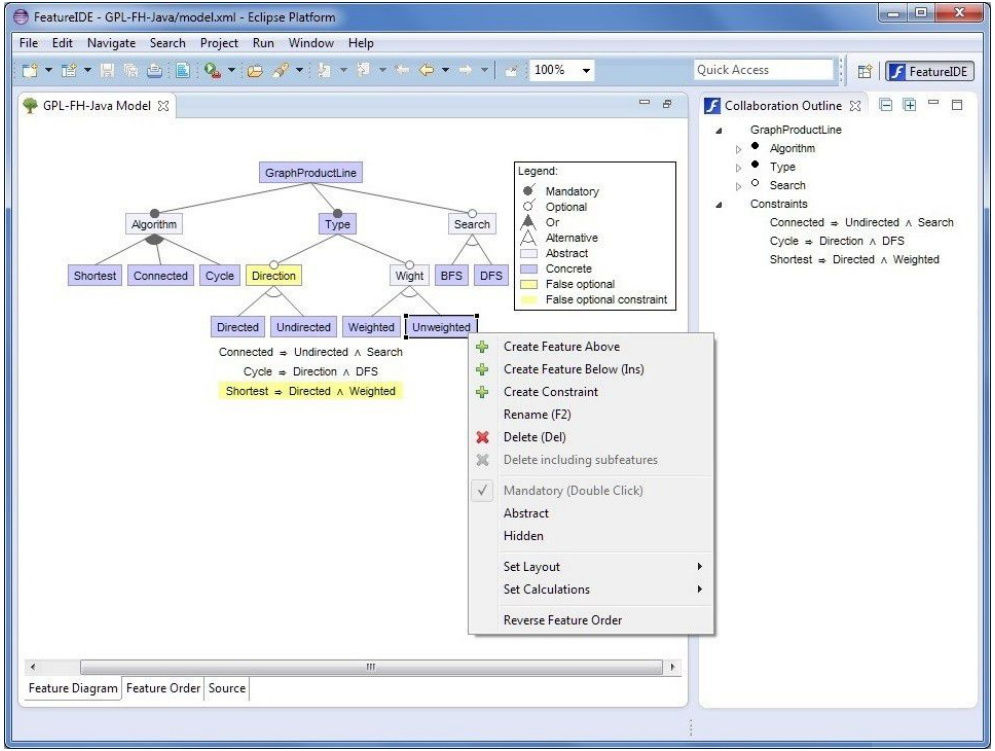
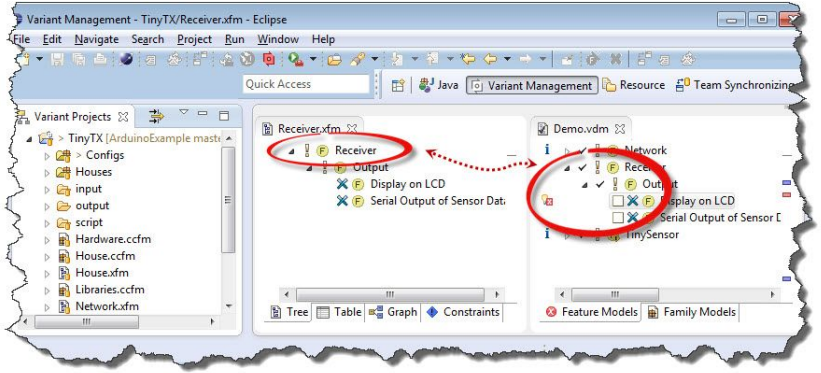
Foundations of Collaborative, Real-Time Feature Modeling

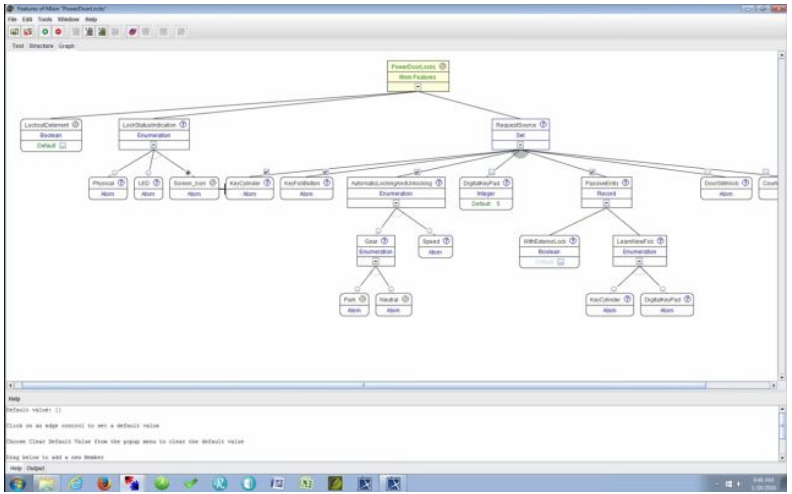
Elias Kuiter | FOSD Meeting 2019

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

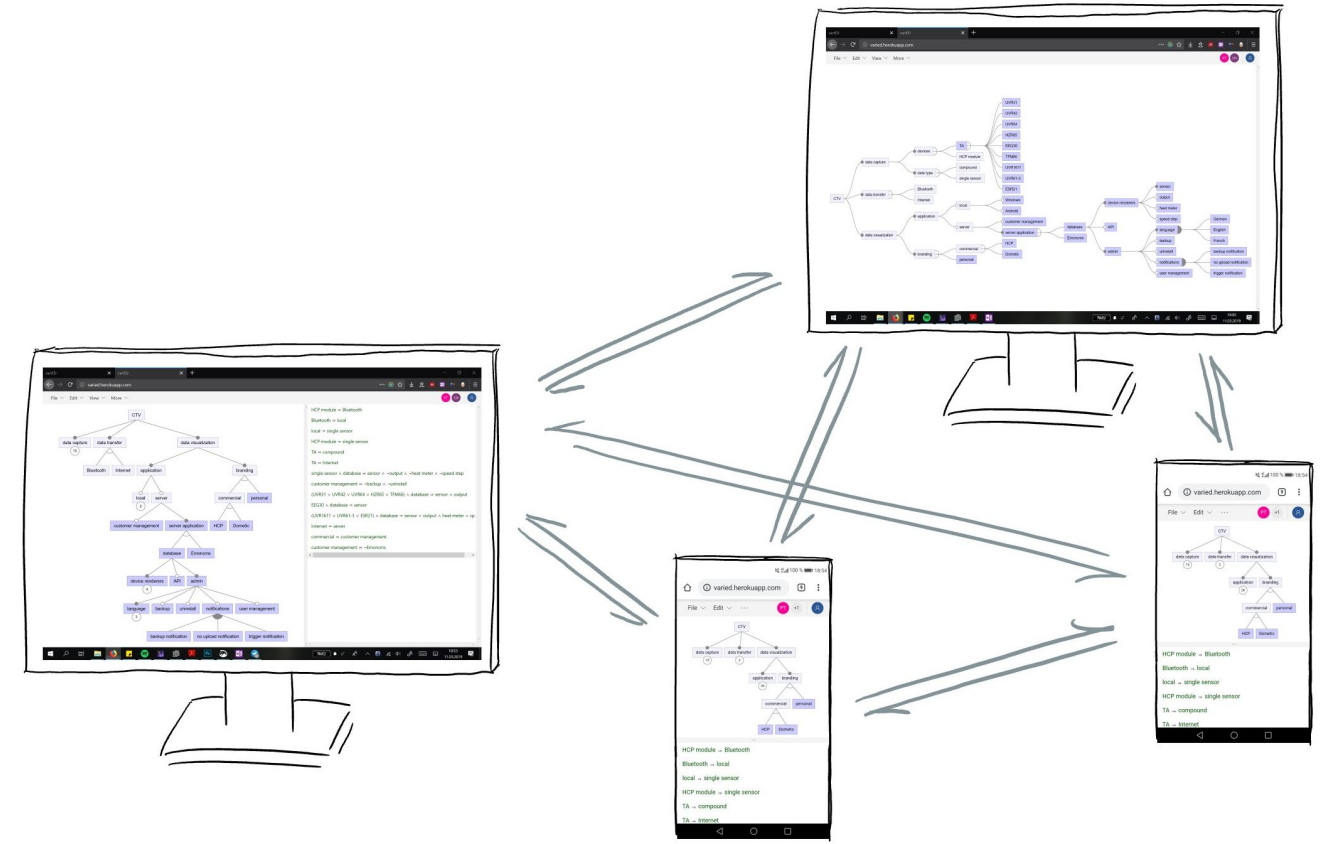
D B
S E
Databases
and
Software
Engineering







