

Lotti's Practical Versioning Scheme (LPVS)



- LPVS describes a concept of document versioning
- Manual implementation is intended, there is no need for a tool
- A major design goal was to keep it simple and stupid (KISS)
- The target audience are humans who care about document reproducibility and genealogy
- Nevertheless the approach is practical, artistic and scientific
- Results are self-explanatory for simple cases and quickly explained for advanced cases

Lotti's Practical Versioning Scheme (LPVS), Release 20100501
<http://lpvs.lotterer.net/>

Copyright © 2010 OpenPKG GmbH
Thomas Lotterer <thl@openpkg.com>, Director Engineering and Production



Creative Commons Attribution-No Derivative Works 3.0 Germany License
<http://creativecommons.org/licenses/by-nd/3.0/de/>

Basics



The text file "foo"

foo.txt

foo



The text file "foo"

foo.txt

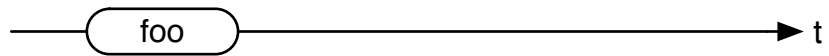
work with the base filename and
leave the file type suffix "as is".

foo



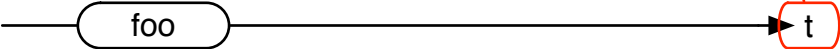
The text file "foo" evolves over time

foo-txt



The text file "foo" evolves over time

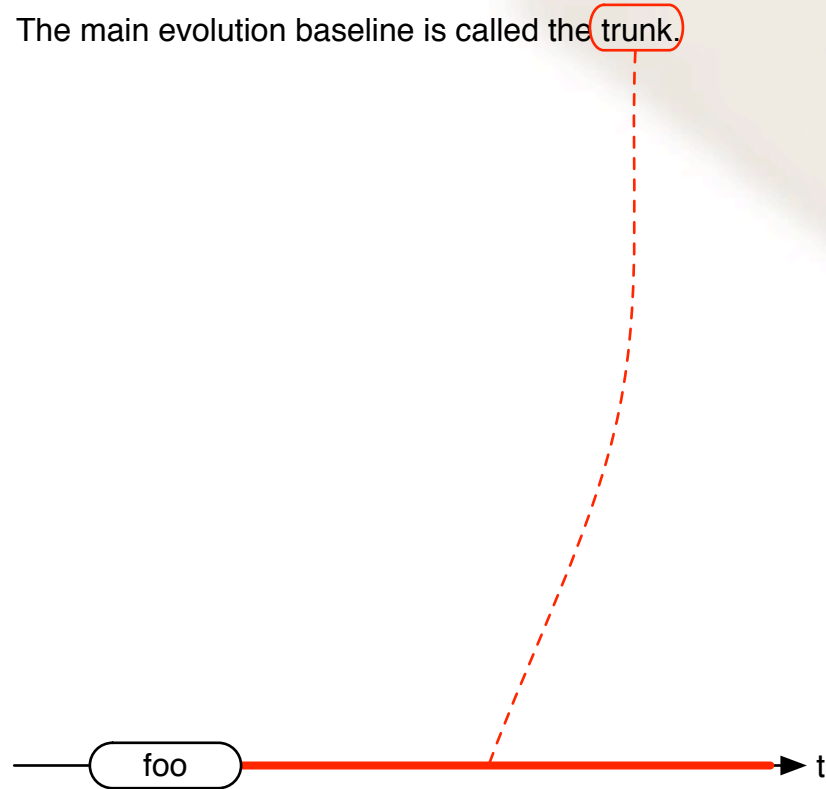
foo-txt



The text file "foo" evolves over time

foo-txt

The main evolution baseline is called the trunk.

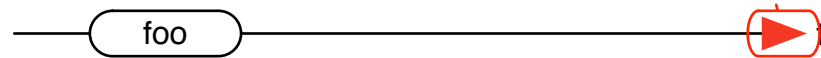


The text file "foo" evolves over time

foo-txt

The main evolution baseline is called the trunk.

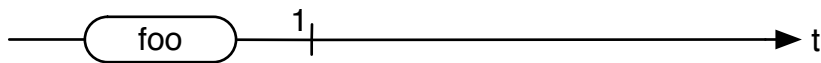
The latest evolution is called the **head**.



Version 1 in the evolution of the text file "foo"

foo-1.txt

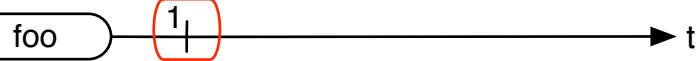
A version is kind of a snapshot. The contents of a version must never change. **NEVER!**
If possible, make a version read-only.



Version 1 in the evolution of the text file "foo"

foo-1.txt

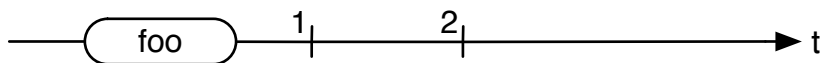
dash 1



Version 2 in the evolution of the text file "foo"

foo-2.txt

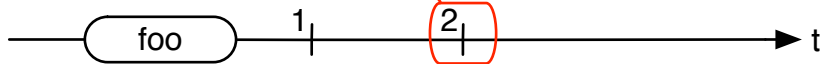
The whole idea of versioning is to distinctively identify a certain content. That implies the version must not be part of the content. It is pure metadata. The examples herein assume use as file names and directory names.



