Variability and Component Composition

Tijs van der Storm

storm@cwi.nl

CWI

Introduction

Traditionally:

- Variability at the level of *product families*
- Components as units of variation

But:

- Not all variability can be factored in component units
- This variability is exhibited by components themselves

$\Rightarrow$ Feature Diagrams on component level.

Problems

But:

- Feature configuration: exponential complexity
- Components may require other components
- Composition may become variable itself

Need to control interaction of:

- dependencies
- feature diagrams

Goals

Goals:

- Establishing conditions for valid composition
- Taming the exponential feature space
- Generating configuration tools (future work)

First steps towards:

- Software Knowledge Base
- Feature Manipulation Environment

Component Interfaces

Component Interfaces:

- Provided interface:
  - feature diagram (FDL)
- Required interface:
  - simple dependencies
  - guarded dependencies

Dependencies address *variants* of components.

NB: guards allow components to be units of variation.

Component Description Language (CDL)

```
component interface ⟨"meta-environment", "1.6"⟩
  provides
    Meta     : all(Rewriter, Type)
    Type     : more-of(batch, studio)
    Rewriter : one-of(asf, elan)
  requires
    when asf {
      ⟨"asf", "1.5"⟩
      ⟨"sdf", "3.0"⟩
    }
    when elan {
      ⟨"elan", "4.5"⟩
      ⟨"sdf", "3.0"⟩ with strategies
    }
```

Valid Composition

Consistency goals:

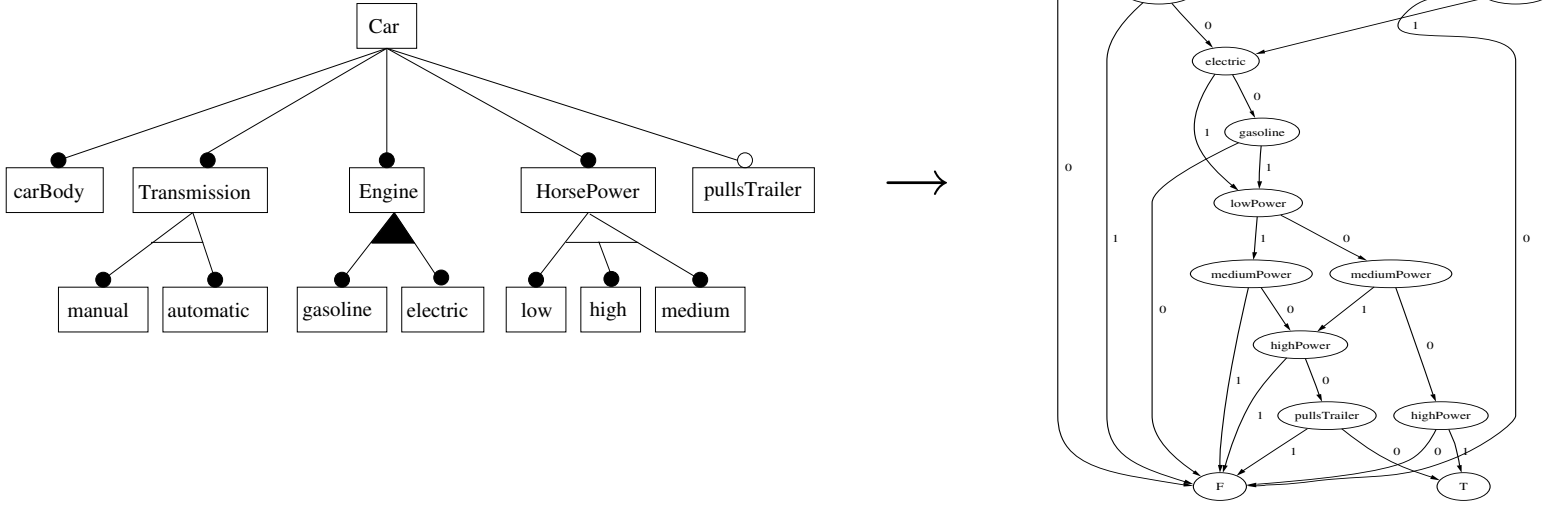
- Consistent feature selection by customer
- Component interface matching:
  - Existing dependencies
  - Consistent switches (features)

$\Rightarrow$ Modelchecking FDL.