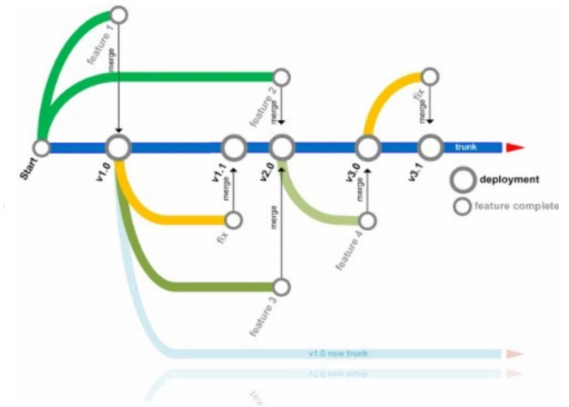
Single-user Feature Modeling Tools



Our Proposal: Collaborative Feature Modeling

Why?

- current tools do not explicitly address collaboration
- VCS allow *asynchronous collaboration*
  but: not real-time, divergence occurs
- imagine using a VCS for pair programming
  → annoying merge conflicts



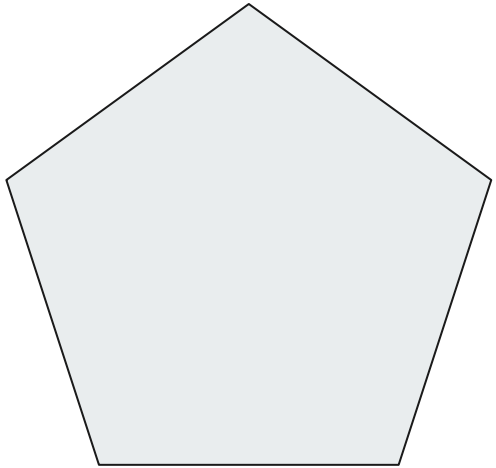
Potential Use Cases

engineers can share and discuss the feature model with domain experts
→ feedback can be used in real time for evolution

domain knowledge is usually spread across many stakeholders
→ synergy effects can help to solve tasks that are difficult for individuals

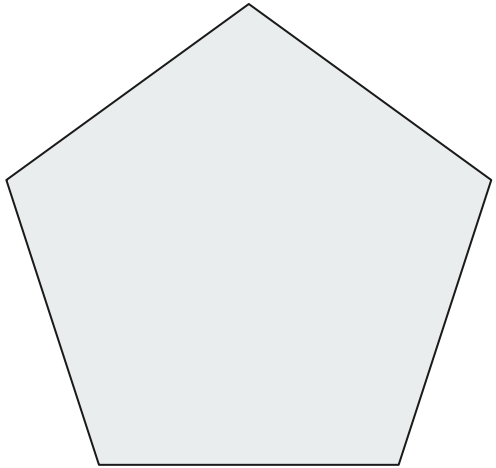
may complement version control systems
→ use a VCS for long-term, and a real-time editor for short-term evolution

What do we need? Requirements Analysis



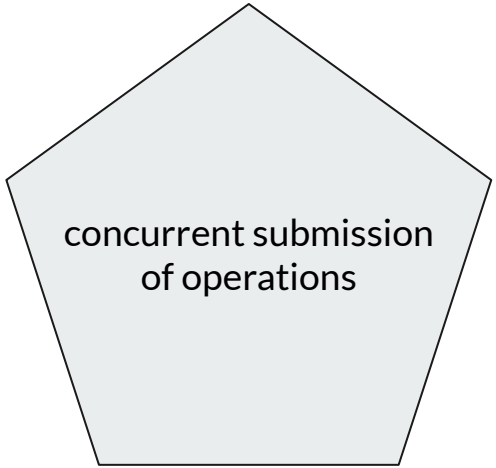
Requirements for a Collaborative FM Editor

