Understanding Decision-Oriented Variability Modeling

Deepak Dhungana **Paul Grünbacher**

{dhungana | gruenbacher} @ase.jku.at

Christian Doppler Laboratory for
Automated Software Engineering
Johannes Kepler University, 4040 Linz, Austria

SIEMENS
VAI

Introduction

- Several decision-oriented approaches exist
  - Synthesis, V-Manage/ESI, Schmid&John, KobrA, etc.
- We have been developing the DOPLER approach for decision-oriented variability modelling
  - Supported by a set of tools
  - Validated with industry partner
- Motivation for this paper
  - More carefully define our modelling language (DoVML)
  - Understand and crystallize the differences between different decision-oriented approaches.

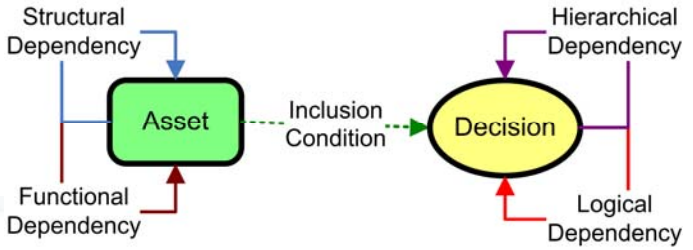
DoVML Application Experiences

- Application domains and contexts
  - Deployment time configuration of steel plant automation (SME 07)
  - Runtime reconfiguration of service oriented systems (VAMOS 08)
  - Runtime adaptation of industrial automation systems (DSPL 08)
  - Configuration management of the DOPLER tool suite (ASE 08)
  - Personalization of enterprise resource planning systems (ICCBSS08)
- Different paradigms (SOC, CBD, IAS)
- Different assets (component, service, function block, ...)
- Different use (design time configuration, runtime adaptation)
- Significant model complexity, eg. Siemens VAI
  - 100+ decisions
  - 500+ artefacts
  - Complex dependencies

