

Multi-level Component Composition

Merijn de Jonge

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

December 3, 2004



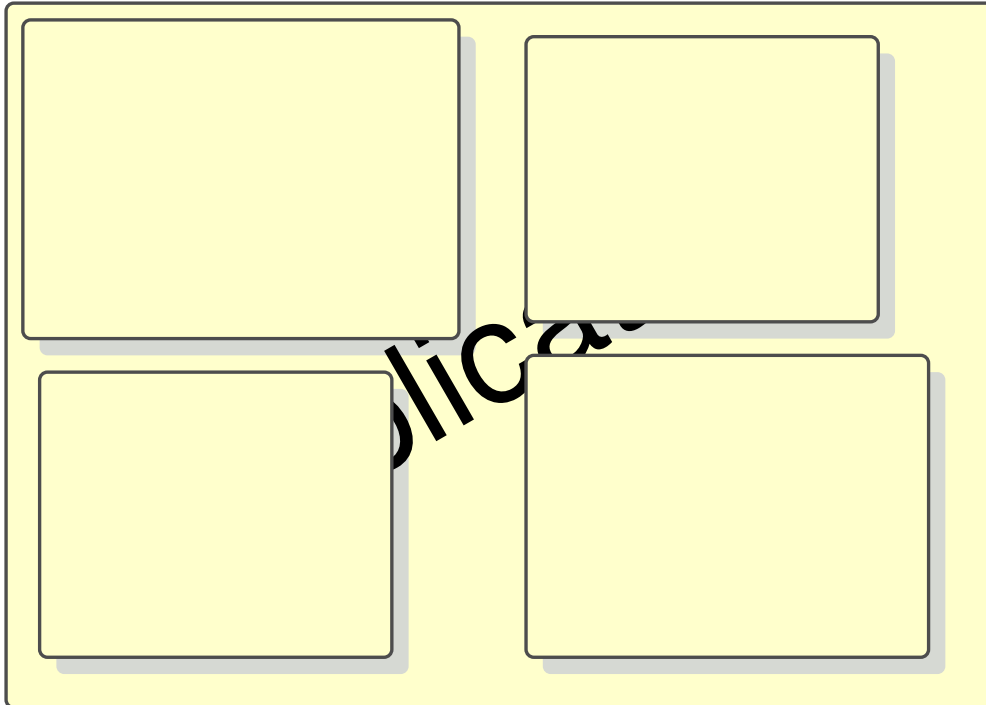
Overview

- Introduction
- Multi-level component composition
- Example architecture

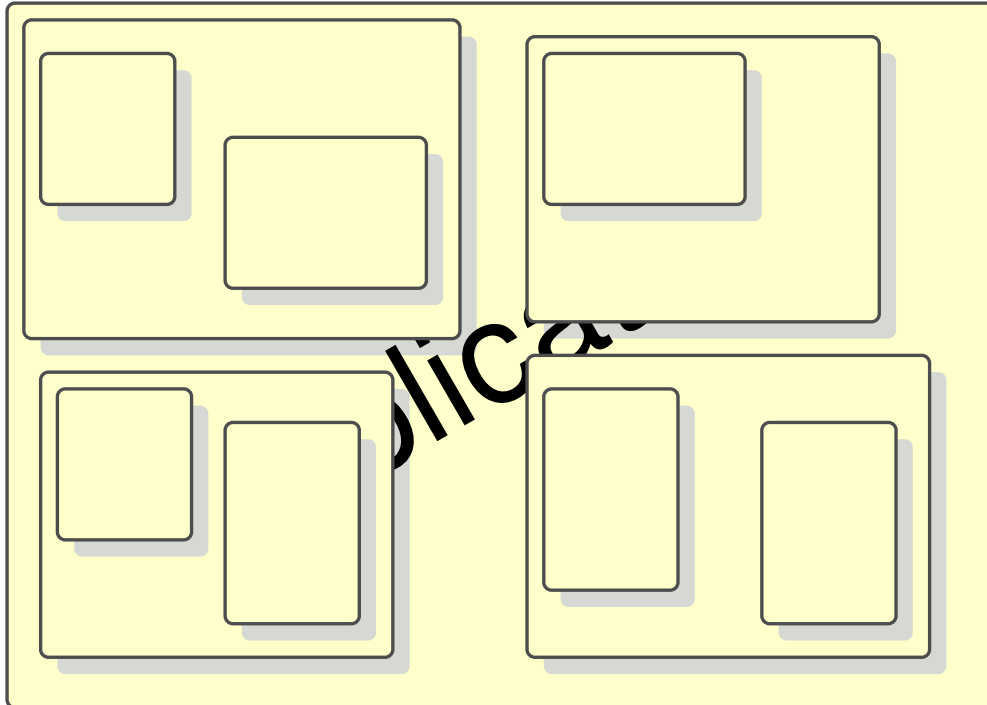
Atoms in Software

Application

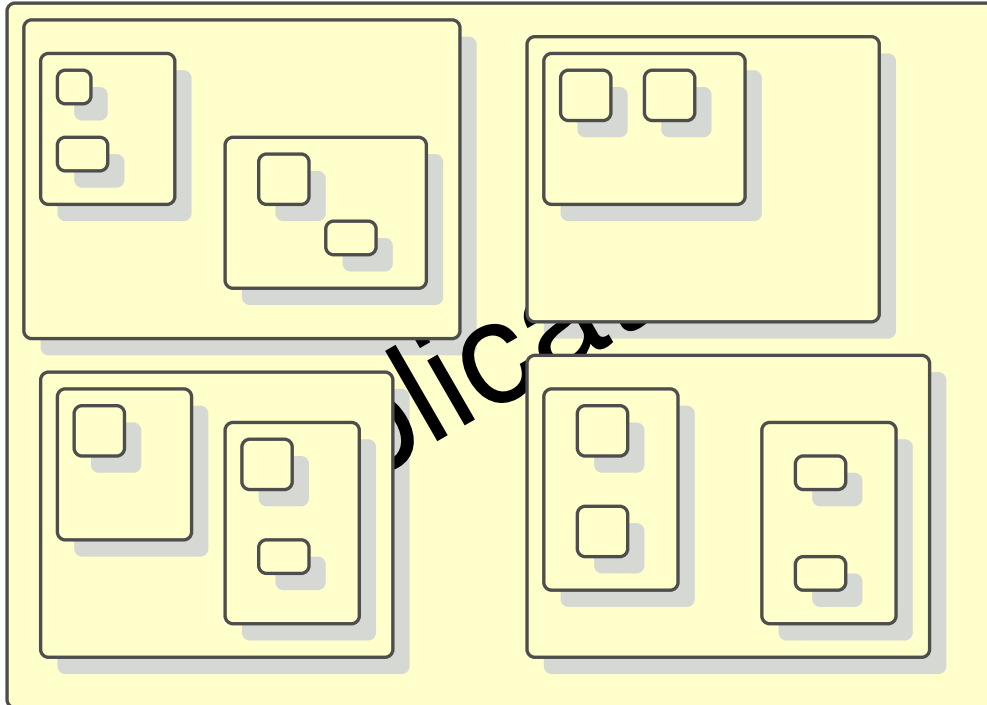
Atoms in Software



Atoms in Software



Atoms in Software



Atomic Components?

- Application
- COM component
- Library
- Module
- Function
- Statement

Different Composition Levels

- Different forms of components
- Different granularity
- Different composition mechanisms
- Different variability mechanisms
- Different modeling mechanisms (if any)

Problems

- No uniform model of components
- No uniform model of variability
- Composition is not transparent
- Composition mechanism prescribes component form
- Components at different levels are orthogonal entities