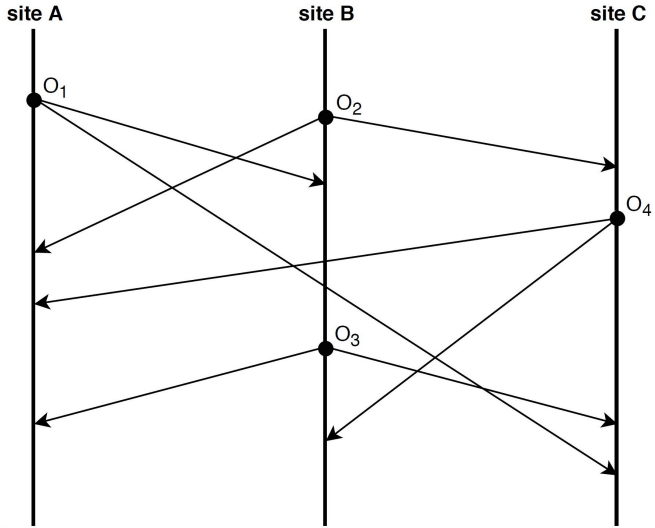
Concurrency

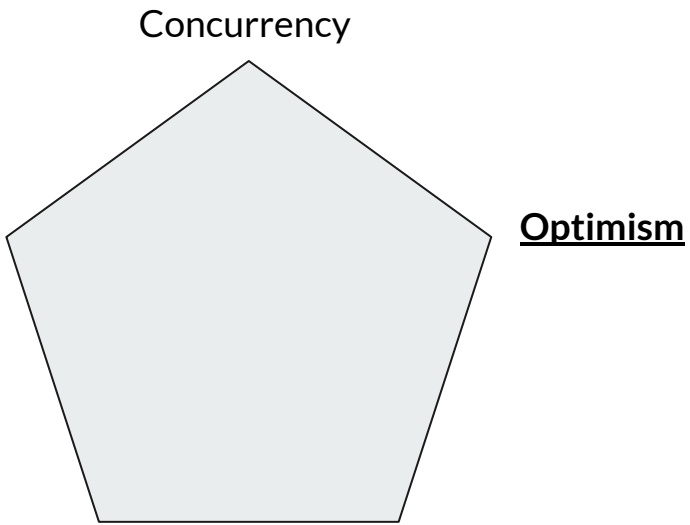


Requirements for a Collaborative FM Editor

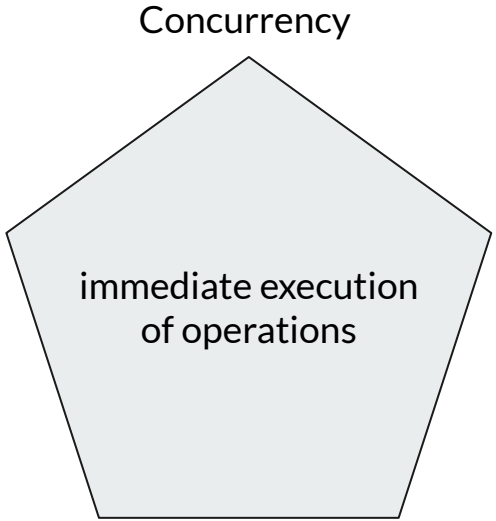
Concurrency





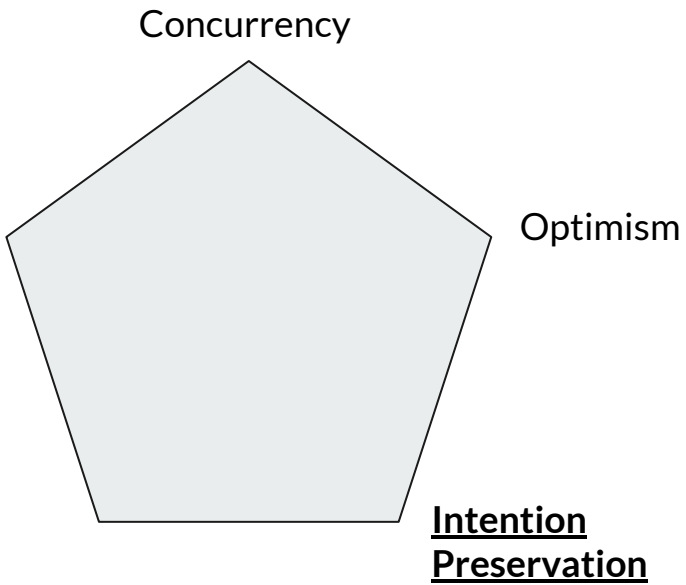


Requirements for a Collaborative FM Editor

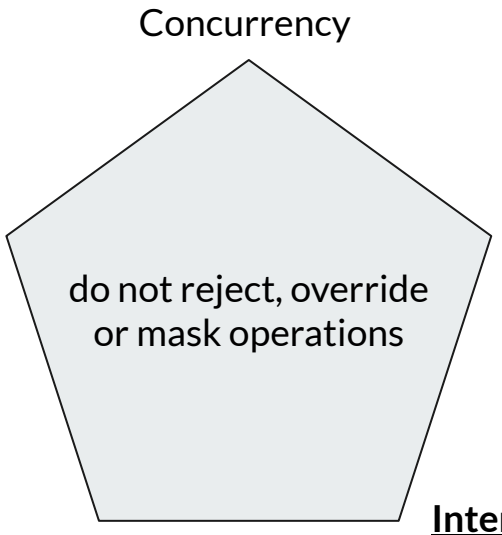

Concurrency
immediate execution
of operations

Optimism

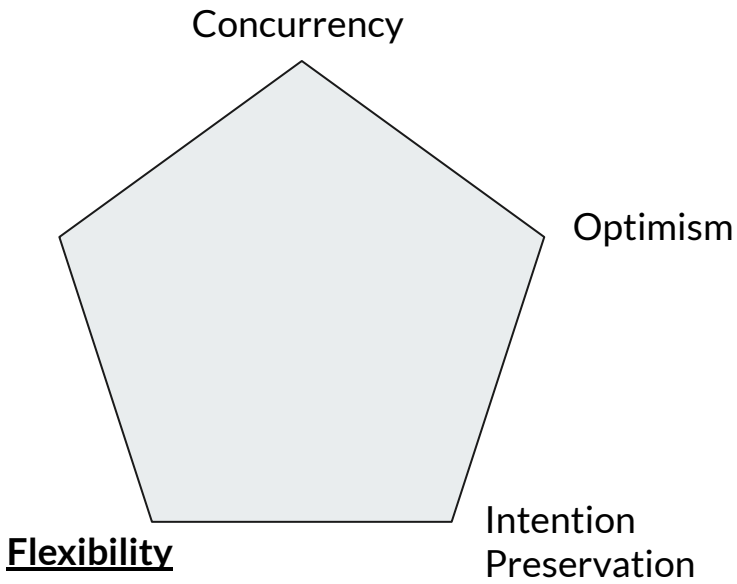

AS SOON AS POSSIBLE
ASAP
AS SOON AS POSSIBLE



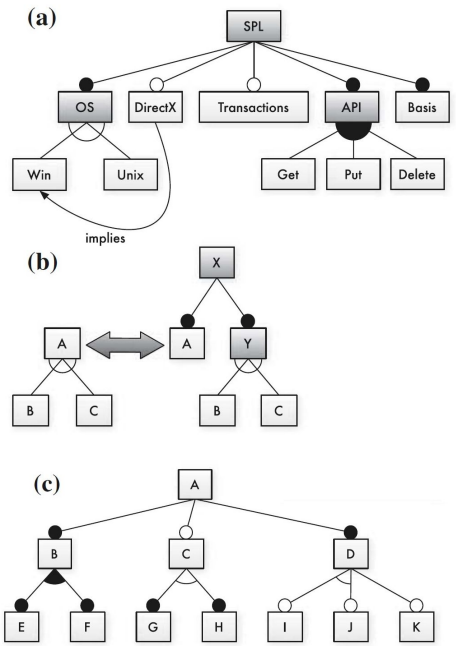
Requirements for a Collaborative FM Editor

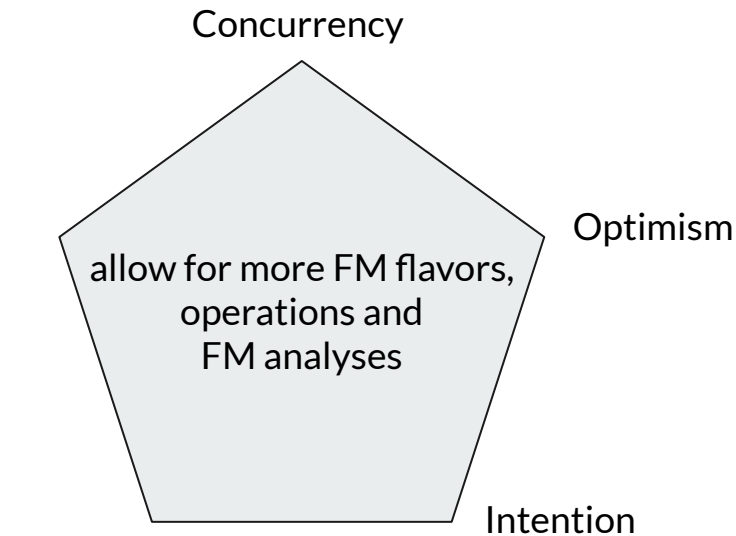

Concurrency
Optimism
do not reject, override
or mask operations
Intention
Preservation


