Evaluating Variability Implementation Mechanisms

BOSCH

International Workshop on Product Line Engineering The Early Steps

November 5, 2002, Seattle, WA

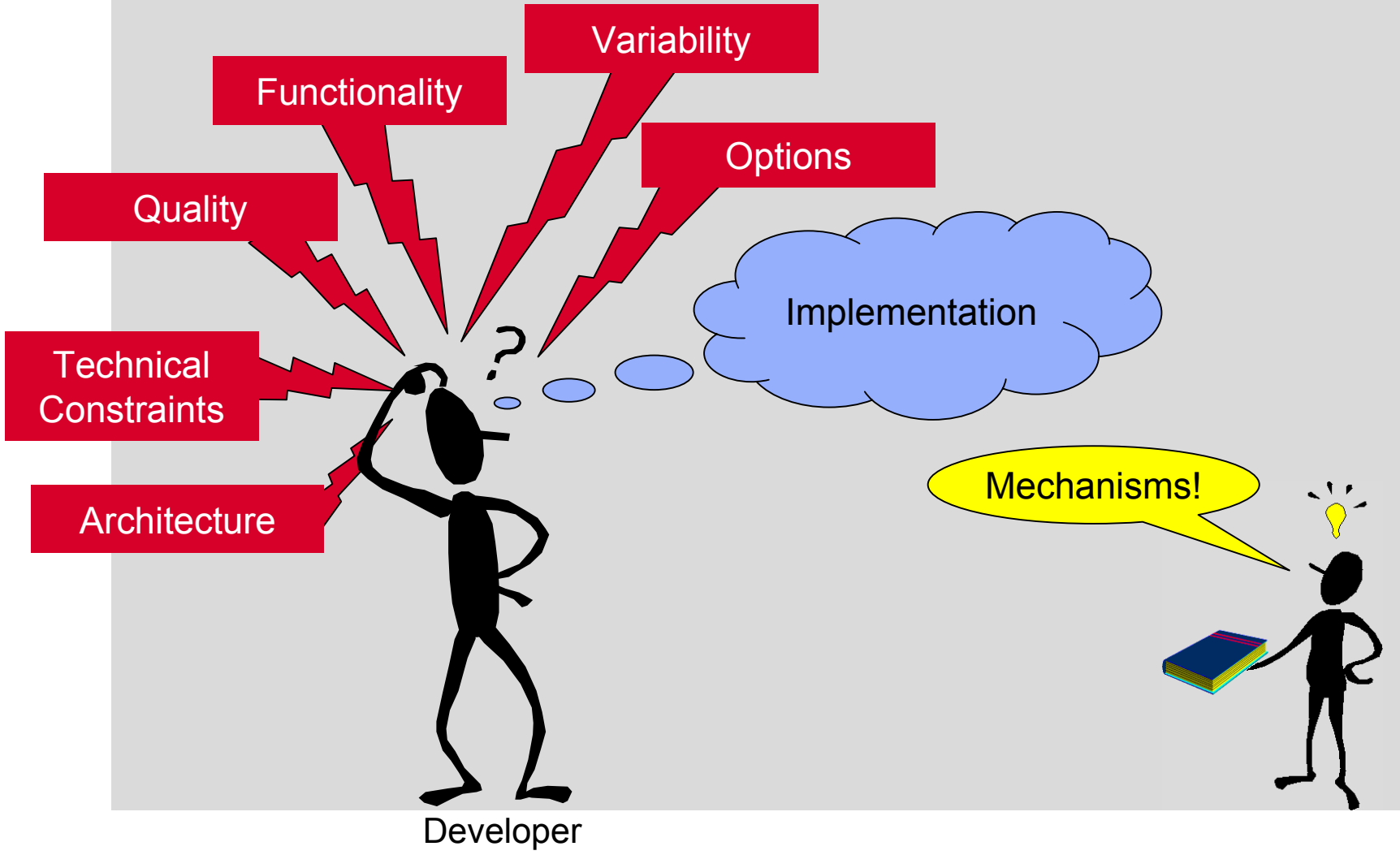
Claudia Fritsch

Robert Bosch GmbH
Corporate Research and Development Software Technology
Frankfurt, Germany

