VICCI

Visual and Interactive Cyber-Physical Systems Control and Integration

Capturing Variability in Space and Time with Hyper Feature Models

Christoph Seidl

Ina Schaefer

Uwe Aßmann

Europa fördert Sachsen. ESF Europäischer Sozialfonds SACHSEN

TECHNISCHE UNIVERSITÄT DRESDEN

> Agenda

- Background
  - Software Ecosystems (SECOs) and Variability
  - Formalization of Feature Models
- Problem: No Variability in Time in Feature Models
- Contributions
  - Hyper Feature Models (HFMs)
  - Version-Aware Constraint Language
  - Automatic Version Selection
- Conclusion: Capturing Variability in Space and Time with HFMs

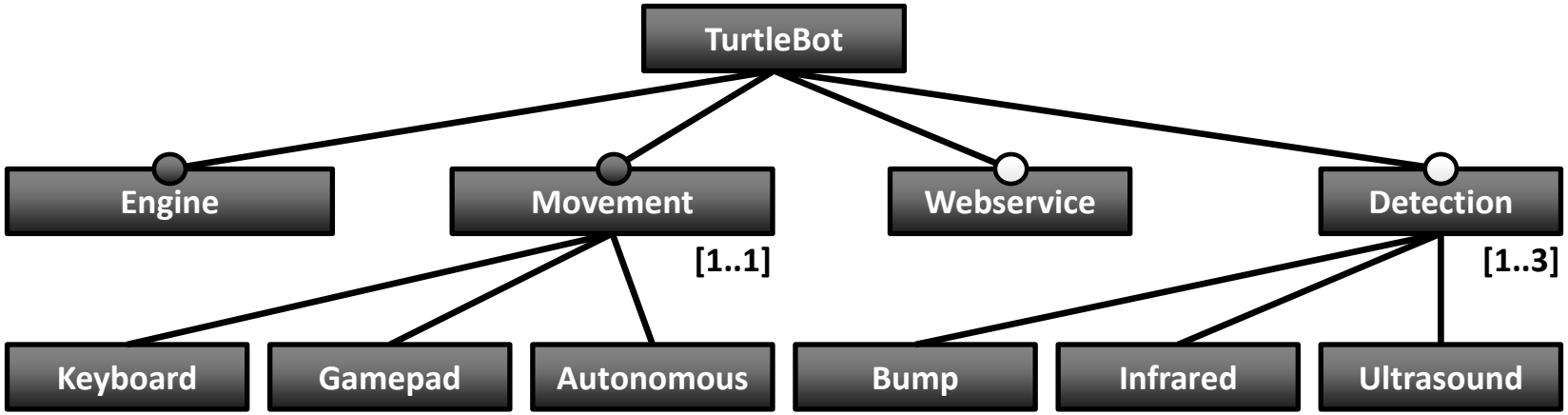
Background

- Software Ecosystems (SECOs)
  - Family of closely related software systems
  - E.g., Eclipse, Android, ROS etc.
  - Similar to SPLs, but…
    - **Multiple** contributors
    - **Frequent (unsynchronized) evolution** of variable assets



- Variability in Space
  - Configuration
  - ➢ <u>Variant</u> of system
- Variability in Time
  - Evolution
  - ➢ <u>Version</u> of system