Version 2 in the evolution of the text file "foo"

foo-2.txt

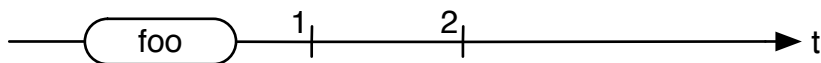
dash 2



Version 2 in the evolution of the text file "foo"

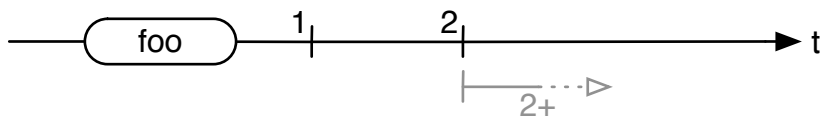
foo-2.txt

While improving version 2 and working towards version 3, how should the "work in progress" be named? It's obviously no longer version 2 but still not version 3.



The '+' Suffix marks "work in progress"

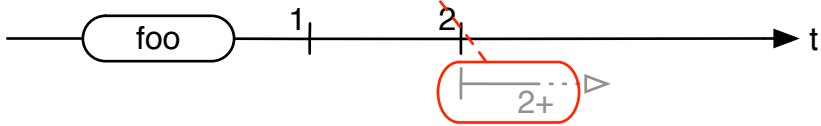
foo-2+.txt



The '+' Suffix marks "work in progress"

foo-2+txt

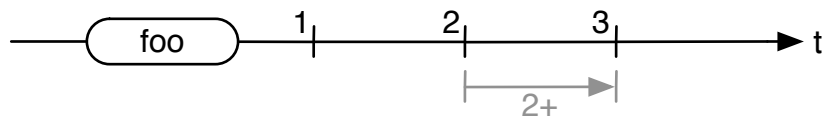
dash 2 plus



Version 3 in the evolution of the text file "foo"

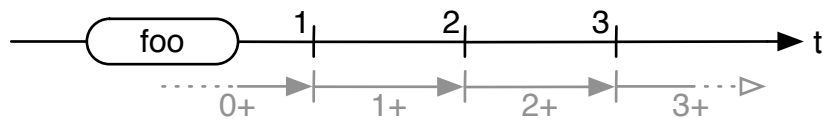
foo-3.txt

If the file format allows identification of the changes, it's useful to keep the '+' version to allow reviewers to more easily find the difference between two versions.



Guess the base of "work in progress" before the first version

foo-0+.txt



Basics

by example

`idea-1.odf`

`idea-2.odf`

`idea-3.odf`

`idea-3+.odf`

`idea-4.odf`

`idea-4+.odf`



Security Classification



Not really versioning, but closely related in the identification space, the '=' Suffix introduces "security classification"

foo-2=**CONFIDENTIAL**.txt

TOPSECRET

SECRET

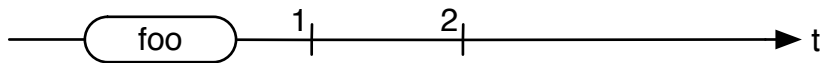
CONFIDENTIAL

RESTRICTED (default when no classification given)

PUBLIC

Use upper case only.

See [International Programs Security Handbook](#), [Appendix E - Foreign & US Classifications](#)



Security Classification

by example

`idea-1.xls`

`idea-2.xls`

`idea-3=CONFIDENTIAL.xls`

`idea-3+=CONFIDENTIAL.xls`

`idea-4=CONFIDENTIAL.xls`

`idea-4+=SECRET.xls`



Branching



The text file "foo" diverges into multiple variants

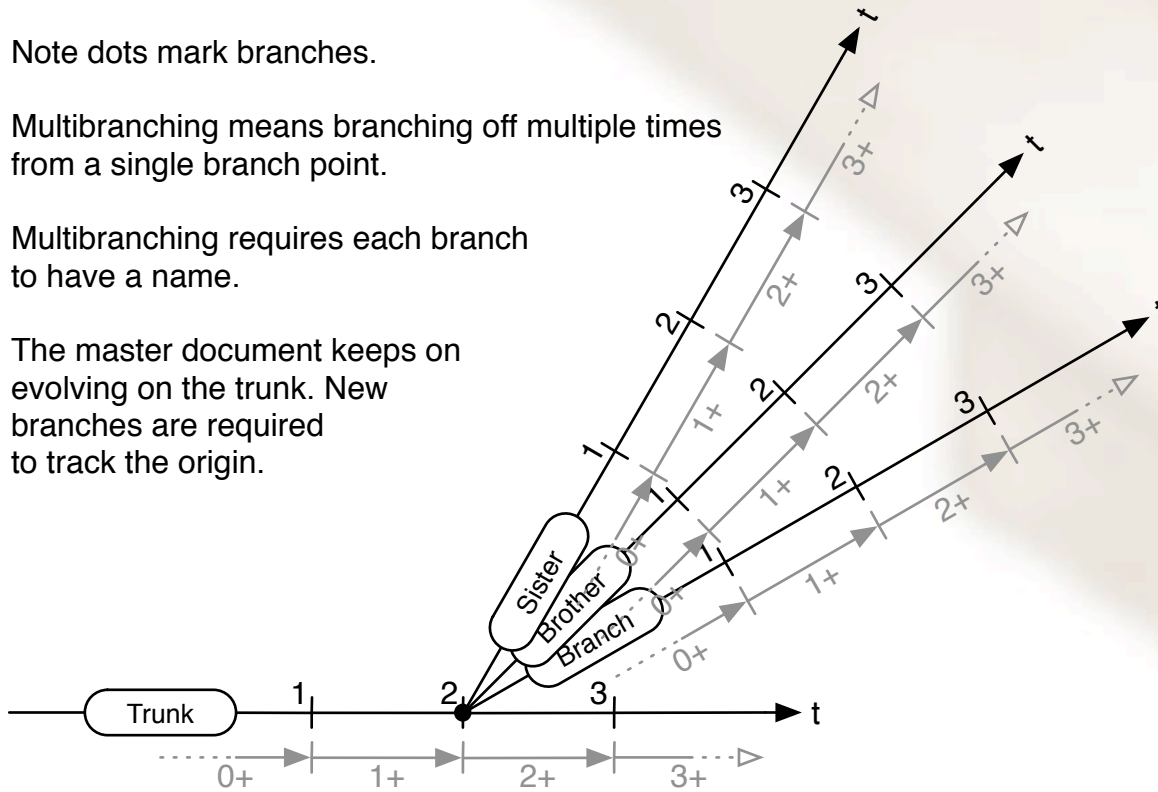
trunk-2.branch-3.txt
trunk-2.brother-3.txt
trunk-2.sister-3.txt

Note dots mark branches.