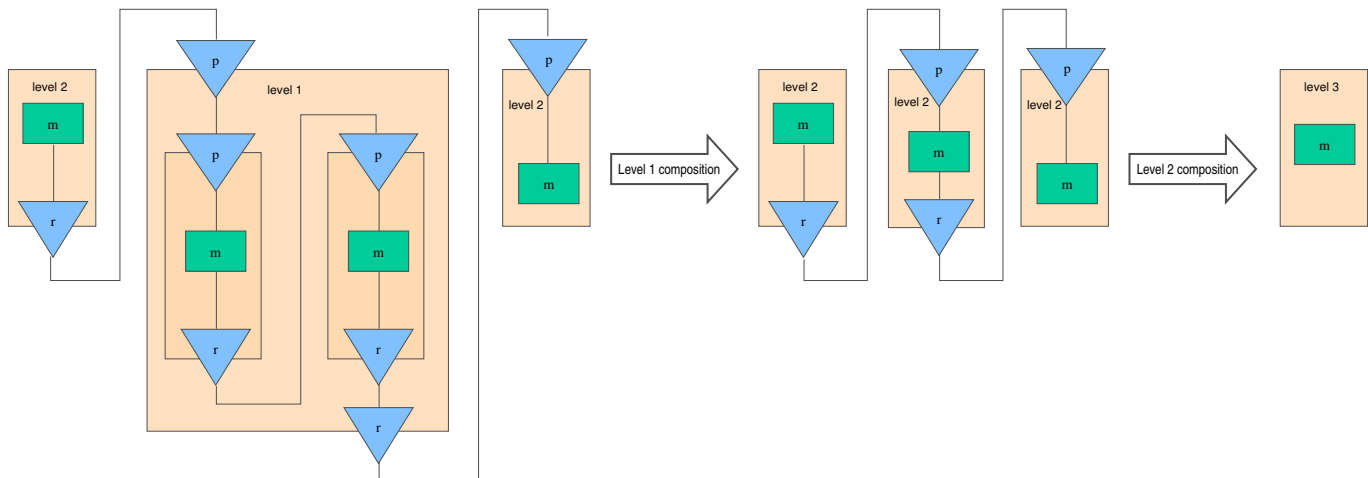
Multi-level Component Composition

Composition and variability are cross-level concerns

- Uniform component model
- Uniform variability model
- Composition at level i forms component at level $i + 1$
- Variability may span multiple levels

Cross-level Composition

- Multi-level composition allows that components can be treated uniformly at different levels
- Cross-level composition combines all levels in a single architecture



Approach

- Identify basic composition concepts
- Use uniform architectural description language (Koala)
- A generic notion of abstraction. (i.e., Koala with generalized module concept)
- 2-phase composition
 - Level (and language) independent normalization phase at architectural level (Koala)
 - Realization phase with level-specific composer tools
- A new level or programming language can be supported iff the composition concepts can be mapped

Example Architecture

- Preliminary work
- Static component composition
- Components:
 - module
 - program = module+ (level 1)
 - package = program+ (level 2)
 - bundle = package+ (level 3)
- Technology
 - Koala
 - Stratego/XT
 - GNU Autotools

Koala Component Model

- Architectural description language
- Interfaces
 - Provides
 - Requires
 - Diversity
- Component: unit of reuse
- Module: interface-less component (abstraction for C module)
- Cable: binding between interfaces

Koala for C

- Module corresponds to C module
- Composition represents composition of C modules (= program or library)
- Variability is mapped to C constructs (or evaluated in Koala model)
- Cables are mapped to C bindings (`#defines`) between function definitions and function calls

Generalize Koala to enable

- composition of programs, . . .
- composition with non-C modules

Level-1 Composition

- Composition is program
 - Variability interface: command-line switches
 - Variability binding: option passing
 - Component binding?

- Program transformation with Stratego/XT
 - Uniform data exchange format eases data exchange across levels
 - Stratego to show that Koala can also be used for other languages
 - Variability interface: ArgOption library
 - Component binding: module imports mechanism

Level-2 Composition

- Composition is package
 - Variability interface?
 - Variability binding?
 - Component binding?
- Autoconf/Automake
 - Variability interface: configure options
 - Variability binding: configure tool with option passing
- XTC
 - Component binding: XTC repositories

Level-3+ Composition

- Composition is bundle
 - Variability interface?
 - Variability binding?
 - Component binding?
- Autoconf/Automake
 - Variability interface: configure options
 - Variability binding: configure tool with option passing
- XTC
 - Component binding: XTC repositories

State of Art

- Software
 - Koala C-composition (philips.com)
 - Autobundle (program-transformation.org)
 - 2-phase Koala compiler (program-transformation.org)
 - Koala source tree composition (program-transformation.org)
 - Open source Koala implementation (to appear)
 - Stratego/XT framework (program-transformation.org)
- Articles
 - Source tree composition (ICSR 2002)
 - Decoupling source trees into build-level components (ICSR 2004)
 - Build-level components (submitted)
 - Build-level component-based software engineering (submitted)
 - XTC: Dynamic component composition with XTC (in preparation)