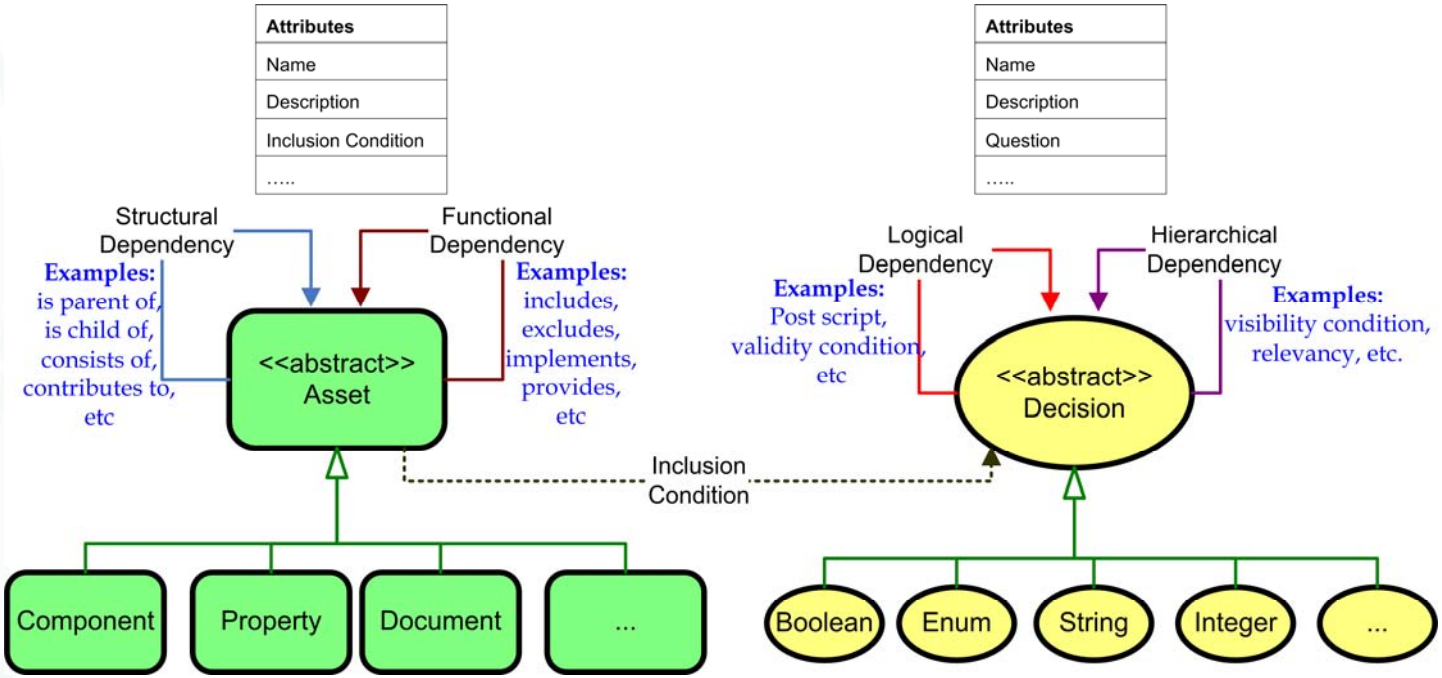
Decisions in PLE

- Decision making is the process of judging the merits of multiple options and selecting one of them for action.
- Variability in PLE is also about choices, alternatives and decisions.
- In our modelling approach, **decisions** are **first class citizens** and not just add-ons to a general concept.
- A decision can represent
  - A step of configuration process
  - Variation points in the artefact space

So, How Does it Work?



Variability modeling involves modeling the problem space and the solution space.



