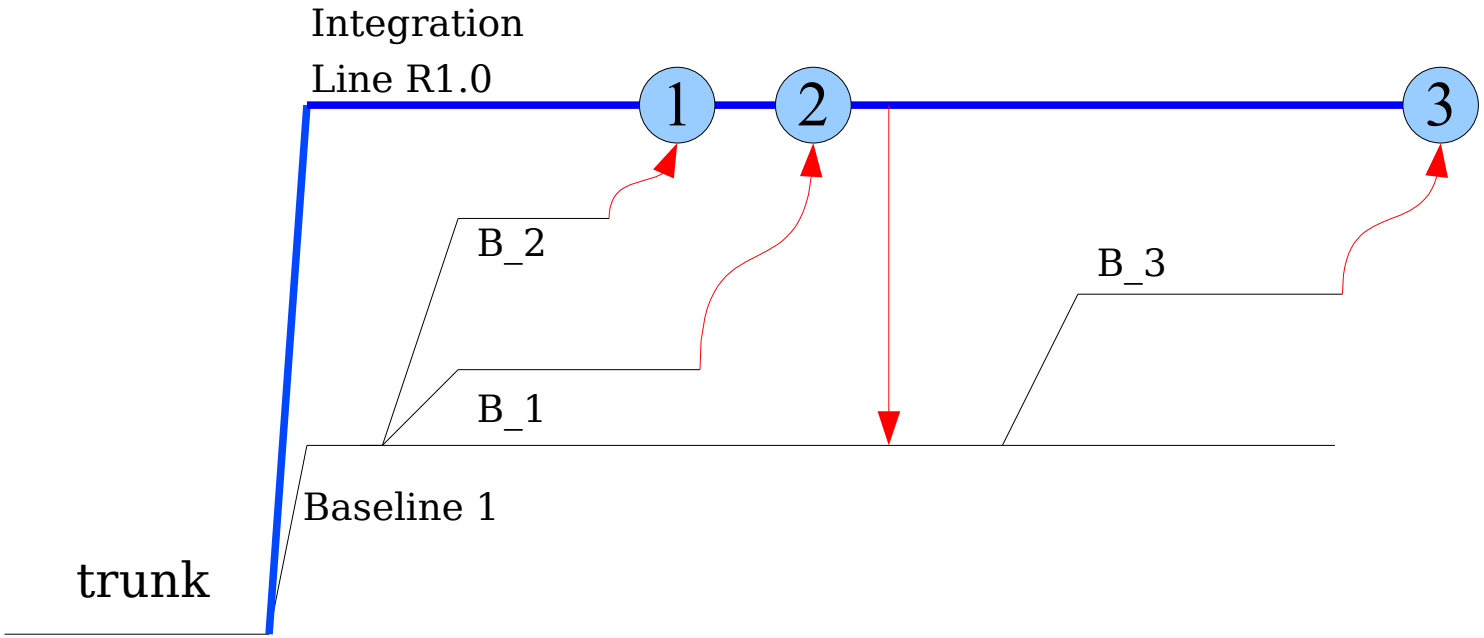
5. Branching Concepts Baselining

- Serialized Releases

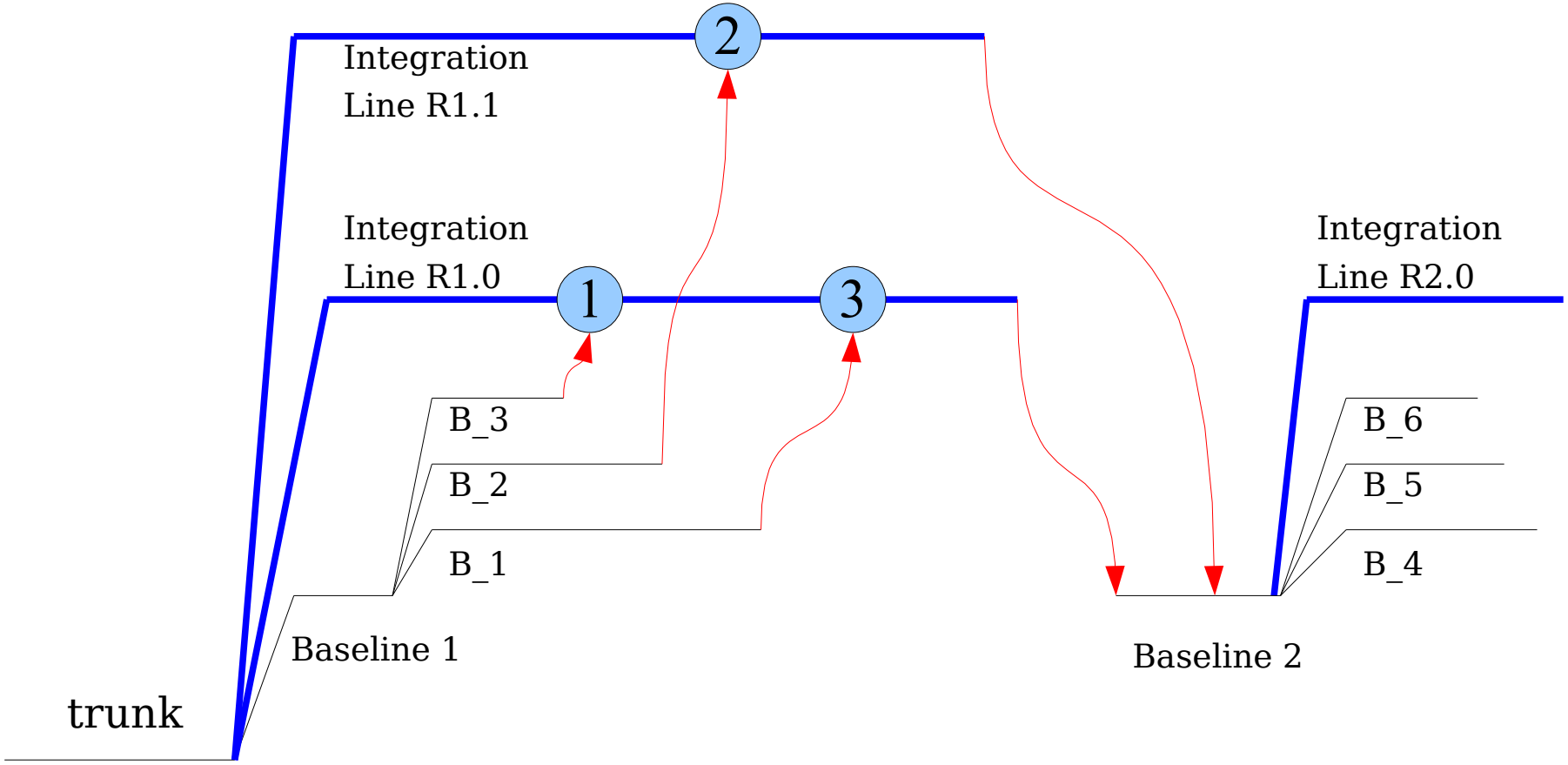


5. Branching Concepts Baselining

- With reintegration into Baseline from time to time.



