Uniform Composition of Software and Documents

Uwe Assmann
Research Center for Integrational Software Engineering (RISE)
PELAB
Linköping University, Sweden
EASYCOMP Consortium
Nov 2002
Overview

Some problems in document processing
Uniform composition
  Invasive composition
  Component model for uniform composition
  Composition operations

EASYCOMP's role
  UNICOMP framework for uniform composition
One of the central insights of the nineties is:

*Separate architecture (composition) from the base components*
Purpose

Get a second level of variability
architecture and components can be varied independently of each other
Scale better by different binding times of composition programs
Get coarse-grain descriptions of a system

However, how to be uniform also for documents?
As already \cite{mcilroy.68} has shown, we need components for a ripe industry.

@InProceedings{mcilroy.68b,  
author = "M. Douglas McIlroy",  
title = "Mass-Produced Software Components",  
booktitle = "Software Engineering Concepts and Techniques (1968 {NATO} Conference of {S}oftware Engineering)",  
editor = "J. M. Buxton and Peter Naur and Brian Randell",  
publisher = {NATO Science Committee},  
pages = "88--98",  
month = oct,  
year = "1968"  
}
Usual Solution

latex writes \citation to .aux-file
bibtex greps them and produces a .bbl file
.bbl file is included

How does the architecture of a latex document look like?
May Be Like This…

\cite
\cite
\cite
\cite
\bibliography{}}

.bib file

.bbl file

bibtex
2 – \texttt{\begin{deliverables} in \LaTeX\end{deliverables}}

Project Plan

\begin{tabular}{|l|l|l|l|l|c|l|}
\hline
\begin{verbatim}
\begin{deliverables}
\end{verbatim} & EASYCOMP workshop I & \DIS.1.1 & \UKA & 12 & W & PU & 18 \\ & EASYCOMP workshop II & \DIS.1.2 & \UKA & 12 & W & PU & 30 \\ & Web-based Composition Centre & \DIS.2 & \UKA & 3 & H & PU & 36 \\ & Composition Handbook & \DIS.3 & \UKA & 14 & R & PU & 24 \\ & Final Report & \DIS.4 & \UKA & 6.5 & R & CO/PU & 36 \\\end{tabular}

\end{tabular}
Purpose

extract deliverables by perl script
concat to latex table
include table

How does the architecture of that document look like?
Like This...

\begin{deliverable}
\begin{deliverable}
\begin{deliverable}
\begin{deliverable}
\input{deliverables}

perl

{pattern "\begin{deliverable}"}
By The Way - It Should Use an LAG(1)
3 – A Simple Web Page, Generated By a Database

<html>
..  
<table>
   <tr> <td> Employee </td> <td> Address </td> </tr>
   <tr> <td> Uwe Assmann </td> <td> Farhagsvägen 128 </td> </tr>
   <tr> <td> Robert Kaminski </td> <td> Platensgatan 9 </td> </tr>
   <tr> <td> Jens Gustavsson </td> <td> Stora Torget 14 </td> </tr>
</table>

</html>
run the embedded script of an HTML template
start SQL query in MySQL
transform (with XSLT) the plain text to HTML
include table and replace the embedded script

How does the architecture of that document look like?
Like This...

Uwe Assmann: Fårhagsvägen 128
Robert Kaminski: Platensgatan 9
Jens Gustavsson: Stora Torget 14

MySQL

XSLT

is replaced by

<script>

<table>
Conclusion

Why don't we define document architectures?
That allows for extracting the architecture and separating it from „components“

Software architecture and composition have been successful for
developing in the large, reuse,
and the notion of architecture

Why don't we define a document architecture language?
That allows for expressing the coarse grain structure of documents?
and unify it with software architecture / software composition?
But..