This Way
No This Way

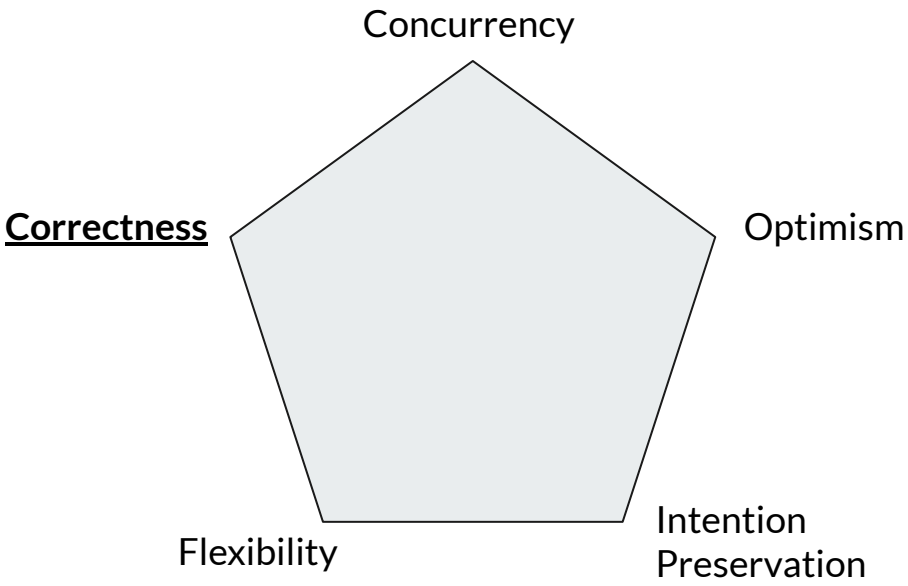


Requirements for a Collaborative FM Editor

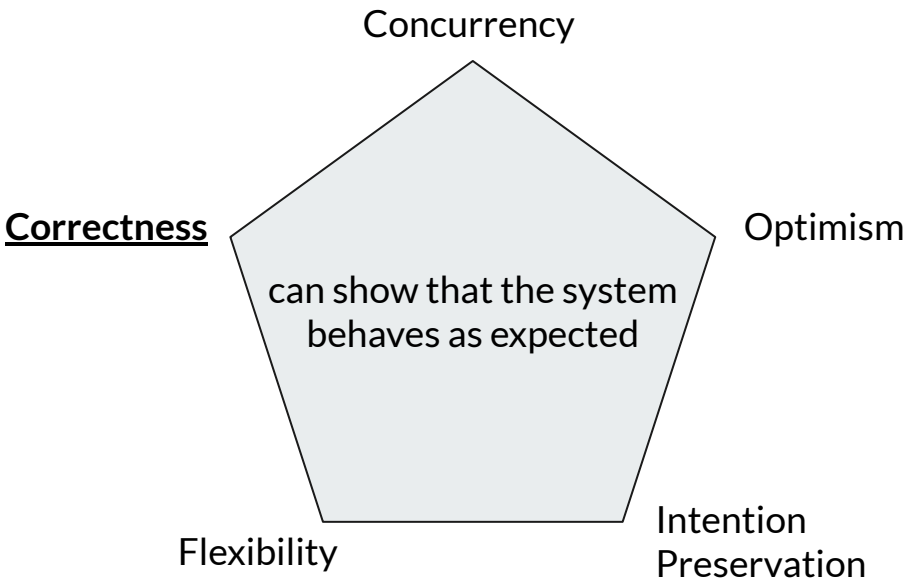




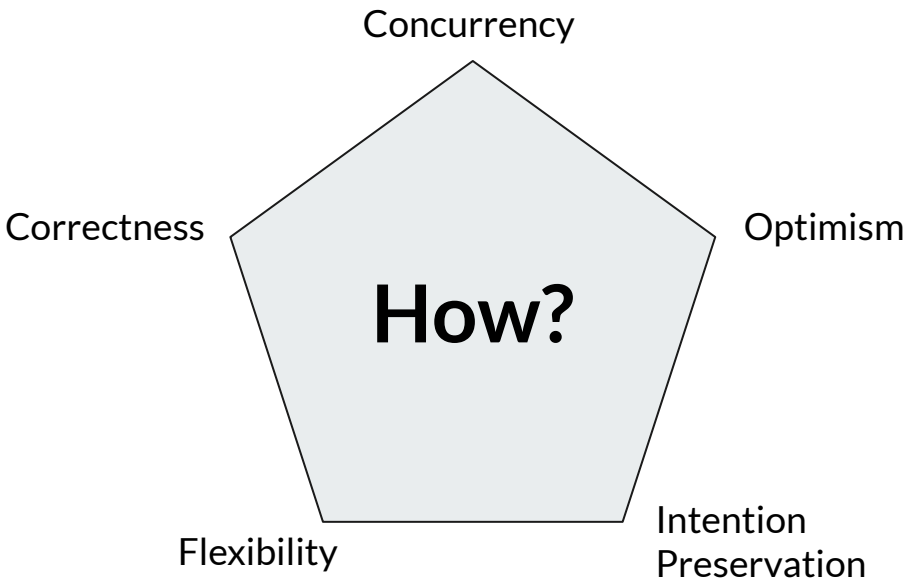
Requirements for a Collaborative FM Editor



Requirements for a Collaborative FM Editor



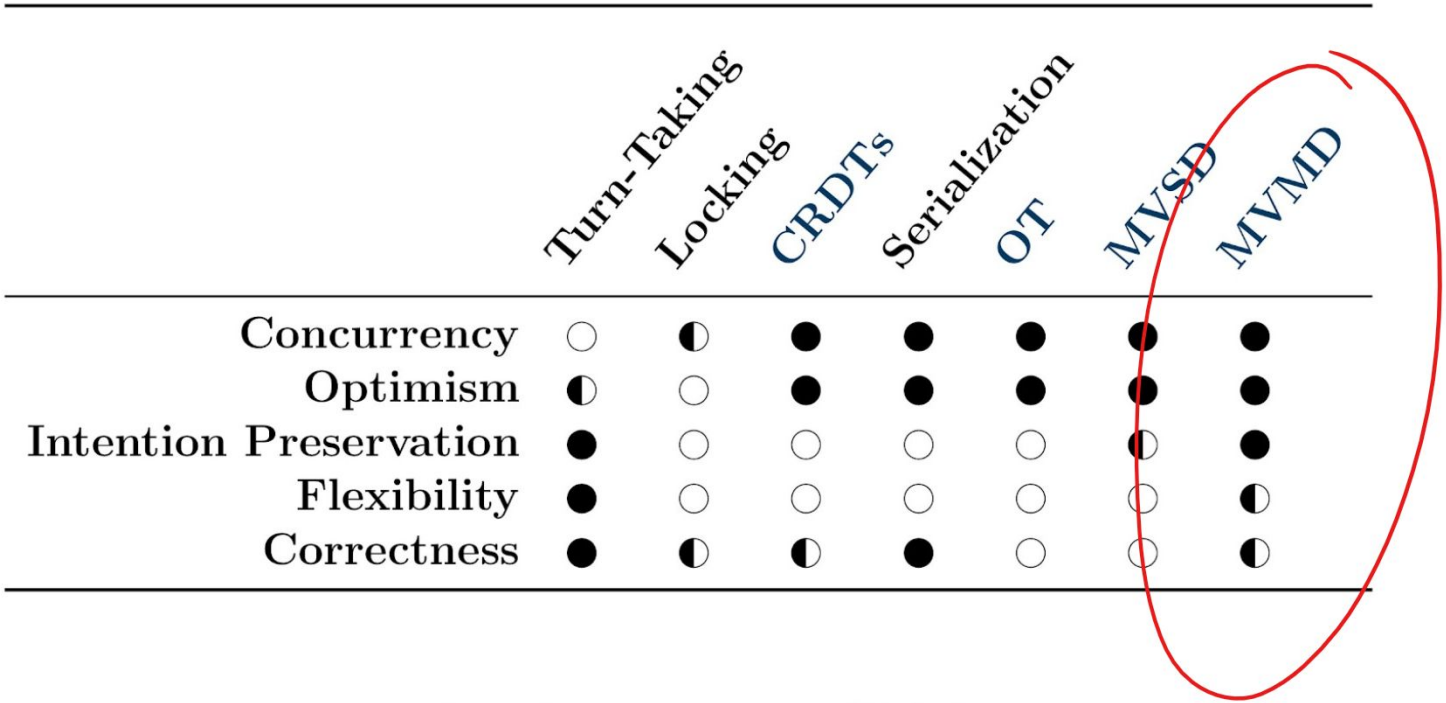
Requirements for a Collaborative FM Editor



