Databases and Software Engineering

Hochschule Harz
Harz University of Applied Sciences

MENSCH | TECHNIK
ORGANISATION | PLANUNG

Foundations of Collaborative, Real-Time Feature Modeling

Elias Kuiter*, Sebastian Krieter*, Jacob Krüger*, Thomas Leich, Gunter Saake

University of Magdeburg, Harz University of Applied Sciences, METOP GmbH

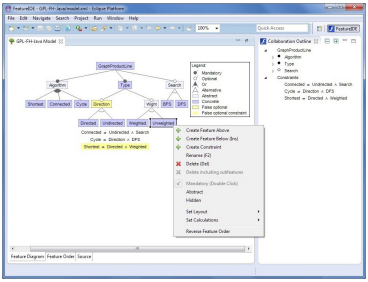
* Supported by pure-systems GmbH

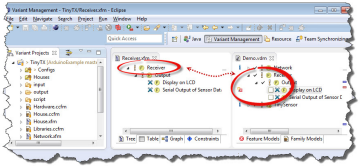
SPLC 2019

September 9–13 | Paris, France

Motivation

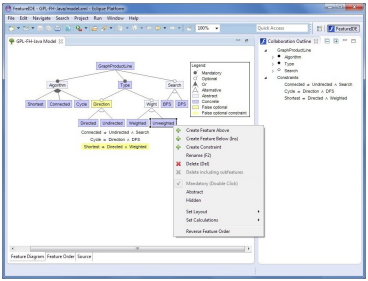
- State of the art: **single-user feature modeling**

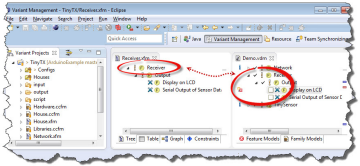




Motivation

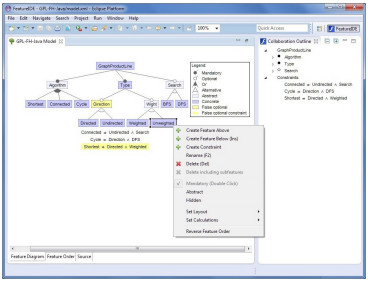
- State of the art: **single-user feature modeling**
- Multiple engineers may want work together

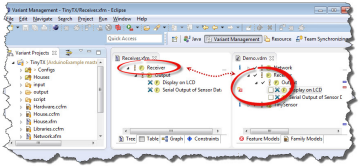




Motivation

- State of the art: **single-user feature modeling**
- Multiple engineers may want work together
- **But:** No dedicated support for collaboration

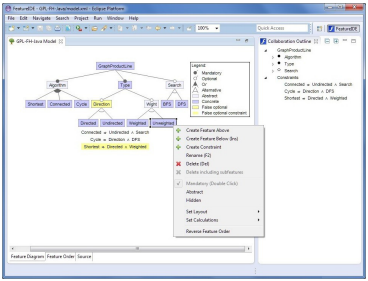


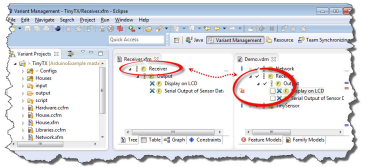


Motivation

- State of the art: **single-user feature modeling**
- Multiple engineers may want work together
- **But:** No dedicated support for collaboration

- One solution: *Asynchronous collaboration* with VCS

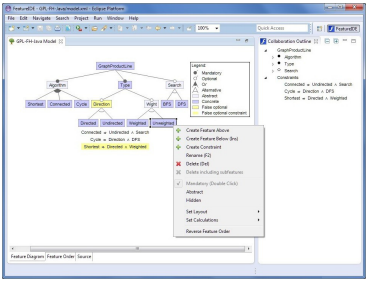


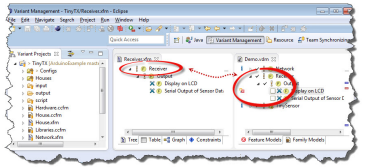


Motivation

- State of the art: **single-user feature modeling**
- Multiple engineers may want work together
- **But:** No dedicated support for collaboration

- One solution: *Asynchronous collaboration* with VCS
- **But:**
  - not real-time
  - promotes divergence





Motivation

Why collaborate?

- Domain knowledge is typically spread across **different collaborators**
  $\Rightarrow$ Leverage group synergies for problem solving

Motivation

Why collaborate?

- Domain knowledge is typically spread across **different collaborators**
  $\Rightarrow$ Leverage group synergies for problem solving

Why real-time?

