

# **Creating Adaptive Applications with AHA! AHA! 3.0 Tutorial**

Natalia Stash

Paul De Bra

ABIS 2004

October 5, 2004

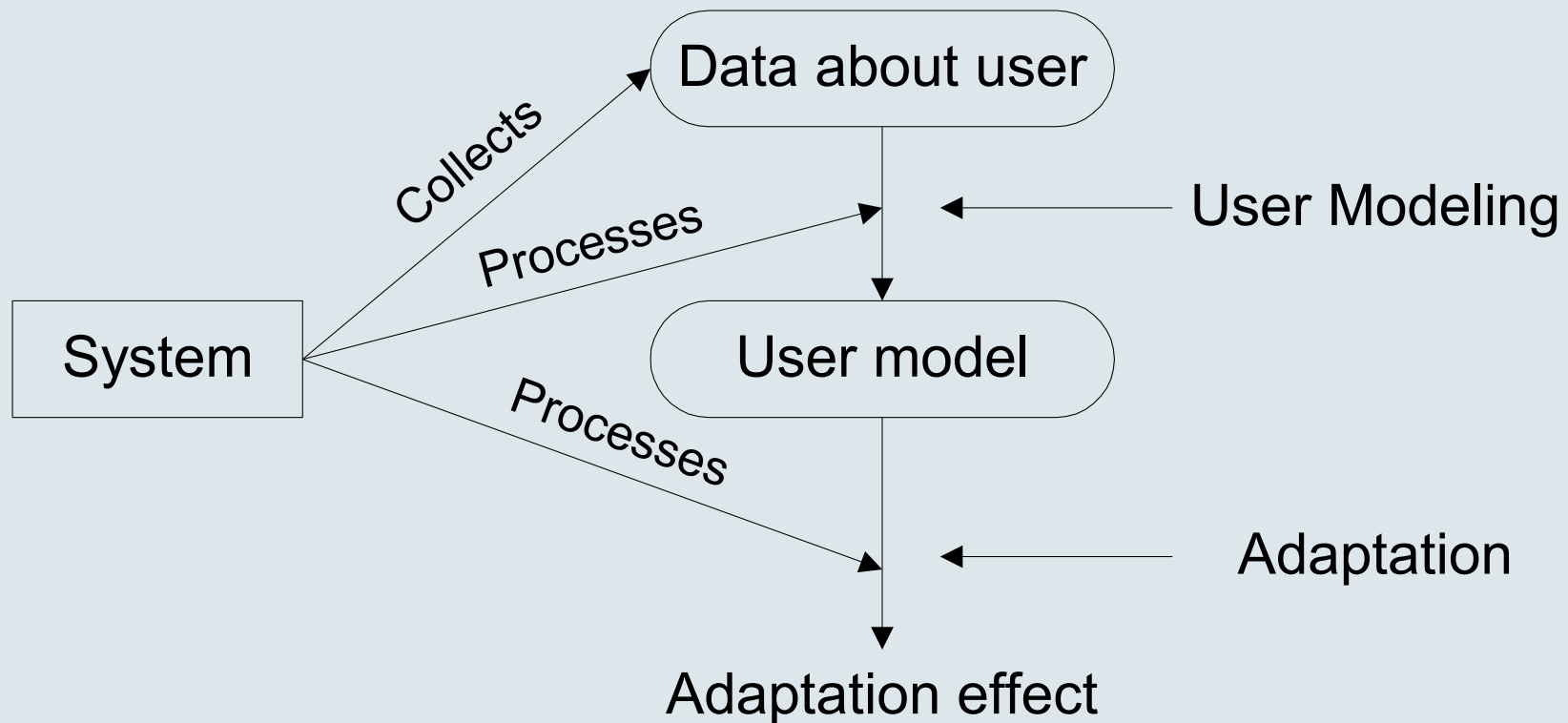
# Why Adaptive Hypermedia (AH)?



*Opportunities with adaptive hypermedia:*

- guide users towards *relevant* information
- at the same do not restrict users
- change the presentation so that it fits the user

# Classic loop “user modeling-adaptation” in Adaptive Systems



# What Do We Adapt in AH?

- Adaptive presentation:
  - adapting the *information*
  - adapting the *presentation* of that information
  - selecting the *media* and *media-related* factors such as image or video quality and size
- Adaptive navigation:
  - adapting the *link anchors* that are shown
  - adapting the *link destinations*
  - giving “overview” for *navigation support* and for *orientation support*

# Content adaptation types

- *Additional* (or prerequisite or comparative) *explanations*: Under a given set of circumstances some additional content is presented
- *Explanation variants*: Different versions of an explanation exist and are selected depending on the user
- *Sorting*: The most relevant information for a user is presented first

# Adaptive Navigation Support

- Direct guidance
- Adaptive link generation
- Adaptive link annotation
- Adaptive link hiding
  - link hiding
  - link disabling
  - link removal
- Map adaptation

## Example from 2L690

1. Before reading about history of hypermedia the URL page shows:

- ...

In Xanadu (a fully distributed hypertext system, developed by Ted Nelson at Brown University, from 1965 on) there was only one protocol, so that part could be missing.

...

2. Before reading about Xanadu the URL page shows:

- ...

In [Xanadu](#) (a fully distributed hypertext system, developed by Ted Nelson at Brown University, from 1965 on) there was only one protocol, so that part could be missing.

...

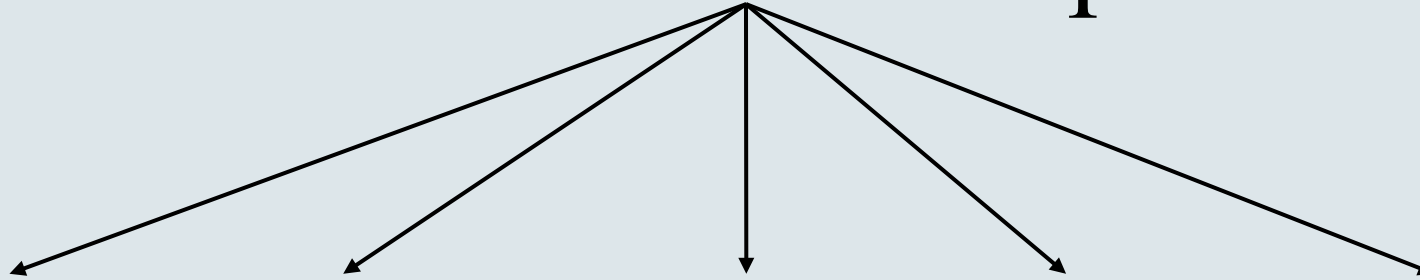
3. After reading about Xanadu this becomes:

- ...

In [Xanadu](#) there was only one protocol, so that part could be missing.