Well, connectors don't suffice. It must be composition operations or other mechanisms (such as AG) that glue the components together. We need composition languages for uniform composition.
The Elements of Composition
The Elements of Composition

Component Model

Composition Technique

Composition Language
Systems with Composition Operators

Classical Component Systems

Object-Oriented Systems

Modular Systems

Software Composition Systems

Uniform Composition Systems

EASYCOMP

Composition Operators

Composition Language

Composition Filters

Hyperslices

Aspect Separation

Architecture as Aspect

Standard Components

Objects as Run-Time Components

Modules as Compile-Time Components

Invasive Composition (COMPOST)

N-calculus

DCOM CORBA Beans/EJB

C++ Java Sather

Modula Ada-85 C++
A Component Model For Uniform Composition

Interface: what do the composition points look like?

- hooks (variation points)
- definitions

What are the values?

- fragments of data and code

A *definition language* (markup language) for definitions
Interrupt: Invasive Composition – What's This?

Invasive Composition adapts and extends components at hooks by transformation
Components have Hooks

Hooks are variation points of a box: fragments which are subject to change.

XML Variation Points
- beginning/end of tag lists
- anchors

Software Variation Points
- method entries/exits
- generic parameters
Implicit Hooks

Example Method Entry/Exit

Given by the programming language
Implicit Hooks For XML

Example List Entry/Exit

Given by the DTD or Xschema
The Composition Technique of Invasive Composition

Invasive Composition adapts and extends components at hooks by transformation

A composer is a tag transformer from unbound to bound hooks
composer: box with hooks --> box with tags
<UL>
  <LI>...</LI>
  <LI>...</LI>
</UL>

box.findHook("ListEntry").extend("<LI>... </LI>");

box.findHook("ListExit").extend("<LI>... </LI>");
m (){ 
    abc..
    cde..
}

box.findHook(".MethodEntry").extend("print("enter m");");
box.findHook(".MethodExit").extend("print("exit m");");
Invasive Composition

Invasive Composition works uniformly over code and data.
Allows to compose XML documents uniformly.
Extend operation implements what we need for document architectures.
Basic Operations on Hooks

bind (parameterize)
extend
rename
copy
But For Documents, We Need More: Definitions

A definition is an item the component defines and exposes to the external world. The programmer writes the definition in a definition language.
Definition Language 1

\cite

\cite

\cite

\cite{.*}

\bibliography{}

Definition language

Regular expressions

\cite{.\*}
Definition language LAG, together with regular expressions \cite{.*}
Definition language
Relational algebra,
started by script
Another Definition Language is XSLT

```html
<html>
  ..
  <table>
    <tr><td>Employee</td><td>Address</td></tr>
    <tr><td>Uwe Assmann</td><td>Farhagsvägen 128</td></tr>
    <tr><td>Robert Kaminski</td><td>Platensgatan 9</td></tr>
  </table>
</html>
```
Basic Operation on Definitions

\[
\text{match(<ExprInDefinitionLanguage>)} \rightarrow \text{Definitions}
\]
The Architecture of Case 1

\cite
\cite
\cite
\cite
\cite
\bibliography{ }

match("\cite{.*}"")
bibtex
extend

.match("@InBook{.*}"")

.bib file

.bbl file
The Architecture of Case 1

\cite
\cite
\cite
\cite
\bibliography{}

match("@InBook{.*}"")

match("\cite{.*}""")

extend

.bib file

.bbl file

bibtex
Advantages

We have extracted the document's architecture
latex becomes simpler
extraction is separated into the composition level
Standard language to write the compositions
no architectural language required
Documents are real components, with a composition interface
The Architecture of Case 2

\begin{deliverable}
\begin{deliverable}
\begin{deliverable}
\begin{deliverable}
\begin{deliverable}
\begin{deliverable}
\input{deliverables}
\end{deliverable}
\end{deliverable}
\end{deliverable}
\end{deliverable}
\end{deliverable}
\end{deliverable}
\end{deliverable}
\end{deliverable}

extend

LAG

deliverable table
Advantages

latex cannot extract LAG patterns, only regular ones
and cannot treat relational algebra either
We can employ many different definition (extraction, markup) languages
We can employ many different connection and composition languages
and write connectors with them
Flexible composition approach
The Architecture of Case 3

Uwe Assmann: Fårhagsvägen 128
Robert Kaminski: Platensgatan 9
Jens Gustavsson: Stora Torget 14

MySQL

XSLT

bind

<table>
Advantages

Better reuse

- scripts are removed from HTML pages
- The template can be reused in other contexts where the table expansion is not required

A lot of embedded scripts in HTML is *composition code*

- let's move it out!