Multibranching means branching off multiple times from a single branch point.

Multibranching requires each branch to have a name.

The master document keeps on evolving on the trunk. New branches are required to track the origin.



The text file "foo" diverges into multiple variants

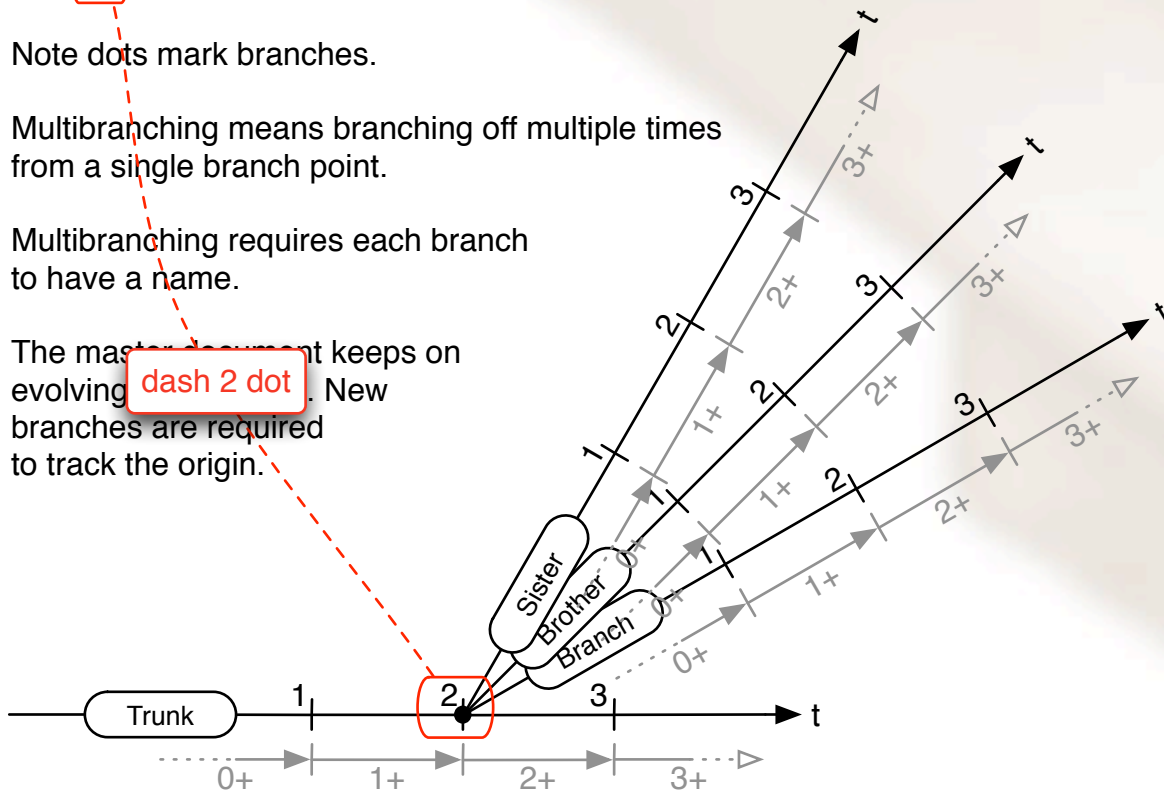
trunk-2.branch-3.txt
trunk-2.brother-3.txt
trunk-2.sister-3.txt

Note dots mark branches.

Multibranching means branching off multiple times from a single branch point.

Multibranching requires each branch to have a name.

The master trunk keeps on evolving
dash 2 dot . New branches are required to track the origin.



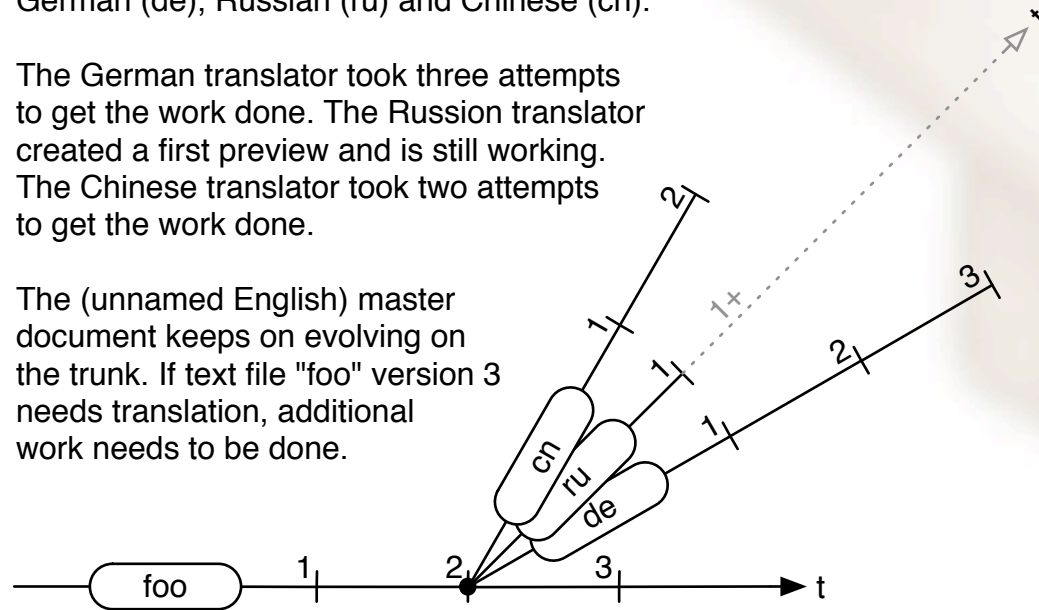
The text file "foo" diverges into multiple variants

foo-2.de-3.txt
foo-2.ru-1+.txt
foo-2.cn-2.txt

In this example, the text file "foo" version 2 needed translation into three languages German (de), Russian (ru) and Chinese (cn).

