# Conflicts in the Collaborative Development of Variability-Intensive Software

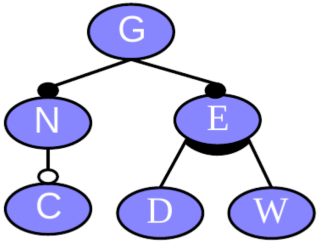
Sandra Greiner[1] Jan Koch[1] Timo Kehrer[1] Christoph Seidl[2]

[1]Software Engineering Group – University of Bern, Switzerland

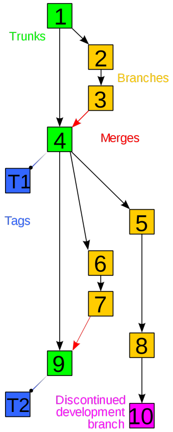
[2]SQUARE Group – ITU Copenhagen, Denmark

**FOSD Meeting, April 2024, Eindhoven, NL**

## Management of Variability in Space ...



# Management of Variability in Space ... and Time



https://commons.wikimedia.org/w/index.php?curid=9562807

https://www.pngall.com/wp-content/uploads/2016/07/Team-Work-PNG-File.png

*By Revision_controlled_project_visualization.svg: *Subversion_project_visualization.svg: Traced by User:Stannered, original by en:User:Sami Keroladerivative work: Moxfyre (talk)derivative work: Echion2 (talk) - Revision_controlled_project_visualization.svg, CC BY-SA 3.0*

## Distributed and Collaborative Single-Product Development

```
public class Graph {
  List<Node> getNodes()
    { ... }
  List<Edge> getEdges()
    { ... }
}
```

```
public class Graph {
  List<Node> getNodes(String l)
    { ... }
  List<Edge> getEdges()
    { ... }
}
```

T: Timo

```
public class Graph {
  List<Node> getNodes()
    { ... }
  List<Edge> getEdges(double w)
    { ... }
}
```

A: Anna

⇒ Conflicts may arise, VCS with many strategies to resolve them

# Distributed and Collaborative Multi-Variants Development

```
public class Graph {
  List<Node> getNodes()
    { ... }
  List<Edge> getEdges()
    { ... }
}
```

```
public class Graph {
  List<Node> getNodes()
    { ... }

  // #IFDEF Edges
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF
}
```

T: Timo

**resolution??**

```
public class Graph {
  List<Node> getNodes()
    { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF
}
```

A: Anna

$\Rightarrow$ do not always occur, may involve different semantics, ...

$\Rightarrow$ more expensive to handle

# Merging Variability-Intensive Software

Conceptually

1) Matching
2) Conflict detection (and classification)
3) Conflict resolution

## Theoretic Merge Scenarios

single-product development

```
public class Graph {
  List<Node> getNodes()
    { ... }
  List<Edge> getEdges()
    { ... }
}
```

```
public class Graph {
  List<Node> getNodes(String l)
    { ... }
  List<Edge> getEdges()
    { ... }
}
```

```
public class Graph {
  List<Node> getNodes()
    { ... }
  List<Edge> getEdges(double w)
    { ... }
}
```

three-way merging
unstructured vs. semi-structured vs. structured

→ mostly well-studied (theoretically and in practice)

variability-intensive software development

```
public class Graph {
  List<Node> getNodes()
    { ... }
  List<Edge> getEdges()
    { ... }
}
```

```
public class Graph {
  List<Node> getNodes()
    { ... }

  // #IFDEF Edges
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF
}
```

```
public class Graph {
  List<Node> getNodes()
    { ... }

  // #IFDEF Weighted
  List<Edge> getEdges(double w)
    { ... }
  // #ENDIF
}
```

resolution??

syntactic conflict vs. 'semantic' conflict?
what kind of semantics:
Boolean expressions or with feature model?

## Merging Variability-Intensive Software

In Practice?

**RQ1** How prevalent are merge conflicts in real-world variability-intensive software?

**RQ2** Are there tendencies in solving the merge conflicts?

$\Rightarrow$ Can we find patterns and automated resolutions?