... / department of mathematics and computer science

# What Can We Adapt To?



knowledge



goals, tasks  
or interest



preferences

background

experience

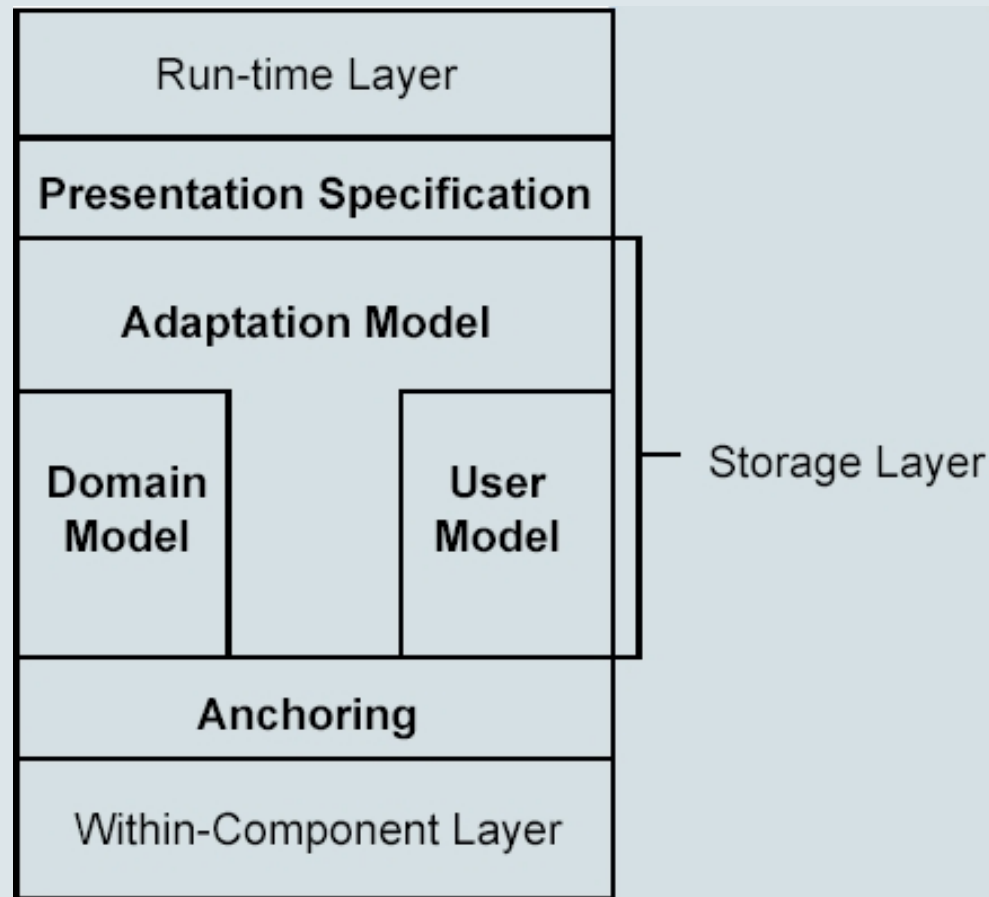
context

environment



# AHAM:

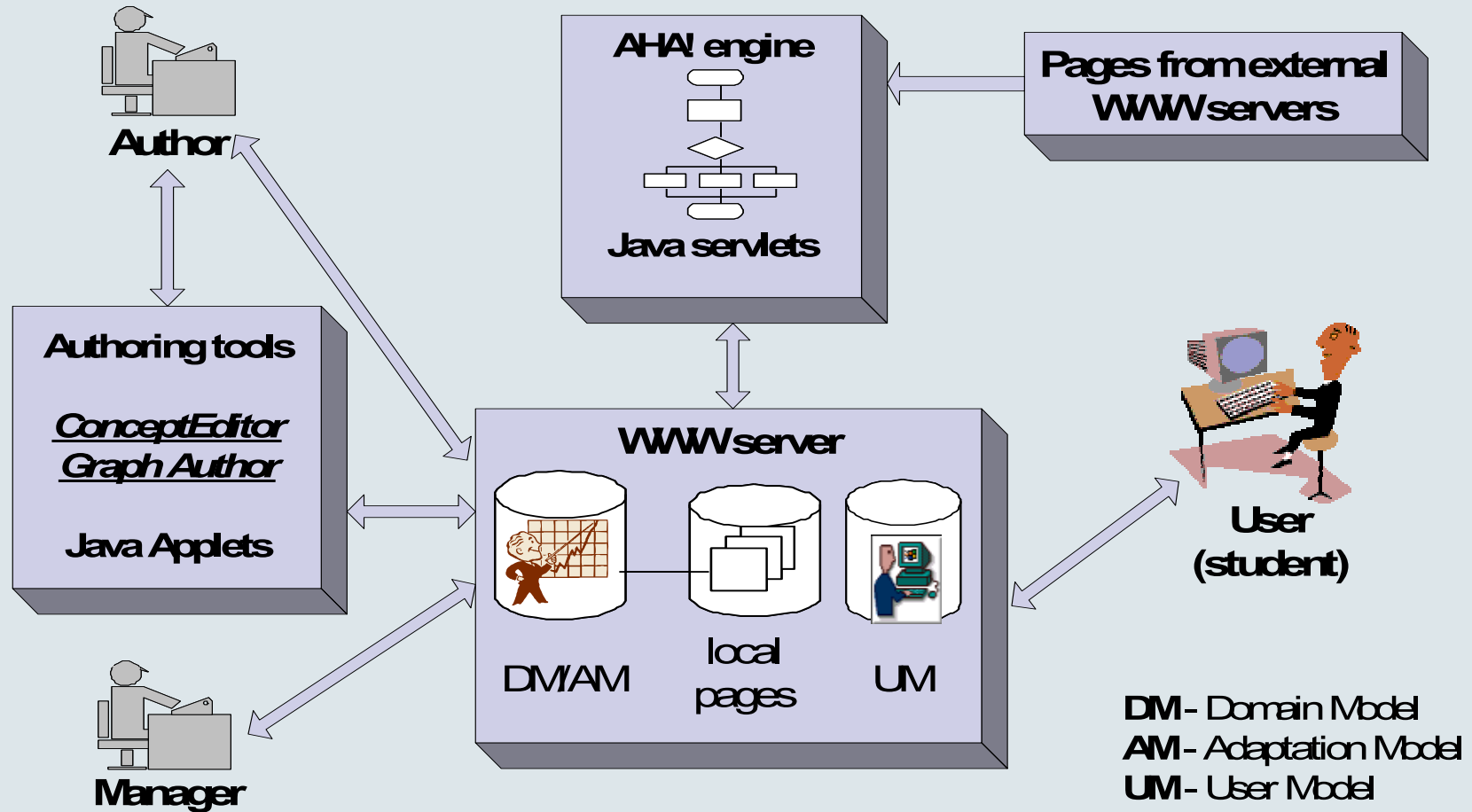
## Adaptive Hypermedia Application Model



# AHA! Adaptive Hypermedia Architecture

- Main characteristics:
  - adaptive web server extension
  - conditional inclusion of fragments
  - adaptive link hiding or annotation
  - adaptation in local and remote pages
  - pages in html or xml format
  - flexible user model (arbitrary concepts and attributes)
  - event-condition-action rules
  - graphical authoring tool for concept relationships
  - forms and multiple-choice tests

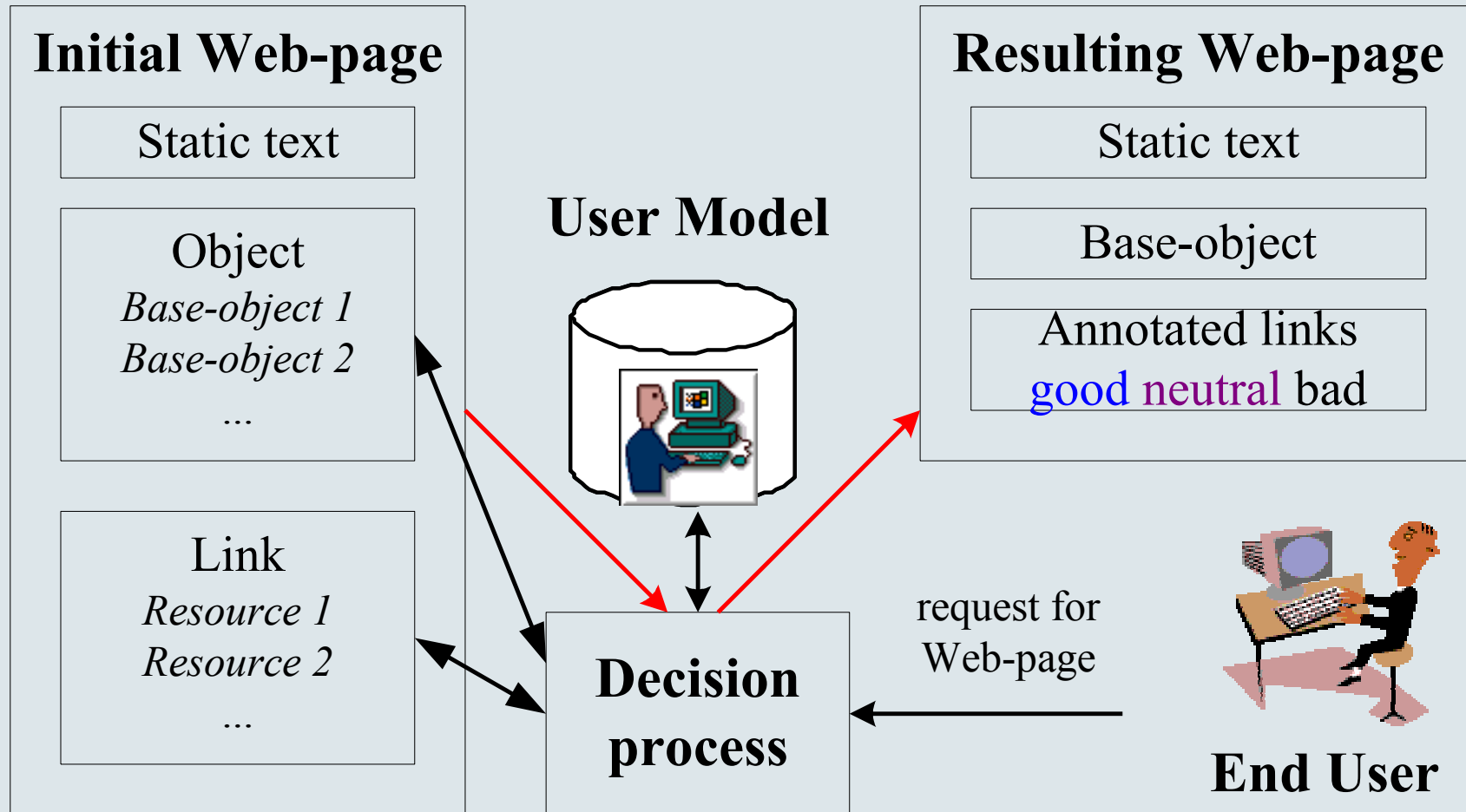
# AHA! Architecture



# User model

1. Domain concepts: attribute-value pair  
(*concept + knowledge value, interest value...*)
2. Concept “personal”: user-related information  
(*name, login-id, password...*)