The German translator took three attempts to get the work done. The Russian translator created a first preview and is still working. The Chinese translator took two attempts to get the work done.

The (unnamed English) master document keeps on evolving on the trunk. If text file "foo" version 3 needs translation, additional work needs to be done.



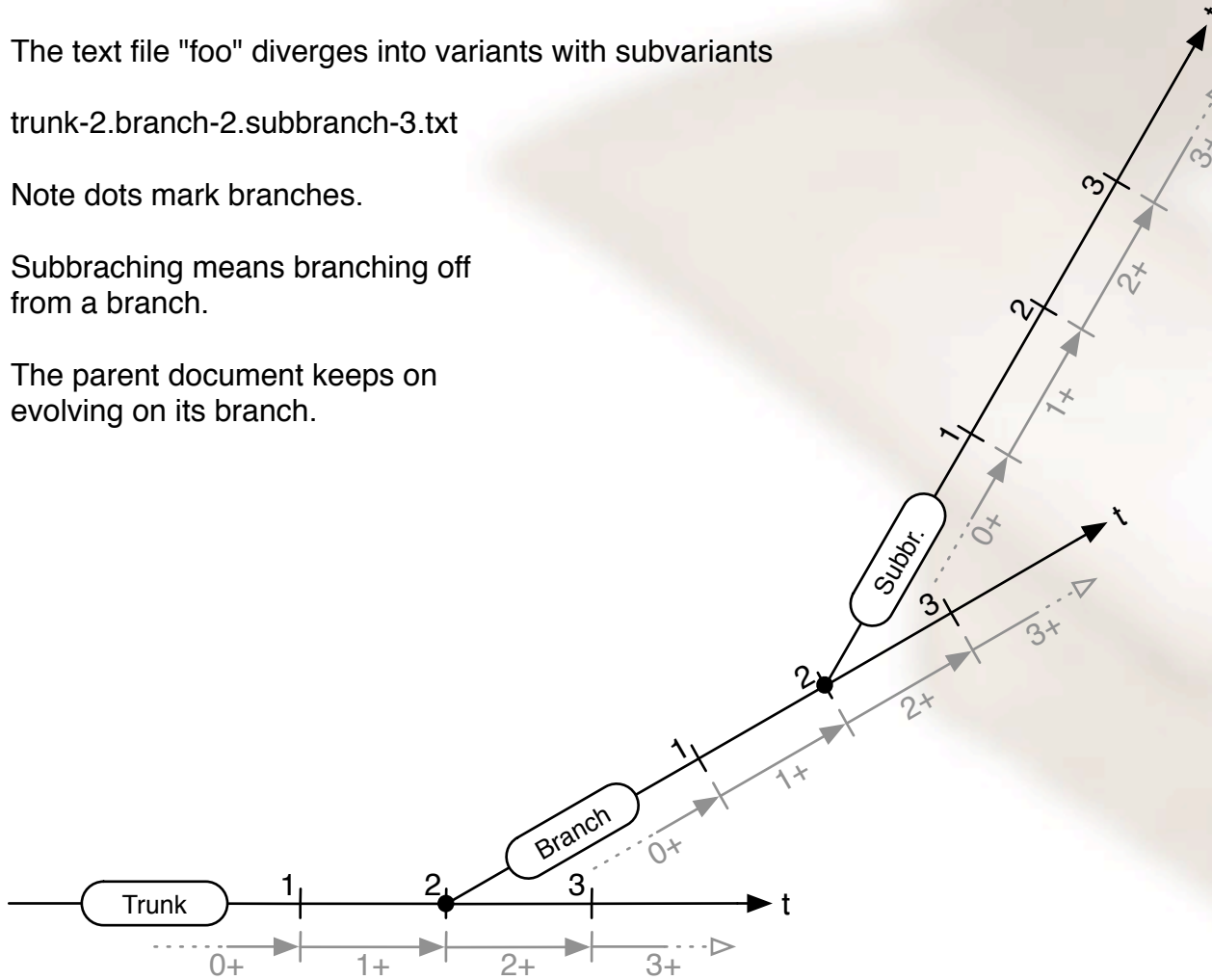
The text file "foo" diverges into variants with subvariants

trunk-2.branch-2.subbranch-3.txt

Note dots mark branches.

Subbranching means branching off from a branch.

The parent document keeps on evolving on its branch.