Representing Decisions in a Model

- A decision can be compared to a typed variable in programming languages, enriched with
  - The set of possible values (including infinite sets, range constraints)
  - The specification of its position in the decision hierarchy (in relation to other available decisions)
  - The specification of the implications of taking the decision (on other decisions) and
  - Labels and annotations (information for the user to better understand the decision).

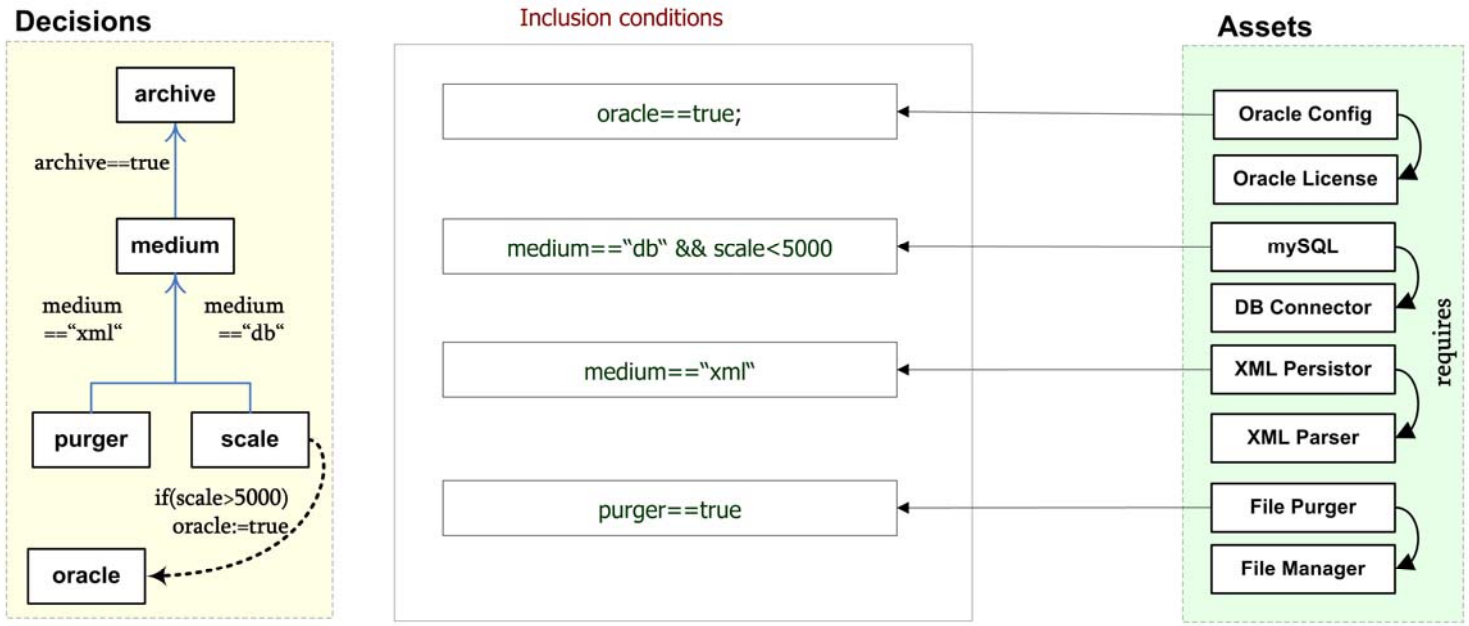
Therefore, taking a decision is equivalent to binding a variable to a value.