# AHA! Adaptive Functionality



# Conditional Inclusion of Objects

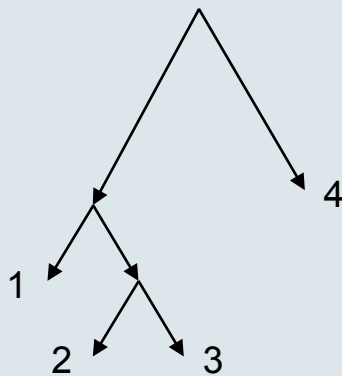
- When AHA! retrieves a page it creates a parse stream for it
- AHA! includes objects when an `<object>` tag is encountered in parse stream:
  1. The adaptation rules of the object concept are executed
  2. A resource to include is selected
  3. The resource is inserted into the parse stream
- The included data may contain `<object>` tags itself
  - as a result objects may include other objects
  - there is a danger of infinite recursive object inclusion

# Creating complex object structure

## Inclusion of objects in objects

- Depth first order processing
- Danger for Self inclusion

Depth-first proces order



Recursive inclusion of objects



# Object Inclusion in DM/AM

- Here is an example of part of a DM/AM for object inclusion:

```
<concept>
  <name>tutorial.object1</name>
  ...
  <attribute name="showability" type="int" isPersistent="true" isSystem="true">
    <casegroup>
      <defaultfragment>file:/tutorial/xml/empty.xhtml</defaultfragment>
      <casevalue><value>1<value>
        <returnfragment> file:/tutorial/xml/frag1.xhtml</returnfragment>
      </casevalue>
      <casevalue><value>2<value>
        <returnfragment> file:/tutorial/xml/frag2.xhtml</returnfragment>
      </casevalue>
    </casegroup>
  </attribute>
</concept>
```



# Adaptive Link Destinations

- Here is an example of part of a DM/AM for choice of link destination:

```
<concept>
  <name>tutorial.conceptname</name>
  ...
  <attribute name="showability" type="int" isPersistent="true" isSystem="true">
  <casegroup>
    <defaultfragment>file:/tutorial/xml/page1.xhtml</defaultfragment>
    <casevalue><value>1<value>
    <returnfragment> file:/tutorial/xml/page2.xhtml</returnfragment>
    </casevalue>
    <casevalue><value>2<value>
    <returnfragment> file:/tutorial/xml/page3.xhtml</returnfragment>
    </casevalue>
  </casegroup>
  </attribute>
</concept>
```

# Adaptive Link Hiding and Annotation

- AHA! recognizes two classes of links: “conditional” and “unconditional:  
`<a href=“page.xhtml” class=“conditional”>link anchor</a>`
- The engine translates links to *good*, *neutral* and *bad*:  
`<a href=“page.xhtml” class=“good”>link anchor</a>`
- AHA! inserts a stylesheet in the (x)html page to define the link colors:  

```
<style type="text/css">
a.Good:link { text-decoration: none; color: #0000ff }
a.Good:visited { text-decoration: none; color: #0000ff }
a.Neutral:link { text-decoration: none; color: #7c007c }
a.Neutral:visited { text-decoration: none; color: #7c007c }
...
</style>
```

## XHTML+AHA page format

- This format is inherited from AHA! 2.0. It has embedded conditional fragments:

```
<!DOCTYPE html SYSTEM "aha/AHAStandard/xhtml-ahaext-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <object data="header.xml" type="text/xml" />
  ...
  <a href="page.xhtml" class="conditional">link anchor</a>
  <a href="conceptname" class="conditional">link destination</a>
  <if expr="tutorial.intro.knowledge>50">
    <block>here something for knowledgeable users</block>
    <block>here something for beginners</block>
  </if>
  ...
  <object data="header.xml" type="text/xml" />
</html>
```

# AHA! and Standard XHTML

- AHA! 3.0 adds support for standard XHTML, with conditional object inclusion.

```
<!DOCTYPE html SYSTEM "aha/AHAStandard/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Example of AHA! with XHTML</title>
  </head>
  <body>
    ...
    <a href="page.xhtml" class="conditional">link anchor</a>
    <a href="conceptname" class="conditional">link destination</a>
    <object name="tutorial.object1" type="aha/text" />
    ...
  </body>
</html>
```

# Setting up an adaptive site with AHA!

- To start using AHA! you should perform the following steps:
  1. Download the Tomcat webserver and start at <http://localhost:8080/admin>
  2. Install AHA! 3.0. We assume you use directory `c:/aha` on a Windows machine (but any directory in Windows or Unix should work)
  3. Start Tomcat (with disconnected network) and create a new context for AHA!
  4. Perform automatic configuration at <http://localhost:8080/aha/Config>
  5. Restart the server and go to the configuration page again to create authors and applications

# Configuring AHA!

- The configuration tool lets you create and edit manager properties, authors and applications

## AhaConfig

The AHA configurator

[Configure DataBase](#)

[Manager Configuration](#)

[Authors](#)

[Convert concept list from internal format to XML file](#)

[Convert concept list from XML file to internal format](#)

[Logout](#)



ACCESS GRANTED

- Note the conversion options between the XML authoring formats and the AHA! engine's internal format

# AHA! Directory Structure

- Structure in the AHA! directory tree:
  - AHAStandard: contains all publicly needed DTDs
  - WEB-INF: AHA! configuration files and classes subdirectory tree (for servlets)
  - lib: class subdirectory tree for authoring applets
  - xmlroot: XML database storage for DM/AM and UM
  - author/authorfiles: all files used and created by the authoring tools:
    - list of authors and for each author a directory with the author's applications
    - concept templates (types of concepts with their attributes)
    - concept relationship types (with their adaptation rules)
  - for each AHA! application a subdirectory tree

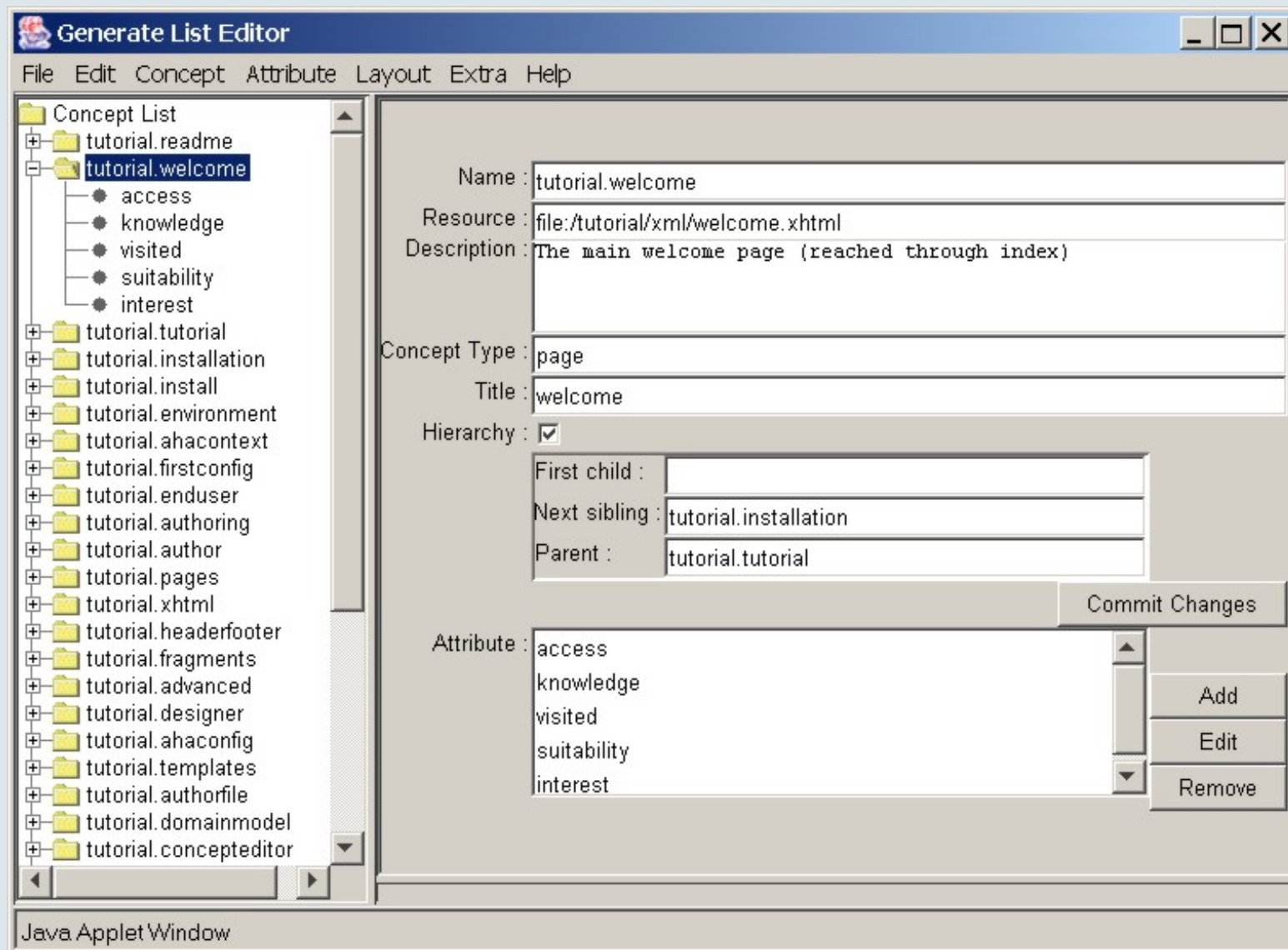
# Authoring

- Authoring process is about content alternatives, adaptation techniques and ultimately the whole user-interaction mechanism design.
- It is crucial to support the authors during this process.
- AHA! authoring tools for creating the DM/AM:
  - Concept Editor (low-level tool)
  - Graph Author (high level tool)
- Currently there is no tool for creating the application content