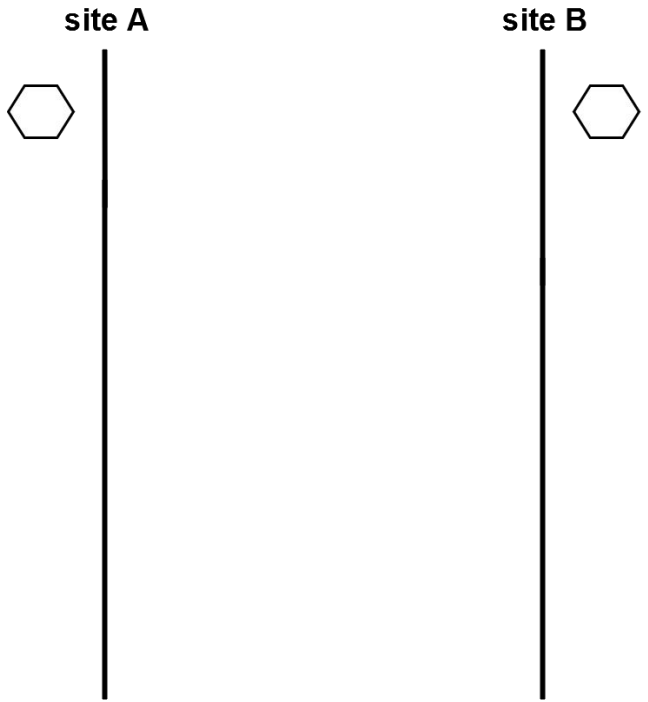
Requirements for a Collaborative FM Editor

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

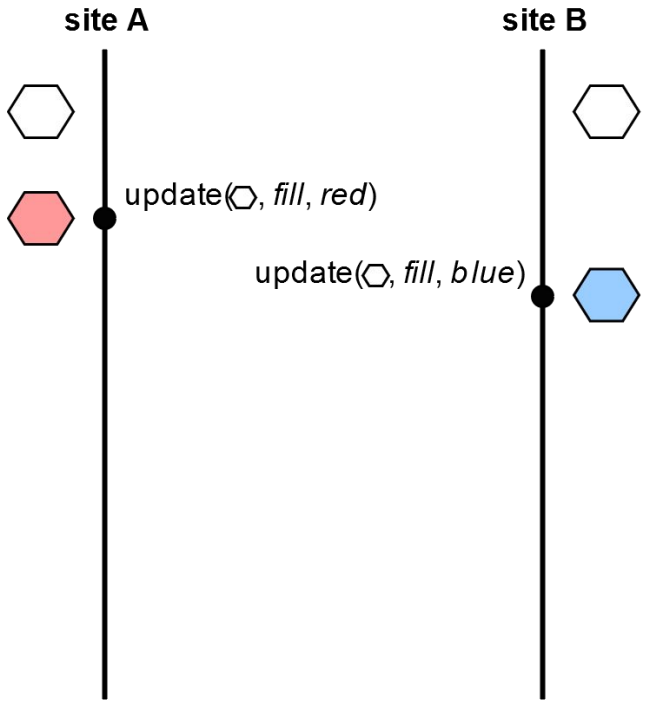
Comparison of Concurrency Control Techniques



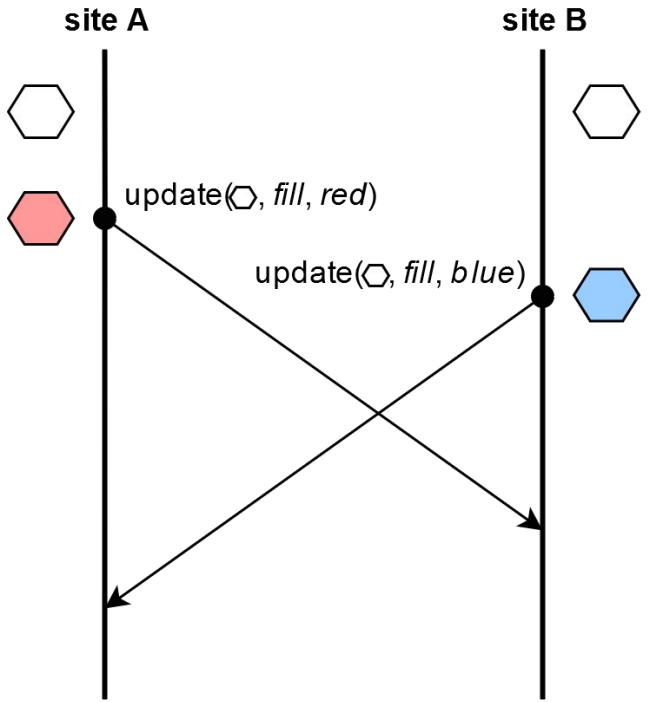
Comparison of Concurrency Control Techniques



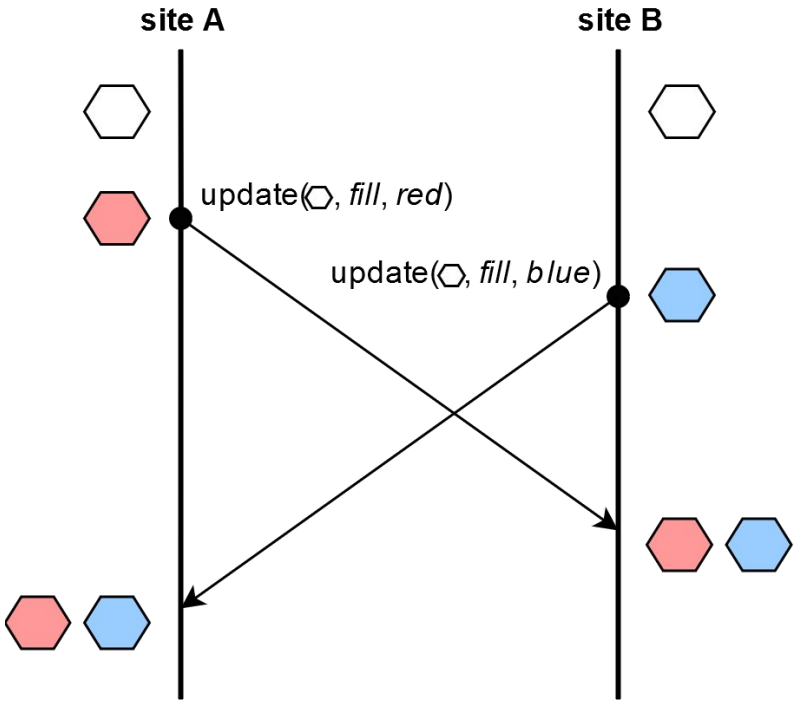
Multi-Version Multi-Display (MVMD)



Multi-Version Multi-Display (MVMD)



Multi-Version Multi-Display (MVMD)



Multi-Version Multi-Display (MVMD)

Applying MVMD to Feature Modeling

Major Tasks

Feature Model Representation

How to represent feature models?

Operation Model

What modeling operations are supported?

Conflict Detection

How to determine whether operations are in conflict?

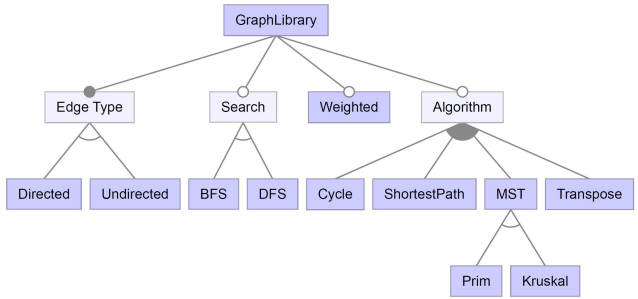
Conflict Resolution

How to proceed when a conflict has been detected?