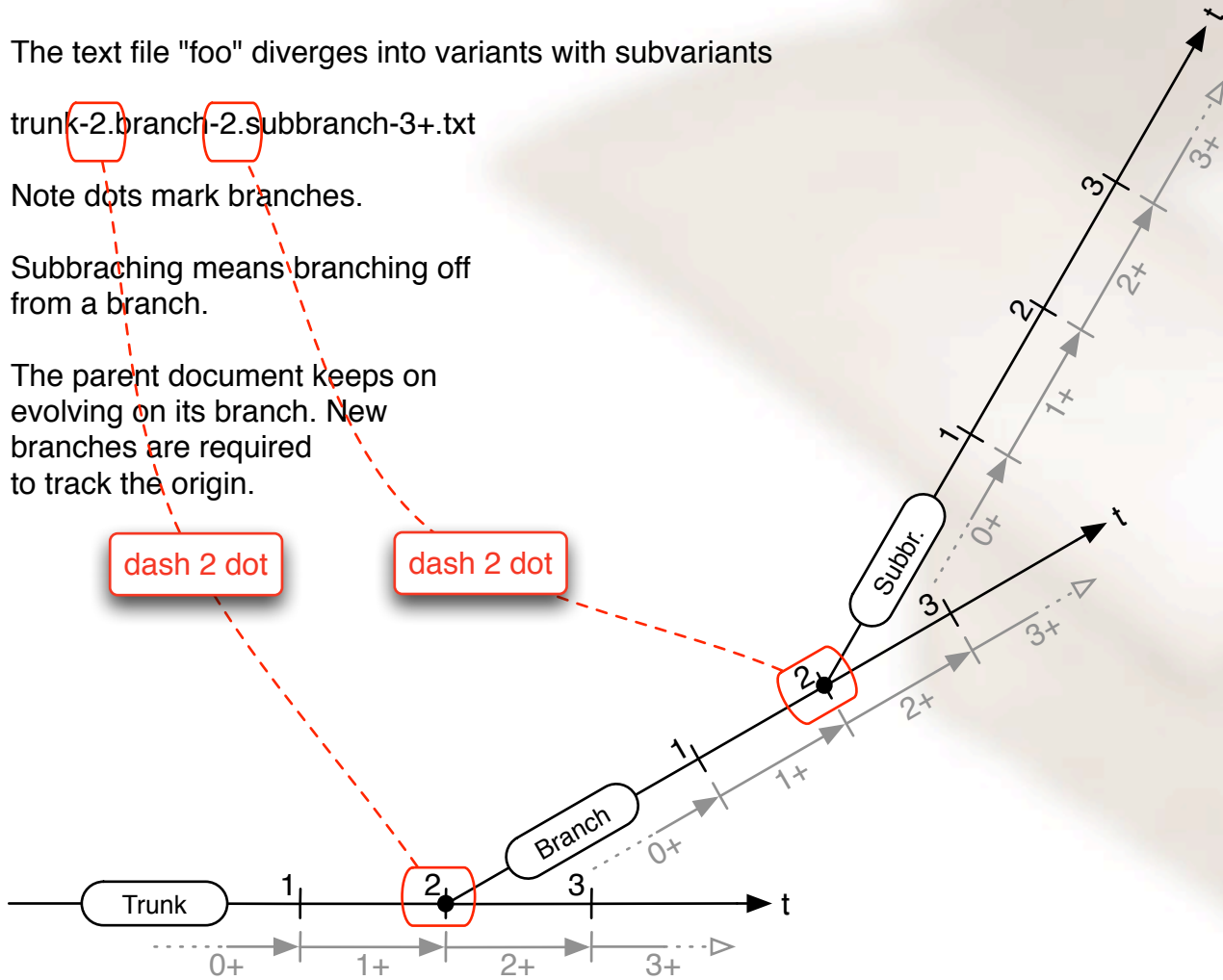
The text file "foo" diverges into variants with subvariants

trunk-2.branch-2.subbranch-3+.txt

Note dots mark branches.

Subbranching means branching off from a branch.

The parent document keeps on evolving on its branch. New branches are required to track the origin.