# The Concept Editor

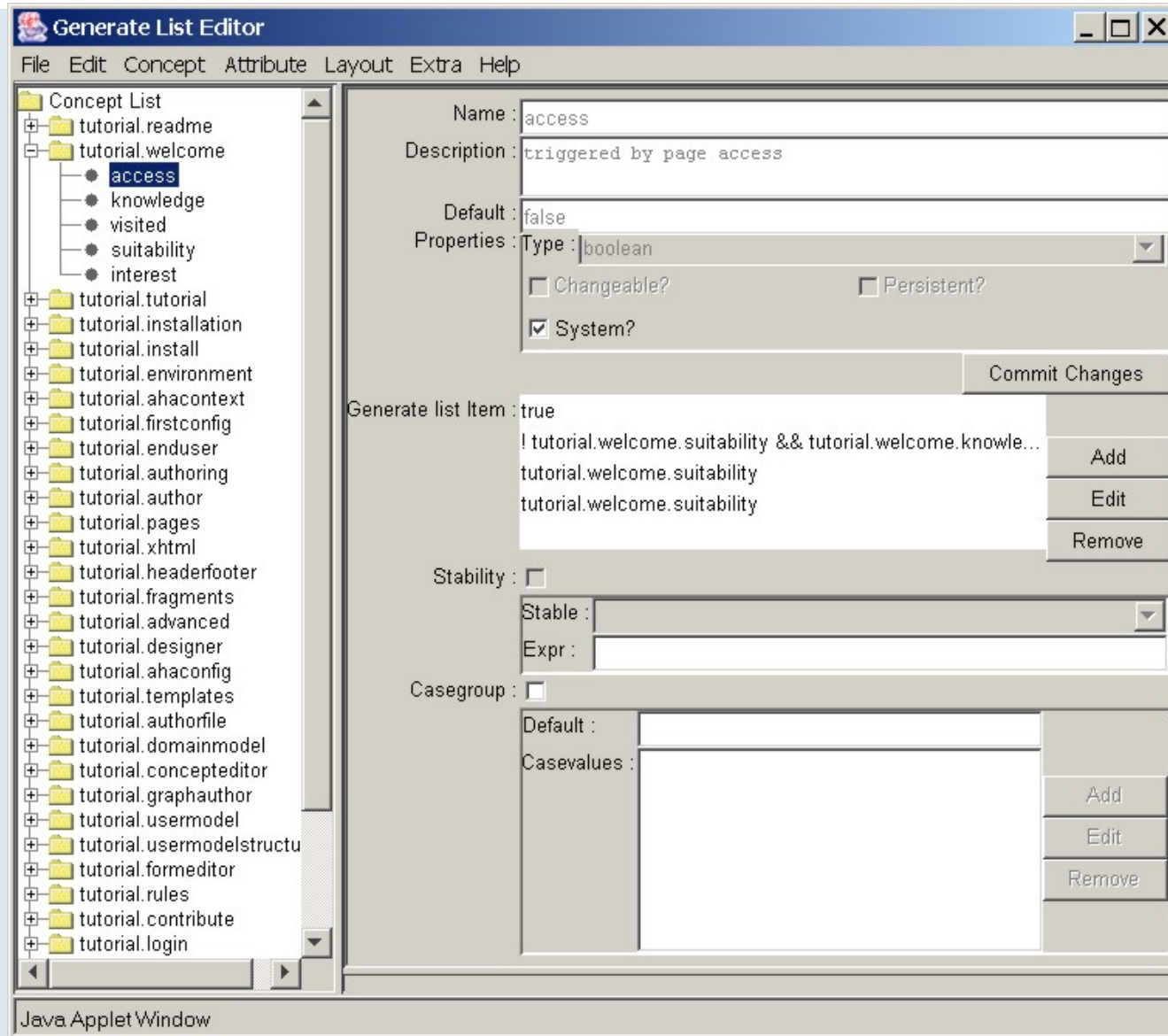
- Lets you edit every aspect of DM/AM for an application:
  - add/remove/edit a concept
  - add/remove/edit attributes of a concept
  - add/remove/edit the suitability requirement of a concept
  - add/remove/edit a casegroup for the conditional inclusion of objects
  - add/remove/edit adaptation rules:
    - each rule is tied to a triggering attribute
    - each rule has a condition
    - each rule has a series of actions executed when the condition is true
    - each rule has a (possibly empty) series of actions executed when the condition is false



The screenshot shows a Java Applet Window titled "Generate List Editor". The window has a menu bar with "File", "Edit", "Concept", "Attribute", "Layout", "Extra", and "Help". On the left is a tree view of a "Concept List" with folders like "tutorial.readme", "tutorial.welcome", "tutorial.tutorial", etc. The "tutorial.welcome" folder is selected, showing sub-items: "access", "knowledge", "visited", "suitability", and "interest". The main area contains a form for editing the selected concept:

- Name : tutorial.welcome
- Resource : file:/tutorial/xml/welcome.xhtml
- Description : The main welcome page (reached through index)
- Concept Type : page
- Title : welcome
- Hierarchy : 
  - First child : [empty]
  - Next sibling : tutorial.installation
  - Parent : tutorial.tutorial
- Attribute : [List of attributes: access, knowledge, visited, suitability, interest]

Buttons for "Commit Changes", "Add", "Edit", and "Remove" are visible. The status bar at the bottom says "Java Applet Window".



**Generate List Editor**

File Edit Concept Attribute Layout Extra Help

Concept List

- tutorial.readme
- tutorial.welcome
  - access
  - knowledge
  - visited
  - suitability
  - interest
- tutorial.tutorial
- tutorial.installation
- tutorial.install
- tutorial.environment
- tutorial.ahacontext
- tutorial.firstconfig
- tutorial.enduser
- tutorial.authoring
- tutorial.author
- tutorial.pages
- tutorial.xhtml
- tutorial.headerfooter
- tutorial.fragments
- tutorial.advanced
- tutorial.designer
- tutorial.ahaconfig
- tutorial.templates
- tutorial.authorfile
- tutorial.domainmodel
- tutorial.concepteditor
- tutorial.graphauthor
- tutorial.usermodel
- tutorial.usermodelstructu
- tutorial.formeditor
- tutorial.rules
- tutorial.contribute
- tutorial.login

Name : access

Description : triggered by page access

Default : false

Properties : Type : boolean

Changeable?  Persistent?

System?

Commit Changes

Generate list Item : true

! tutorial.welcome.suitability && tutorial.welcome.knowle...

tutorial.welcome.suitability

tutorial.welcome.suitability

Add

Edit

Remove

Stability :

Stable :

Expr :

Casegroup :

Default :

Casevalues :