5. Branching Concepts Baselining



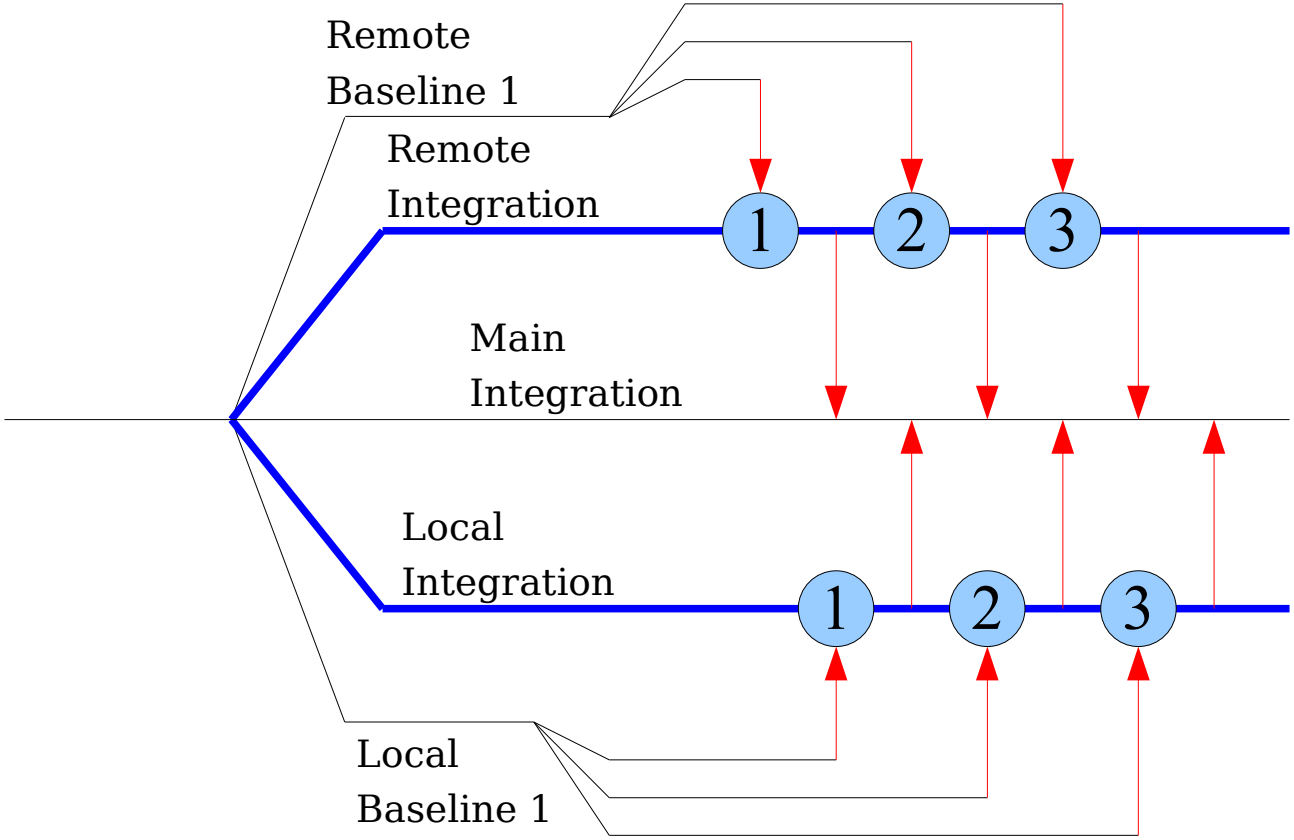
5. Branching Concepts Baselining

- Advantages
  - Exact association between change/defect Management and the branches
  - Stabilized Code line

5. Branching Concepts Baselining

- Disadvantages
  - May be problematic if code refactoring is needed.
  - Sometimes problems occur if branches depend on each other.

5. Branching Concepts Distributed Developm.



6. Recipes for successful Branching/Merging

- Before you think about a Branching concept, think about a **Merging** concept instead.
  - Who should do the merge?
    - The developers.

6. Recipes for successful Branching/Merging

- What should be part of the merging process?
  - Unit Tests
  - Integration Test
  - System Test
  - Continuous Integration (CI) which is triggered by commits can be a good support on the integration lines.

6. Recipes for successful Branching/Merging

- Think about your branching concept before project start. These concepts are part of the project planning phase.
- Do not write your branch concept in **stone**!
- Observe your experience with your branching concept and……
  - **If it is necessary just change it!**

6. Recipes for successful Branching/Merging

- Naming Conventions for branch names.
  - Very important

    Write it down instead of just thinking about it!

6. Recipes for successful Branching/Merging

- Naming Conventions for Branches:
  - Integration Line (IL_...)
    - Release number as supplemental (IL_RELEASE-1.0.0)
  - Bug Fix Branches (BFB_...)
    - Bug tracker id should appended (BFB_TICKET31)

6. Recipes for successful Branching/Merging

- Naming Conventions for Branches:
  - Feature Branches (FB_...) (FB_TICKET66)
  - If you have different tools for Change Management and Bug Tracking, add the information too.
    - For example FB_CMTICKET42

6. Recipes for successful Branching/Merging

