Using UML and XMI for Generating Adaptive Navigation Sequences in Web-Based Systems

Peter Dolog and Wolfgang Nejdl

Presenter: Jens Gößner

{dolog, nejdl}@learninglab.de

http://www.learninglab.de/~dolog

Learning Lab Lower Saxony (L3S),
University of Hannover
Hannover

UML 2003 Conference

Outline

Motivation

Related Work

The Process of Adaptive Navigation Modelling

Prototype Implementation

Conclusions and Further Work

Introduction & Motivation

Introduction to Adaptation & Personalization

Web applications enable a large number of users to access information.

This is
- a benefit

but
- it requires to handle new conditions.

One size fits all approach is not suitable !
- because, applications should reflect different requirements of users with different background

Personalization and adaptation appears in many application domains (e.g., CRM-systems).

These applications adapt themselves to user features.

System models can help the developers to overcome the increasing complexity caused by the need to maintain different variants of information.

Motivation

- Browsing represents a path through the hypertext graph
- Different users – different paths
- Following user‘s path – following interaction
- Different interactions demand different navigation possibilities

Question:

- How to model the dynamics of adaptable navigation ?

Related Work

Existing Methodologies or Modelling Languages for Navigation Modelling:

- WebML – Modelling approach for data intensive web applications
- UWE – UML extension for web engineering
- OOHDM – object oriented method for hypermedia based web applications
- OO-H

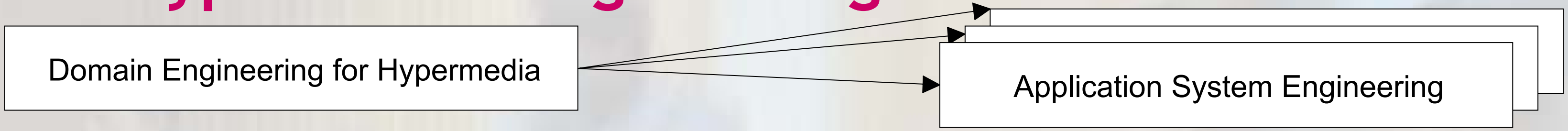
Generation Methods

- From WCML
- From WebML

But:

→ **Navigation modelling based on user interaction with focus on behaviour and user observation is missing in these methods !!**

Overview of DEAHE - Domain Engineering Approach to Hypermedia Engineering



Domain Engineering for Hypermedia

Application System Engineering

Define Conceptual Domain Model
  - Real World
  - Representation

Define Feature Model
  - Real World
  - Representation

Define Domain Architecture
  - Information
  - Navigation
  - Presentation
  - User
  - Environment
  - Adaptation

Implement Domain
  - Reusable Components
  - Tools
  - Reuse Infrastructure
  - Domain Specific Languages