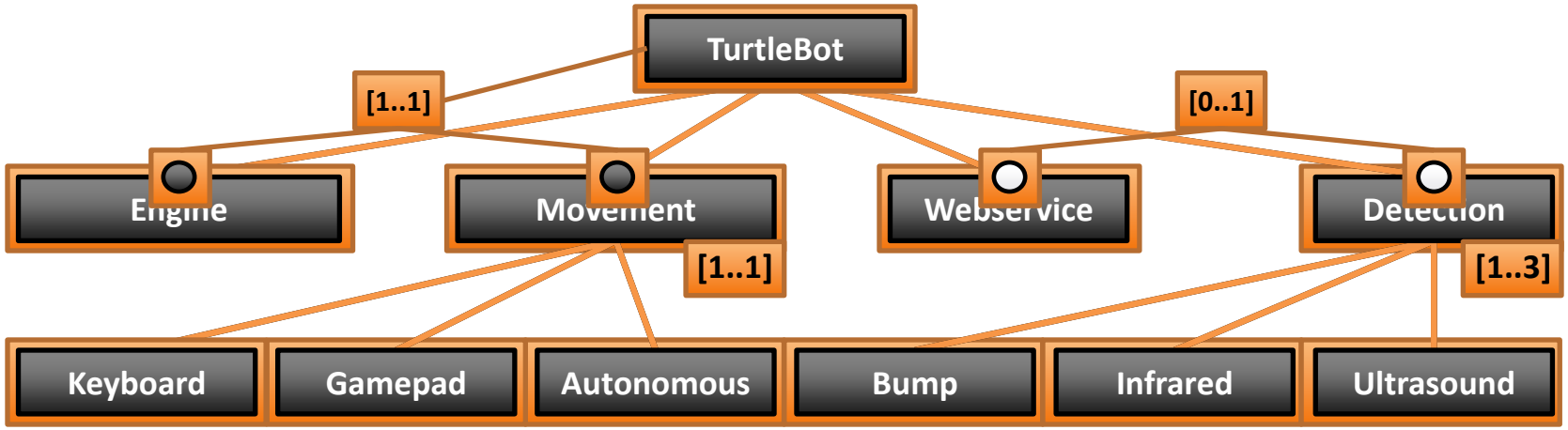
Running Example: TurtleBot Driver



Cross-Tree Constraints

1. Autonomous $\rightarrow$ Detection
2. Keyboard $\vee$ Gamepad $\rightarrow$ Webservice

Definition 1: Feature Model



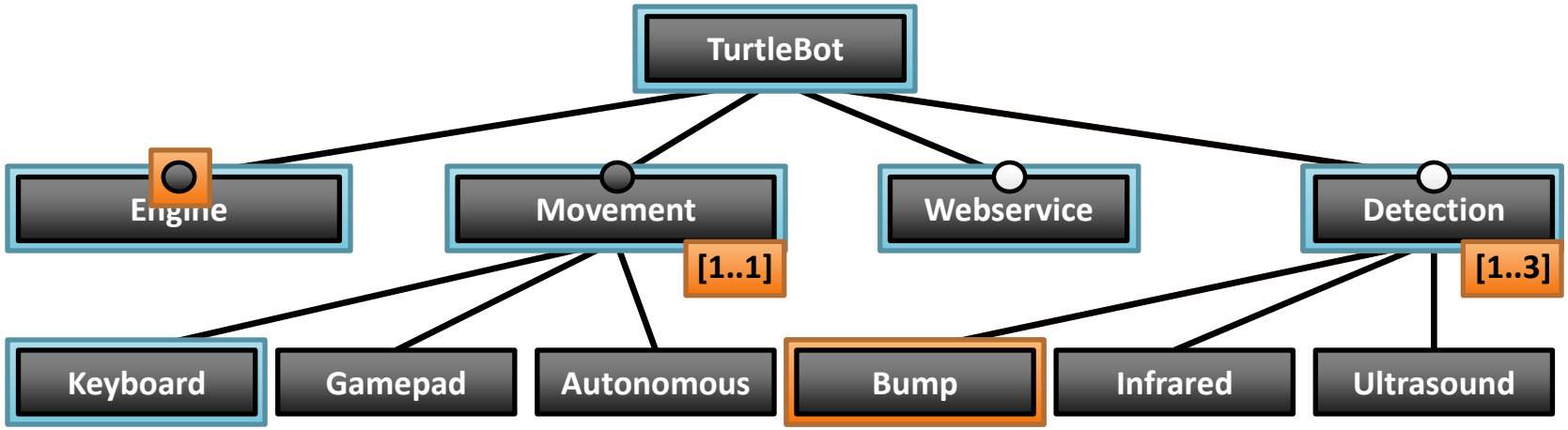
Cross-Tree Constraints

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

Definition 1 (Feature Model). *A feature model is a 4-tuple* $FM = (\mathcal{F}, \prec, \lambda, \Phi)$ *with*

1. $\mathcal{F}$: *a finite set of features*
2. $\prec \subseteq \mathcal{F} \times \mathcal{F}$: *a decomposition relation on* $\mathcal{F}$ *forming a rooted tree of features*
3. $\lambda : \mathcal{P}(\mathcal{F}) \rightharpoonup \mathbb{N}_0 \times \mathbb{N}_0$: *a function assigning minimum and maximum cardinality to features and feature groups*
4. Φ: *a set of propositional formulas over* $\mathcal{F}$ *representing cross-tree constraints*

