VICCI

Visual and Interactive Cyber-Physical Systems Control and Integration

Deriving Variants with Variability in Space and Time

Using Hyper Feature Models and Delta Modeling

Christoph Seidl

Ina Schaefer

Uwe Aßmann

Europa fördert Sachsen. ESF Europäischer Sozialfonds SACHSEN

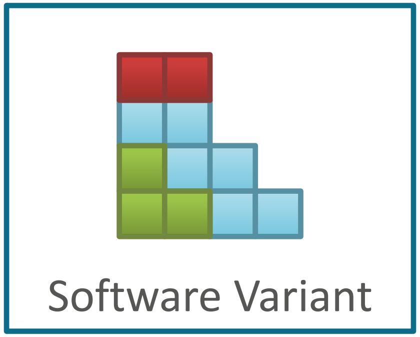
TECHNISCHE UNIVERSITÄT DRESDEN

> Background

- Variability in Space
  - Configuration
  - ➢ Variant of system
- Variability in Time
  - Evolution
  - ➢ Version of system

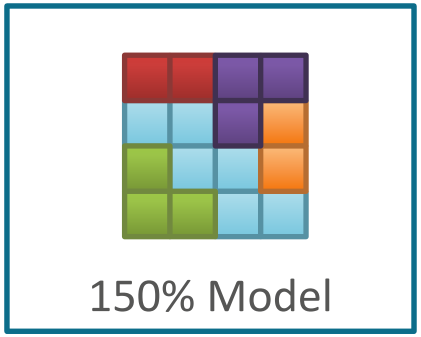
- Variability Realization Mechanisms (Excerpt)