- Engineers can discuss the feature model with domain experts
  $\Rightarrow$ **Real-time feedback**

Motivation

Why collaborate?

- Domain knowledge is typically spread across **different collaborators**
  $\Rightarrow$ Leverage group synergies for problem solving

Why real-time?

- Engineers can discuss the feature model with domain experts
  $\Rightarrow$ **Real-time feedback**
- Allows **tight collaboration** on shared model elements
  (similar to pair programming)

Motivation

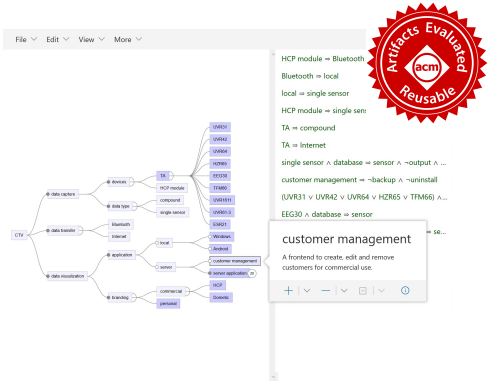
Our contribution:

1. Formal approach to
   collaborative real-time feature modeling
   with focus on
   - Consistency maintenance
   - Conflict detection (& resolution)

Motivation

Our contribution:

1. Formal approach to **collaborative real-time feature modeling** with focus on
   - Consistency maintenance
   - Conflict detection (& resolution)
2. Open-source research prototype



Requirements

Assumptions

- (Potentially) simultaneous edits

Requirements

Assumptions

- (Potentially) simultaneous edits
- Remotely connected

Requirements

Assumptions

- (Potentially) simultaneous edits
- Remotely connected
- Small group of collaborators

Requirements

Assumptions

- (Potentially) simultaneous edits
- Remotely connected
- Small group of collaborators

Which leads to ...

Requirements

- Concurrency

Requirements

- Concurrency ⇒ conflict detection

Requirements

- Concurrency $\Rightarrow$ conflict detection
- Intention Preservation

Requirements

- Concurrency ⇒ conflict detection
- Intention Preservation ⇒ accommodate conflicts in versions

Requirements

- Concurrency ⇒ conflict detection
- Intention Preservation ⇒ accommodate conflicts in versions
- Optimism

Requirements

- Concurrency ⇒ conflict detection
- Intention Preservation ⇒ accommodate conflicts in versions
- Optimism ⇒ avoid wait time due to network latency

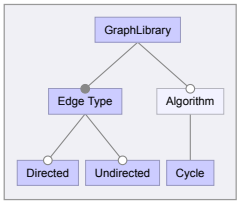
Requirements

Requirements

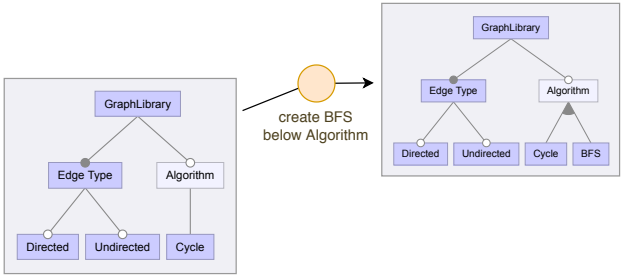
- Concurrency ⇒ conflict detection
- Intention Preservation ⇒ accommodate conflicts in versions
- Optimism ⇒ avoid wait time due to network latency

⇒ **Multi-Version Multi-Display**
(Sun and Chen, 2002)