Definition 2: Feature Model Semantics



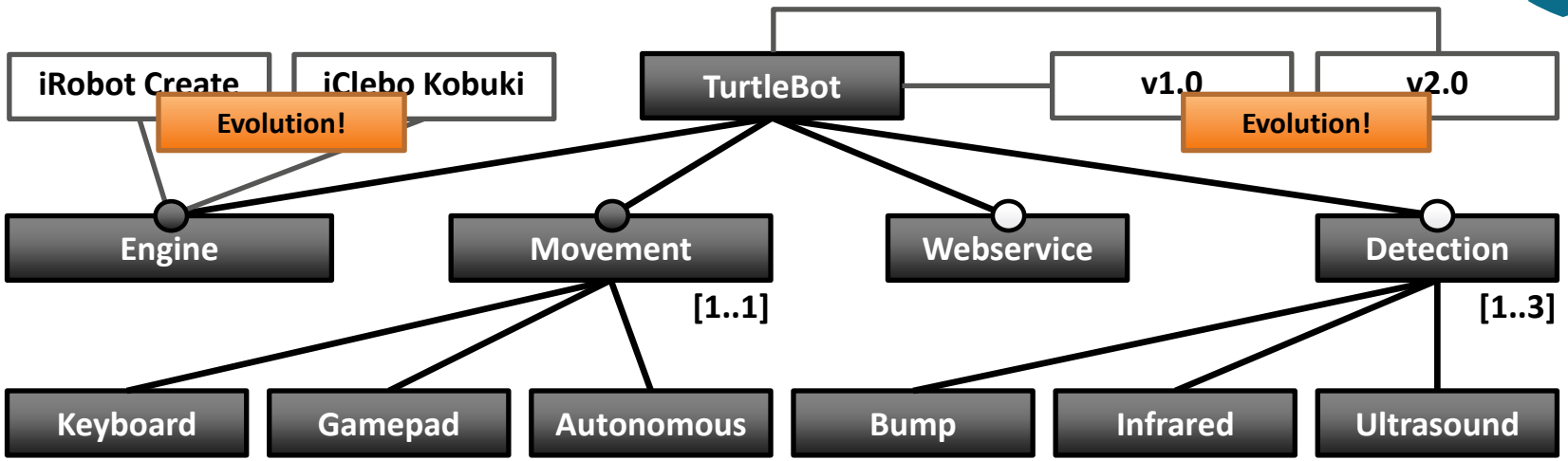
Cross-Tree Constraints

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

Definition 2 (Feature Model Semantics). *A configuration $\mathcal{C} \subseteq \mathcal{F}$ of a feature model has to satisfy the following conditions to be valid:*

1. *Cardinality constraints of features and feature groups are respected by the configuration.*
   $f \in \mathcal{C} \wedge f \prec F \wedge \lambda(F) = (k, l) \Rightarrow k \leq |F \cap \mathcal{F}| \leq l$
2. *For each selected feature, the parent feature has to be in the configuration.*
   $f_1 \in \mathcal{C} \wedge f_2 \in \mathcal{F} \wedge f_2 \prec f_1 \Rightarrow f_2 \in \mathcal{C}$
3. *All propositional formulas of the cross-tree constraints have to be satisfied by the configuration.*
   $\mathcal{C} \models \bigwedge_{\phi_i \in \Phi} \phi_i$

> Problem: No Variability in Time in Feature Models



- Constraints caused by evolution (excerpt)
  - iClebo Kobuki is incompatible with TurtleBot v1.0
  - iClebo Kobuki requires at least TurtleBot v2.0
- Problem
  - Feature model captures only variability in space
  - Only exactly one version of variable asset in feature model (no variability in time)

Requirements on Solution

- **Requirement 1**
  Support for feature versions as units of configuration.

- **Requirement 2**
  Specification of the relation of versions in logical development lines (i.e., predecessor and successor versions including branching).