Desired Variant



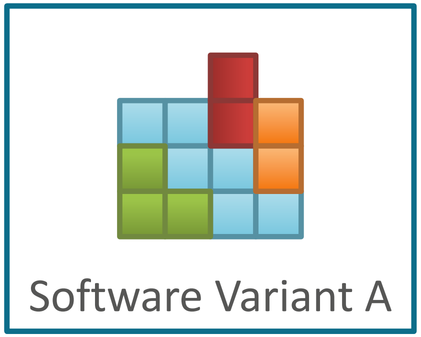
Software Variant

Subtractive Variability



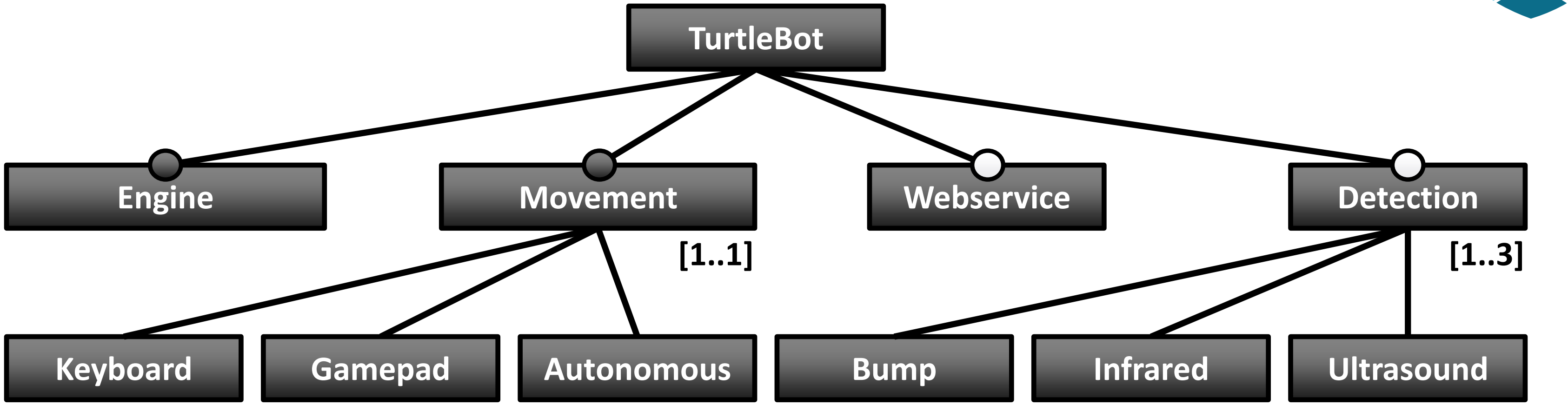
150% Model

Delta Modeling



Software Variant A

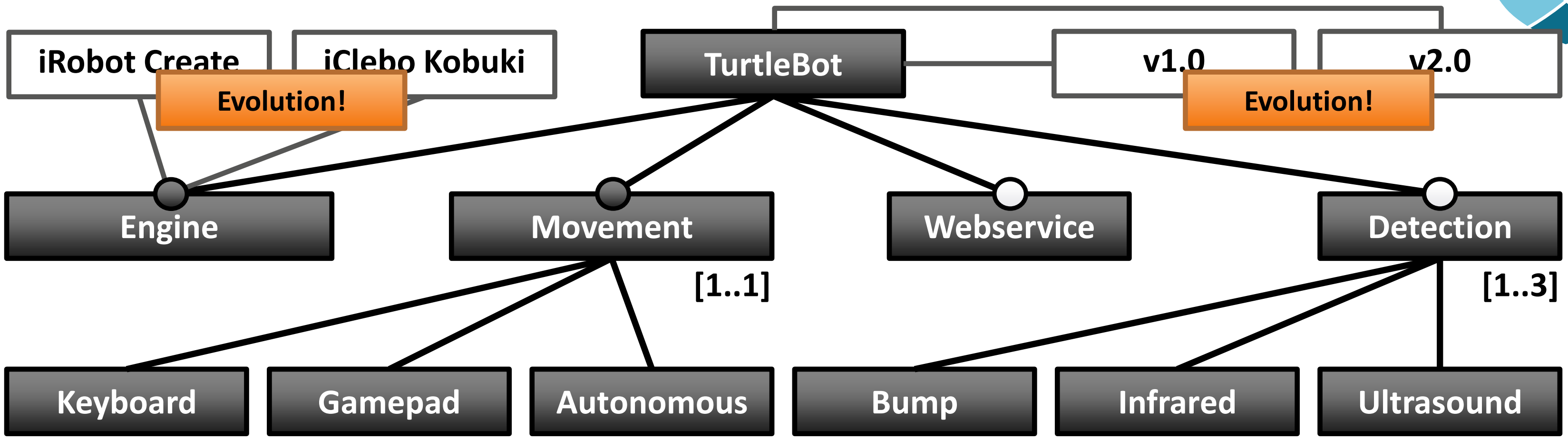
> Running Example: TurtleBot Driver



Cross-Tree Constraints

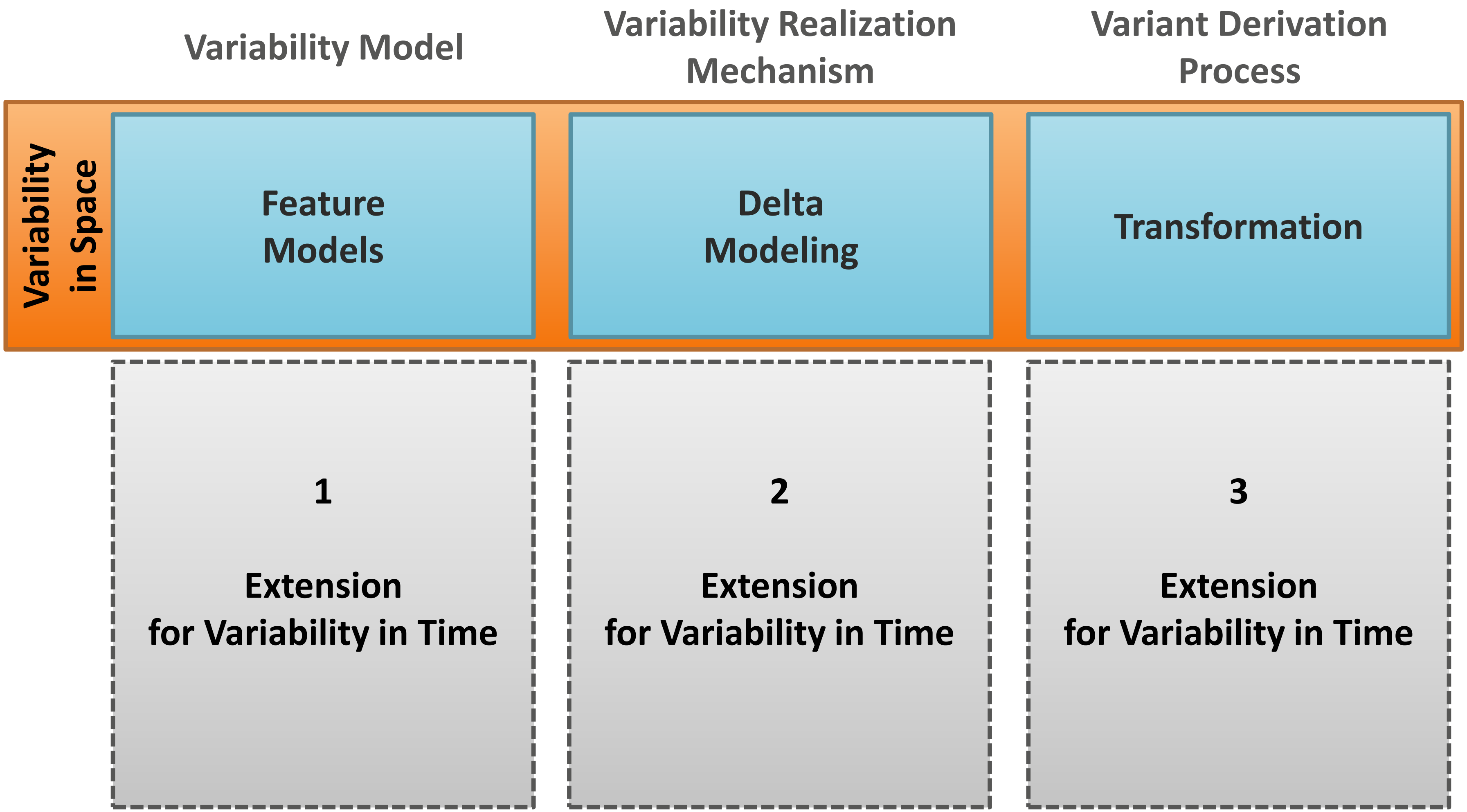
1. Autonomous → Detection
2. Keyboard ∨ Gamepad → Webservice

> Feature Models: No Variability in Time



- Constraints caused by evolution (excerpt)
  - iClebo Kobuki is incompatible with TurtleBot v1.0
  - iClebo Kobuki requires at least TurtleBot v2.0
- Problem
  - Cannot model the evolution on conceptual level
  - **Cannot derive "older" variants!**