Claudia.Fritsch@de.bosch.com

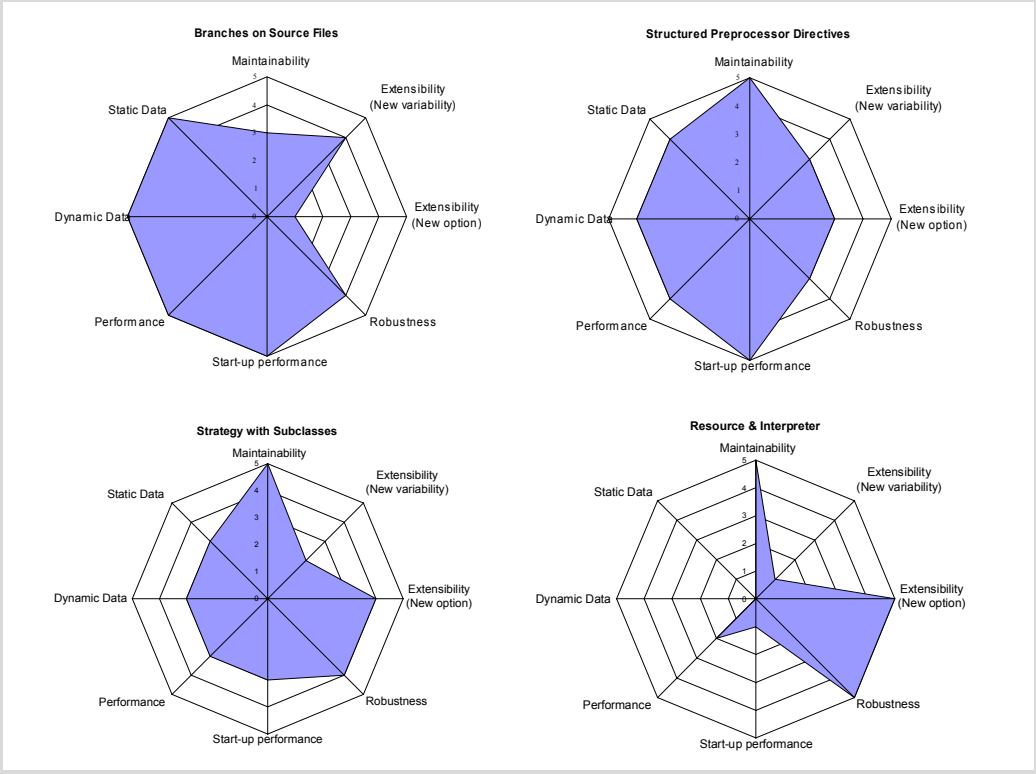
Evaluating Variability Implementation Mechanisms

BOSCH



BOSCH

A Catalog of Evaluated Variability Implementation Mechanisms



Catalog contains

- mechanism descriptions
- mechanism evaluations according to relevant qualities
- Kiviat diagrams as index

BOSCH

Way to the Catalog

- Identify and document mechanisms
- Identify and document qualities
- Evaluate the mechanisms according to the qualities

Our intention is

- to enable a software development team to build their catalog of VIMs
- to start an exchange of knowledge and techniques

1. Identify and Document Mechanisms – Example

Mechanism: **Scattered Preprocessor Directives**

Solution: Use preprocessor directives for conditional compiling.