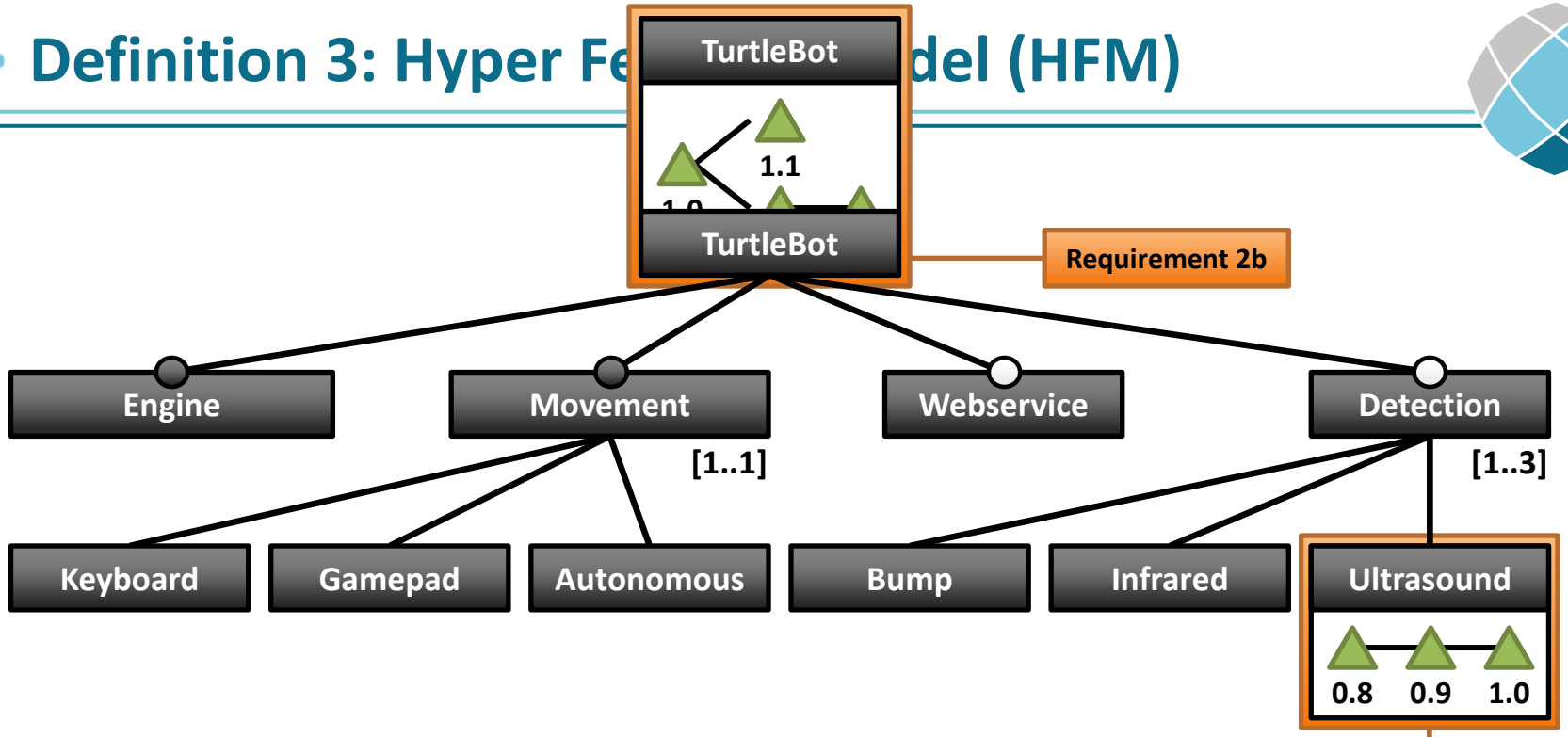
- **Requirement 3**
  Expression of dependencies on and incompatibilities with version ranges.

- **Requirement 4**
  Support for users in selecting suitable combinations of versions from a selection of features.

Contribution 1

Hyper Feature Models (HFMs)