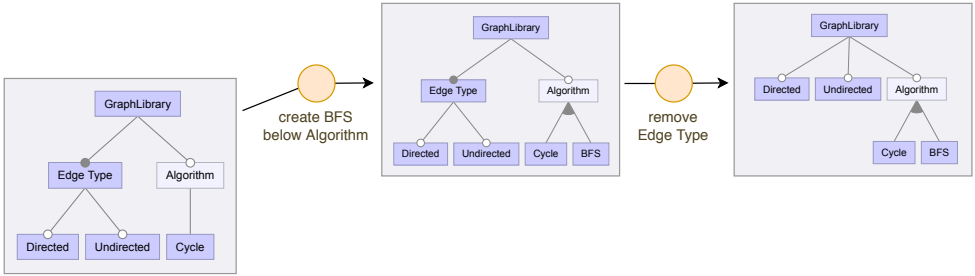
Running Example



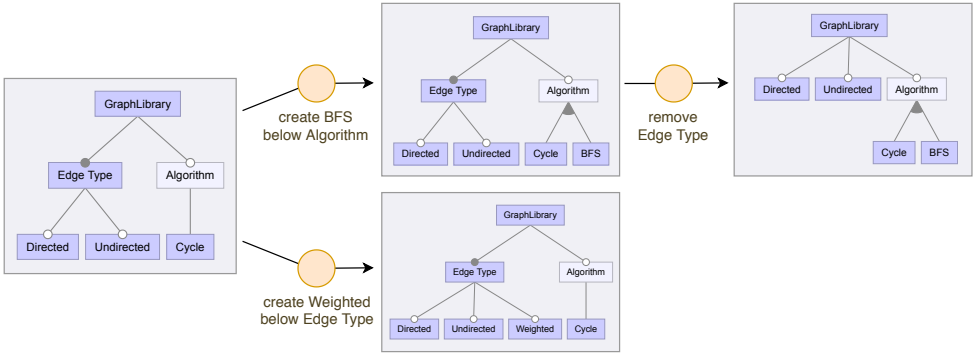
Running Example



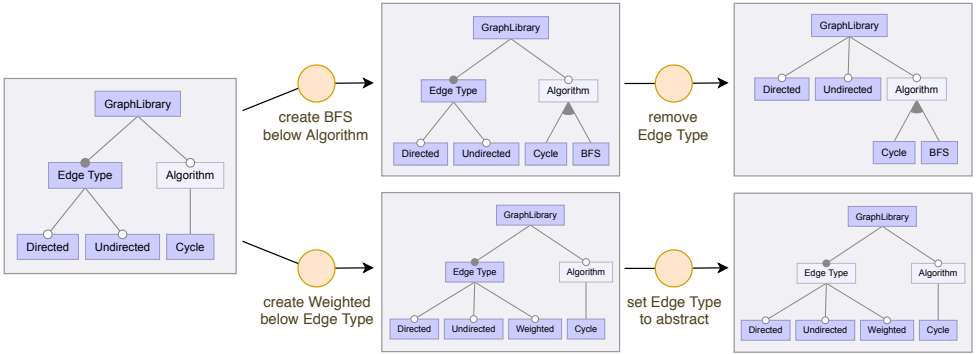
Running Example



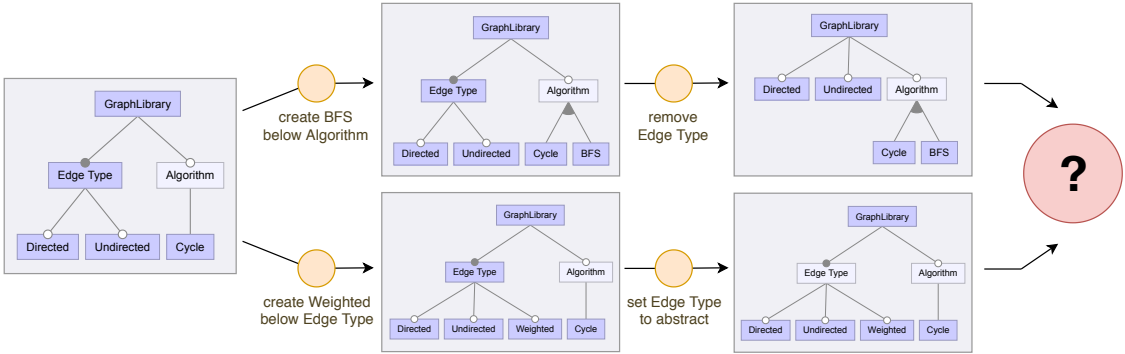
Running Example



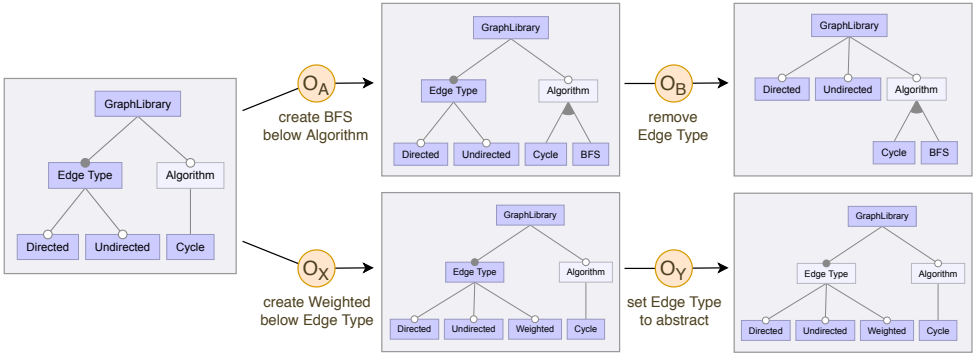
Running Example



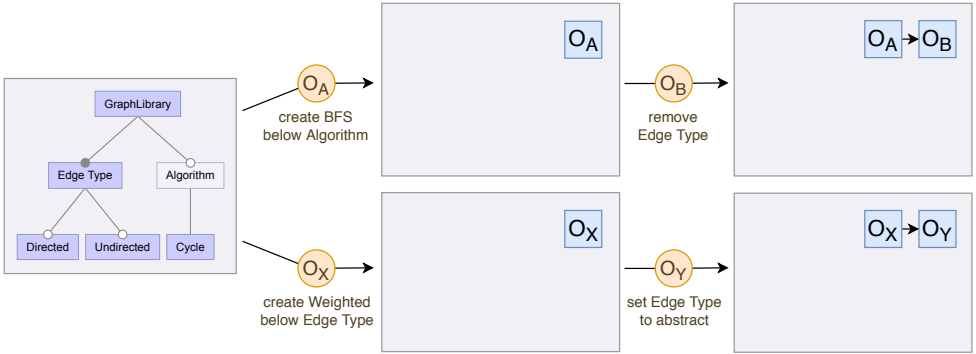
Running Example



Running Example

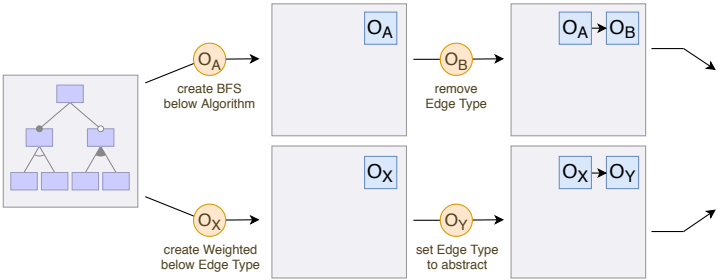


Running Example