Major Tasks

Feature Model Representation

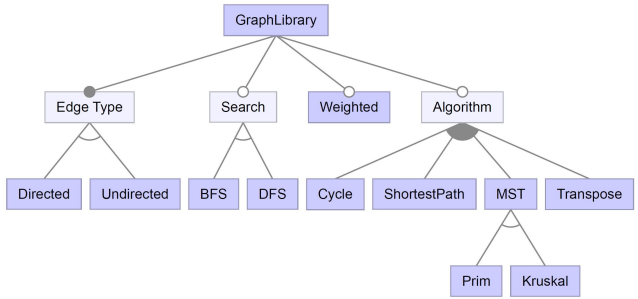
How to represent feature models?



Major Tasks

Feature Model Representation

How to represent feature models?



Operation Model

What modeling operations are supported?

We initially support these operations:

- **create and remove** feature (subtrees) and constraints
- **set** feature and constraint **attributes** (mandatory, ...)
- batch operations on multiple targets

Major Tasks

Feature Model Representation

How to represent feature models?

Operation Model

What modeling operations are supported?

Conflict Detection

How to determine whether operations are in conflict?

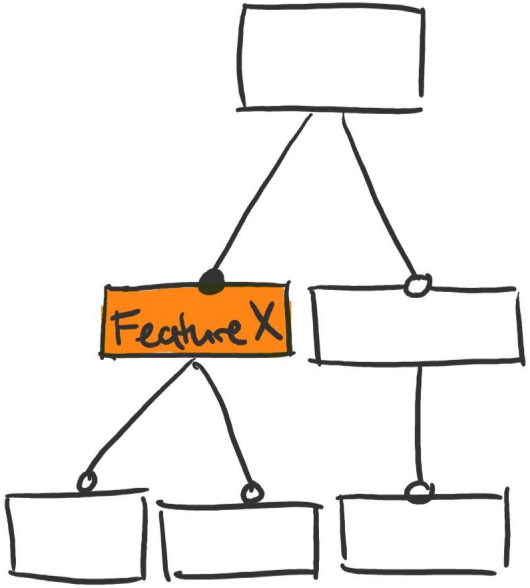
Conflict Resolution

How to proceed when a conflict has been detected?

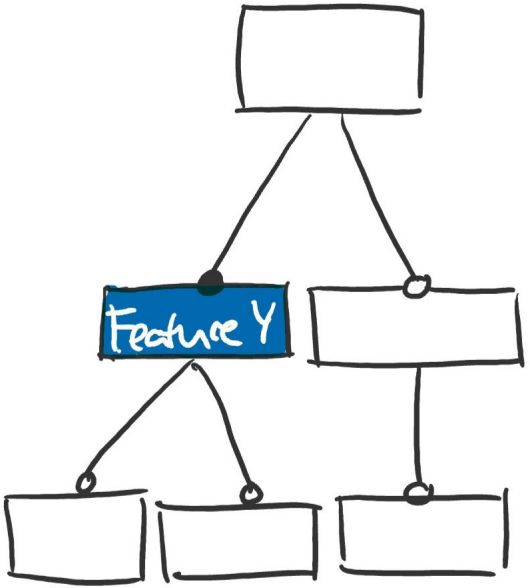
Conflict Detection

We extend the MVMD approach with a set of *conflict detection rules* specific to feature modeling.

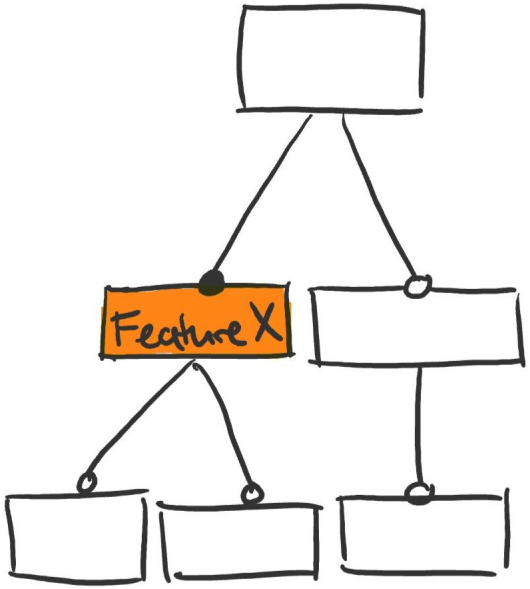
If any rule applies, multiple versions are created.



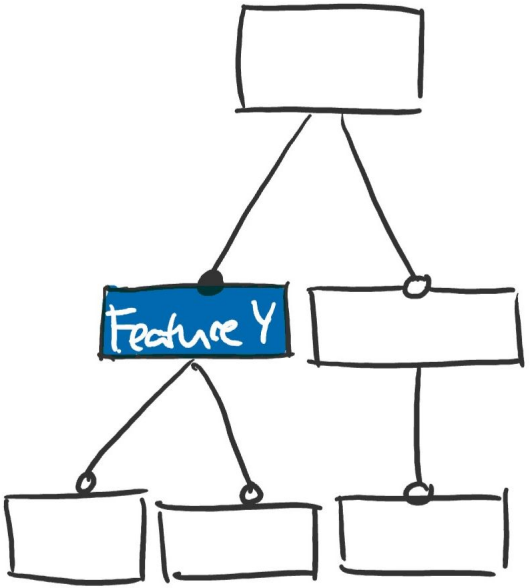
set feature name to *FeatureX*



set feature name to *FeatureY*

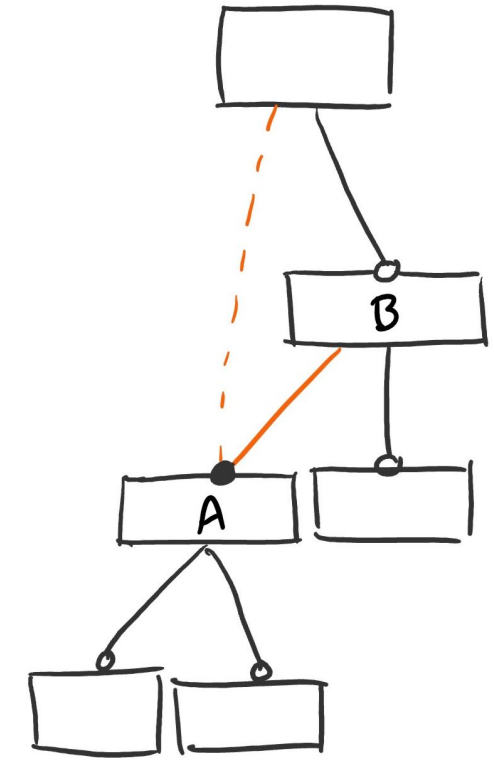


set feature name to *FeatureX*

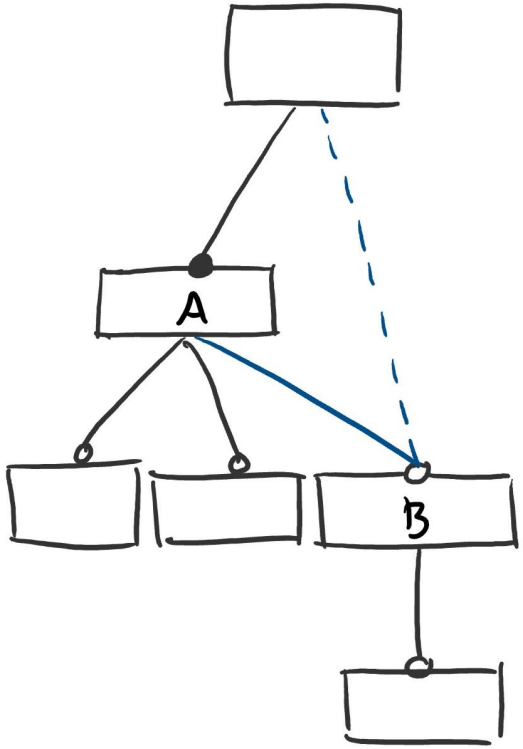


set feature name to *FeatureY*

Rule #1: No writes to the same feature attribute.