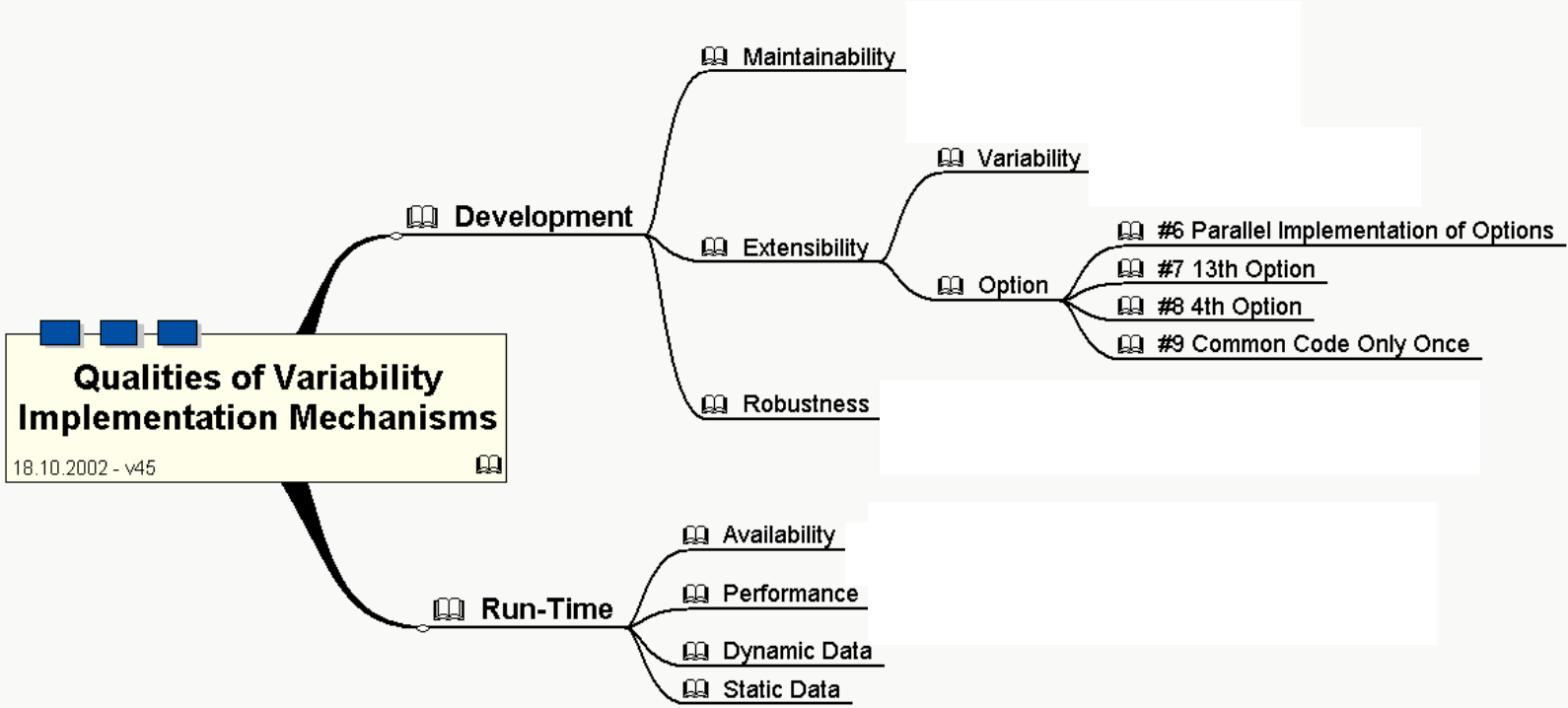
Example code: ...

Implement the options: Enclose option-specific code in `#if defined(...) ... #endif`.

Select an option: A product-specific config.h defines the option which the product should have. Thus, the preprocessor will select the corresponding code, and will discard the code enclosed in `#if`s which are NULL. The configuration files should be assigned to product-related views, such that automatic build processes for the different products can easily be achieved.

Binding time: Compilation of platform code for a certain product.

2. Identify and Document Qualities – Example

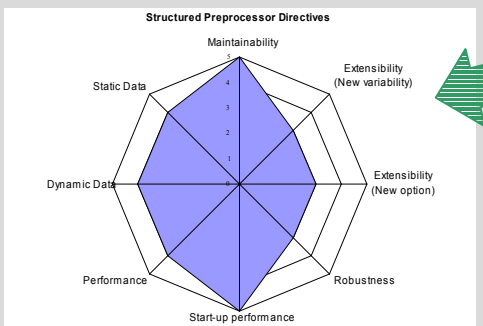


#7, 13th Option: For a variability, 12 options are implemented. A 13th option has to be implemented. Introducing the 13th option is as easy as introducing the 2nd.