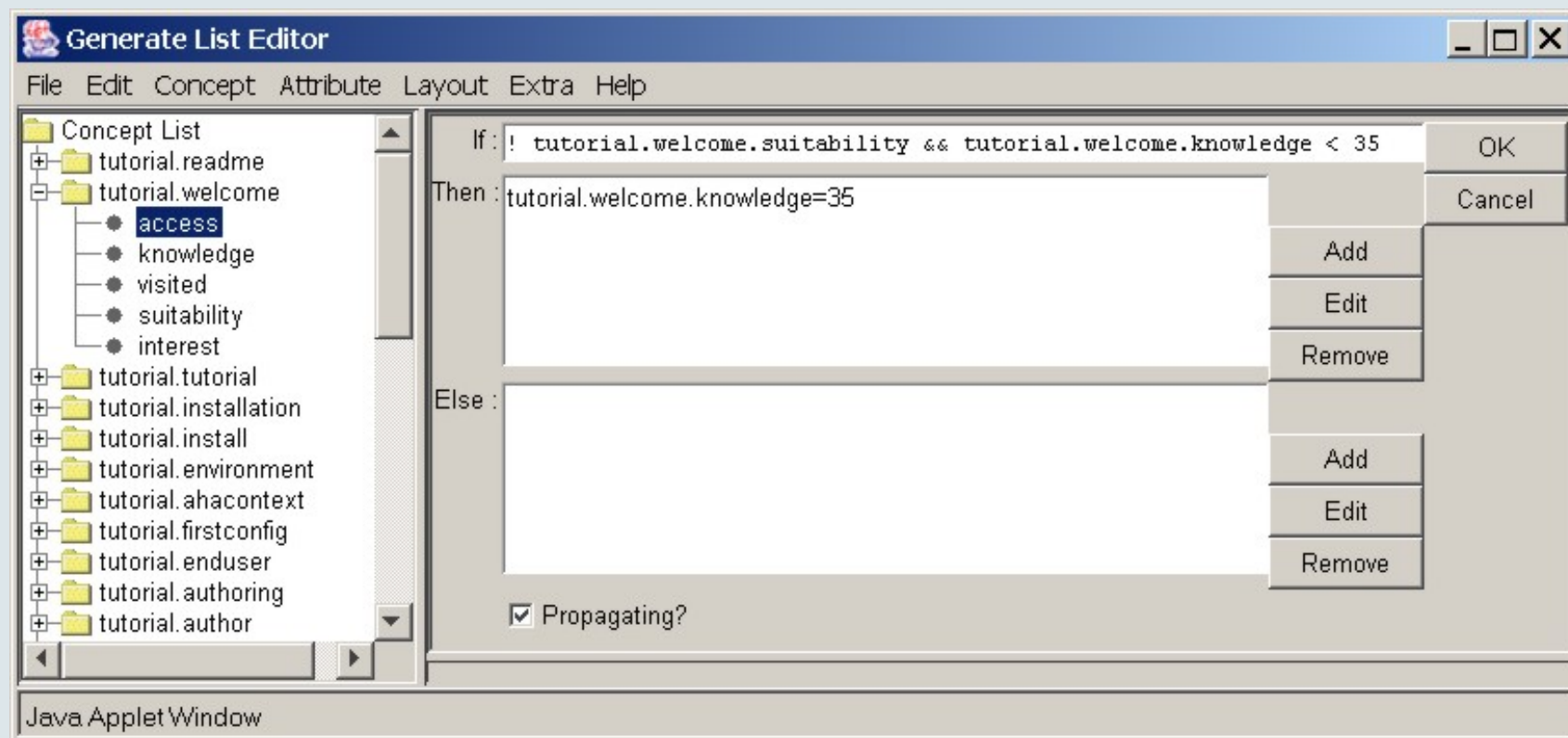
Add

Edit

Remove

Java Applet Window

# The Concept Editor: Adaptive Rules Definition



# The Concept Editor: “Stability” and “Casegroup” properties

Stability : <input checked="" type="checkbox"/>	Stable : always	
	Expr :	
Casegroup : <input checked="" type="checkbox"/>	Default : file:/tutorial/xml/empty.xhtml	
	Casevalues : object.knowledge<100 - file:/tutorial/xml/object1.xhtml	
		Add
		Edit
		Remove

# The Concept Editor: Summary

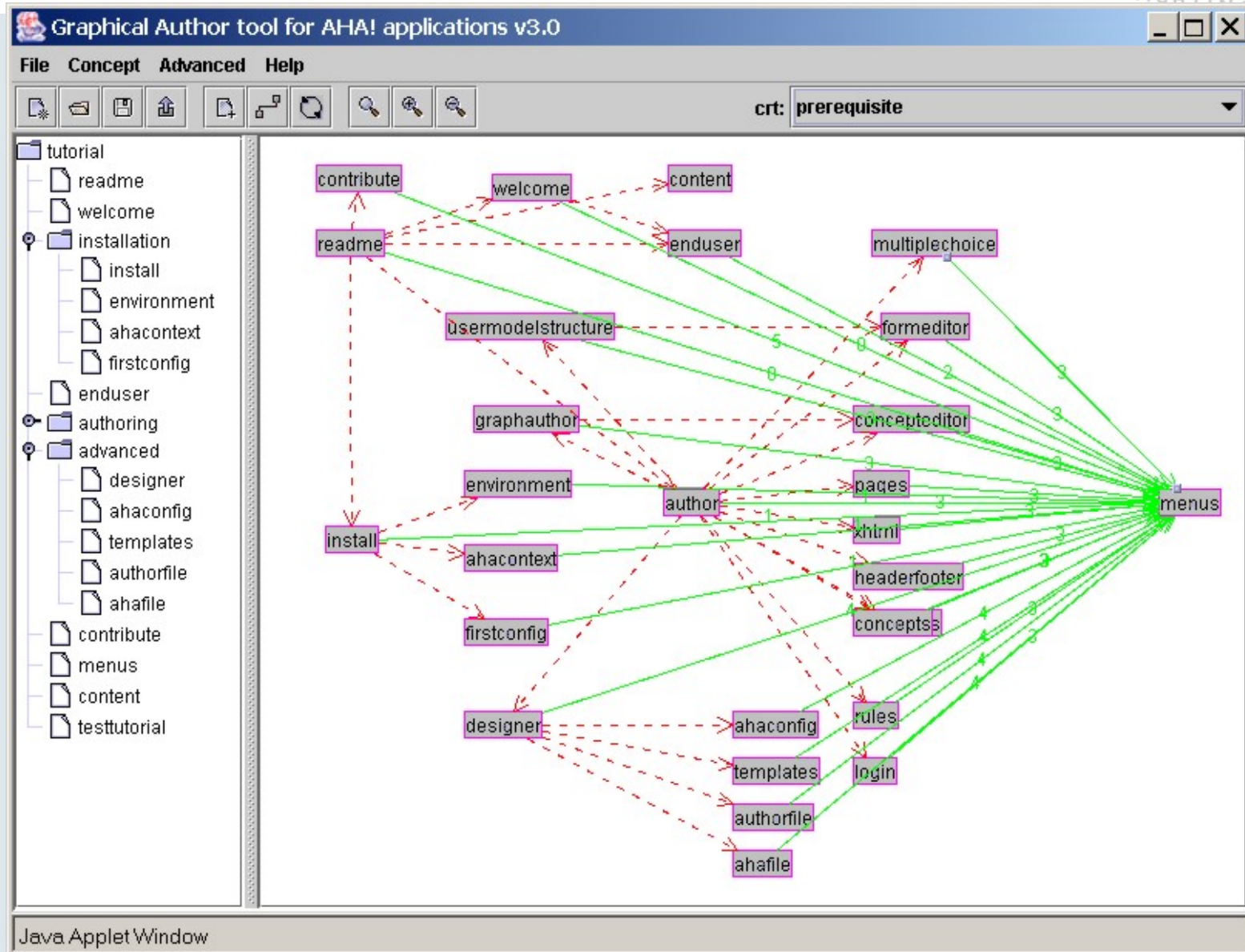
- The Concept Editor shows many aspects of the AHAM reference model:
  - arbitrarily many concepts
  - each concept may have different attributes
  - event-condition-action rules used to provide “specific adaptation rules”
  - resource tied to concepts for page selectors
  - casegroup used for page constructors

The Concept Editor is also missing AHAM structures:

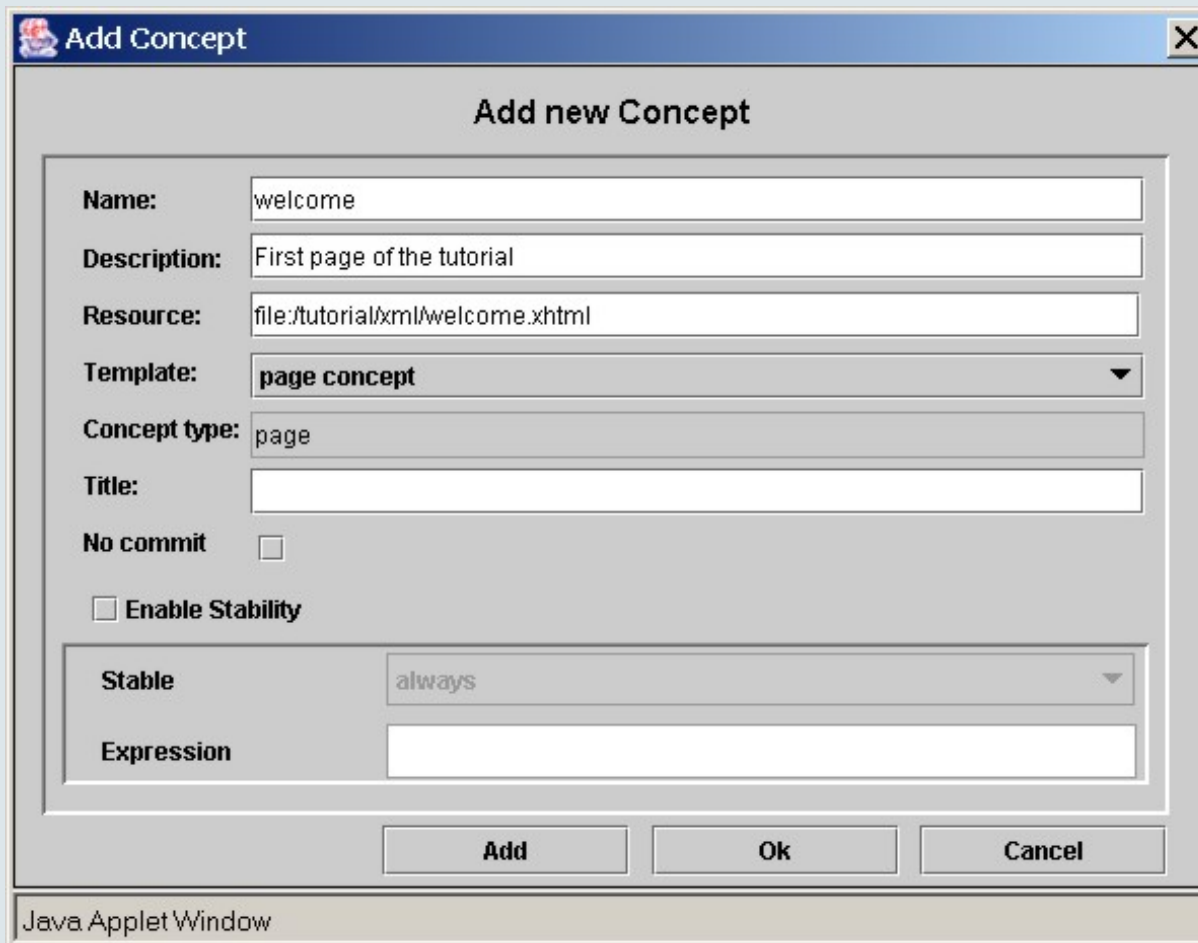
- there are no concept relationships
- there are no “generic adaptation rules”

The Graph Author Tool alleviates these shortcomings





# The Graph Author: Adding and Editing a Concept



**Add Concept**

**Add new Concept**

**Name:** welcome

**Description:** First page of the tutorial

**Resource:** file:/tutorial/xml/welcome.xhtml

**Template:** page concept

**Concept type:** page

**Title:**

**No commit**

**Enable Stability**

**Stable** always

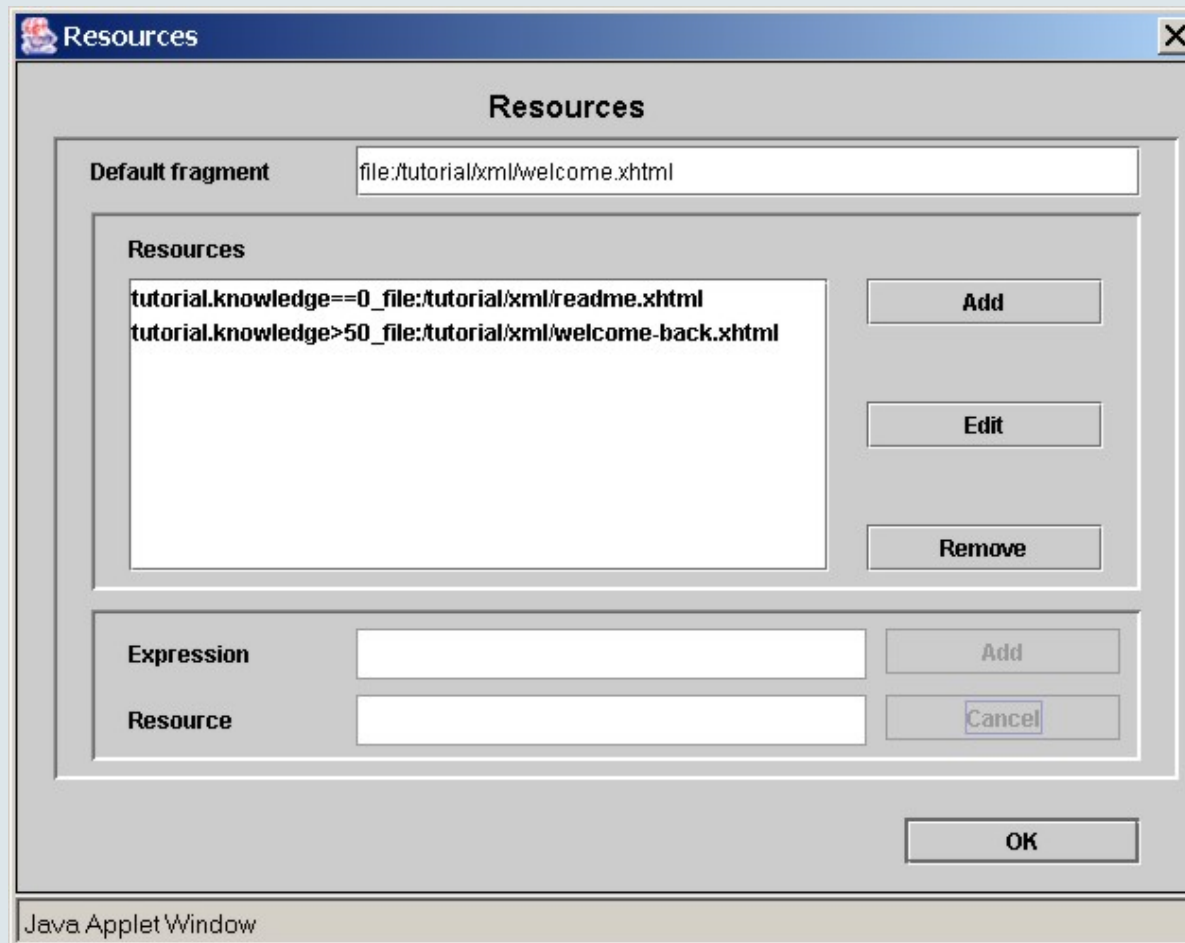
**Expression**

**Add** **Ok** **Cancel**

Java Applet Window



# The Graph Author: Defining Resource Selection



The screenshot shows a Java Applet Window titled "Resources". The window contains a "Resources" section with a "Default fragment" field set to "file:/tutorial/xml/welcome.xhtml". Below this is a list of resources with two entries: "tutorial.knowledge==0\_file:/tutorial/xml/readme.xhtml" and "tutorial.knowledge>50\_file:/tutorial/xml/welcome-back.xhtml". To the right of the list are "Add", "Edit", and "Remove" buttons. Below the list are two input fields: "Expression" and "Resource", with "Add" and "Cancel" buttons respectively. An "OK" button is at the bottom right.

Resources

Default fragment: file:/tutorial/xml/welcome.xhtml

Resources

- tutorial.knowledge==0\_file:/tutorial/xml/readme.xhtml
- tutorial.knowledge>50\_file:/tutorial/xml/welcome-back.xhtml

Buttons: Add, Edit, Remove

Expression: [ ] Add

Resource: [ ] Cancel

OK

Java Applet Window

# Concept Relationship Types

- In the Graph Author concept relationship types are tied to *generic adaptation rules*
  - Each rule can have a parameter to replace default
  - Only binary relationship types are possible
  - When a *specific adaptation rule* is needed a generic rule (template) must be created and instantiated
  - There is currently no authoring interface to create:
    - concept templates
    - concept relationship templates

There is no termination problem detection in the Graph Author Tool (but planned) and there is no confluence problem because authors have no control over the execution order of the rules.