> Solution Overview



Variability Model

Variability Realization Mechanism

Variant Derivation Process

Variability in Space

Feature Models

Delta Modeling

Transformation

1

Extension for Variability in Time

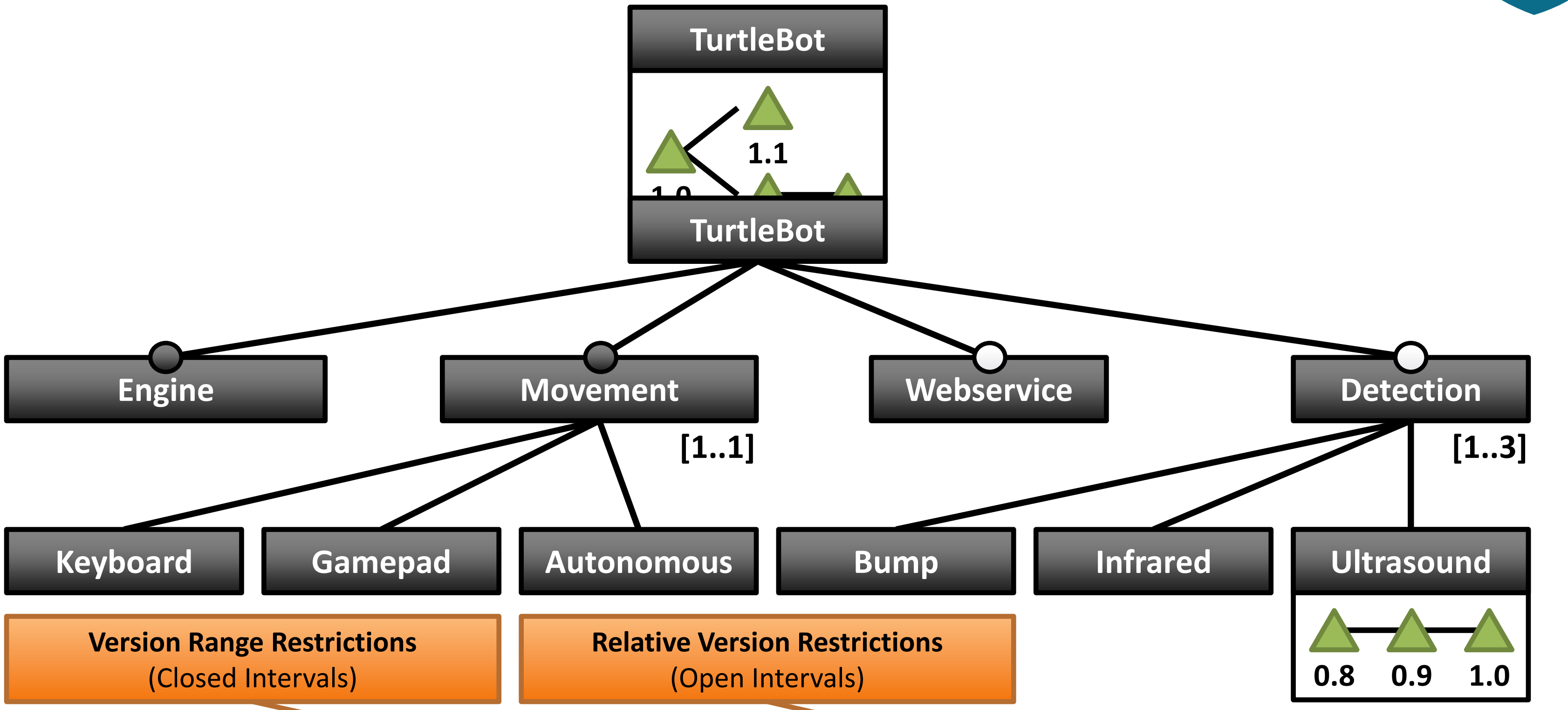
2

Extension for Variability in Time

3

Extension for Variability in Time

> Hyper Feature Models (HFMs) for Versions



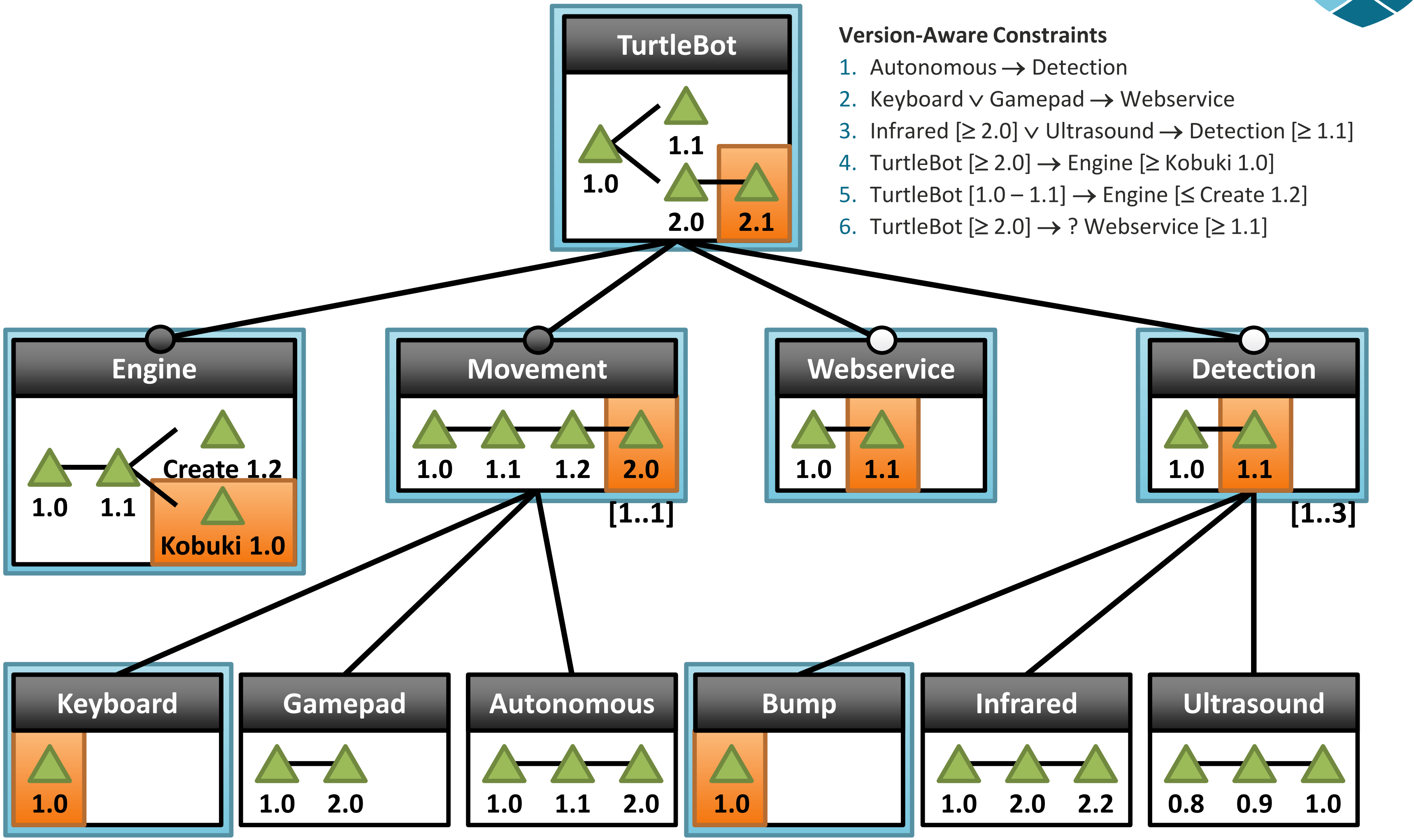
Version Range Restrictions
(Closed Intervals)

Relative Version Restrictions
(Open Intervals)

Version-Aware Constraints

1. Autonomous → Detection
2. Keyboard ∨ Gamepad → Webservice
3. Infrared [≥ 2.0] ∨ Ultrasound → Detection [≥ 1.1]
4. TurtleBot [≥ 2.0] → Engine [≥ Kobuki 1.0]
5. TurtleBot [1.0 – 1.1] → Engine [≤ Create 1.2]
6. TurtleBot [≥ 2.0] → ? Webservice [≥ 1.1]

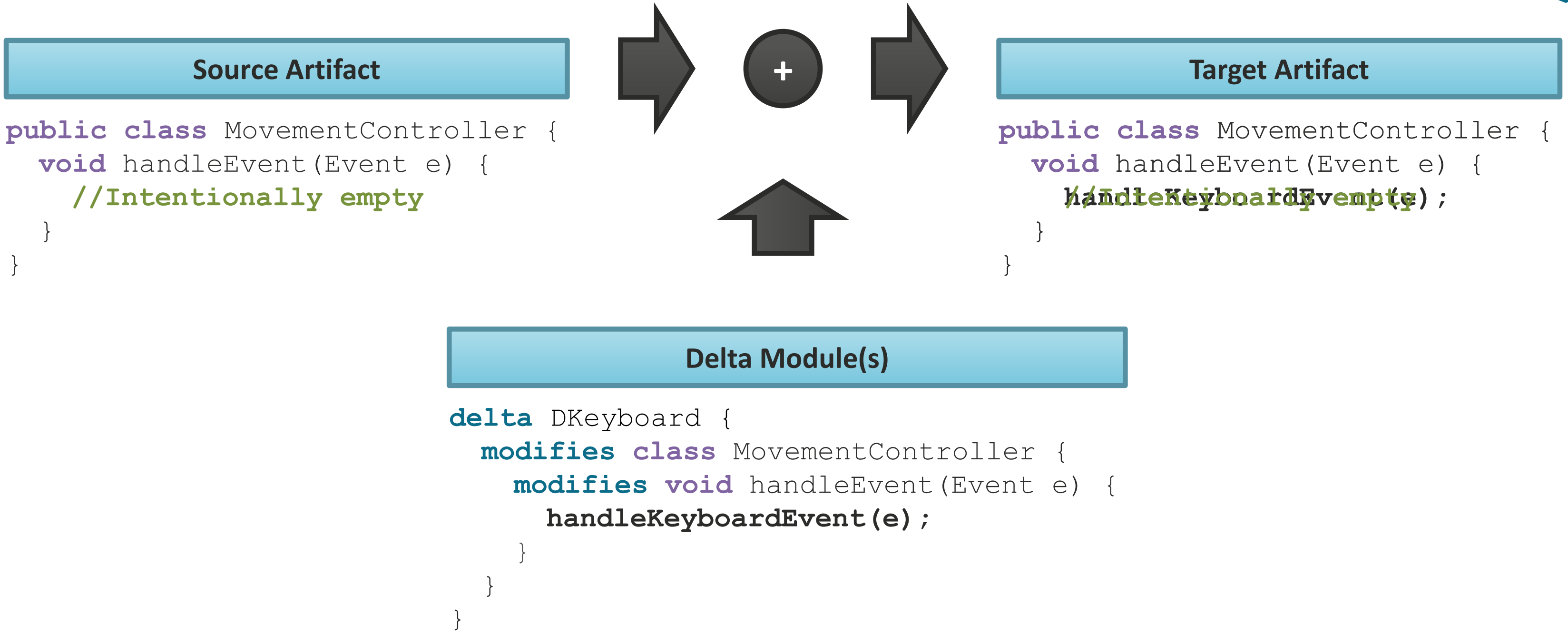
> Configurations of Hyper Feature Models



Version-Aware Constraints

1. Autonomous → Detection
2. Keyboard ∨ Gamepad → Webservice
3. Infrared [≥ 2.0] ∨ Ultrasound → Detection [≥ 1.1]
4. TurtleBot [≥ 2.0] → Engine [≥ Kobuki 1.0]
5. TurtleBot [1.0 – 1.1] → Engine [≤ Create 1.2]
6. TurtleBot [≥ 2.0] → ? Webservice [≥ 1.1]

Delta Modeling: No Variability in Time (Yet)



Source Artifact

```
public class MovementController {
  void handleEvent(Event e) {
    //Intentionally empty
  }
}
```

Target Artifact

```
public class MovementController {
  void handleEvent(Event e) {
    handleKeyboardEvent(e);
    //Intentionally empty
  }
}
```

Delta Module(s)

```
delta DKeyboard {
  modifies class MovementController {
    modifies void handleEvent(Event e) {
      handleKeyboardEvent(e);
    }
  }
}
```

- Delta Modeling (currently) handles only variability in space
- Delta Modeling is a form of transformation
  - May be extended for variability in time
  - ➢ Explicit notion of **evolution delta modules**

> Example: Using Evolution Delta Modules

```
public class Movement {
  void handleEvent(Event e) {
    //Lengthy code here...
  }
}
```