Analyse Requirements of Stakeholders

Perform Selection of Reusable Components

Define and Scope Requirements not Covered by Reusable Components

Do Design of Custom Components

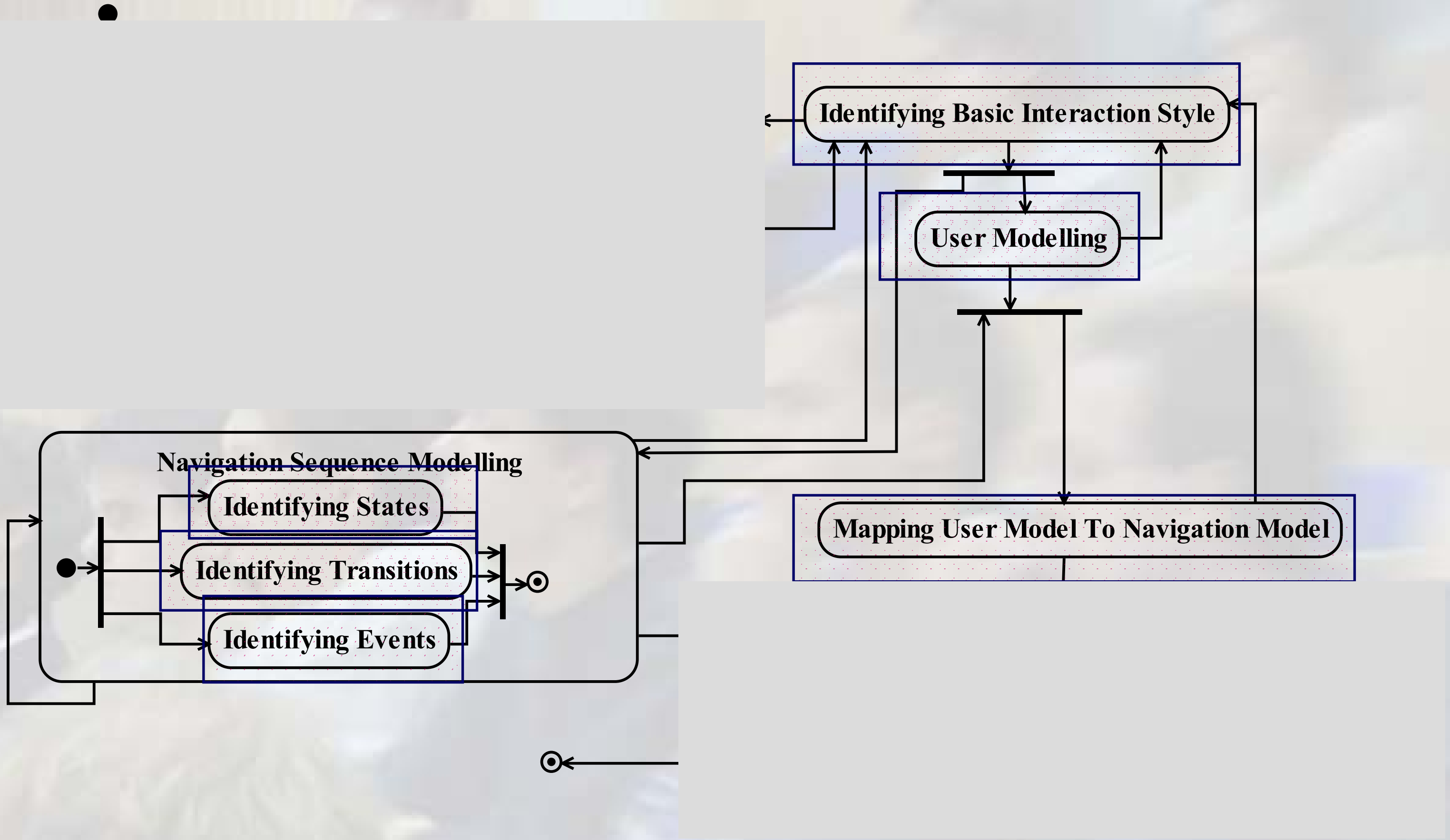
Specialise and Integrate Resusable and Custom Components

Consider Variability and Reuse at Reuse Time and at Bind Time

Peter Dolog and Mária Bieliková: Towards Variability Modelling for Reuse in Hypermedia Engineering. In Proc. of 6th East-European Conference on Advances in Databases and Information Systems. September 2002, Slovakia, LNCS 2435.