Simplifying web engineering
What We Really Need: Transclusion

„hot update“
every change in a definition is immediately synchronized in all uses
Semantics is between call by name and call by value
Invented by Ted Nelson, the inventor of hypertext
What We Really Need: Transclusion

\begin{deliverable}
\begin{deliverable}
\begin{deliverable}
\begin{deliverable}
\input{deliverables}

Transclusion Operator

LAG

AG-Extractor

extend

deliverable table
Uniform Composition in the EASYCOMP Project
EASYCOMPS Goal

Provide a uniform composition technology (Level 8)
  Easy
  Powerful

EU IST FET Basic Research Project
  2.5 Mio Euro Funding
  U Karlsruhe (coordinator), U Hagen, U Linköping,
  Ecole des Mines Nantes, TU Vienna, U Twente
  HEI Mannheim, QLABS Paris, ILOG Paris
EASYCOMP Workplan

Core composition operator library UNICOMP
  COMPOST is an implementation
Composition tools for XML active components
  Component specifications
  Consistency checkers
Aspect Composition
  Event-based AOP
Case studies
Uniform component model UNICOMP

Generalized parameter model *hooks*

Uniform composition operators

**Component models for**

XML components

* XHTML, HeiTML, SVG, SMIL,...

Software components

* Java, C, ...

**Build on top of a XML refactorer and Java refactorer**
Uniformity in 3 Dimensions

Component Model

Composition Time

Language
UNICOMP Architecture

Abstract Composition

Interfaces for exchange and checking

Time specific

Language specific

UNICOMP Components

Components (Boxes)

Fragment Components

Run Time Components

Fragment Components

Java Recoder

X-HTML Recorder

SVG Recorder

JDRUMS

...
Instances of UNICOMP for Different Composition Times

UNICOMP

Fragment component model

Generic COTS component model

Run time component model

Code Fragment Components

COTS components

Run time components
Uniform Component and Composition Specification

DAML www.daml.org for
  Component properties
  Generic Semantics
  Composition validation
In the ontology environment SWEDE
Checking Active Documents Uniformly with SWEDE

Document in XML Dialect

XML Dialect Ontology

Constraint
Outlook on 2005

Uniform XML composition technology
   Components
   Composition operators and languages
   Generic semantic descriptions

Simplified construction of Documents, integrated with Software
Web Sites, Active Paper, Wearables, Traders, Agents easier
www.easycomp.org
Software Composition Workshop at ETAPS
Warsaw, Poland, April 7, 2003
Deadline Dec 20, 2002
www.ida.liu.se/~uweas/sc2003
http://i44www.ipd.info.uni-karlsruhe.de/~compost
www.ida.liu.se/~uweas/compost-distributions
Inheritance works uniformly over code and data

if regarded as invasive composition

inheritance :=

copy first super document extend with second super document
public static void main(String argv[]) {
    // Prepare the composition by allocating a composition system.
    if (JavaComposition) {
        JavaCompositionSystem cs = new JavaCompositionSystem(outputPath);
    } else if (XHTMLComposition) {
        XHTMLCompositionSystem cs = new XHTMLCompositionSystem(outputPath);
    } else { // Runtime Composition
        RuntimeCompositionSystem cs = new RuntimeCompositionSystem();
    }
    // Load base active documents
    IComponent human = cs.createComponent("Human");
    IComponent baby = cs.createComponent("Baby");

    // Use framework code
    IComponent humanBaby = inherit(cs, human, baby, "HumanBaby");
    // publish
    cs.publish(true);
}
A framework method for multiple inheritance

```java
/**
   * A framework method for multiple inheritance
   */
   public IComponent inherit(CompositionSystem cs,
                               IComponent father,
                               IComponent mother,
                               String newName) {

   IComponent newComponent = father.copy(newName);
   newComponent.extend(mother);
   return newComponent;
   }
```