1. Rename class
2. Extract method
   a) Create new method
   b) Move code to new method
   c) Call new method

eδ

Evolution Delta Operations

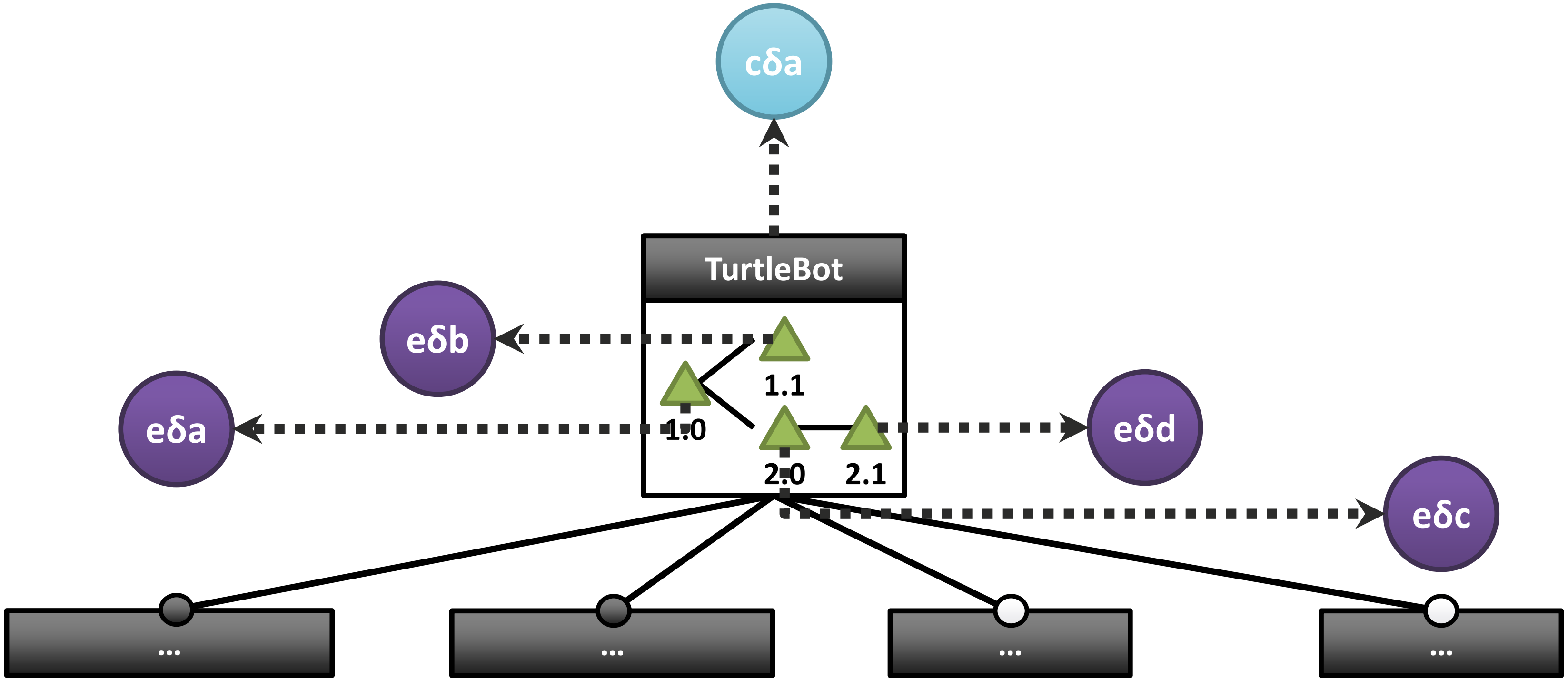
Configuration Delta Operations

Modifying Identifiers

Refactorings

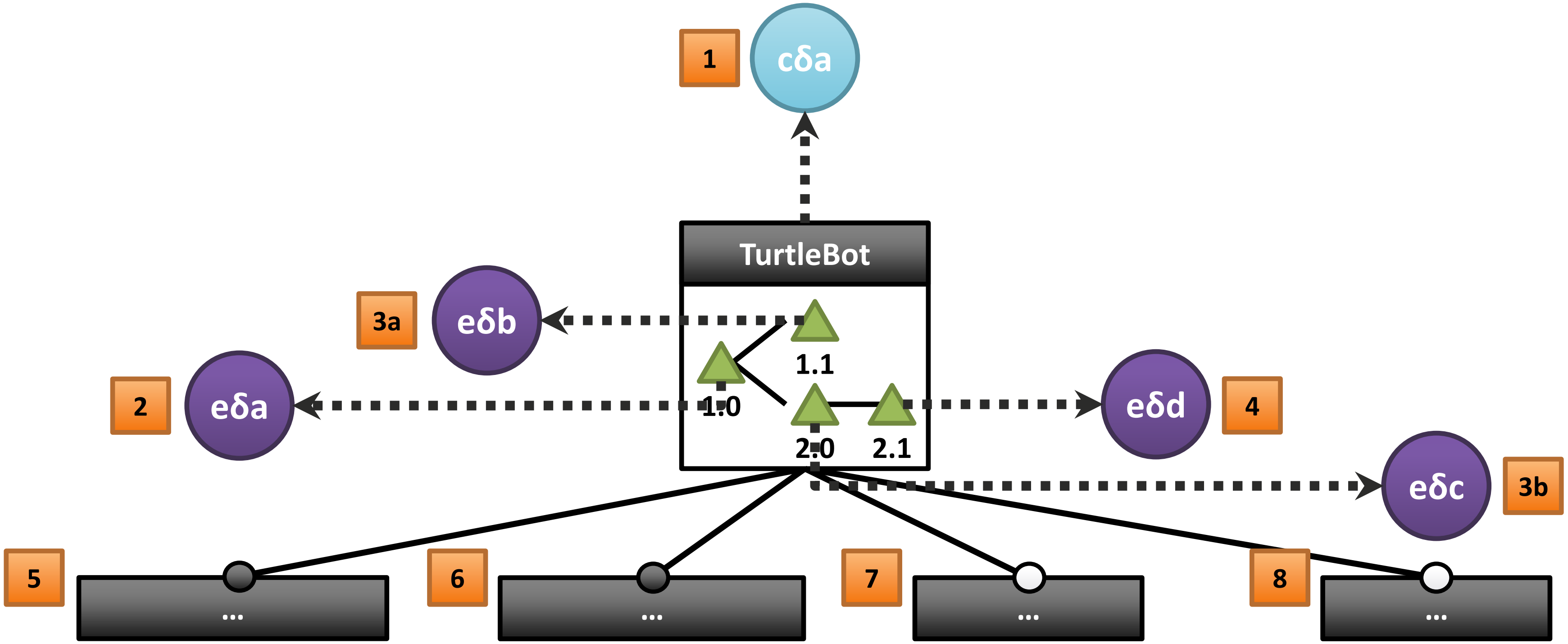
```
public class MovementController {
  void handleEvent(Event e) {
    handleKeyboardEvent(e);
  }

  private void handleKeyboardEvent(Event e) {
    //Lengthy code here...
  }
}
```

> Mapping HFMs to Delta Modules



- Features are mapped to configuration delta modules
- Versions are mapped to evolution delta modules