The Process of Adaptive Navigation Modelling

The process of the approach



Identifying Basic Interaction Style

User Modelling

Navigation Sequence Modelling

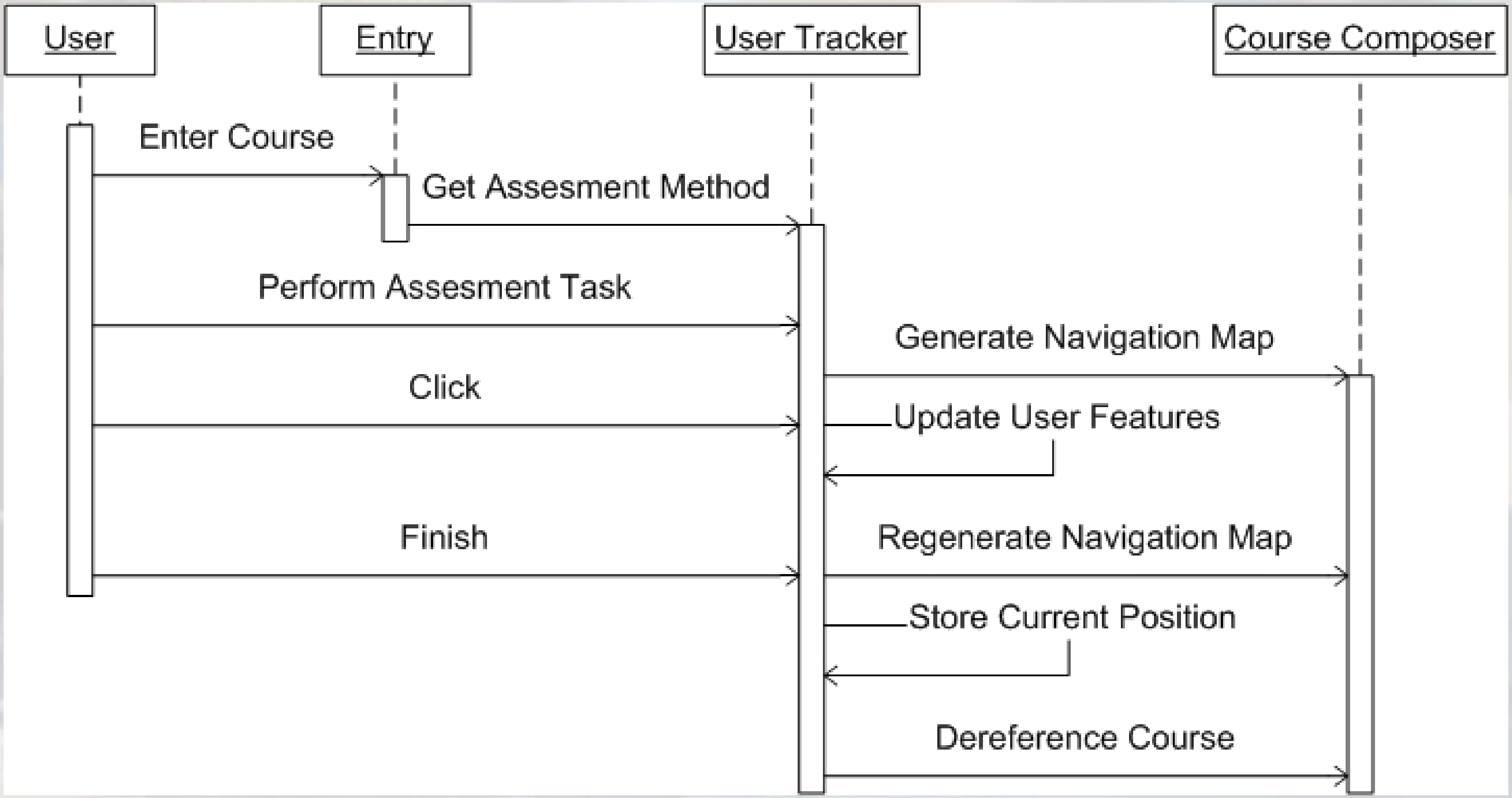
Identifying States

Identifying Transitions

