Easy and Powerful Composition of XML-based Active Documents

Dr. U. Aßmann
PELAB, Linköpings Universitet
Sweden
Talk at European Media Lab
Heidelberg
Feb. 2001
Overview

- Introduction
  - XML and Active Documents
- Future Applications of Active Documents
  - Web Objects, Active Paper and Wearables
  - Traders, Agents
- What does Software Engineering look like today?
- A Method for Uniform Composition: Invasive Composition
- EASYCOMP
Entering the Age of Active Documents

- XML is the ASCII of the 21st century
  - It will represent software and data uniformly
  - Tags will have active semantics <if> <then> <else>
- XML documents will be active (*active documents, active components*)
- Active XML-based documents will penetrate
  - the future Web,
  - future documents, and
  - future software
Everything Will Be An Active Document

- Orfali/Harkey: Instant CORBA
- Documents
  - Hypertext
  - Multimedia, e.g., Courses for Online Learning
  - Web sites, server objects
  - Simulated objects
- Traders
- Wearables/Active Paper
- Agents
Future Applications of Active Documents
Overview of Active Documents

Mobile Reflective Agents

Shippable (Active paper, Wearables)

Reflective (Agents)

Introspective (Traders)

Stationary (Web sites, Documents)
Why is a web site different from a web page?
- Only a bit more complex

Why is a desktop different from a web site?
- Representation

What the difference of a web site and
- a CAD model
- a roller bearing
- a simulated robot?

What's the difference between a browser and an active web site?
- Client vs server
Shippable Active Documents (Wearables, Active Paper)

- A shippable active document can be shipped easily to other physical places
  - web page, web site, desktop, browser
- Problem: Shippable Active Documents can be copied and must be integrated again
- Examples
  - Wearable computers require shippable active documents
  - Active paper require shippable active documents
Mr. Spielberg has married in May, Ms Anita Söderberg, from Linköping, Sweden

Now, it's June and he tries to reuse his E-paper tax form from last year (in 2010, the tax authorities don't accept paper declarations)

His wife Anita also has an E-paper tax form, but from Sweden

How to both combine their papers?
Wearables Rely on Composition Technology

- Prof. Assmann works for two months in Karlsruhe
- He carries his active desktop as wrist watch
- In Karlsruhe, many sub-documents change
  - also those that are edited in parallel in Linköping by his colleagues
- When he comes home to Linköping, he wants to integrates wrist watch with the others
Active Documents Introspecting Others (Traders)

Traders deal with components or services

Traders are active documents which reason about other active documents

Mediator Pattern

Export Functionality

Import Functionality

1

2

3

Client

Service

Trader
Introspective Active Documents Rely on Semantic Properties

- Introspective Active Documents have a **meta-level architecture**, relying on semantic descriptions
  - properties, protocol specifications
- Traders:
  - Services described by semantic properties
  - Clients specify desired properties
  - Traders match desired and offered properties (e.g., by DOOD technology)
  - Traders delegate to other traders if they do not find matches
- Similar to book management in libraries
Reflective Active Documents (Agents)

- Agents are autonomous active documents which reflect and reason about themselves and their environment
  - have a meta-level architecture with reflective features
- **Splitt-off self agents** are agents that represent a human being and represent him for tasks
  - Buying things automatically
  - Finding interesting people to meet
- www.agentlink.org
Overview of Active Documents

- Mobile Reflective Agents
- Shippable (Active paper, Wearables)
- Reflective (Agents)
- Introspectivte (Traders)
- Stationary (Web sites, Documents)
How Do We Build Such Systems?

- These systems will be complex
  - They will migrate
  - They will contact other unknown documents
  - They will negotiate

- Reuse of components of active documents would be nice...

- Semantics is an issue...

- Security is an issue...

- Activeness is an issue...
Active Documents are Software

Active documents are Software!

Apply Software Engineering Techniques!