Implicit Order of Delta Modules

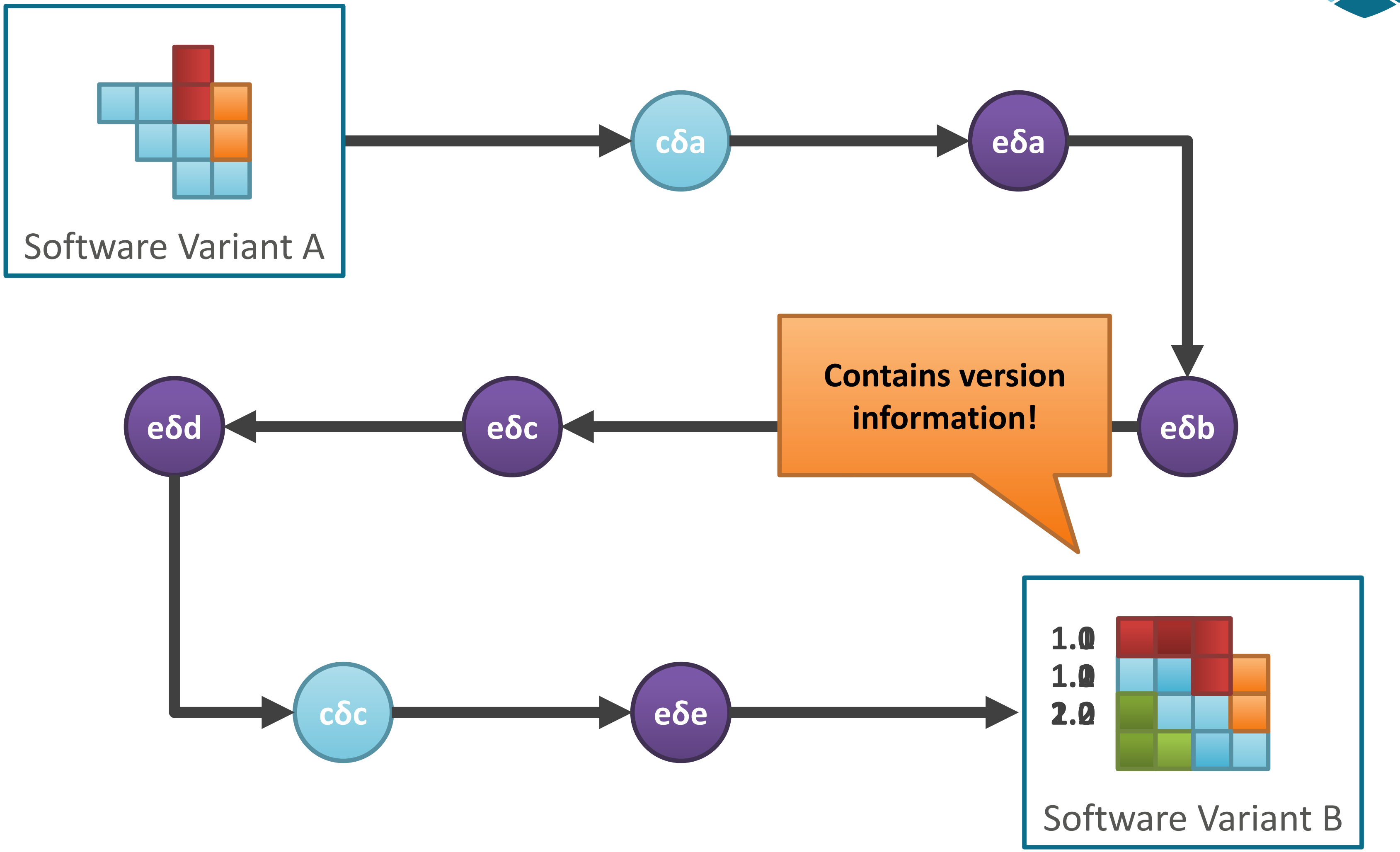


a) Configuration delta modules

b) Evolution delta modules in order of development lines

c) Continue with remaining features of HFM

> Applying Delta Modules

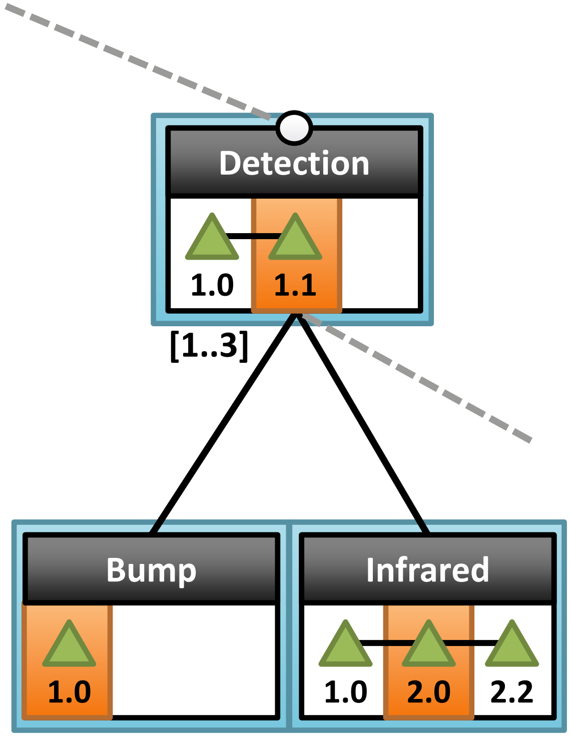


Summary Current Work

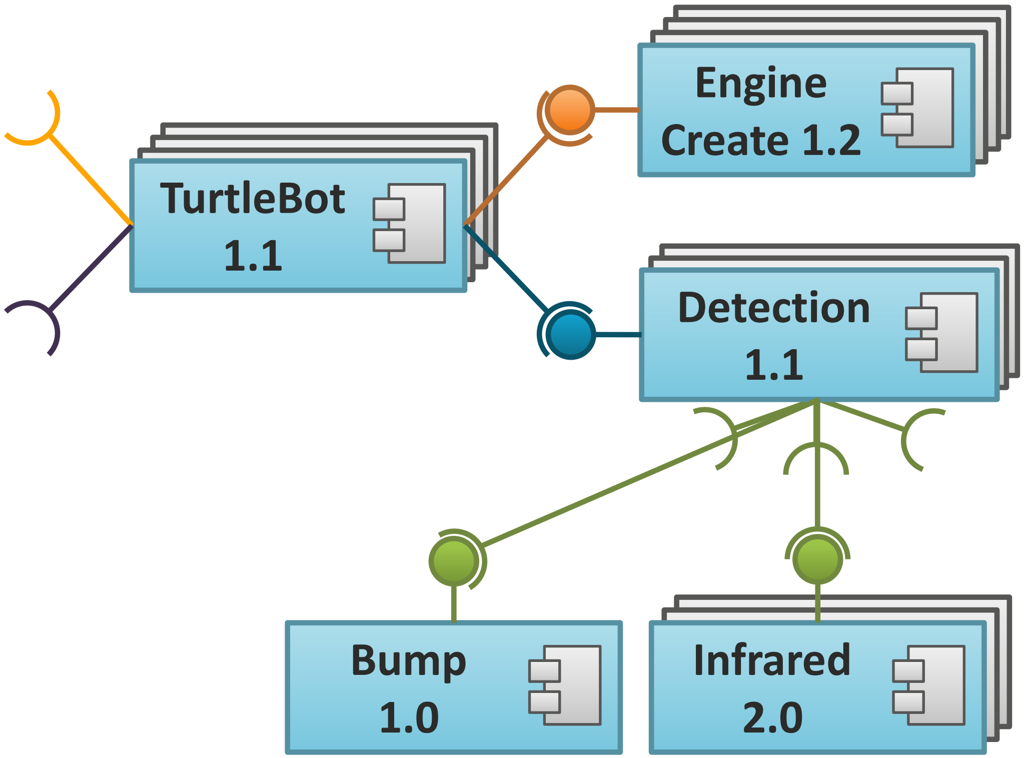
	Variability Model	Variability Realization Mechanism	Variant Derivation Process
Variability in Space	Feature Models	Delta Modeling	Transformation
Variability in Time	Hyper Feature Models	Evolution Delta Modules	Feature and Version to Delta Module Mapping
	Version-Aware Constraint Language		Implicit Delta Module Evaluation Order