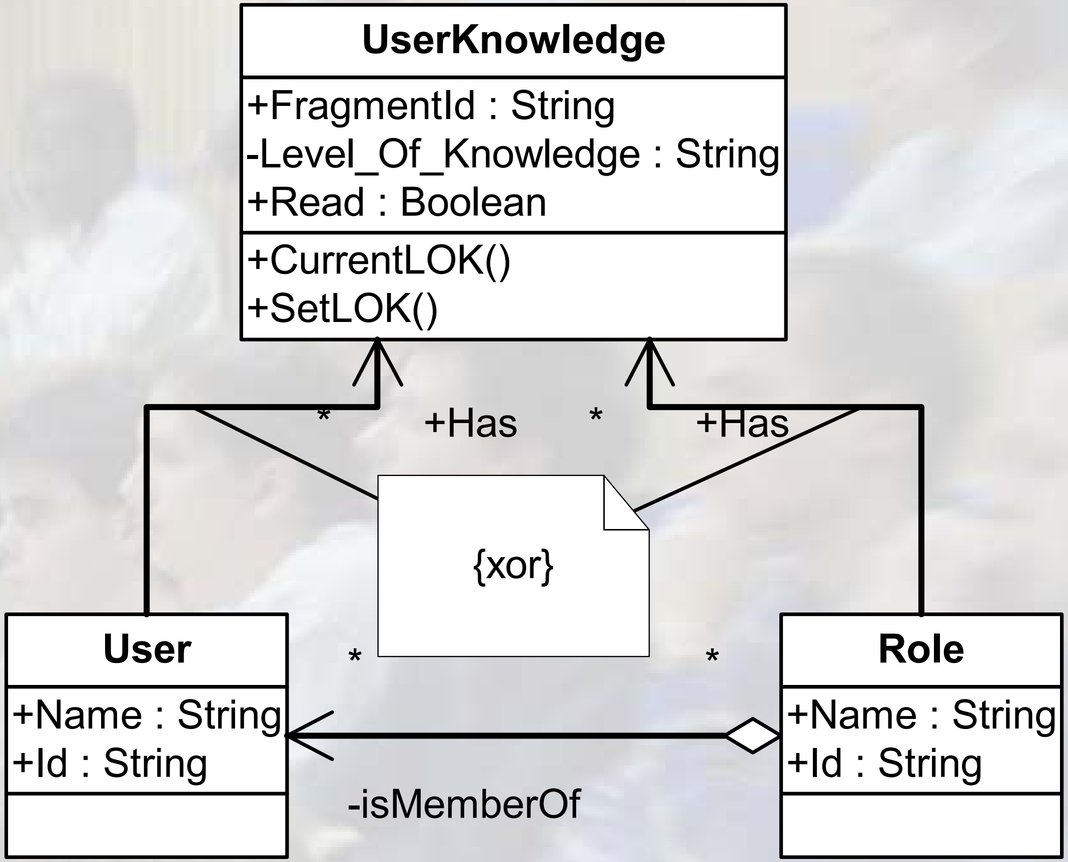
Identifying Events

Mapping User Model To Navigation Model

Basic Interaction Scheme



User Model



UserKnowledge

+FragmentId : String
-Level_Of_Knowledge : String
+Read : Boolean

+CurrentLOK()
+SetLOK()

