

Working Session

Implementation Mechanisms for Variability (summary)

Merijn de Jonge

(<http://www.cs.uu.nl/~mdejonge>)

December 3, 2004



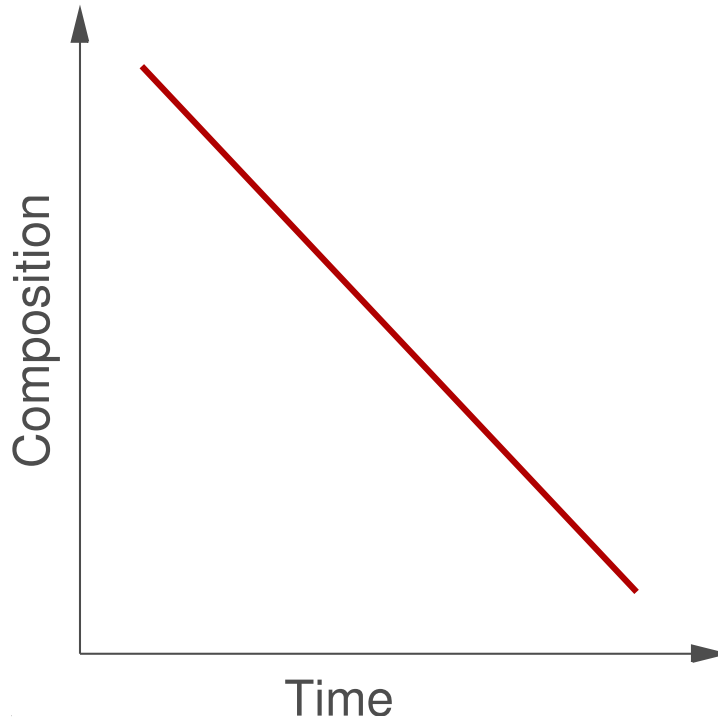
Overview

- Binding time
- Implementation mechanisms
- Configuration realization
- Programming language support

Binding Time

- A variation point can be bound at different moments in time (*timeline variability*).
- E.g., compile-time, deployment-time, startup-time, run-time
- Different benefits: efficiency (compile-time) ↔ flexibility (run-time)
- Not all variation points can be bound at any time
- Binding time(s) should be modeled explicitly
- Does anyone know a feature model that does this?
- State updates:
 - run-time: data structures
 - deployment-time: configuration files

Late Compile-time Binding



(Late binding \neq dynamic binding)

Implementation Mechanisms

- There are plenty
 - preprocessor, pointers, if-then-else, strategy patterns, reflection, configuration files, command-line switches, property files, generators, dynamic linking, . . .
- Depend on binding time
- Multiple binding times are hard to combine
- Many mechanisms are just alternatives, yet they are often used together
- Complicates software understanding/development/maintenance
- Fewer mechanisms is better
- Can we come-up with some minimal but sufficient set of mechanisms?
- Do we want to choose variation mechanisms anyway?

Configuration Realization

- Product lines: From a configuration in the problem space, automatically derive an implementation in the solution space.

Which known approaches do exist?

- Koala
 - Linux kernel (low-level)
 - GEARS
 - (MOCCA)
 - Spring framework
 - EVCL
- Product software (deployment-time variability)
 - Microsoft office suite
 - Exact Software

Programming Language Support

Do we want to choose variation mechanisms anyway?

- Only very limited support of variability in programming languages
- C basically has only run-time variability
- Koala abstracts over binding time
- In Java: use configuration class to query bindings
 - dynamic approach
 - does not cater for compile-time optimization
- Proposal: make variation points first-class in programming languages
- Compiler can use knowledge of (compile-time) bindings for e.g., program optimization
- Also beneficial at other binding-times
- Which forms of variability are essential and should be supported?