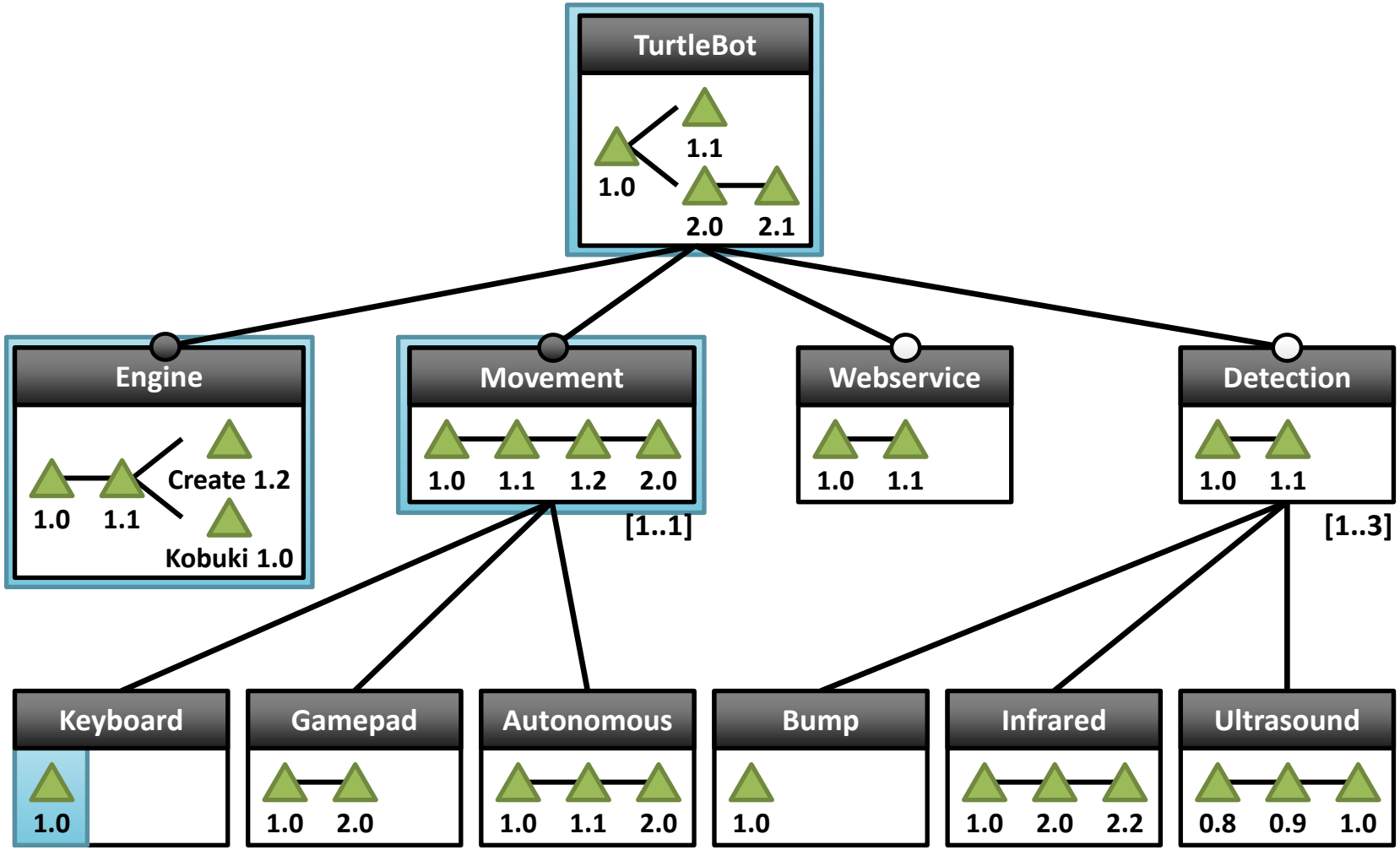
Definition 3: Hyper Feature Model (HFM)



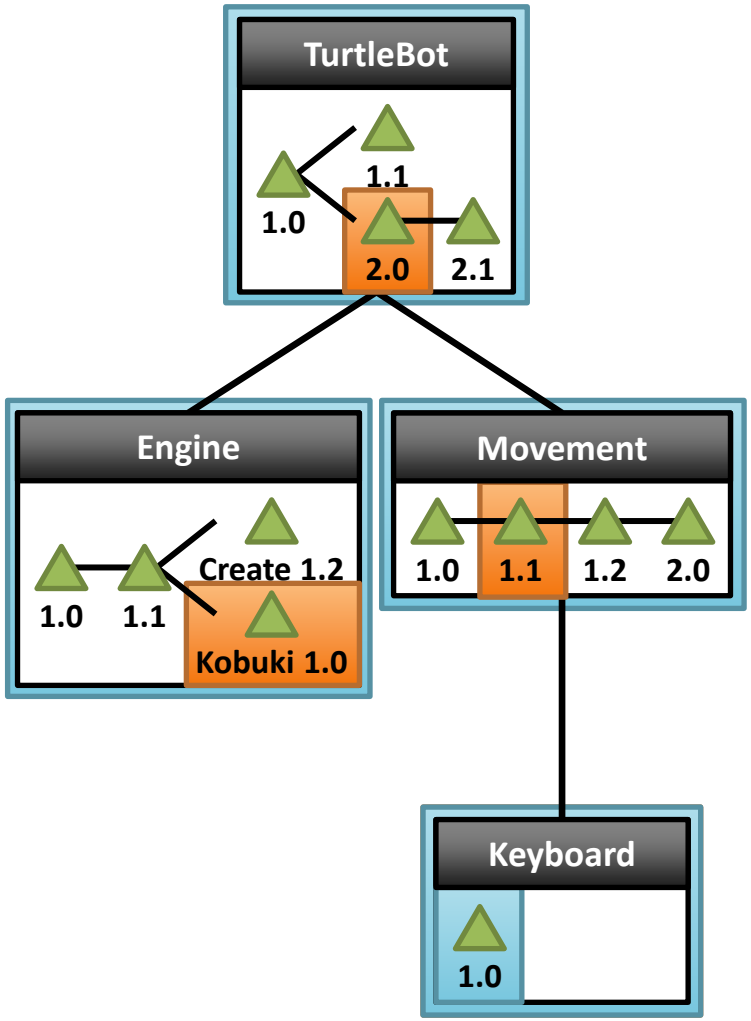
Feature Model (Definition 1)