*

+Has

*

+Has

{xor}

*

*

User

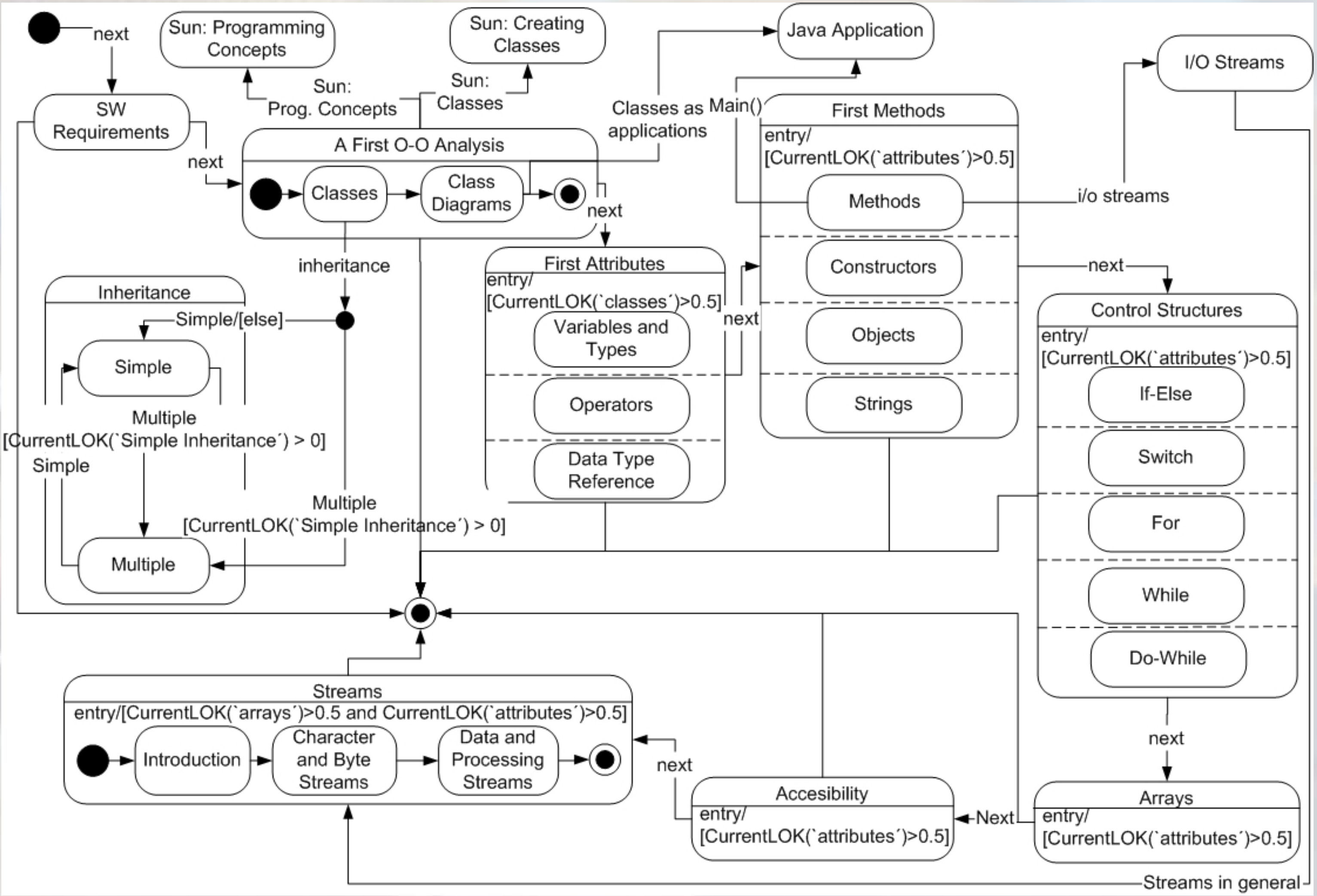
+Name : String
+Id : String

Role

+Name : String
+Id : String

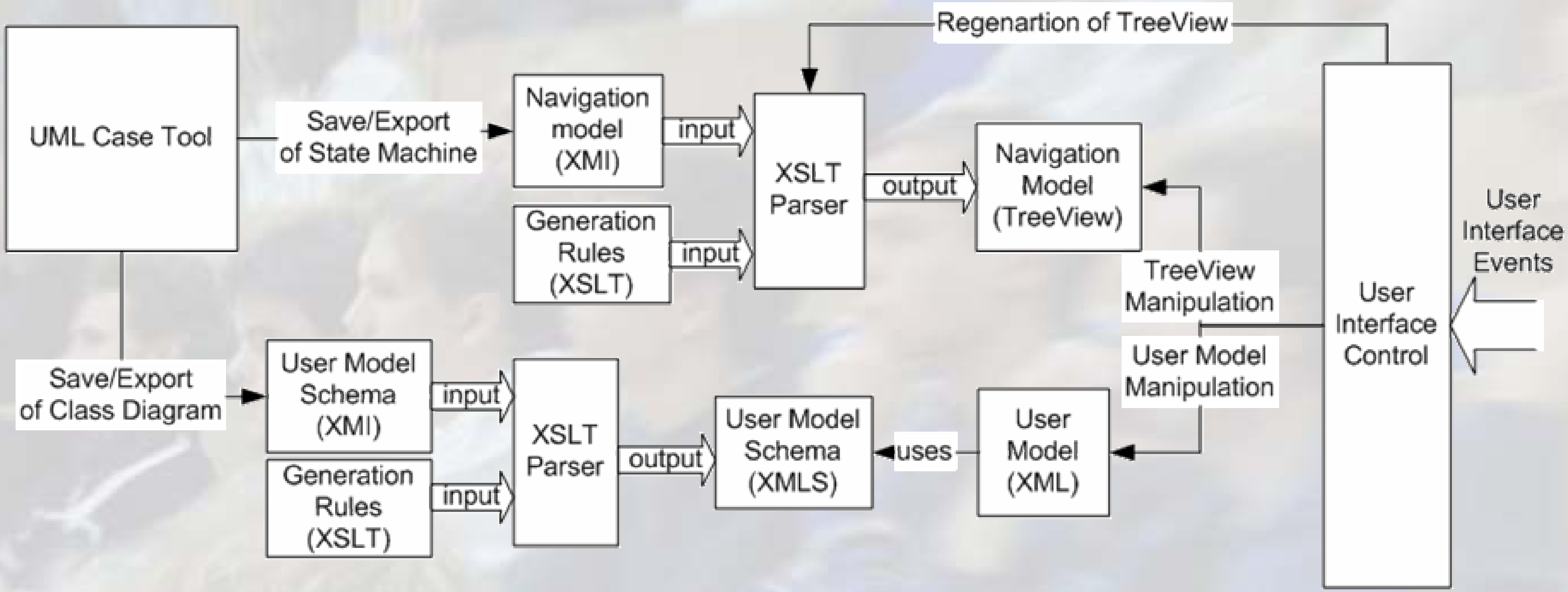
-isMemberOf

Navigation Model of an Adaptive Java e-Lecture


next
Sun: Programming Concepts
Sun: Creating Classes
Java Application
I/O Streams
SW Requirements
Sun: Prog. Concepts
Sun: Classes
Classes as Main() applications
First Methods
entry/ [CurrentLOK(`attributes´)>0.5]
A First O-O Analysis
next
Classes
Class Diagrams
next
Methods
i/o streams
inheritance
First Attributes
entry/ [CurrentLOK(`classes´)>0.5]
Constructors
next
Inheritance
Simple/[else]
Variables and Types
next
Objects
Control Structures
entry/ [CurrentLOK(`attributes´)>0.5]
Simple
Operators
Strings
If-Else
Multiple [CurrentLOK(`Simple Inheritance´) > 0]
Simple
Data Type Reference
Switch
Multiple [CurrentLOK(`Simple Inheritance´) > 0]
For
Multiple
While
Do-While
Streams
entry/[CurrentLOK(`arrays´)>0.5 and CurrentLOK(`attributes´)>0.5]
Introduction
Character and Byte Streams
Data and Processing Streams
next
next
Accesibility
entry/ [CurrentLOK(`attributes´)>0.5]
Next
Arrays
entry/ [CurrentLOK(`attributes´)>0.5]
Streams in general