The text file "foo" diverges into variants with subvariants

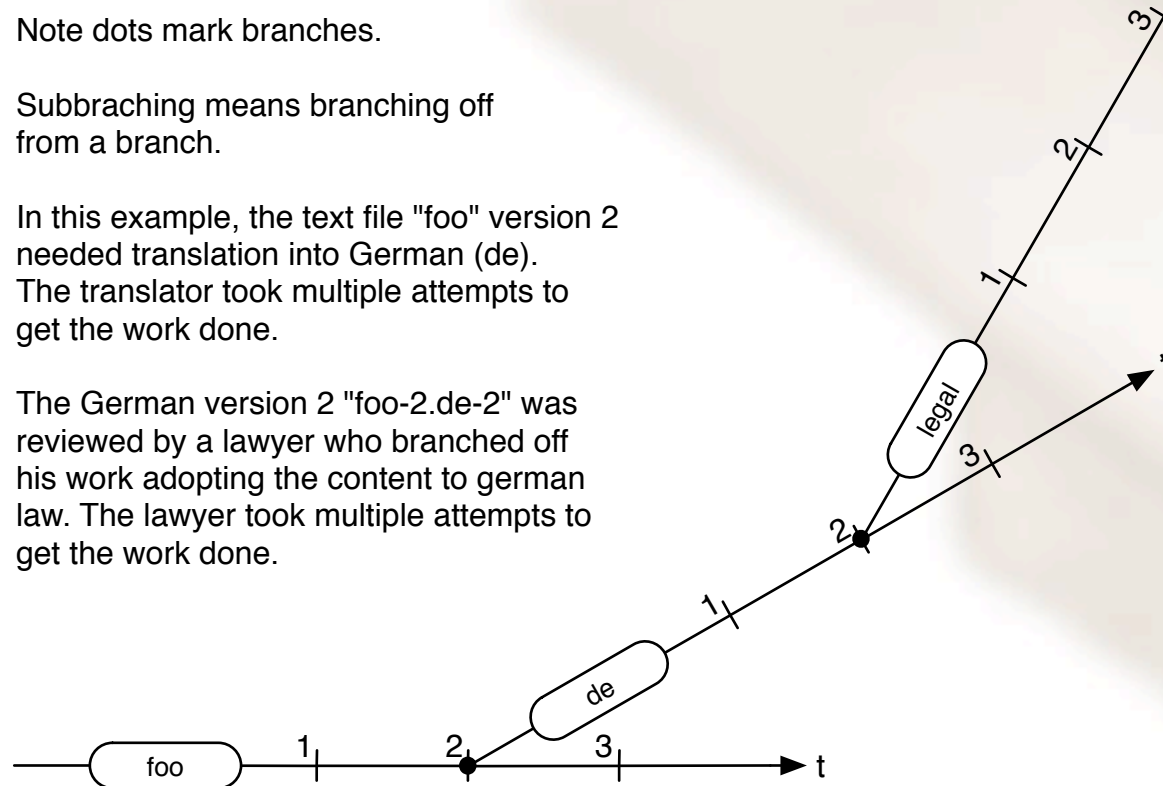
foo-2.de-2.legal-3.txt

Note dots mark branches.

Subbranching means branching off from a branch.

In this example, the text file "foo" version 2 needed translation into German (de). The translator took multiple attempts to get the work done.

The German version 2 "foo-2.de-2" was reviewed by a lawyer who branched off his work adopting the content to german law. The lawyer took multiple attempts to get the work done.

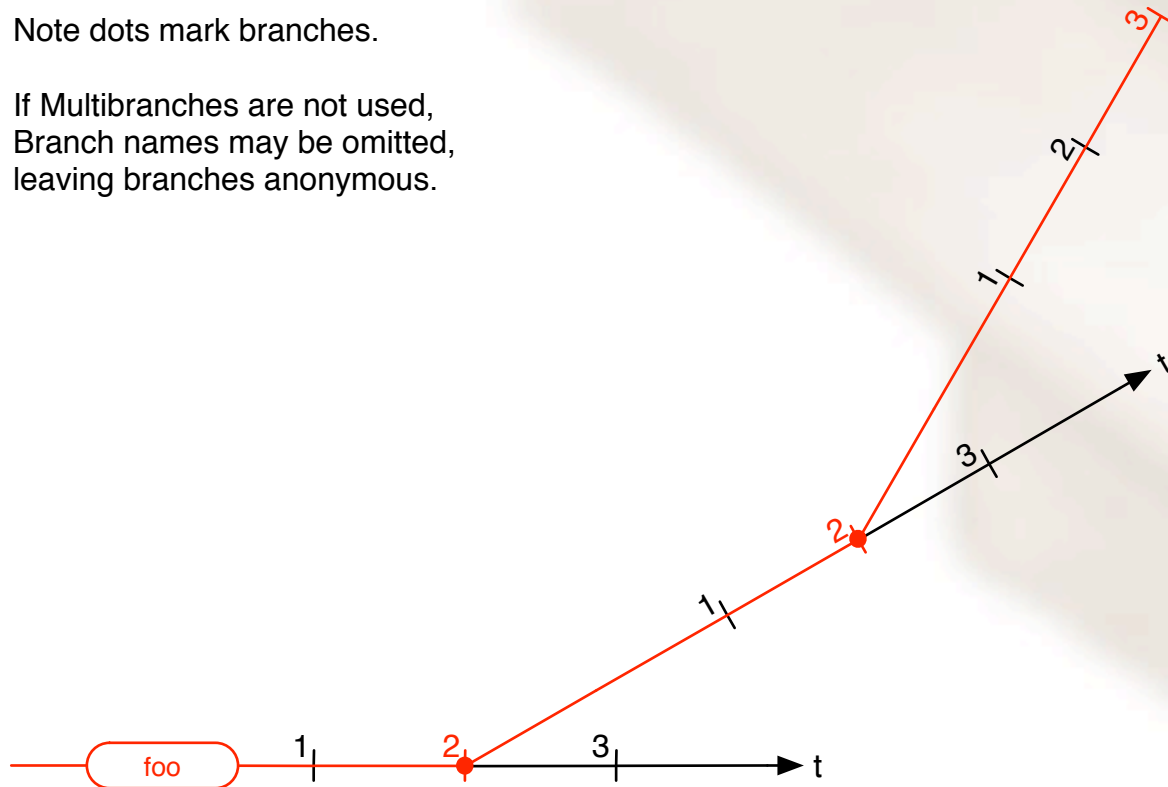


Simplification for Subbranches

foo-2.2.3.txt

Note dots mark branches.

If Multibranches are not used,
Branch names may be omitted,
leaving branches anonymous.



Branching

by example

```
animals-1.jpg  
animals-2.jpg  
animals-2.cats-0+.jpg  
animals-2.dogs-1.jpg  
animals-2.dogs-2.jpg  
animals-2.dogs-2.dalmatians-1.jpg  
animals-2.dogs-2.dobermann-1.jpg  
animals-2.dogs-2.dobermann-2.jpg  
animals-2.dogs-2.dobermann-3.jpg  
animals-3.jpg
```