> Future Work: HFMs with Component Systems

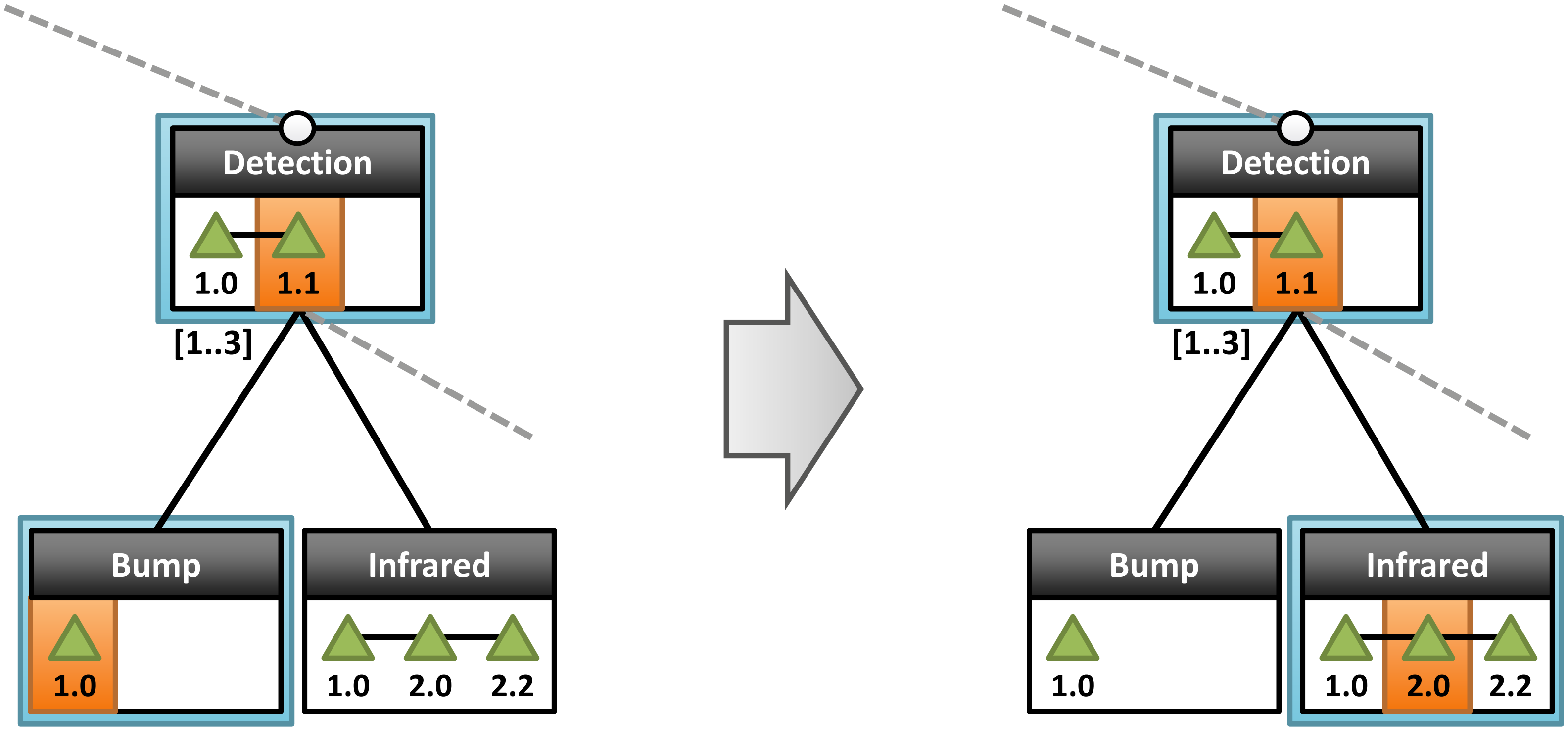
Problem Space



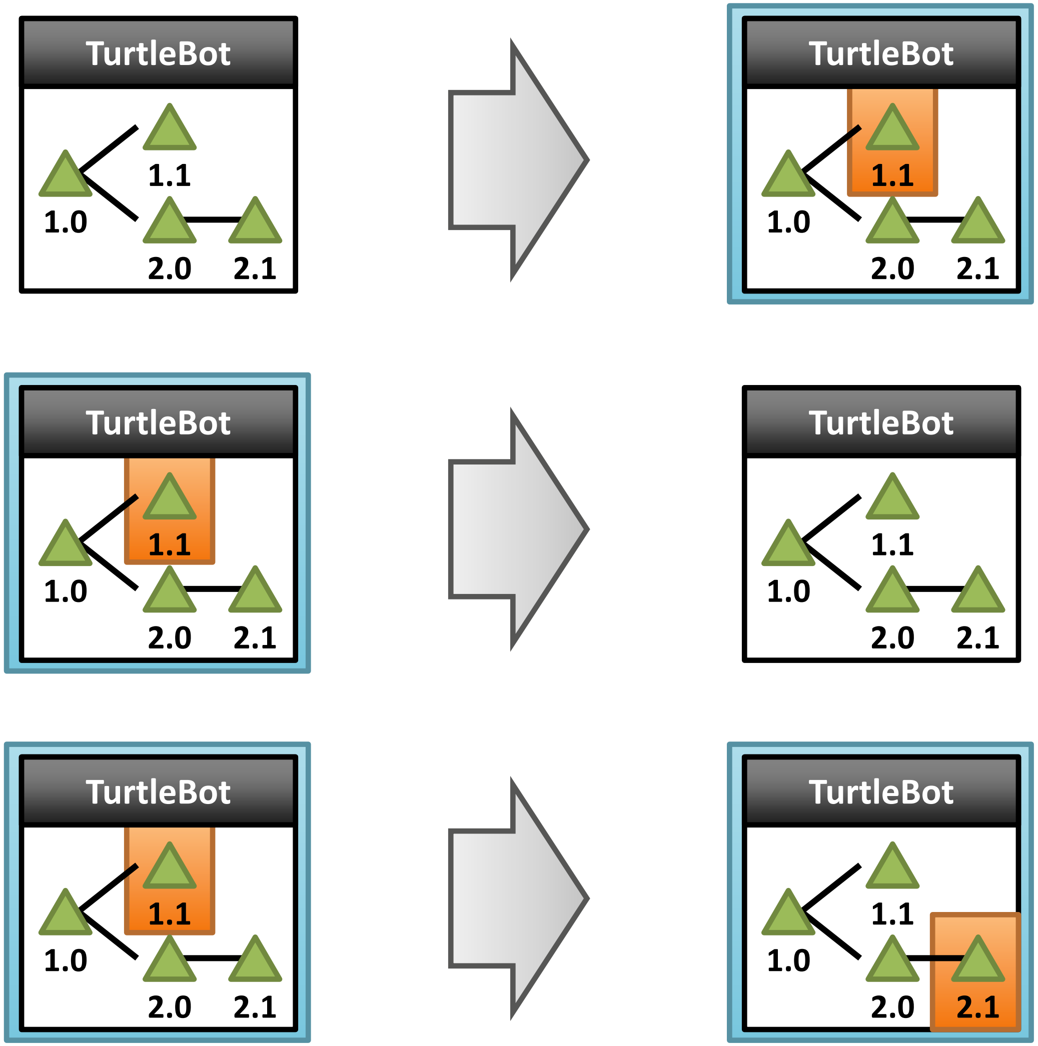
Solution Space



> Future Work: Reconfiguration = Migrating Configurations