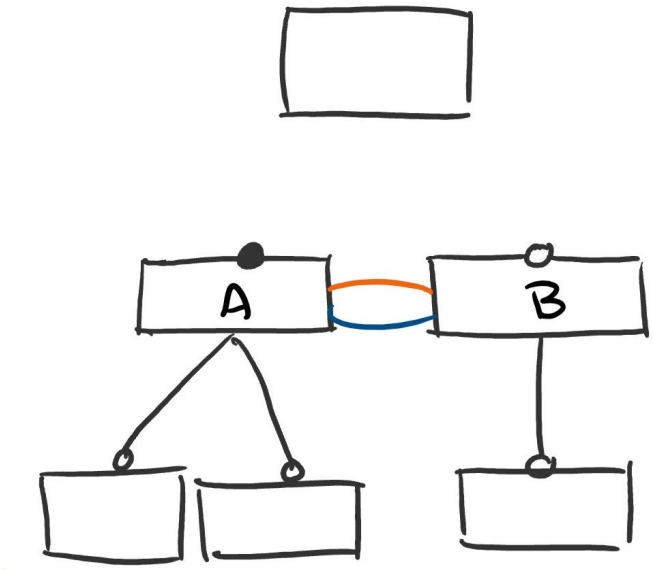


move feature subtree *A* below *B*

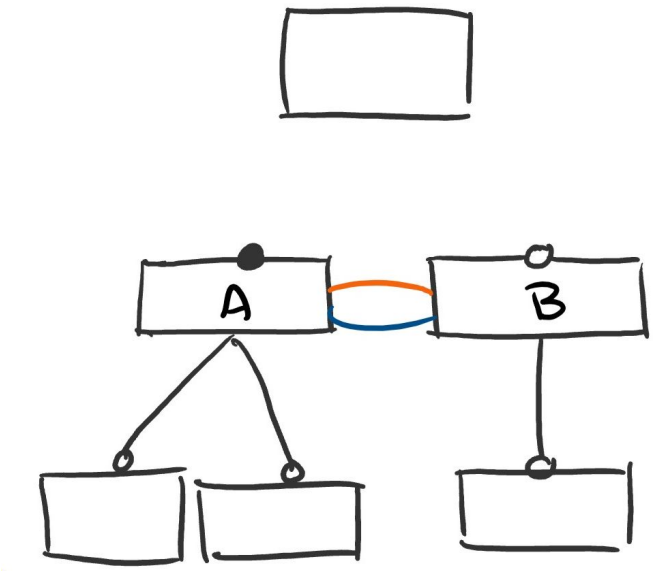


move feature subtree *B* below *A*

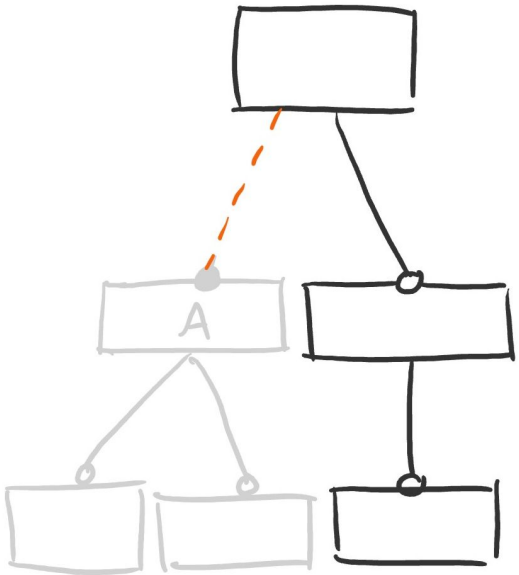
Result:



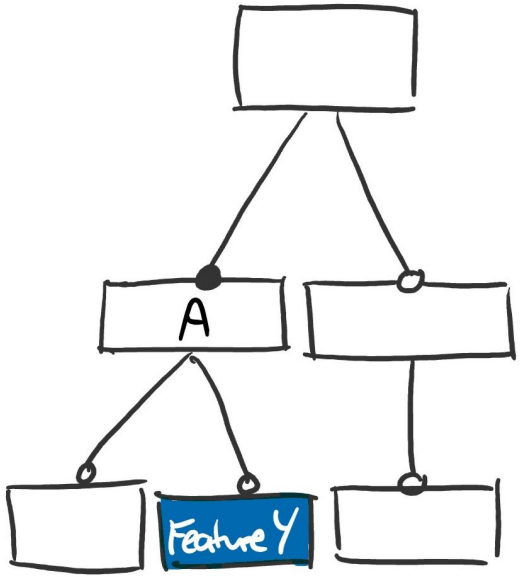
Result:



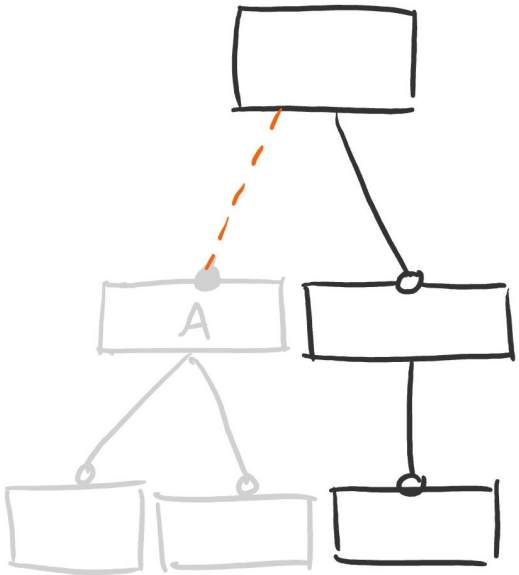
Rule #2: May not introduce cycles.