Representing Assets in a Model

- Basically Boolean variables
  - Either included or excluded from the final product
  - Value determined by the inclusion condition
- Importance of Asset types
  - Enables domain-specific abstraction for the modeler
  - Component, Test case, .., Document, etc
- Importance of Asset dependencies
  - Not just decisions but also assets are related/dependent on each other
  - Enable domain-specific nomenclature for basic concepts
    - Parent, child, inclusion, exclusion, predecessor, successor, etc.

Different domain-specific tools deal with the list of included assets and their dependencies to create/generate a customer-specific solution.

Example



Partial declaration of decisions (Problem Space)

Boolean decision **archive** visible if true;
Question: Do you want to archive daily records?
Enumeration decision **medium** visible if archive==true;
Question: Which medium should be used for archiving records?
Boolean decision **purger** visible if medium==“xml“;
Question: Do you want old xml files to be automatically deleted?
Integer decision **scale** visible if medium==“db“; valid if scale>=0;
Question: Enter the anticipated number of daily records!
Post-script: if(scale>5000) oracle:=true;
Integer decision **oracle** visible if false;
// state variable, not visible to the user

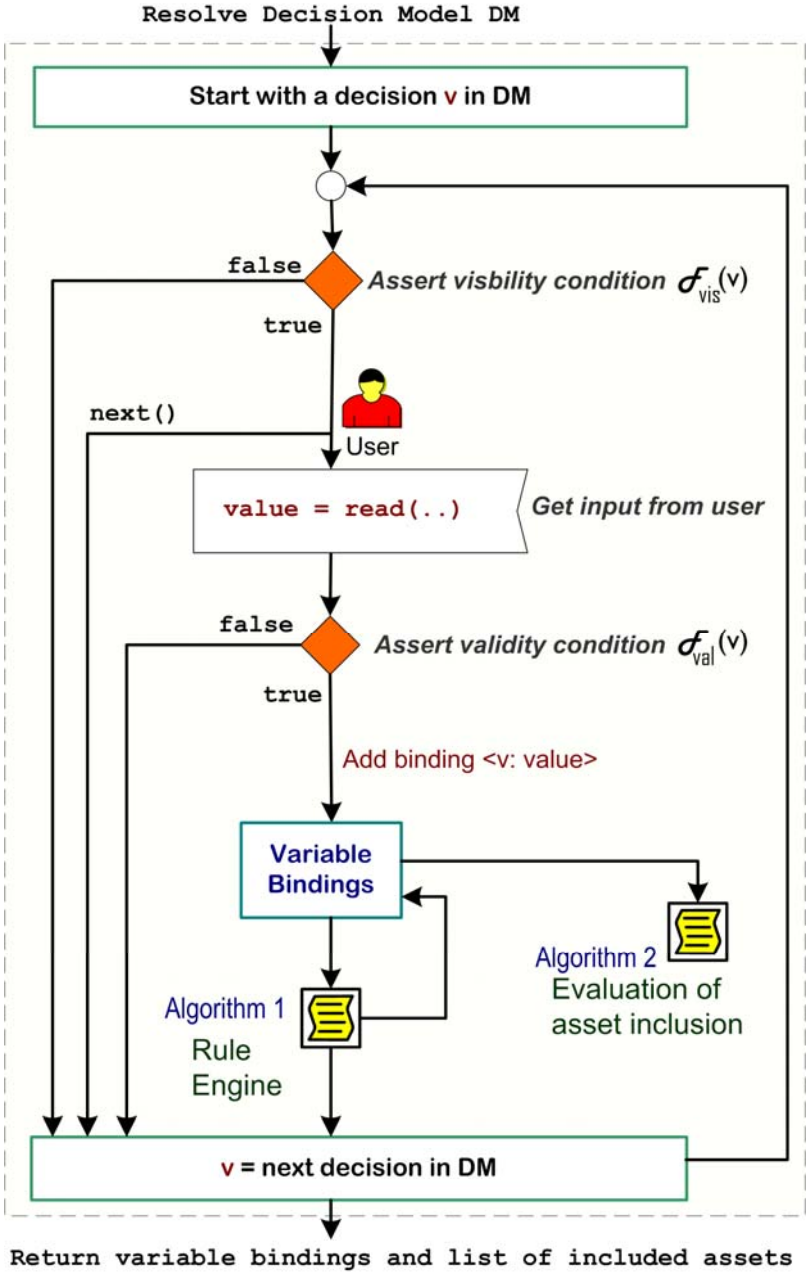
Partial declaration of assets (Solution Space)