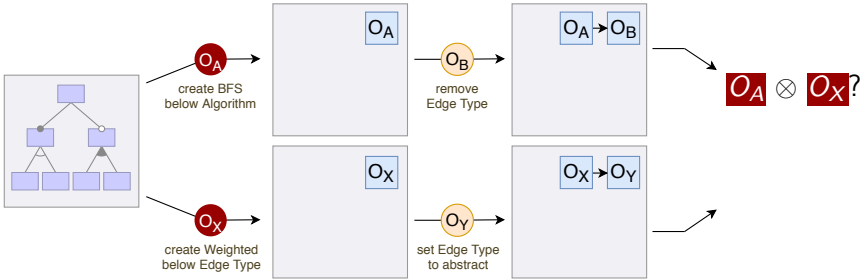
Feature Modeling Conflicts

How to determine conflicts algorithmically?



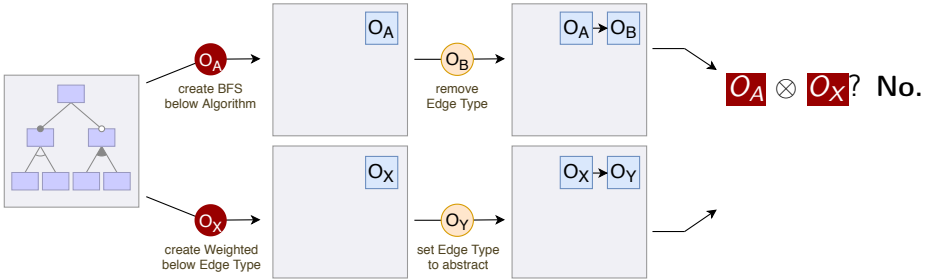
Feature Modeling Conflicts

How to determine conflicts algorithmically?



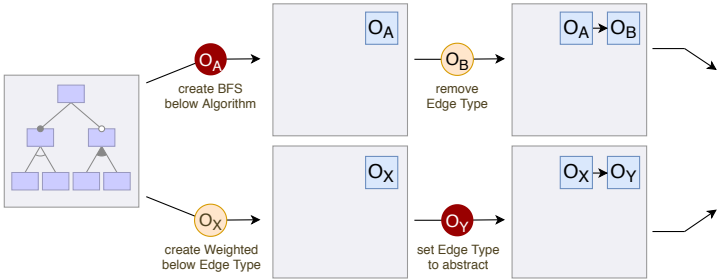
Feature Modeling Conflicts

How to determine conflicts algorithmically?



Feature Modeling Conflicts

How to determine conflicts algorithmically?