Prototype Implementation

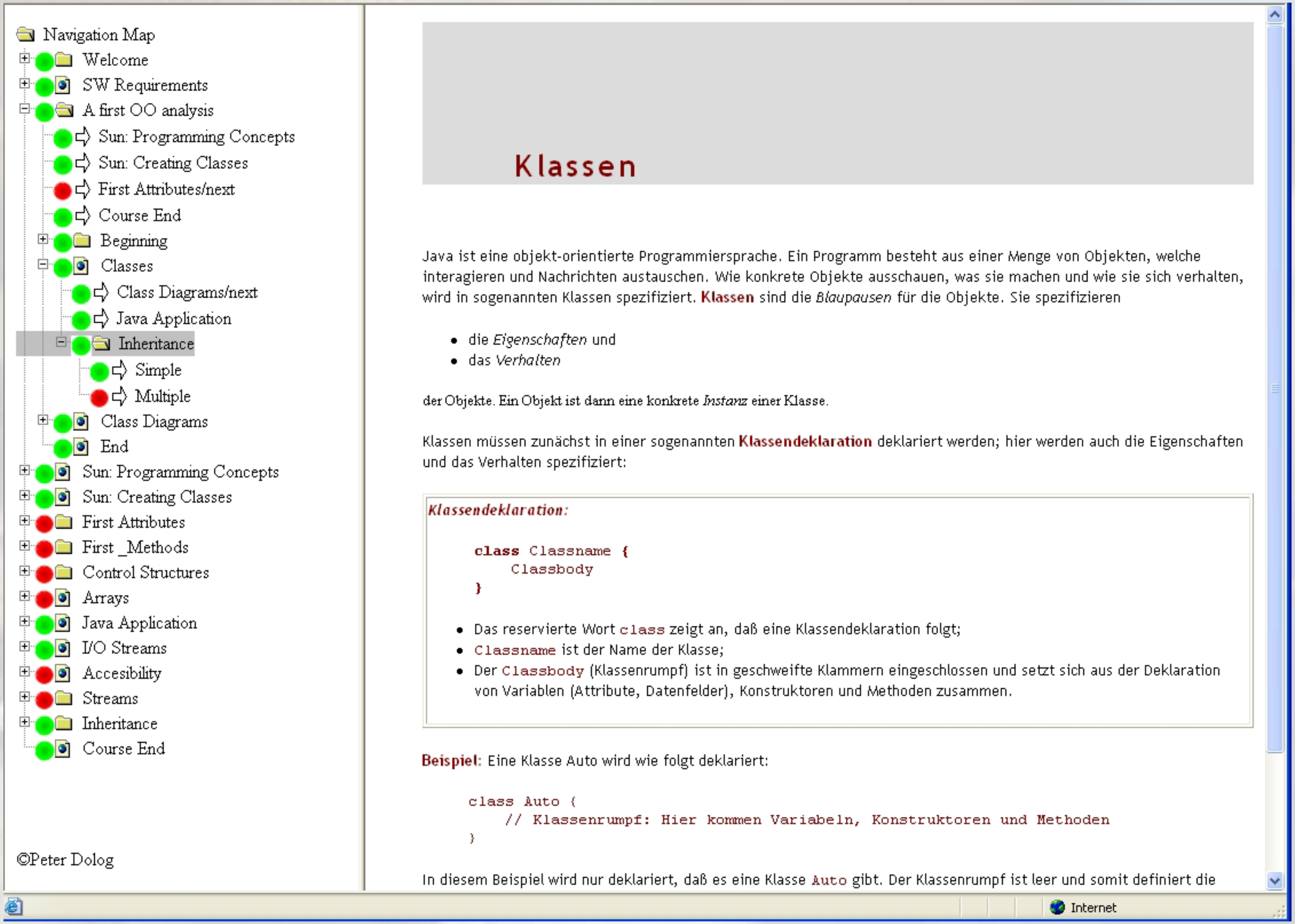
Implementation Prototype - From the State Diagram to the Navigation Map



Example of XSLT - Part of Composite State Template

```
...
 <xsl:if test="$fldname!='state_machine_top'">
    <xsl:choose>
      <xsl:when test="$region='True'">
        <xsl:choose>
         <xsl:when test="$level=0">
            <xsl:value-of select="concat('aux', $level+1)"/> = insFld(foldersTree,
    gFldConc("<xsl:value-of select="$fldname"/>", "<xsl:apply-templates
    select="UML:ModelElement.taggedValue"/>", "green", "<xsl:value-of
    select="$stateid"/>"))
         </xsl:when>
         <xsl:otherwise>
            <xsl:value-of select="concat('aux', $level+1)"/> = insFld(<xsl:value-of
    select="concat('aux', $level)"/>, gFldConc("<xsl:value-of
    select="$fldname"/>", "<xsl:apply-templates
    select="UML:ModelElement.taggedValue"/>", "green", "<xsl:value-of
    select="$stateid"/>"))
         </xsl:otherwise>
        </xsl:choose>
    </xsl:when>
...
```