# Concept Templates

```
<!DOCTYPE template SYSTEM 'template.dtd'>
<template>
<name>page concept</name>
<attributes>
  <attribute>
    <name>access</name><description>triggered by page access</description>
    <default>>false</default><type>bool</type>
    <isPersistent>>false</isPersistent><isSystem>>true</isSystem>
    <isChangeable>>false</isChangeable>
  </attribute>
  ...
</attributes>
<hasresource>>true</hasresource><concepttype>page</concepttype>
<conceptrelations>
  <conceptrelation>
    <name>knowledge_update</name><label>35</label>
  </conceptrelation>
</conceptrelations>
</template>
```

# Concept Relationship Types

- Concept relationship types are defined by two files:
  - one defines how the Graph Author presents it (color, arrow style) and whether it must be acyclic
  - the other one defines how it is translated to AHA! adaptation rules:

```
<!DOCTYPE aha_relation_type SYSTEM 'aha_relation_type.dtd'>
<aha_relation_type>
  <name> prerequisite </name>
  <listitems>
    <setdefault location ="destination.suitability" combination="AND">
      source.knowledge &gt; var:50 </setdefault>
  </listitems>
</aha_relation_type>
```

# Concept Relationship Types

- Here is the knowledge propagation relationship, used to propagate knowledge through the concept hierarchy:

```
<!DOCTYPE aha_relation_type SYSTEM 'aha_relation_type.dtd'>
```

```
<aha_relation_type>
```

```
  <name>knowledge_propagation</name>
```

```
  <listitems>
```

```
    <generateListItem isPropagating="true" location="child.knowledge" >
```

```
      <requirement> true </requirement>
```

```
      <>trueActions>
```

```
        <action combination="DIV_S">
```

```
          <conceptName> parent </conceptName>
```

```
          <attributeName> knowledge </attributeName>
```

```
          <expression>parent.knowledge + (var:DIVIDE * _child.knowledge)</expression>
```

```
        </action>
```

```
      </trueActions>
```

```
    </generateListItem>
```

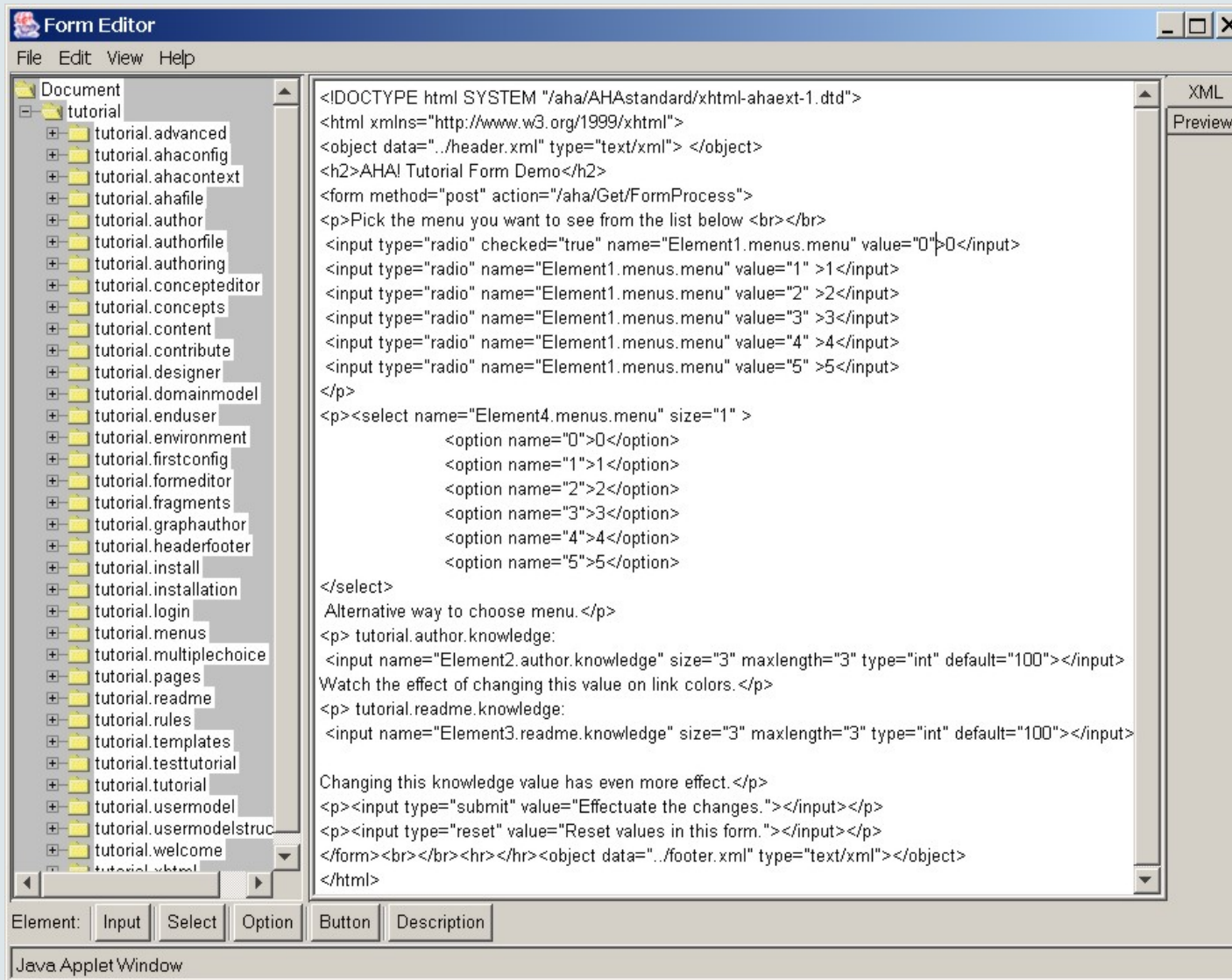
```
  </listitems>
```

```
</aha_relation_type>
```

# Forms and progress reports

- AHA! offers standard forms and reports:
  - The end user can choose link colors through the “color configuration” form. By choosing colors a choice is made between link hiding (default) or link annotation
  - The end-user can change the knowledge value for concepts where the knowledge attribute is marked as “changeable”
  - AHA! can produce a list of pages the user has read or has not read yet (from this application). This is based on the “visited” attribute value
  - AHA! can also present some other information like the user’s name, id and e-mail address

# The Form Editor



The screenshot shows the 'Form Editor' application window. On the left is a file tree under 'Document' with a sub-folder 'tutorial' containing various sub-folders like 'tutorial.advanced', 'tutorial.ahaconfig', etc. The main area is an XML editor with the following content:

```

<!DOCTYPE html SYSTEM "/aha/AHAstandard/xhtml1-ahaext-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<object data=" ../header.xml" type="text/xml"> </object>
<h2>AHA! Tutorial Form Demo</h2>
<form method="post" action="/aha/Get/FormProcess">
<p>Pick the menu you want to see from the list below <br></p>
<input type="radio" checked="true" name="Element1.menus.menu" value="0">0</input>
<input type="radio" name="Element1.menus.menu" value="1" >1</input>
<input type="radio" name="Element1.menus.menu" value="2" >2</input>
<input type="radio" name="Element1.menus.menu" value="3" >3</input>
<input type="radio" name="Element1.menus.menu" value="4" >4</input>
<input type="radio" name="Element1.menus.menu" value="5" >5</input>
</p>
<p><select name="Element4.menus.menu" size="1" >
      <option name="0">0</option>
      <option name="1">1</option>
      <option name="2">2</option>
      <option name="3">3</option>
      <option name="4">4</option>
      <option name="5">5</option>
    </select>
    Alternative way to choose menu.</p>
<p> tutorial.author.knowledge:
    <input name="Element2.author.knowledge" size="3" maxlength="3" type="int" default="100"></input>
    Watch the effect of changing this value on link colors.</p>
<p> tutorial.readme.knowledge:
    <input name="Element3.readme.knowledge" size="3" maxlength="3" type="int" default="100"></input>

    Changing this knowledge value has even more effect.</p>
<p><input type="submit" value="Effectuate the changes."></input></p>
<p><input type="reset" value="Reset values in this form."></input></p>
</form><br></hr><object data=" ../footer.xml" type="text/xml"></object>
</html>
  