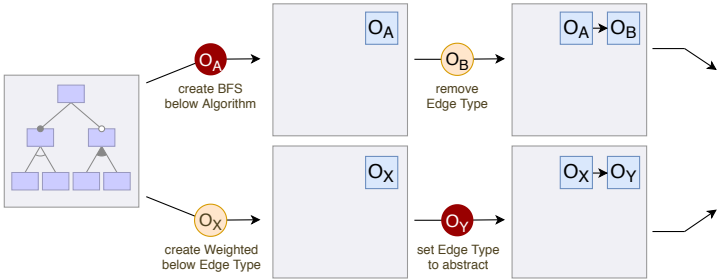


$O_A \otimes O_X$? **No.**

$O_A \otimes O_Y$, based on O_X?

Feature Modeling Conflicts

How to determine conflicts algorithmically?

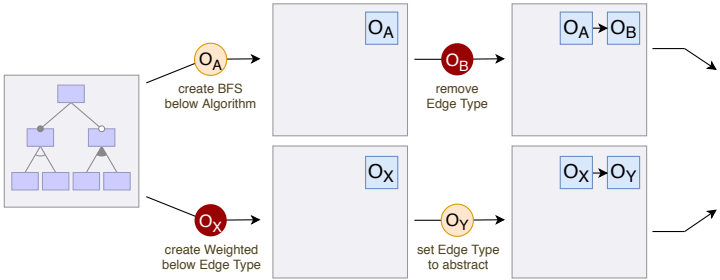


$O_A \otimes O_X$? **No.**

$O_A \otimes O_Y$, based on O_X? **No.**

Feature Modeling Conflicts

How to determine conflicts algorithmically?



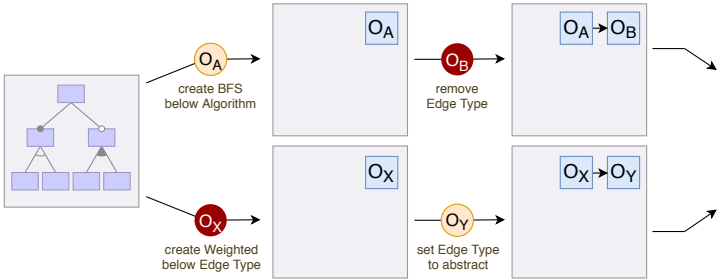
$O_A \otimes O_X$? **No.**

$O_A \otimes O_Y$, based on O_X? **No.**

$O_X \otimes O_B$, based on O_A?

Feature Modeling Conflicts

How to determine conflicts algorithmically?



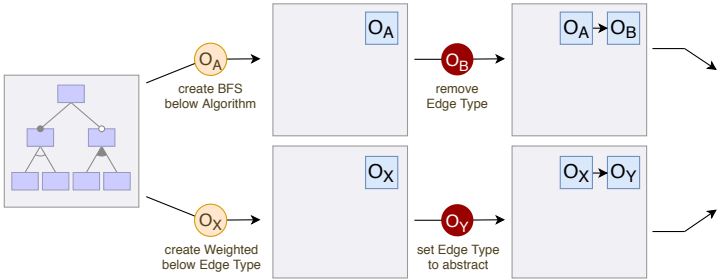
$O_A \otimes O_X$? **No.**

$O_A \otimes O_Y$, based on O_X? **No.**

$O_X \otimes O_B$, based on O_A? **Yes.**

Feature Modeling Conflicts

How to determine conflicts algorithmically?



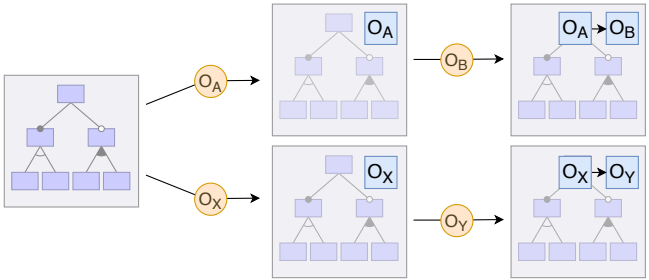
$O_A \otimes O_X$? **No.**

$O_A \otimes O_Y$, based on O_X? **No.**

$O_X \otimes O_B$, based on O_A? **Yes.**

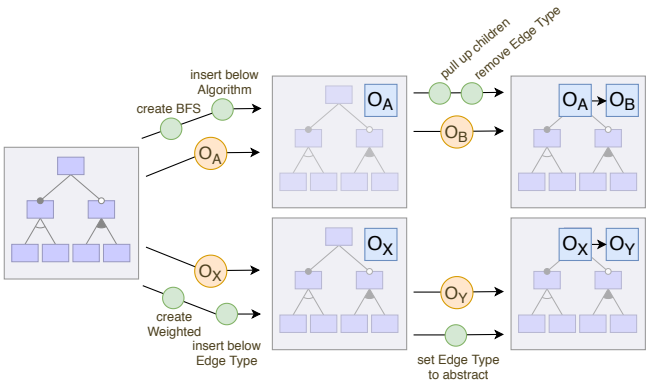
Thus, also $O_Y \otimes O_B$.