> Future Work: Configuration Migration Operations

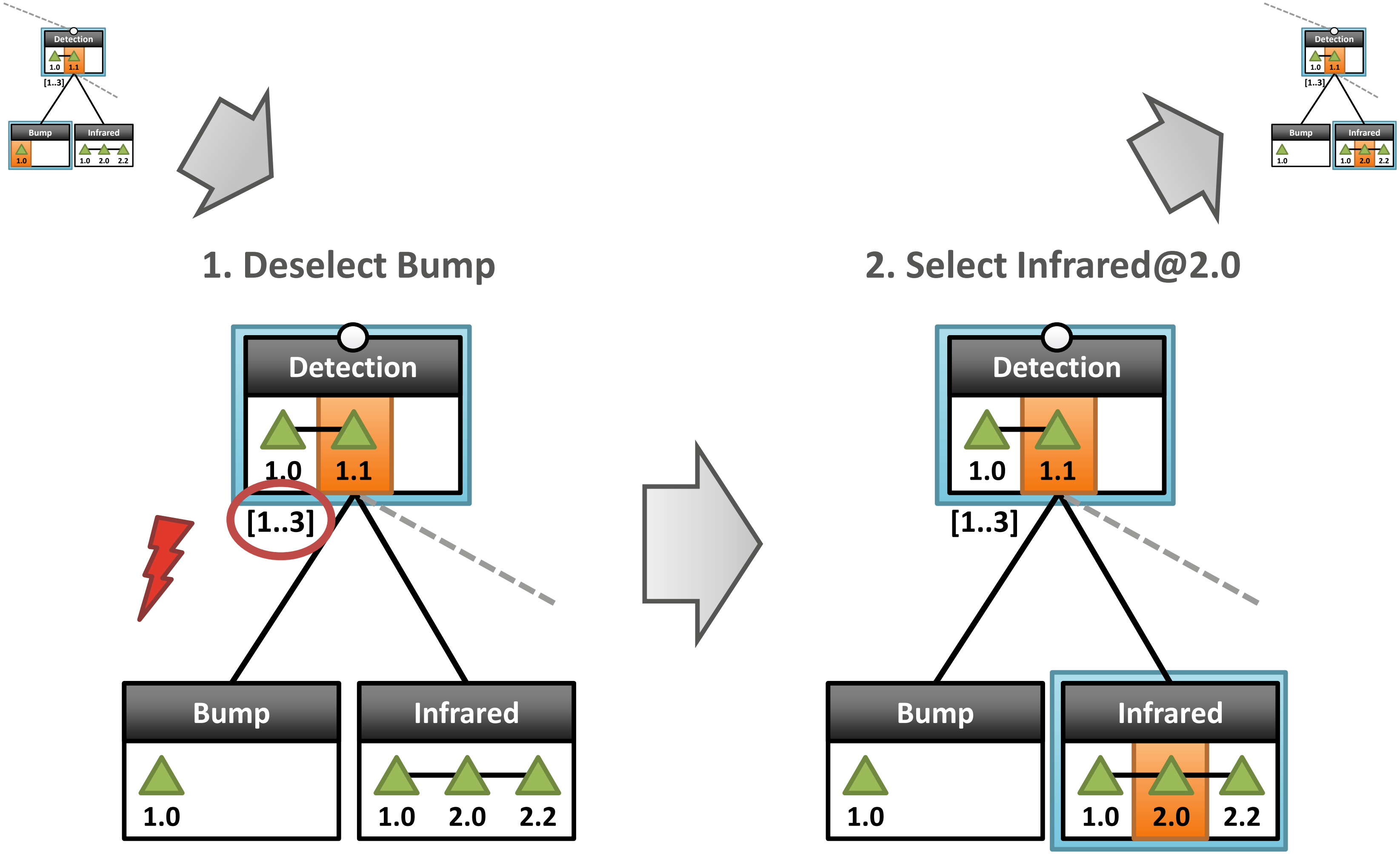


Select Feature (at Version)

Deselect Feature (and Version)

Change Version

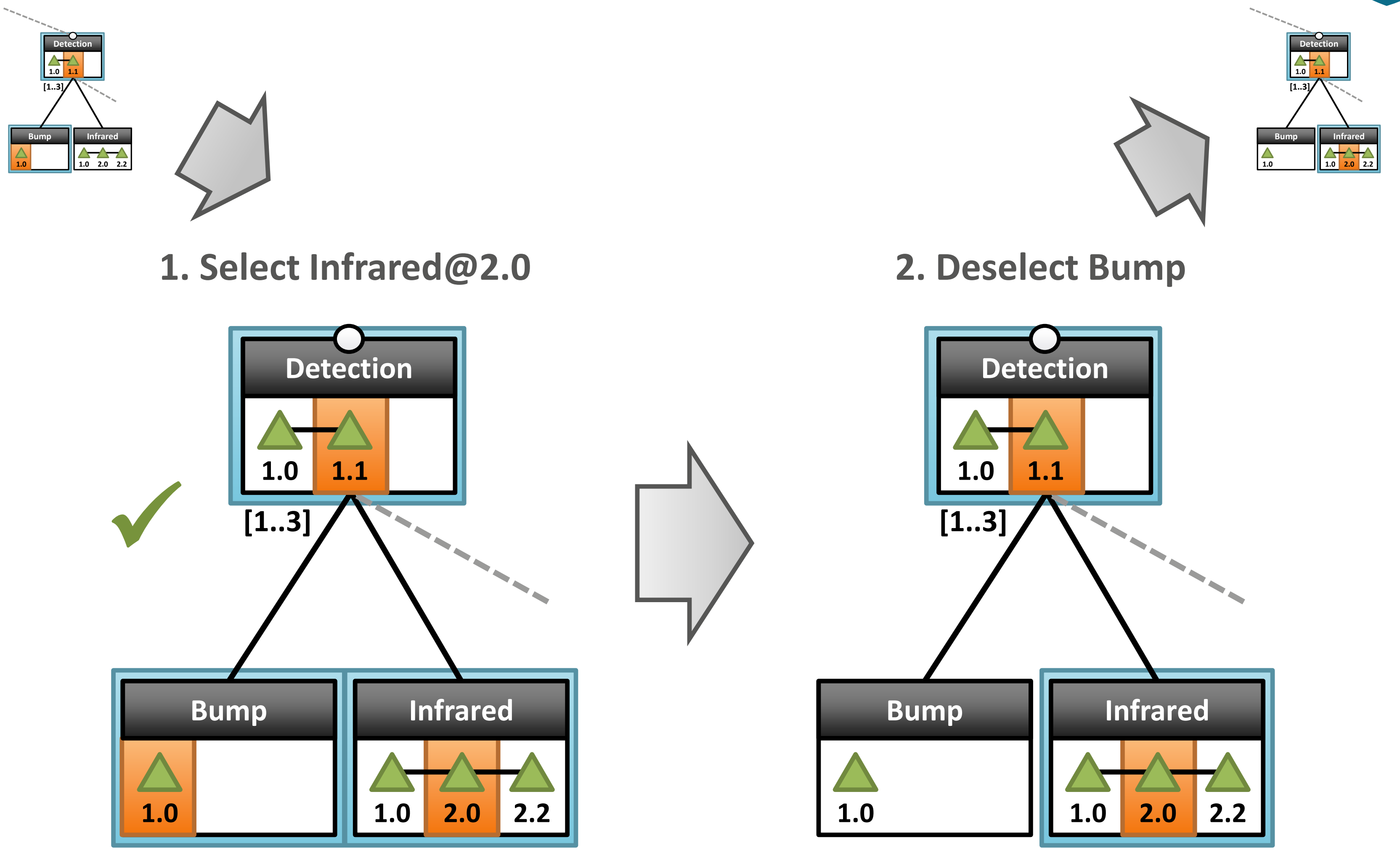
> Future Work: Ordering of Operations Matters (1)