Implementation

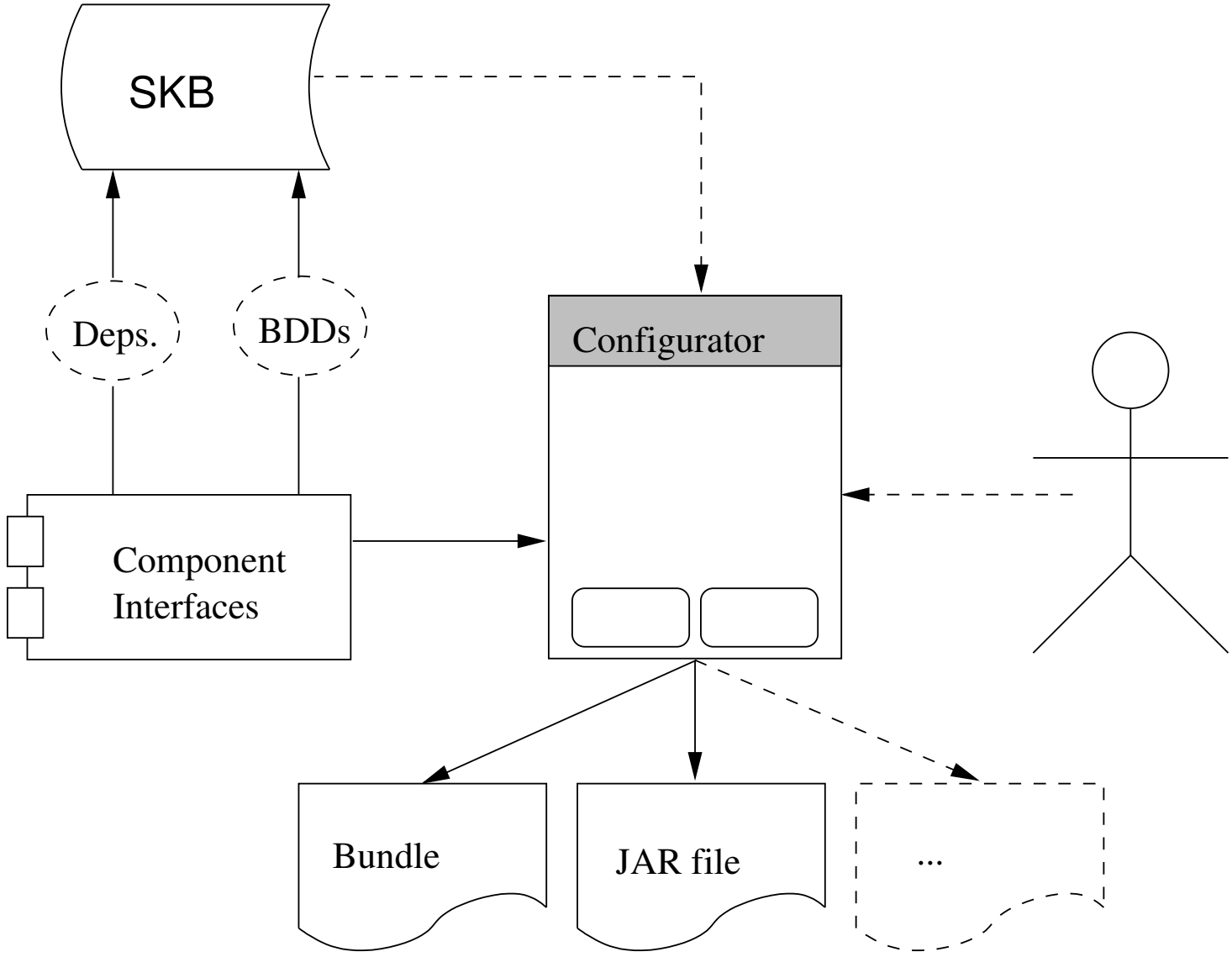
Translate component interfaces to Relations.

FDL to Relations via Binary Decision Diagram:



Consistency checks formulated as *queries*.

A Vision


SKB
Deps.
BDDs
Configurator
Component
Interfaces
Bundle
JAR file
...

Concluding...

We have discussed:

- Components as products need variability
- CDL allows the expression of this
- Composition guaranteed through BDDs and RSCRIPT

CDL enables automatic variability management for CBSE.

Many directions still to explore...

Future Work

Future work:

- Components *inheriting variability* of their dependencies
- Combining *binding time* with *partial evaluation*.
- Feature *propagation* patterns
- Operational semantics for CDL
- Generation of tools from interfaces

Questions?