Merging

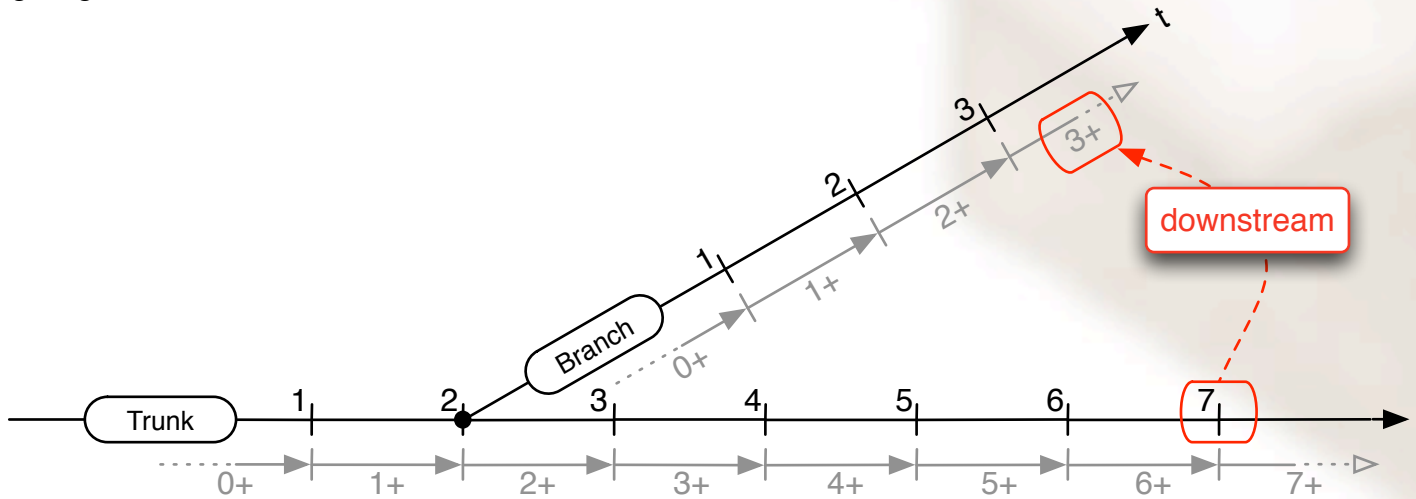


Merging changes from ancestor downstream to descendant

This approach assumes evolution creates major changes on trunk while maintenance creates minor changes on stable branches.

Useful evolution is then cherry-picked from ancestor (Trunk) and merged to descendant (Branch).

Here changes between Trunk-2 and Trunk-7 are merged downstream after Trunk-2.Branch-3 giving Trunk-2.Branch-3+

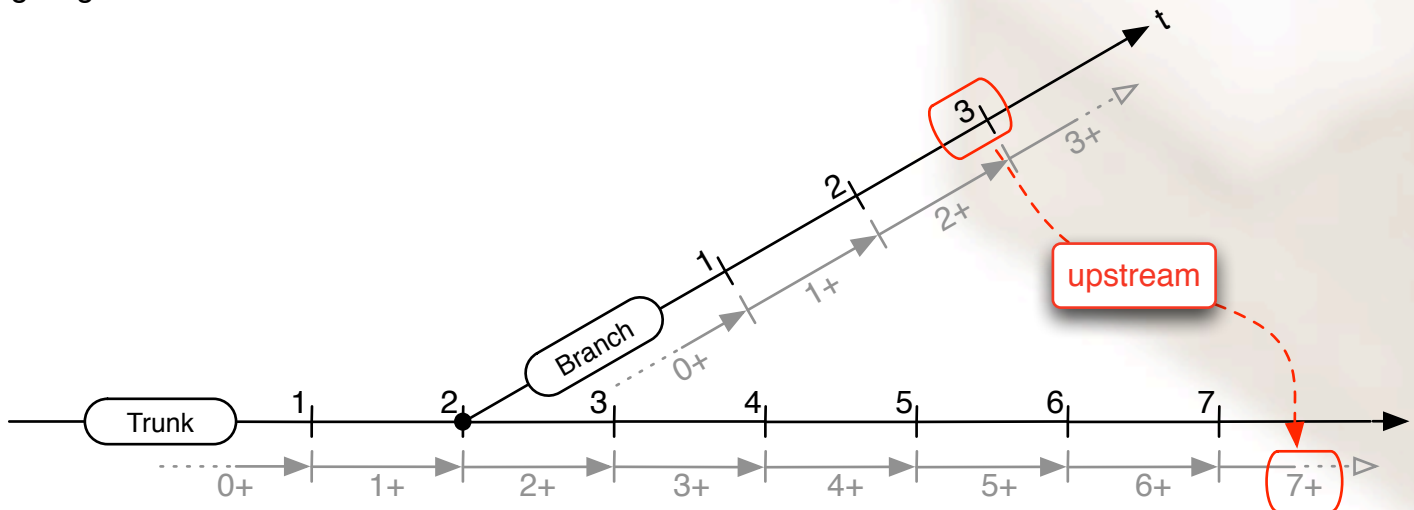


Merging changes from descendant upstream to ancestor

This approach assumes evolution creates major changes on branches while maintenance creates minor changes on stable trunk.

Useful evolution is then cherry-picked from descendant (Branch) and merged to ancestor (Trunk).