Feature Modeling Conflicts



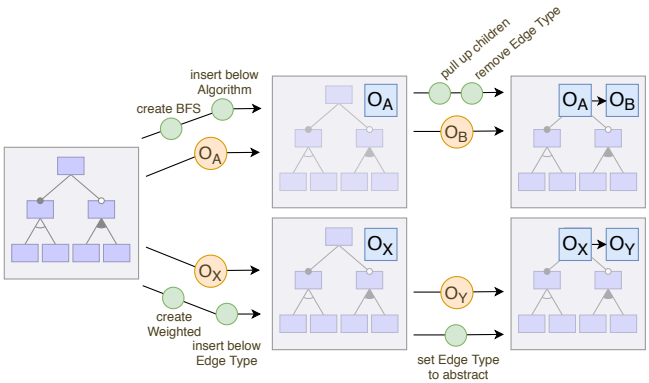
Decompose into *primitive operations*.

Feature Modeling Conflicts



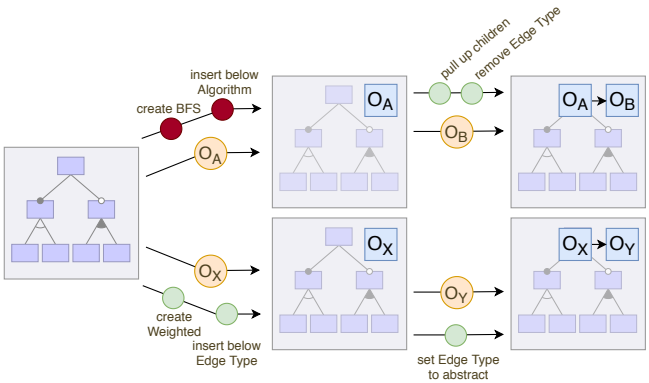
Decompose into *primitive operations*.

Feature Modeling Conflicts



Decompose into *primitive operations*.
$O_X \otimes O_B$ because:

Feature Modeling Conflicts

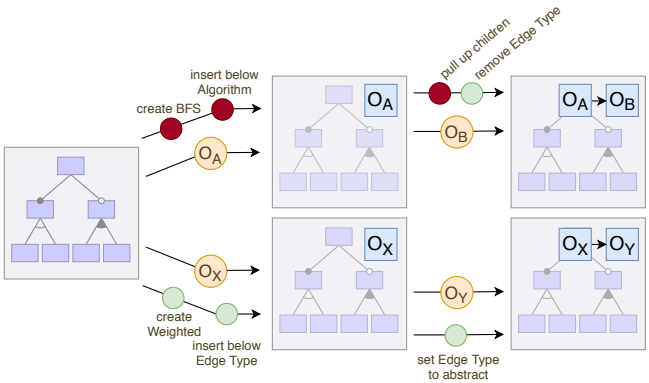


Decompose into *primitive operations*.

$O_X \otimes O_B$ because:

Based on O_A,

Feature Modeling Conflicts

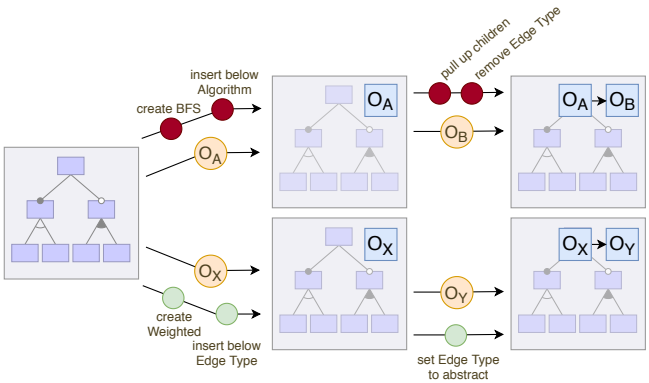


Decompose into *primitive operations*.

$O_X \otimes O_B$ because:

Based on O_A, apply O_B.