BOSCH

3. Evaluate Mechanisms According to Qualities

- there is no absolute measure
- we used relative measuring
- find for one quality
  the best and the worst mechanism
- rank the remaining mechanisms
- use a scale, e.g.,
  ++ + o - --
- explain the ranking



Mechanism		Quality: Maintainability (Bug Fix One Option)	Extensibility (2nd Variability)	Extensibility (13th Option)	Robustness	Availability	Performance	Dynamic Data	Static Data
[B]	Branches on Source Files	o	+	--	+	++	++	++	++
[P1]	Scattered Preprocessor Directives	o	+	--	--	++	++	++	++
[P2]	Structured Preprocessor Directives	++	o	o	o	++	+	+	+
[C]	C++ Conditional Compilation	+	o	o	o	++	+	+	+
[O]	Polymorphism with Subclasses	++	-	+	+	o	o	o	o
[SS]	Strategy with Subclasses	++	-	+	+	o	o	o	o
[ST]	Strategy with Template Parameters	++	-	+	+	+	o	o	o
[Se]	Selection on Configuration Data	+	o	o	o	-	o	-	--
[RI]	Resource & Interpreter	++	--	++	++	--	-		

Issue: Do You Know more Mechanisms?

We have analyzed

- Buschmann et al.: Pattern-Oriented Software Architecture
- Gamma et al.: Design Patterns
- Coplien: Multi-Paradigm Design for C++
- Czarnecki, Eisenecker: Generative Programming
- Svahnberg: A Taxonomy of Variability Realization Techniques
- Kiczales: Aspect-Oriented Programming

So far, our mechanisms only require known techniques and existent tools.

Issue: How Do We Transfer Knowledge about Mechanism Qualities?

We want to offer evaluated mechanisms because

- deciding for and applying a mechanism introduces qualities into a system while withdrawing others.
- engineers like to know (or should know) a priori which qualities a mechanism affects.

But: Evaluating proved to be difficult.

How would you measure a mechanism's quality?

- before–after: unsuitable
- absolute: unfeasible
- Are Kiviat diagrams the most suitable presentation?

Issue: Implementation Counts

Architecture is important.

Understanding commonalities & variabilities is important.

Modeling is important.

But: Implementation is essential.

You need to know adequate implementation techniques.

Ideally, we would like to offer a non-stop variability management, continuous through all process steps.

Issue: Do We Need such a Catalog?

Benefits:

- Developers exchange knowledge.
- Catalog is a reference book.
- Code will be more homogeneous.
- Catalog enables exchange of knowledge across business units.

Risks:

- It takes time to make the catalog.
- Is the cost of making such a catalog worth the effort?

Issue: Why Do We Need the Catalog Early?

Finally, your software product line manifests itself in code.

You need guidelines and standards for coding – early.

It takes time to make such a catalog – as soon as you have started to write production code, you won't have the time.

You need to disseminate knowledge about PL techniques – early.

Useful information creates interest and helps to reduce fear.

BOSCH

Recapitulation

- I have presented a method to
  - identify,
  - document, and
  - evaluate

  mechanisms to implement variability.
- Issues
  - Do you know more mechanisms?
  - How do we transfer knowledge about mechanism qualities?
  - Implementation counts!
  - Do we need such a catalog?
  - Why do we need the catalog early?

Additional Slides ...

Terminology

A variability is a difference between products of a product line.

The differences are defined by a set of options.

Variability manifests itself in

- optional features in requirements engineering
- variation points in design
- different implementations

The binding time is the process step, in which a variability gets decided.

A mechanism is

- an architectural or design pattern
- a guideline
- an idiom
- a technical trick