Works for all scenarios!
The Composition Language of Invasive Composition

Using standard languages
  - XML itself
  - Java or others

Meta-composition possible
  - Composition of composition programs
Uniform Inheritance (for Java Runtime Components)

Inheritance works uniformly for runtime components

inheritance :=

- copy first runtime component
- extend with second runtime component
**COMPOST – The Invasive Composition System**

First system to support *invasive composition*
- Composition language
  - Java
- Library of static meta-programs

**Reified concepts**
- Boxes, Hooks, Composers

<table>
<thead>
<tr>
<th>Services</th>
<th>Composers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names &amp; Types</td>
<td>Refactorings, also for XML</td>
</tr>
<tr>
<td>Parsers + pretty printers</td>
<td>Inheritance operators</td>
</tr>
<tr>
<td></td>
<td>Connectors</td>
</tr>
<tr>
<td></td>
<td>Aspects</td>
</tr>
</tbody>
</table>
COMPOST for Everybody

0.76 is out (Uni Linköping/Uni Karlsruhe)

http://i44w3info.uni-karlsruhe.de/~compost.html
http://www.ida.liu.se/~uweas/compost-distributions

COMPOST is on Level 7, but extended to level 8 in EASYCOMP...
Everything Will Be An Active Document

Mobile Reflective Agents

Shippable (Active paper, Wearables)

Reflective (Agents)

Introspective (Traders)

Stationary (Web sites, Documents)
What Does Software Engineering Look Like Today?
The Essence of the 60s-90s: LEGO Software

Modular Systems
Object-oriented Technology and Design Patterns
Component-based Programming (CORBA, DCOM)
Architecture languages

Now: paradigm shift towards composition systems
Where Are We Today?

Component systems (CORBA, DCOM, Beans, OO-frameworks) provide a communication and standard service infrastructure components ... but nothing more

LEGO Software
Software Architecture Systems

Component

Interface

Port

Role

Connector

Component
Filter Components

Composition connectors:
- pipe, tee, file

Black-box composition

Connector-based Programming

Component-based applications

make, shell
The Essence of the Last 5 Years

Aspect-oriented Programming
View-based Programming
Weaver-Tool

Algorithm

Debugging aspect

Persistence aspect

Debugging aspect

Persistence aspect

Debugging aspect

Persistence aspect
Components

Weaver

Composition recipe

System constructed in a component- and composition-based architecture

Aspect-oriented composition

View-based composition
Entering the Era of Composition: The Essence of the 2000s

Composition Systems with Composition Languages

Uniform Composition Systems

Everything will be an active component/document
Software and documents will be composed uniformly with composition operators and languages
Components

Composition Language

Invasive composition

Uniform composition

System constructed in a composition-based architecture

Composition recipe
Classical Component Systems

CORBA
DCOM
Beans/EJB

Architecture Systems

Composition Systems
Composition Language
Meta-object composition Invasive Composition

Systems with Composition Operators
Composition Operators
Composition Filters
Hyperslices

Aspect Systems
Aspect Separation
Aspect/J

Architecture Systems
Architecture as Aspect
Darwin
Aesop

Classical Component Systems
Standard Components
DCOM CORBA Beans/EJB

Object-Oriented Systems
Objects as Run-Time Components
C++
Java
Sather

Modular Systems
Modules as Compile-Time Components
Modula
Ada-85
C..
Uniform Composition Should Be Easy and Powerful

Uniform composition of software and data with XML as basis

Easy composition technology
   Architecture
   Reuse compositions from libraries and languages
   Aspect orientation
   Extensibility and integratability

Powerful system construction
   Semantics for compositions in XML
   Reflection, introspection
Declared Hooks are declared by the box writer as variables in the hook’s tags.
Declaration of Hooks

Markup Tags
Standardized Names (Hungarian Notation)
Comment Tags

<superclasshook> X </superclasshook>
class Set extends genericXSuperClass { }
class Set /* @superClass */