Visualization of the Navigation Map



Navigation Map
- 🟢 Welcome
- 🟢 SW Requirements
- 🟢 A first OO analysis
  - 🟢 Sun: Programming Concepts
  - 🟢 Sun: Creating Classes
  - 🔴 First Attributes/next
  - 🟢 Course End
  - 🟢 Beginning
  - 🟢 Classes
    - 🟢 Class Diagrams/next
    - 🟢 Java Application
    - 🟢 Inheritance
      - 🟢 Simple
      - 🔴 Multiple
  - 🟢 Class Diagrams
  - 🟢 End
- 🟢 Sun: Programming Concepts
- 🟢 Sun: Creating Classes
- 🔴 First Attributes
- 🔴 First _Methods
- 🔴 Control Structures
- 🔴 Arrays
- 🟢 Java Application
- 🟢 I/O Streams
- 🔴 Accesibility
- 🔴 Streams
- 🟢 Inheritance
- 🟢 Course End

©Peter Dolog

Klassen

Java ist eine objekt-orientierte Programmiersprache. Ein Programm besteht aus einer Menge von Objekten, welche interagieren und Nachrichten austauschen. Wie konkrete Objekte ausschauen, was sie machen und wie sie sich verhalten, wird in sogenannten Klassen spezifiziert. **Klassen** sind die *Blaupausen* für die Objekte. Sie spezifizieren

- die *Eigenschaften* und
- das *Verhalten*

der Objekte. Ein Objekt ist dann eine konkrete *Instanz* einer Klasse.

Klassen müssen zunächst in einer sogenannten **Klassendeklaration** deklariert werden; hier werden auch die Eigenschaften und das Verhalten spezifiziert:

Klassendeklaration:

```
class Classname {
    Classbody
}
```

- Das reservierte Wort `class` zeigt an, daß eine Klassendeklaration folgt;
- `Classname` ist der Name der Klasse;
- Der `Classbody` (Klassenrumpf) ist in geschweifte Klammern eingeschlossen und setzt sich aus der Deklaration von Variablen (Attribute, Datenfelder), Konstruktoren und Methoden zusammen.

Beispiel: Eine Klasse Auto wird wie folgt deklariert:

```
class Auto {
    // Klassenrumpf: Hier kommen Variabeln, Konstruktoren und Methoden
}
```

In diesem Beispiel wird nur deklariert, daß es eine Klasse `Auto` gibt. Der Klassenrumpf ist leer und somit definiert die

Conclusions and Further Work

We developed:

- A state diagram based method for navigation modelling
- Generator from the state diagrams to navigation tree view
- System based on such generator

We will further investigate

- Further parametrisation of generator
- Possibilities to employ view mechanism for retrieving appropriate parts of the domain model
- Possibilities to generate state diagrams from such views

Thank you for your attention!!!

References on DEAHE:

Peter Dolog and Mária Bieliková: Hypermedia Systems Modelling Framework. Computing and Infromatics. Vol. 21, No. 3, 221-239, 2002.

Peter Dolog and Mária Bieliková: Towards Variability Modelling for Reuse in Hypermedia Engineering. In Yannis Manolopoulos and Pavol Navrat editors, Proc. of 6th East-European Conference on Advances in Databases and Information Systems. September 2002, Bratislava, Slovakia. LNCS 2435. Pages 388-401.

Peter Dolog and Mária Bieliková: Navigation Modelling in Adaptive Hypermedia. In Paul De Bra, Peter Brusilovsky and Ricardo Conejo editors, Proc. of 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems. May 2002, Malaga, Spain. LNCS 2347. Pages 586-592.

Peter Dolog and Wolfgang Nejdl: Challenges and Benefits of the Semantic Web for User Modelling. In Proc. of AH2003 workshop, 12th WWW Conference, Budapest, May 2003.