1. Deselect Bump

2. Select Infrared@2.0

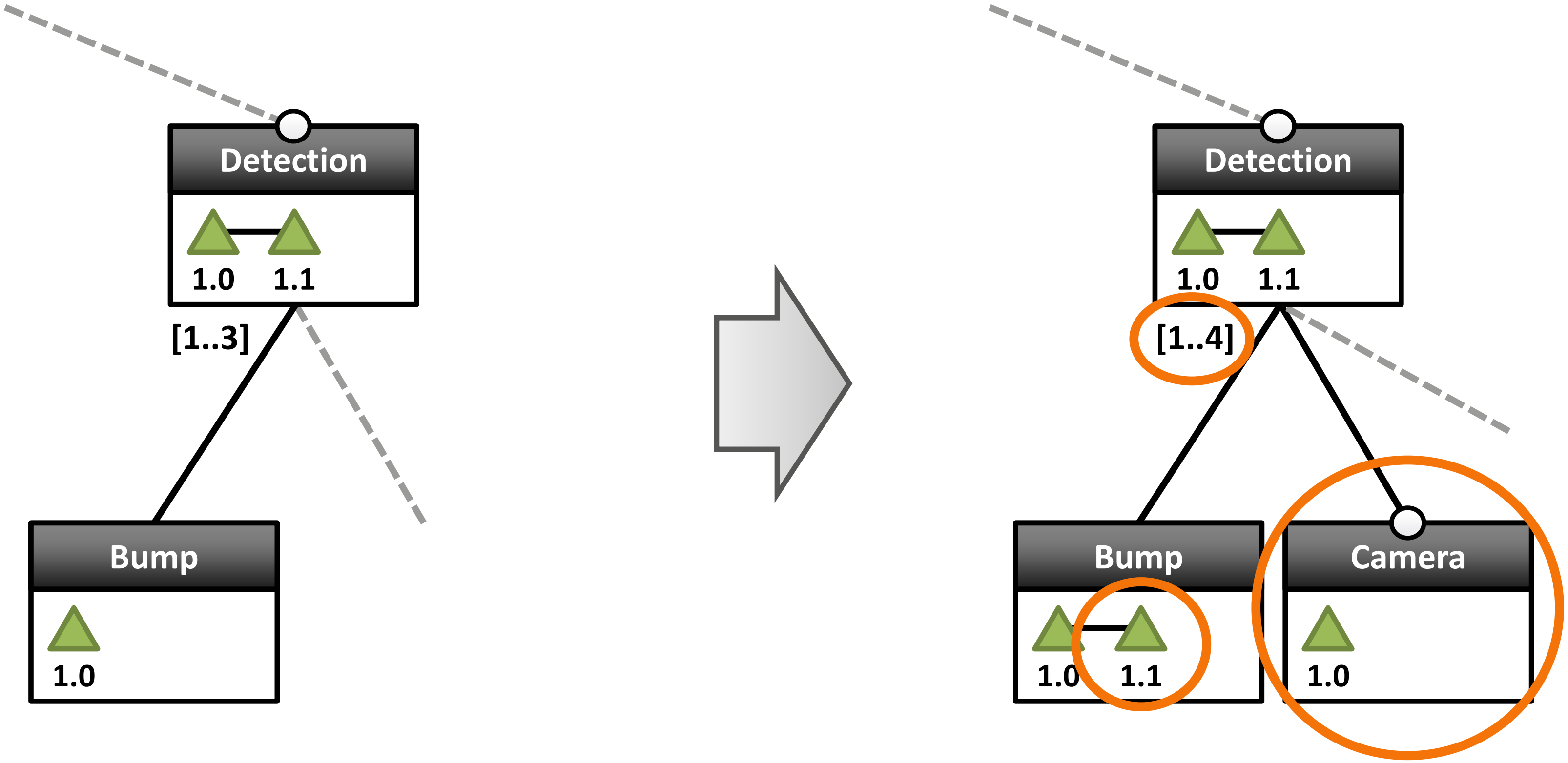
> Future Work: Ordering of Operations Matters (2)



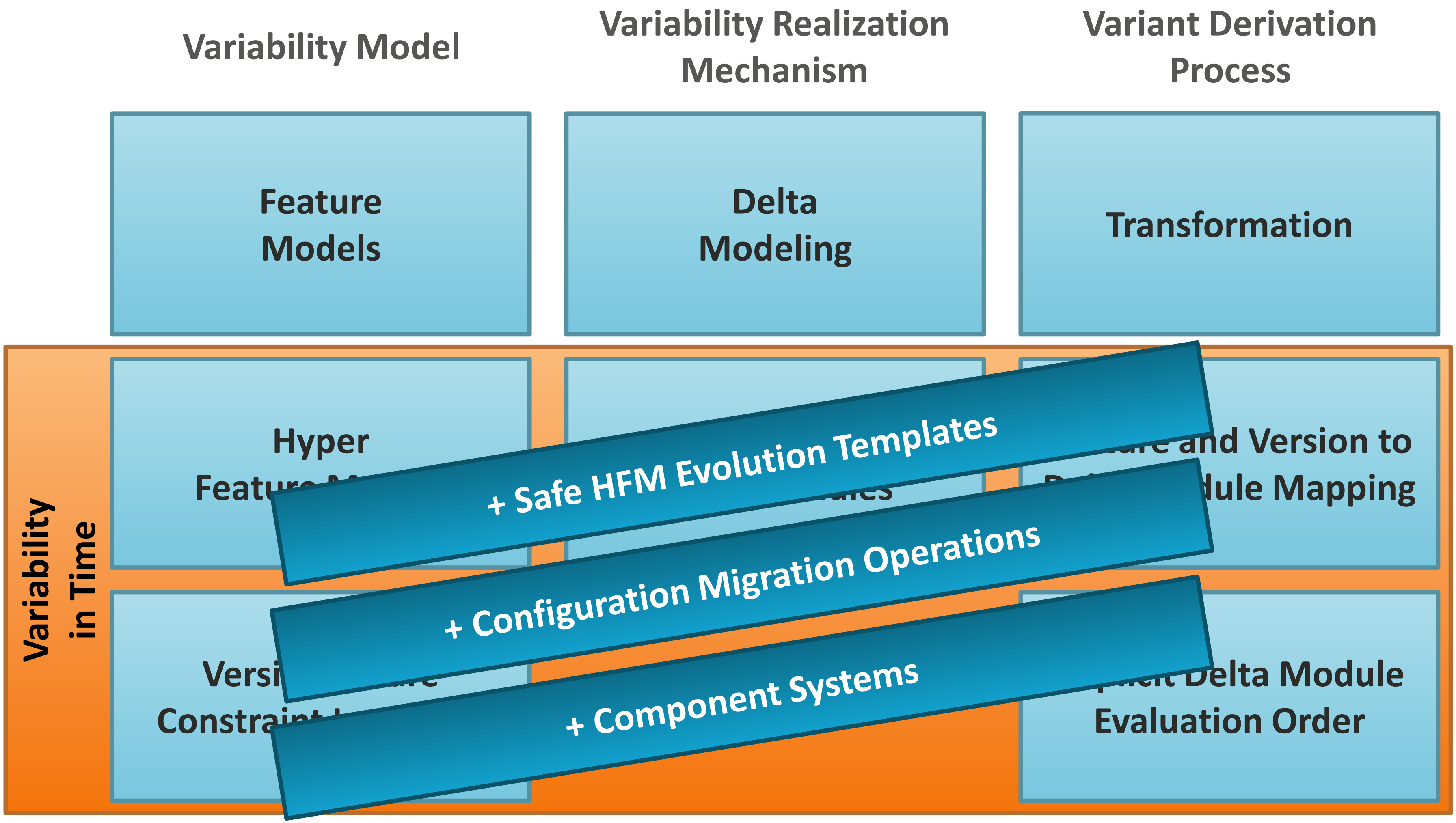
1. Select Infrared@2.0

2. Deselect Bump

> Future Work: Safe Evolution Templates for HFMs



> Future Work



Variability Model

Variability Realization Mechanism

Variant Derivation Process

Feature Models

Delta Modeling

Transformation

Variability in Time

Hyper Featur M

ules

ure and Version to D dule Mapping

Versi ure Constraint L

licit Delta Module Evaluation Order

+ Safe HFM Evolution Templates

+ Configuration Migration Operations

+ Component Systems

Interesting Contacts for Me

- Dynamic Software Product Lines (DSPLs)
- Formal Methods for (Re-)Configuration
- Safe Evolution Templates (for Versions?)
- Evolution in Software Product Lines in General

Thank you for your attention!

Questions, Comments, Feedback?