- In Subversion...
  - Create sub folders in the branches directory
    - integration, bugfixes, features
  - May be for the tags folders is needed the same
    - Build Tags etc.

6. Recipes for successful Branching/Merging

- My final Statement:

  No Unit Tests No Branching!

- But the better Statement is:

  No Unit Tests No Merging!

7. Pitfalls of Branching

- Not merged for a long time.
  - This can result in many conflicts during a following merge.
    - Reduce branch life time
    - Make reintegrations from time to time.
    - Change your Branching Concept

7. Pitfalls of Branching

- Branch-oholic
  - Branch at all costs.
    - May be you have not invested enough effort into a „Branching/Merging Concept“ (Software Configuration Plan).

On-line Sources I

- [1] Brad Appleton's Streamed Lines: Branching Patterns for Parallel Software Development
  - http://cmcrossroads.com/bradapp/acme
- [2] UCM Branching Strategies
  - http://www.snuffybear.com/ucm_branch.htm
- [3] Branching and Merging Primer
  - http://msdn.microsoft.com/en-us/library/aa7308

On-line Sources II

- [4] Branching Strategy Questioned
  - http://blogs.open.collab.net/svn/2007/11/branch
- [5] Parallel Branching Strategies in Software Configuration Management
  - http://whitepapers.techrepublic.com.com/abstra
- [6] Distributed Version Control Systems: A Not-So-Quick Guide Through
  - http://www.infoq.com/articles/dvcs-guide

On-line Sources III

- [7] Homepage of Subversion
  - http://subversion.tigris.org
- [8] Book about Subversion
  - http://www.svnbook.org
- [9] Subversion Forum
  - http://www.svnforum.org
- [10] German Subversion forum
  - http://forum.subversionbuch.de

On-line Sources II

- [11] Forum for Software Configuration Management
  - http://www.xing.com/net/skm
- [12] The SKM Wiki (german)
  - http://www.skmwiki.de

Questions?

subconf2008@soebes.com

- Thank you for your attention.