1. $\mathcal{F}$: *a finite set of features*
2. $\mathcal{V}$: *a finite set of feature versions*
3. $\prec \subseteq \mathcal{F} \times \mathcal{F}$: *a decomposition relation on* $\mathcal{F}$ *forming a rooted tree of features*
4. $\lambda : \mathcal{P}(\mathcal{F}) \rightharpoonup \mathbb{N}_0 \times \mathbb{N}_0$: *a function assigning minimum and maximum cardinality to features and feature groups*

Definition 5: HFM Semantics



Definition 5: HFM Semantics



Definition 5 (Hyper Feature Model Semantics). *A configuration $\mathcal{C} \subseteq \mathcal{F} \cup \mathcal{V}$ of a Hyper Feature Model has to satisfy the following conditions to be valid:*

1. *Cardinality constraints of features and feature groups are respected by the configuration.*
   $f \in \mathcal{C} \wedge f \prec F \wedge \lambda(F) = (k, l) \Rightarrow k \leq |F \cap \mathcal{F}| \leq l$
2. *For each selected feature, the parent feature has to be in the configuration.*
   $f_1 \in \mathcal{C} \cap \mathcal{F} \wedge f_2 \in \mathcal{F} \wedge f_2 \prec f_1 \Rightarrow f_2 \in \mathcal{C}$
3. *For each selected version, the containing feature also has to be part of the configuration.*
   $\forall v \in \mathcal{C} \cap \mathcal{V} : \Upsilon(f) = \mathcal{V}_f \wedge v \in \mathcal{V}_f \rightarrow f \in \mathcal{C}$
4. *For each selected feature, there has to be exactly one version in the configuration.*
   $\forall f \in \mathcal{C} \cap \mathcal{F} : \Upsilon(f) = \mathcal{V}_f \rightarrow \exists v \in \mathcal{V}_f : v \in \mathcal{C} \wedge (v_1, v_2 \in \mathcal{C} \cap \mathcal{V}_f \rightarrow v_1 = v_2)$

Feature Model (Definition 2)

Contribution 2

Version-Aware Constraint Language

Definition 6: Version-Aware Constraint Syntax

Definition 6 (Version-Aware Constraint Syntax). *Let $f \in \mathcal{F}$ be a feature, $v_a, v_b \in \mathcal{V}$ versions and $op \in \{>, \geq, =, \leq, <\}$: $\mathcal{V} \rightharpoonup \mathcal{P}(\mathcal{V})$ a set of operators over versions. The following are basic expressions of the version-aware constraint language:*

Atomic Expressions

1. f	*(feature presence)*
2. $f\ [v_a - v_b]$	*(version range restriction)*
3. $f\ [op\ v_a]$	*(relative version restriction)*
4. $?f\ [v_a - v_b]$	*(conditional version range restriction)*
5. $?f\ [op\ v_a]$	*(conditional relative version restriction)*

Requirement 3

Let ψ_1 and ψ_2 be version-aware constraints. The following are compound expressions of the version-aware constraint language:

Compound Expressions

6. ψ_1	*(basic expression)*
7. (ψ_1)	*(nested expression)*
8. $\neg\psi_1$	*(negation)*
9. $\psi_1 \wedge \psi_2$	*(conjunction)*
10. $\psi_1 \vee \psi_2$	*(disjunction)*
11. $\psi_1 \rightarrow \psi_2$	*(implication)*