Feature Modeling Conflicts

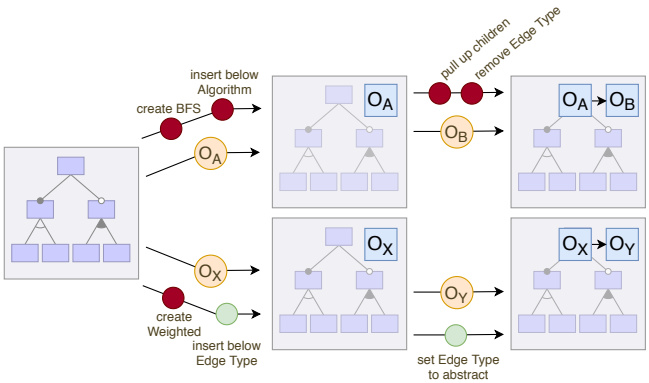


Decompose into *primitive operations*.

$O_X \otimes O_B$ because:

Based on O_A, apply O_B.

Feature Modeling Conflicts



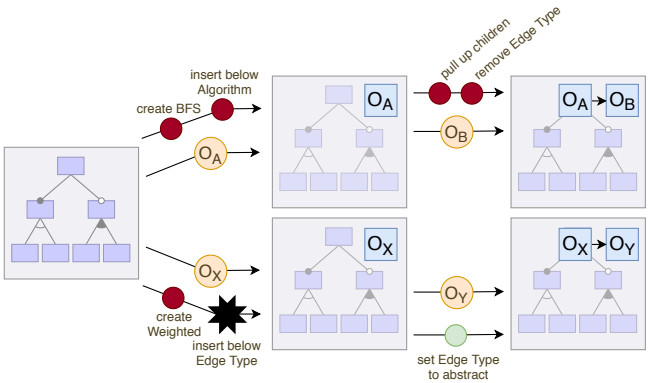
Decompose into *primitive operations*.

$O_X \otimes O_B$ because:

Based on O_A, apply O_B.

Now apply O_X ...

Feature Modeling Conflicts



Decompose into *primitive operations*.

$O_X \otimes O_B$ because:

Based on O_A, apply O_B.

Now apply O_X ...

... but a conflict rule applies.

Feature Modeling Conflicts

Version Creation

- Detected conflicts: $O_X \otimes O_B$ and $O_Y \otimes O_B$

Feature Modeling Conflicts

Version Creation

- Detected conflicts: $O_X \otimes O_B$ and $O_Y \otimes O_B$
- Applying the *Multi-Version Multi-Display* technique yields:

Feature Modeling Conflicts

Version Creation

- Detected conflicts: $O_X \otimes O_B$ and $O_Y \otimes O_B$
- Applying the *Multi-Version Multi-Display* technique yields:

Feature Modeling Conflicts

Version Creation

- Detected conflicts: $O_X \otimes O_B$ and $O_Y \otimes O_B$
- Applying the *Multi-Version Multi-Display* technique yields:

This technique ...

- ... preserves all intentions

Feature Modeling Conflicts

Version Creation

- Detected conflicts: $O_X \otimes O_B$ and $O_Y \otimes O_B$
- Applying the *Multi-Version Multi-Display* technique yields:

This technique ...

- ... preserves all intentions
- ... minimizes the number of versions

Feature Modeling Conflicts

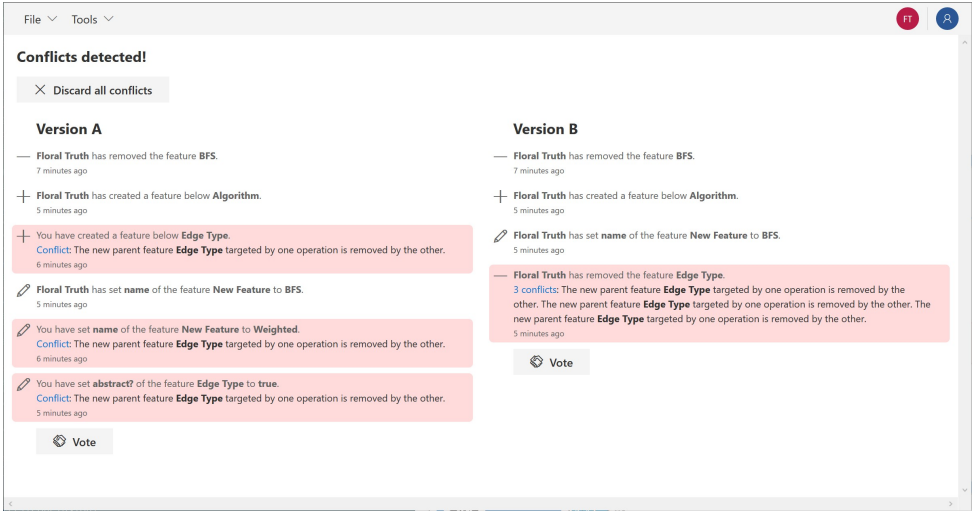
Version Creation

- Detected conflicts: $O_X \otimes O_B$ and $O_Y \otimes O_B$
- Applying the *Multi-Version Multi-Display* technique yields:

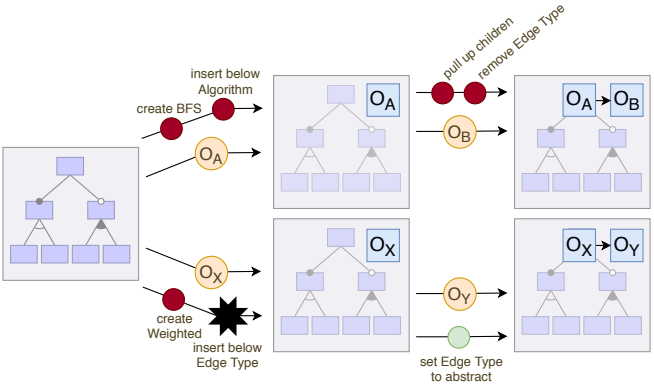
This technique ...

- ... preserves all intentions
- ... minimizes the number of versions
- ... maximizes the number of operations per version

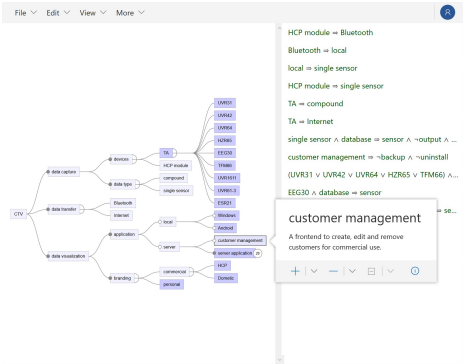
Conflict Resolution GUI



Conclusion



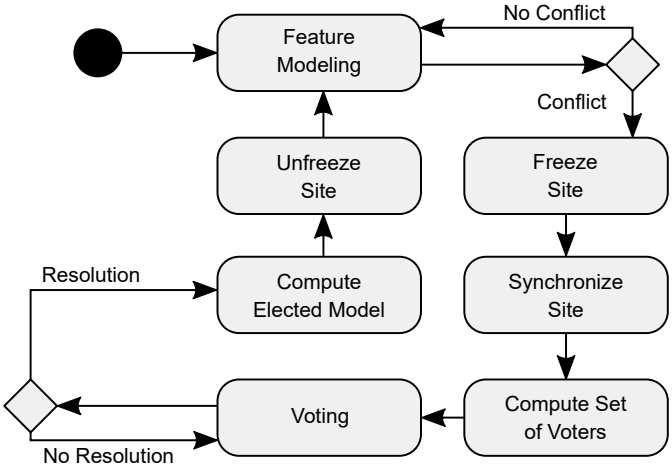
Formal description



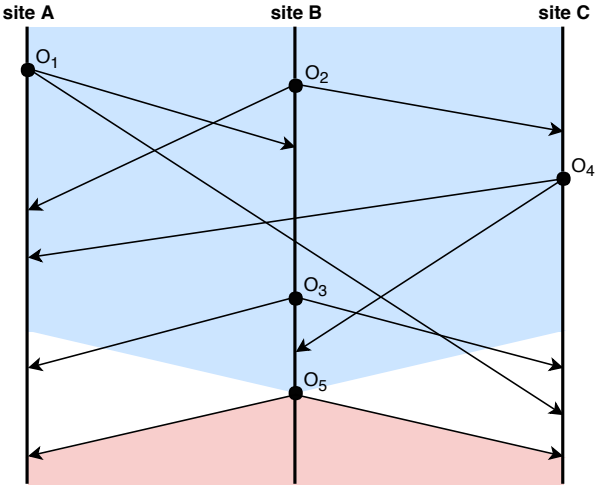
Open-source prototype

`github.com/ekuiter/variED`

Resolution Process



CCI Model



- Convergence
- Causality Preservation
- Intention Preservation

Concurrency Control

	Turn-Taking	Locking	CRDTs	Serialization	OT	MVSD	MVMD
Concurrency	○	◐	●	●	●	●	●
Optimism	◐	○	●	●	●	●	●
Intention Preservation	●	○	○	○	○	◐	●
Flexibility	●	○	○	○	○	○	◐
Correctness	●	◐	◐	●	○	○	◐

Architecture