Component **XML Persistor** included if medium==“xml“;	
requires {XML Parser}	
Component **mySQL** included if medium==“db“ && scale<5000;	
requires {DB Connector}	
Component **File Purger** included if purger==true;	
requires {File Manager}	
Component **Oracle Config** included if oracle==true;	
requires {Oracle Licence}	
Component **DB Connector;**	Component **File Manager;**
Component **Oracle Licence;**	Component **XML Parser;**

You mean, you execute models?

- Our decision models can be interpreted (“executed“) by tools.
- Users are presented with the list of relevant questions.
- The model answers two basic questions
  - What were the options chosen by the user?
  - What assets are required to fulfil the choices?
- The model guarantees:
  - The options chosen by the user are consistent
  - The list of required assets is complete

Executing a Decision Model



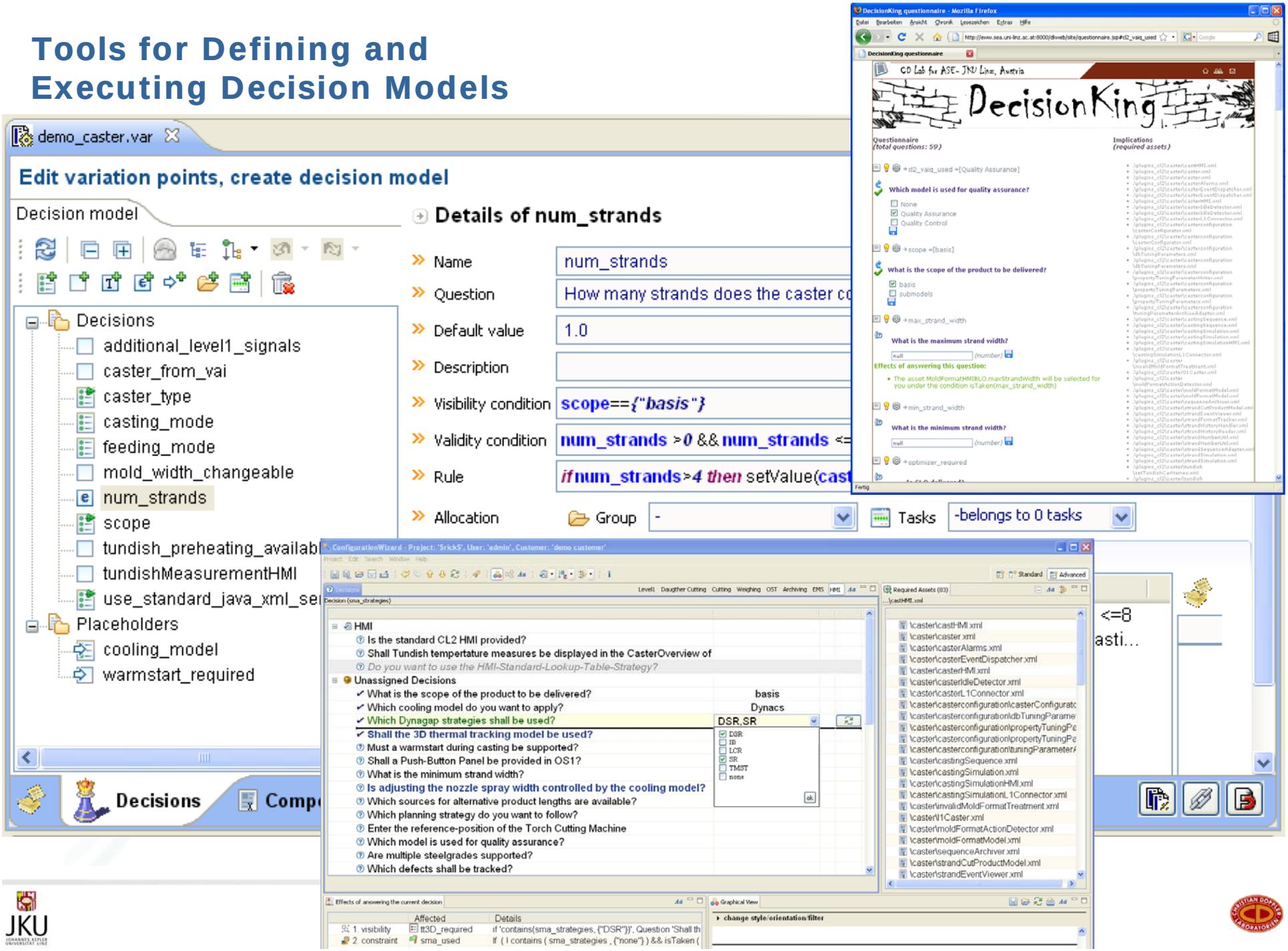
- Currently supported by our tool DecisionKing
  - Decision types: Boolean, Enumeration, String, Number
  - Asset dependency types: inclusion, exclusion, parent, child, predecessor, successor, abstraction, implementation, etc.
- Rule Engine is based on JBoss Engine
  - Custom rule language adding syntactic sugar to JBOSS Drools Rule Language

```
1  if (num_strands>4)then
2    setValue(casting_mode, {"Single"});
3  endif
```

is automatically translated into drools rule:

```
1  rule "0"
2  salience 0
3  no-loop true
4  when
5    num_strands:RuleNumberDecision(
6      name == "num_strands",
7      active==true)and
8      eval(num_strands.getPValue()>4)then
9        ArrayList<String> drools_a;
10       i.identify();
11       drools_a = new ArrayList<String>();
12       drools_a.add("Single");
13       i.set(0,"casting_mode",
14         new ArrayList<String>(drools_a));
15   end
```

Tools for Defining and Executing Decision Models



Summary and Future Work

- DOVML consists of key modeling elements Decisions and Assets
- Tool support is required for
  - Specification of rules
  - Evaluation and execution of models
- We have applied the approach in a number of development contexts
  - Deployment time configuration of steel plant automation (SME 07)
  - Runtime reconfiguration of service oriented systems (VAMOS 08)
  - Runtime adaptation of industrial automation systems (DSPL 08)
  - Configuration management of the DOPLER tool suite (ASE 08)
  - Personalization of enterprise resource planning systems (ICCBSS 08)
- Ongoing Work
  - More formal representations of decision-oriented variability models and their formal semantics
  - Formal comparison of our approach with other decision-oriented approaches
  - Consistency checking and static analysis of decision models

Questions?
Now- or later!

Understanding Decision-Oriented Variability Modeling

Deepak Dhungana **Paul Grünbacher**

{dhungana | gruenbacher} @ase.jku.at

Christian Doppler Laboratory for
Automated Software Engineering
Johannes Kepler University, 4040 Linz, Austria

SIEMENS
VAI