Boolean Algebra

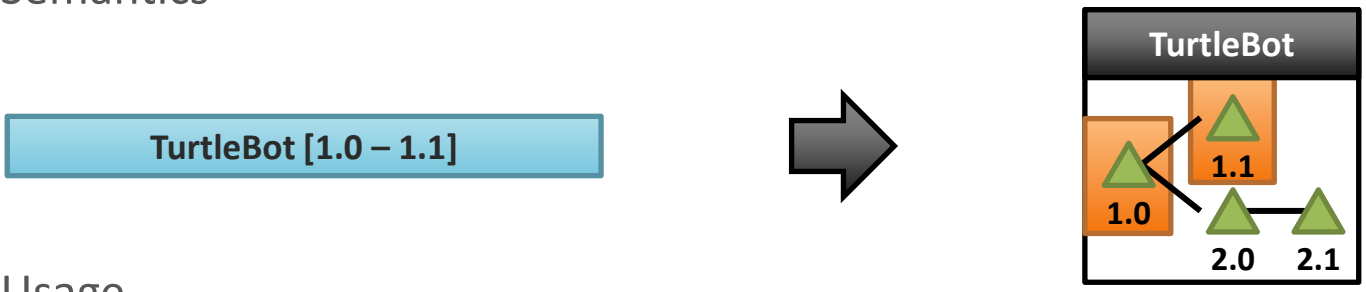
> (Conditional) Version Range Restrictions

$$f\ [v_a - v_b]$$

$$?f\ [v_a - v_b] \equiv (f \rightarrow f\ [v_a - v_b])$$

- Examples
  - **TurtleBot [1.0 – 1.1]** → Engine [≤ Create 1.2]
  - **?Movement [1.2 – 2.0]**

- Semantics

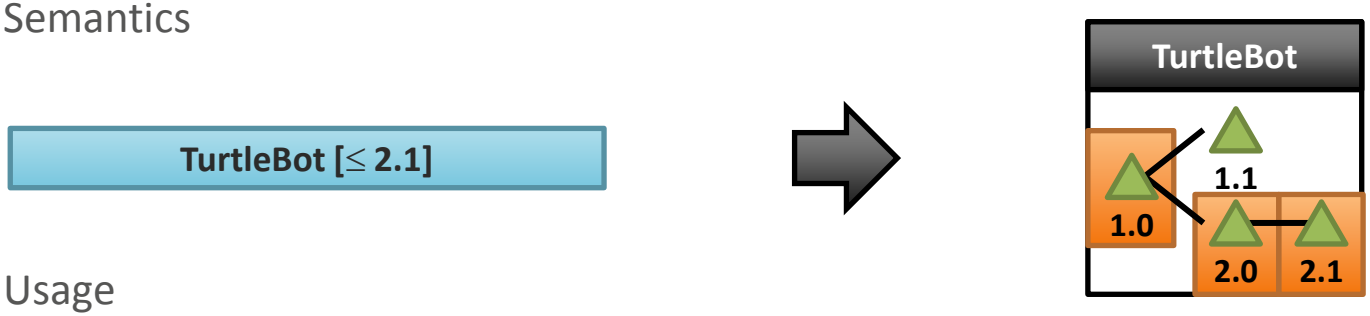


- Usage
  - Closed intervals (fixed even if evolved)
  - E.g., dependencies of Eclipse manifest files

> (Conditional) Relative Version Restrictions

$$\mathbf{f}\ [op\ \mathbf{v_a}]$$
$$?\mathbf{f}\ [op\ \mathbf{v_a}] \equiv (\mathbf{f} \rightarrow \mathbf{f}\ [op\ \mathbf{v_a}])$$

- Operators: op = {>, ≥, =, ≤, <}
- Examples
  - TurtleBot [1.0 – 1.1] → **Engine [≤ Create 1.2]**
  - **?Engine [≥ Kobuki 1.0]**
- Semantics

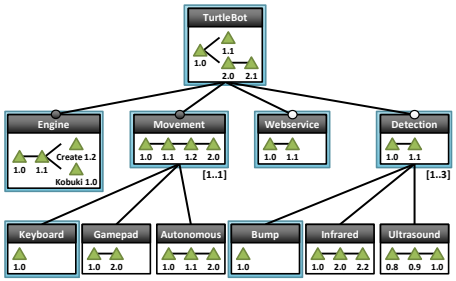


- Usage
  - No need to know all versions in the interval explicitly
  - Open intervals (extended in case of evolution)