But how does Software Engineering look like today?
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, Beans)
- 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

Port

Interface

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
Composition Systems

Components with Composition Operators

Composition Operators

Composition Language

Invasive Composition

\( \pi \)-calculus

\( \lambda N \)-calculus

Hyperslices

SOP

Aspect Systems

Aspect Separation

Aspect/J

Architecture Systems

Architecture as Aspect

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++
Structure

Interfaces

Light Plan

Pipe Plan

Integrated House
Debugging aspect

Persistence aspect

Algorithm

Weaver-Tool

Debugging aspect

Persistence aspect

Debugging aspect
Components

Weaver

Composition recipe

Aspect-oriented composition

View-based composition

System constructed in a component- and composition-based architecture
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
Uniform Composition Systems

Composition Systems

Systems with Composition Operators
Composition Operators

Aspect Systems
Composition Language
Aspect Separation

Architecture Systems
Architecture as Aspect

Classical Component Systems
Standard Components

Object-Oriented Systems
Objects as Run-Time Components

Modular Systems
Modules as Compile-Time Components
Requirements for Uniform Composition: Easy and Powerful

- **Uniform** composition of software and data with XML as basis
  - security
- **Easy** system construction by simple composition technology
  - Architecture
  - Reuse compositions from libraries and languages
  - Aspect orientation
  - Extensibility and integratability
- **Powerful** system construction
  - Semantics for compositions in XML
  - Reflection, introspection
Invasive Composition - a Uniform Composition Technique
Composition

Component Model
Composition Technique
Composition Language
Invasive Composition

Invasive Composition adapts and extends components at hooks by transformation
1. The Component Model of Invasive Composition

- The basic element is a **box**, a set of tag elements
- May be the representation
  - an XML page
  - a XML site
  - a software component
    - a class, a package, a method
  - an aspect
  - a meta description
  - a composition program
Boxes have Hooks

Hooks are arbitrary elements or spots of the representation of a box which are subject to change

- beginning/end of tag lists
- anchors
- method entries/exits
- generic parameters
Implicit Hooks

Example Method Entry/Exit

```
Example Method Entry/Exit

Given by the programming language, the DTD or Xschema
```
Declared Hooks are declared by the box writer as variables in the hook’s tags.
Declaration of Hooks

- Markup Tags
- Language Extensions (keywords..)
- Standardized Names
- Comment Tags

<superclasshook> X </superclasshook>

class Set extends genericXSuperClass {
}

class Set /* @superClass */
2. 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>
  <LI>... </LI>
</UL>

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

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

box.findHook(".Entry").extend("print("enter m");");

box.findHook(".Exit").extend("print("exit m");");
Invasive Composition as Hook Transformations

- Invasive Composition works uniformly over code and data
- Allows to compose active documents uniformly (stationary, shippable, introspective, reflective)
3 The Composition Language of Invasive Composition

- Using standard languages (Java) and XML itself
- Meta-composition possible
The **COMPO$$ition$$ SyTe$$m**

- COMPOST is the first system to support *invasive composition* for Java
- Library of meta-programs
- Composition language Java
- Reifies concepts
  - *Boxes, Hooks, Composers*
- and many other things
  - Names & Types
  - Design Patterns
  - Refactorings
  - Inheritance operators
COMPOST for Everybody

- 0.63 is out (Uni Karlsruhe/Uni Linköping)
  - parsers + pretty printers, name analysis service
  - boxes and hooks
  - papers

http://i44w3info.uni-karlsruhe.de/~compost.html

- COMPOST is still on Level 7, but extended in EASYCOMP...
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, ILOG
EASYCOMP Workplan

- Core composition operator library
- Composition tools for XML-based active components
  - Component specifications
  - Consistency checkers
- Aspect Composition
  - Languages, graph rewriting for automatized weaving
  - Event-based AOP
- Evaluation case studies in
  - Multimedia learning
  - E-commerce
  - Dynamic reconfiguration
Component properties
- DAML www.daml.org and OIL www.ontobroker.org
- Feature logic, F-Logic
- DOOD technology: deductive object-oriented databases

Composition validation
- Protocol checking methods

Security?
- Virus detection
- Sandbox behavior
- Compositional Proof-carrying codes
Outlook on 2005

- Uniform XML composition technology
  - Tooling
  - Components
  - Semantic descriptions
- Simplified construction of Active Documents:
  - Reuse
  - Extensibility, Aspect separation
  - Composition language
  - Reflection
- Web Sites, Active Paper, Wearables, Traders, Agents easier
THE END

- www.easycomp.org
- www.ida.liu.se/~uweas
- i44w3.ipd.info.uni-karlsruhe.de/~compost