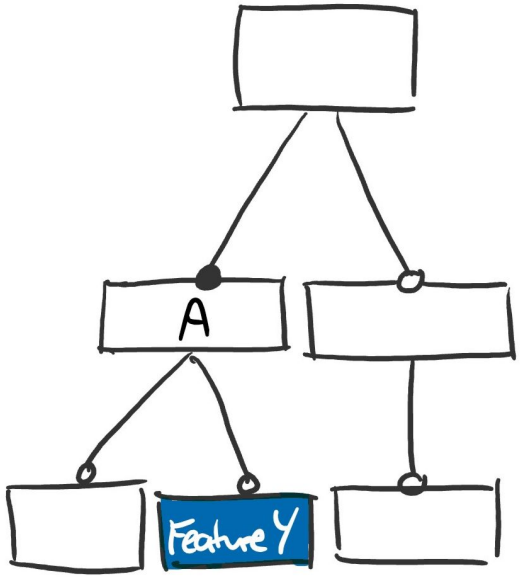
remove feature subtree *A*



set feature name to *FeatureY*

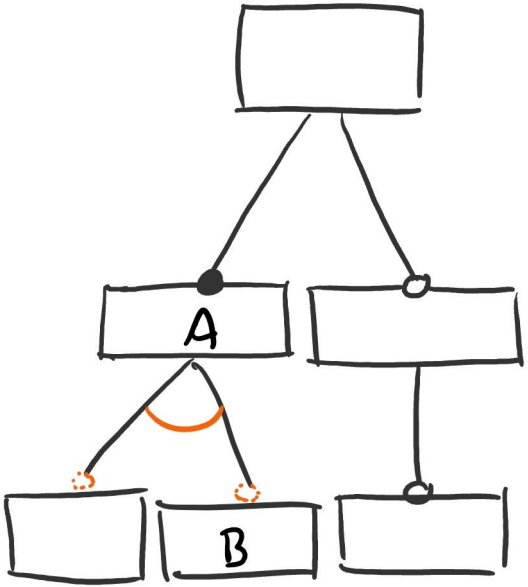


remove feature subtree *A*

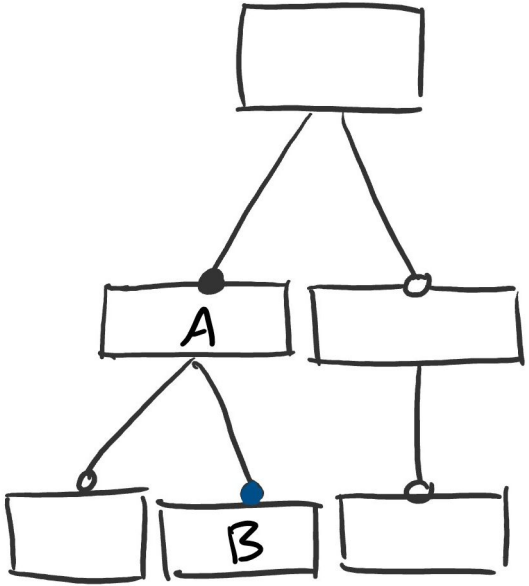


set feature name to *FeatureY*

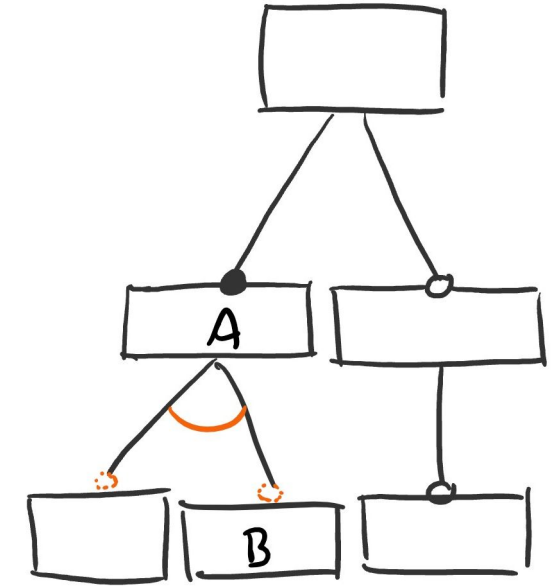
Rule #3: No writes to removed features.



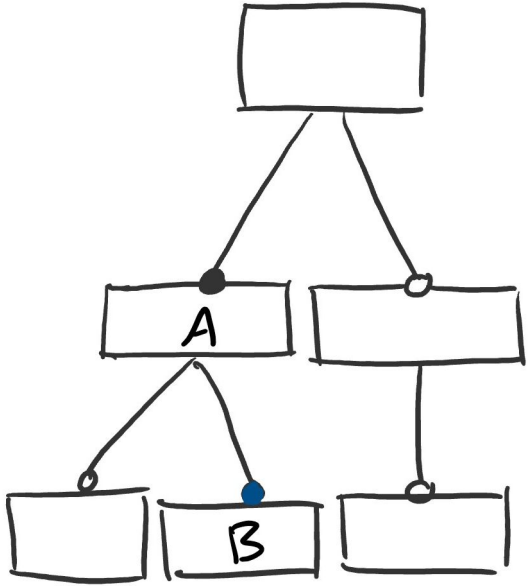
set *A* to alternative



set *B* to mandatory



set *A* to alternative



set *B* to mandatory

Rule #4: No mandatory/optional writes to group children.

Conflict Detection Rules

- **Rule #1:** No writes to the same feature attribute.
- **Rule #2:** May not introduce cycles.
- **Rule #3:** No writes to removed features.
- **Rule #4:** No mandatory/optional writes to group children.
- ...
- extensible with semantic properties such as *no dead features*, *no redundant constraints* etc.

Major Tasks

Feature Model Representation

How to represent feature models?

Operation Model

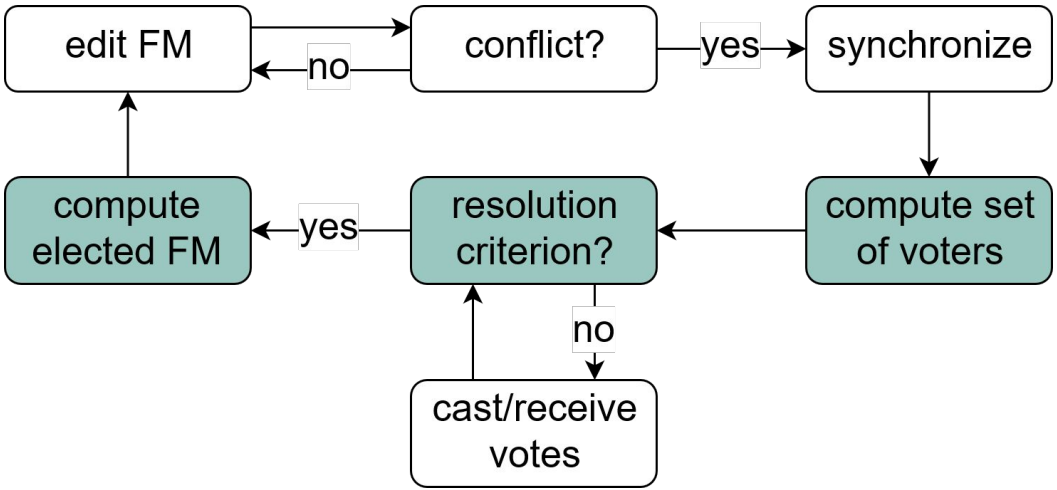
What modeling operations are supported?

Conflict Detection

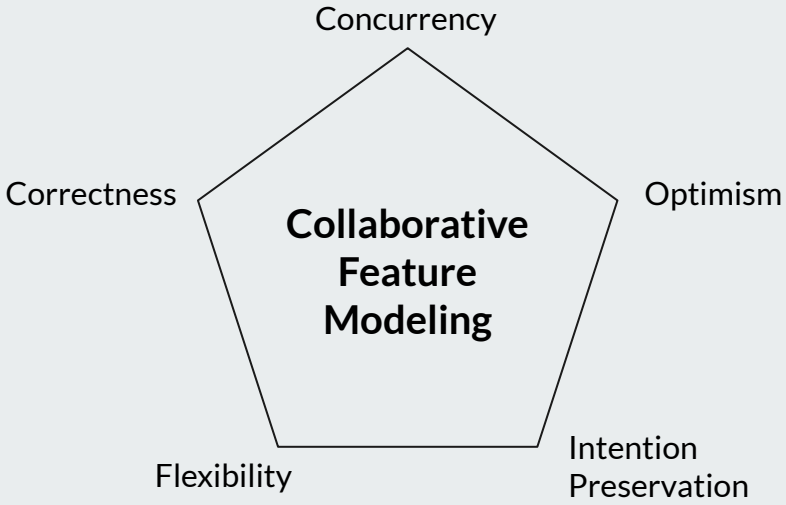
How to determine whether operations are in conflict?

Conflict Resolution

How to proceed when a conflict has been detected?



Conflict Resolution Process



Discussion

- Would you use it?
  - If yes, what for?
  - If not, why?
- What do you value most in editing software?
- Is there any ~~feature~~ functionality you would like to see in a collaborative FM editor?