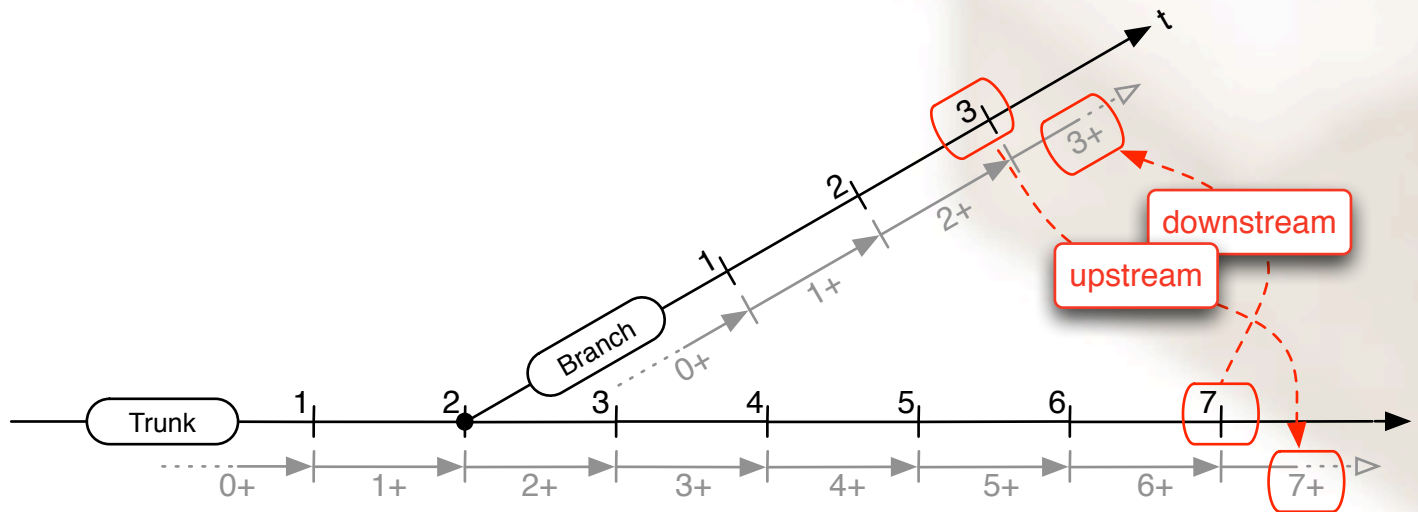
Here changes between Trunk-2.Branch-0 (aka Trunk-2) and Trunk-2.Branch-3 are merged upstream after Trunk-7 giving Trunk-7+



Cross merging changes between ancestor (Trunk) and descendant (Branch)

This approach assumes evolution on trunk and branches.

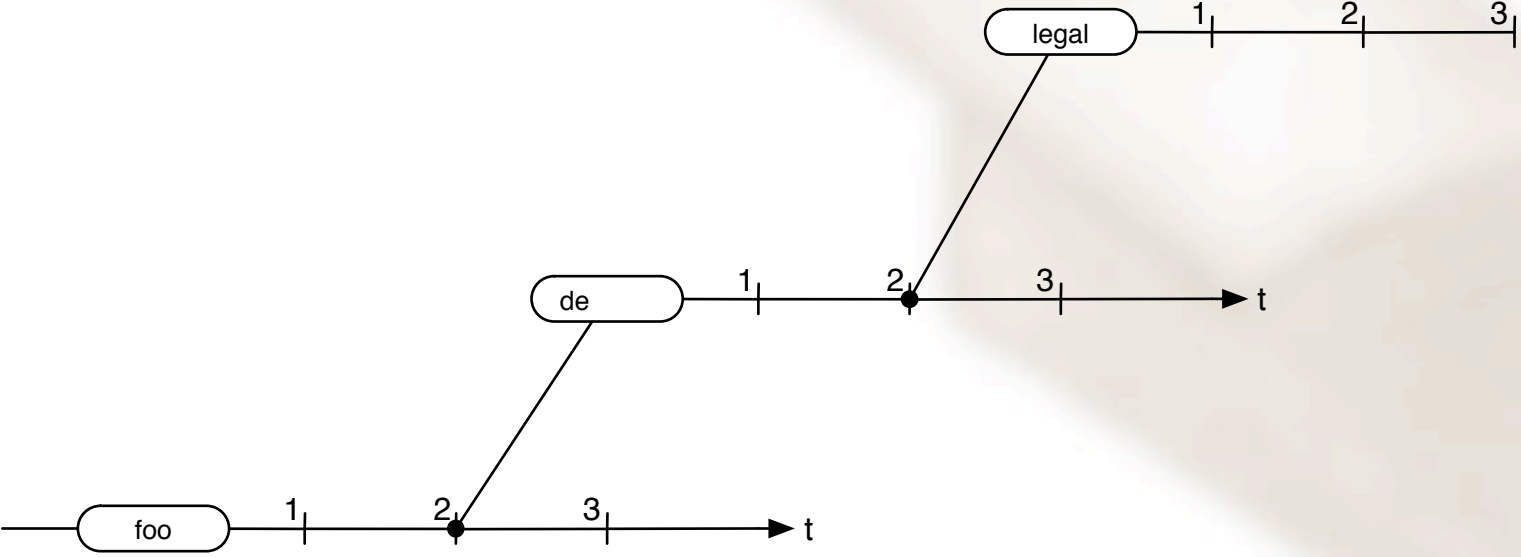
This is very hard to handle!



Science



The more scientific representation of branches is to draw time lines in parallel



Lotti's Practical Versioning Scheme (LPVS) is mentioned as "Document Unified Version Set" in Ralf S. Engelschall's Capgemini sd&m Research paper "Unified Version Identification for Software Configuration Management Artifacts". It boils down to the "Unified Version Set"

$$V = \langle M, P, N, C, T, S, L \rangle$$

and proposes the following simple but efficient versioning scheme V_D for standalone documents:

```
VD = V+(
  <RegExp("[[:alpha:]]+[[:alnum:]]*" ) ∪ ∅, [0, ∞[>j , // major version
  ∅, // phase unused
  ∅, // minor version unused
  <work-in-progress, released>, // class
  ∅, // time unused
  <TOPSECRET, SECRET, CONFIDENTIAL, RESTRICTED, PUBLIC>, // scope
  ∅ // label unused
)
```

The recommended formatting into the corresponding textual representation — which intentionally directly forms the filename of the document — is:

```
v2t(v) = concat(
  join(map(v.m, 1, j, map_m), 1, j, "."),
  if v.c = "work-in-progress" then "+" else "",
  if v.s = "RESTRICTED" then "" else concat("=", v.s)
)
map_m(m) = if m.branch ≠ "" then concat(m.branch, "-", m.version) else m.version
```