$\Rightarrow$ reduced burden for developers, higher automation, less errors

## Experiment

## Method

Behavior:

1) clone C/C++ repos from GitHub
2) iterate commit history
3) for each merge:

    check if the conflicting chunks contain an `#if` or `#define`

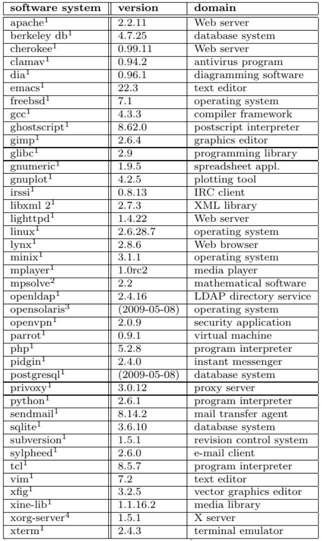
Counted numbers of:

commits
merges (with & without variability)
conflicting files and
conflicting chunks
chunks not taken at least one from parent

not considered:

variable source code (without annotation)

# Setup



| software system | version | domain |
|---|---|---|
| apache[1] | 2.2.11 | Web server |
| berkeley db[1] | 4.7.25 | database system |
| cherokee[1] | 0.99.11 | Web server |
| clamav[1] | 0.94.2 | antivirus program |
| dia[1] | 0.96.1 | diagramming software |
| emacs[1] | 22.3 | text editor |
| freebsd[1] | 7.1 | operating system |
| gcc[1] | 4.3.3 | compiler framework |
| ghostscript[1] | 8.62.0 | postscript interpreter |
| gimp[1] | 2.6.4 | graphics editor |
| glibc[1] | 2.9 | programming library |
| gnumeric[1] | 1.9.5 | spreadsheet appl. |
| gnuplot[1] | 4.2.5 | plotting tool |
| irssi[1] | 0.8.13 | IRC client |
| libxml 2[1] | 2.7.3 | XML library |
| lighttpd[1] | 1.4.22 | Web server |
| linux[1] | 2.6.28.7 | operating system |
| lynx[1] | 2.8.6 | Web browser |
| minix[1] | 3.1.1 | operating system |
| mplayer[1] | 1.0rc2 | media player |
| mpsolve[2] | 2.2 | mathematical software |
| openldap[1] | 2.4.16 | LDAP directory service |
| opensolaris[3] | (2009-05-08) | operating system |
| openvpn[1] | 2.0.9 | security application |
| parrot[1] | 0.9.1 | virtual machine |
| php[1] | 5.2.8 | program interpreter |
| pidgin[1] | 2.4.0 | instant messenger |
| postgresql[1] | (2009-05-08) | database system |
| privoxy[1] | 3.0.12 | proxy server |
| python[1] | 2.6.1 | program interpreter |
| sendmail[1] | 8.14.2 | mail transfer agent |
| sqlite[1] | 3.6.10 | database system |
| subversion[1] | 1.5.1 | revision control system |
| sylpheed[1] | 2.6.0 | e-mail client |
| tcl[1] | 8.5.7 | program interpreter |
| vim[1] | 7.2 | text editor |
| xfig[1] | 3.2.5 | vector graphics editor |
| xine-lib[1] | 1.1.16.2 | media library |
| xorg-server[4] | 1.5.1 | X server |
| xterm[1] | 2.4.3 | terminal emulator |

analysis of 40 preprocessor-based projects `[Liebig'10]`

skip those without any **conflict** involving variability (total: 22)
e.g.,

- Berkley DB (only 7 public commits)
- Apache HTTP
- …

`https://dl.acm.org/doi/10.1145/1806799.1806819`

## Results

## RQ1: Prevalence

*How prevalent are merge conflicts in real-world variability-intensive software?*

Not frequent in most of the projects
but in certain ones, very prominent (e.g., free-bsd)

## RQ2: Tendencies

*Are there tendencies in solving the merge conflicts?*

mostly at least one chunk in resolution stems from parents

unclear: how *many* chunks are taken over

## Discussion

## Discussion

**Points to consider**

- open-source vs closed-source projects
- developing practices and guidelines, e.g.,
  - public branch will be synchronized only with stable updates

**Future Work**

- check in detail where the resolution comes from?
- analyze non-C++ repositories
- influence of user or development habits?
- study of closed-source projects

$\Rightarrow$ derive automated merge resolutions for variability conflicts