Contribution 3

Automatic Version Selection

Automatic Version Selection: General Procedure



+

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]

encode

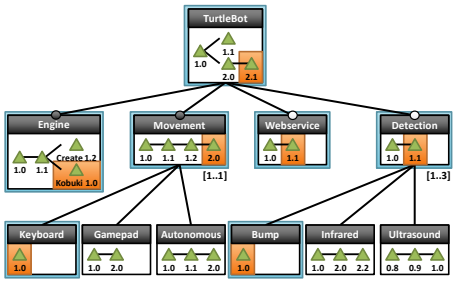
Constraint Satisfaction Problem (CSP) Solver

solve

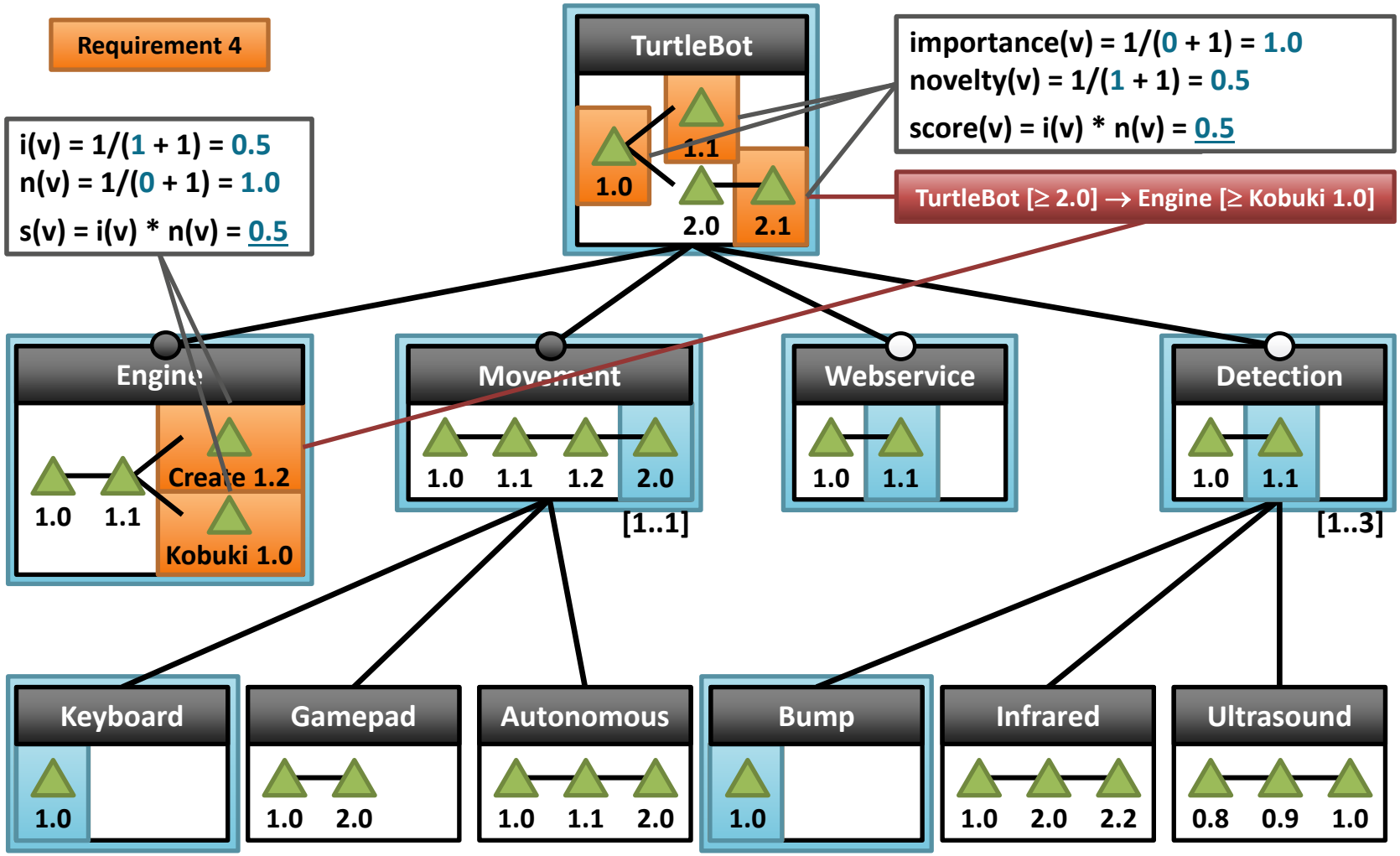
All Possible (Version) Configurations

select

Scoring to find "best" configuration



Scoring to Find "Best" Configuration



> Conclusion

- Problem: No variability in time in Feature Models

- Solution
  - Hyper Feature Models (HFMs)
  - Version-Aware Constraint Language
  - Automatic Version Selection

- Fulfills initially posed requirements
- Evaluated in case study using TurtleBot driver

➢ Capturing variability in space and time in Hyper Feature Models

Thank you for your attention!

Questions, Comments, Feedback?