```

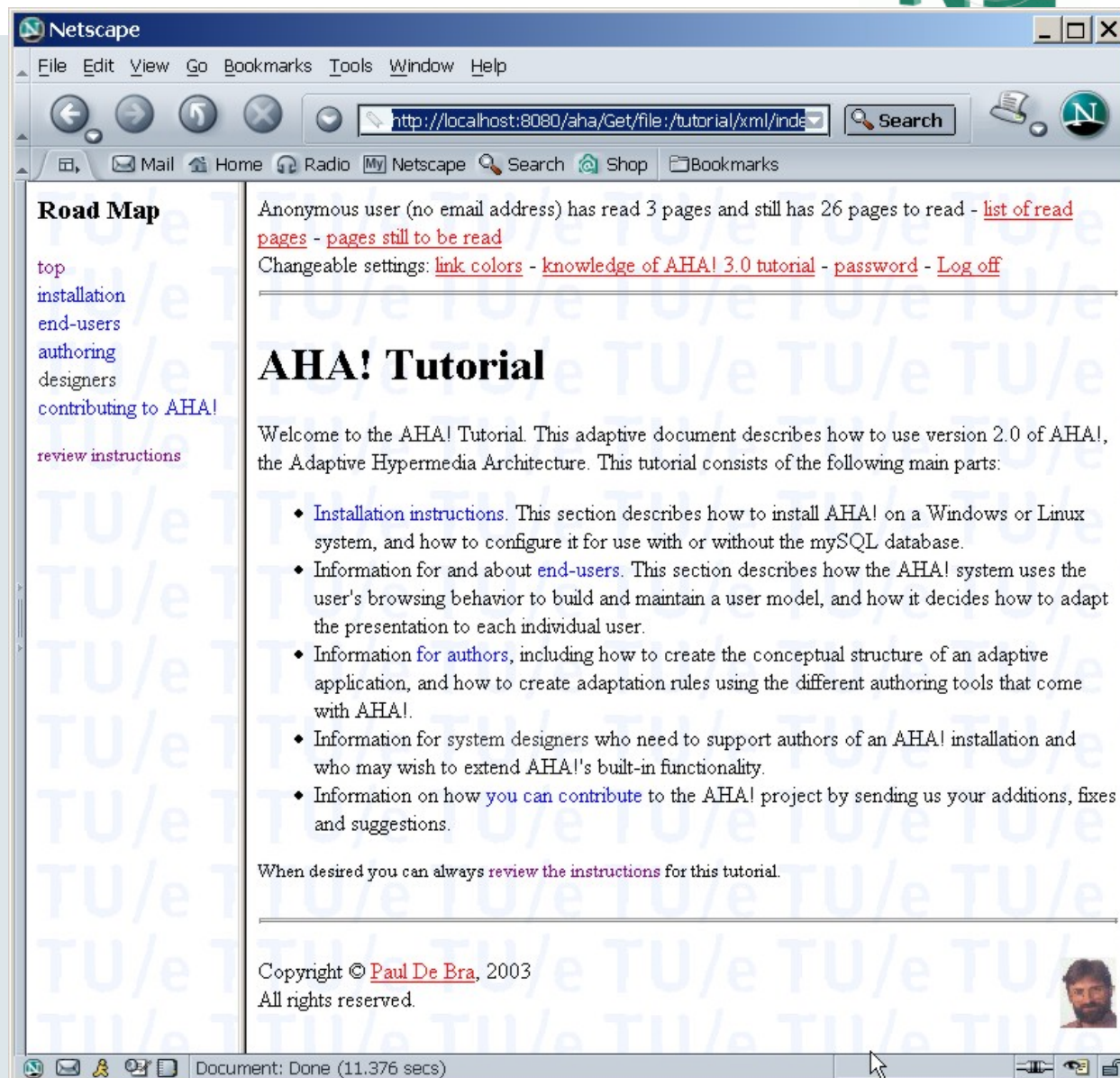
At the bottom of the window, there is a 'Java Applet Window' and a control bar with buttons for 'Element:', 'Input', 'Select', 'Option', 'Button', and 'Description'.

# Multiple-Choice Tests

- AHA! can present multiple-choice tests:
  - Each test may consist of multiple questions.
  - Each question may have one or more correct answers.
  - The test may contain more questions than are asked. The choice is random.
  - A question may have more answers than are shown. The choice is random.
  - Explanations of answers may be given if the author has decided so.
  - A score is given and stored in the user model for the knowledge attribute of the concept associated with the test.
  - There is currently no authoring tool for creating multiple-choice tests.



# Example of old AHA! Tutorial Inteface



# Layout

- Most AHS have a fixed “look and feel”
- The layout of AHA! applications can be configured:
  - **views** are atomic presentation units
  - **viewgroups** define a frame structure
  - a **layout** defines a complete presentation style (a set of viewgroups) for an application
- The presentation of links can also be configured:
  - link presentation and annotation can be defined for each layout

Anonymous user (no email address) has read 9 pages and still has 25 pages to read - [list of read pages](#) - [pages still to be read](#)  
Changeable settings: [link colors](#) - [knowledge of AHA! 3.0 tutorial](#) - [password](#) - [Log off](#)

---

## The AHA! Page Format

Pages in AHA! applications are written in **XHTML**, with or without an extension for special AHA! tags. Every AHA! page that uses the special tags must start with the following line:

```
<!DOCTYPE html SYSTEM "/aha/AHAstandard/xhtml1-ahaext-1.dtd">
```

(where the /aha prefix is the *path* parameter in the **AHA! configuration**). The content of the page uses a default namespace of **xhtml**:

```
<html xmlns="http://www.w3.org/1999/xhtml">
...
</html>
```

The content of a page must strictly adhere to the XHTML or the XHTML+AHA standard. We have a short description of the [differences between HTML and XHTML](#). AHA! pages may contain four types of special elements: [headers and footers](#), [conditional fragments](#), [conditional objects](#) and [conditional links](#).

- **Headers and footers** are included using the `object` tag, for example:  

```
<object data="../header.xml" type="text/xml" />
```
- The syntax of headers and of footers is the same. Headers and footers are optional.
- **Fragments** are conditionally included using the `if` tag. The content of a fragment must be a syntactically correct piece of XHTML+AHA code (possibly including other fragments), in which all the tags are properly nested.
- Content can also be included using **conditional objects**. These are defined through the standard XHTML `object` tag, with a special type of `aha/text`. The advantages over conditional fragments are that no extension of XHTML is needed and that the same piece of content can be included in different pages without having to copy it.
- AHA! distinguishes three types of links:
  - Links that are of the class "conditional", like in  

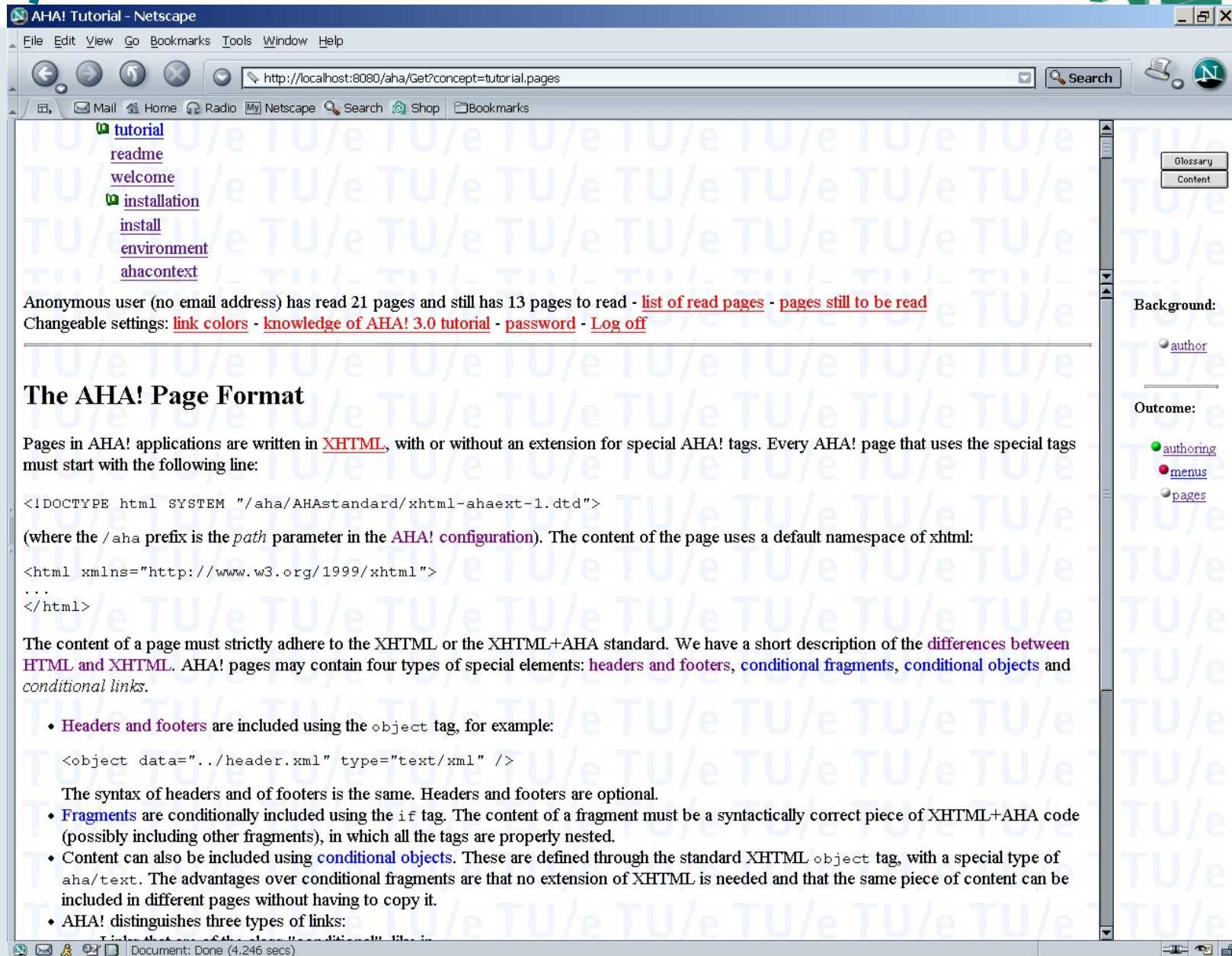
```
<a href="pagename.xhtml" class="conditional">...</a>
```
  - or  

```
<a href="http://site/pagename.xhtml" class="conditional">...</a>
```
  - are conditionally shown to or hidden from the end-user. The condition is part of the [concept structure](#). The link color of

Glossary  
Content

Document: Done (1.933 secs)





aha! tutorial - Netscape

File Edit View Go Bookmarks Tools Window Help

http://localhost:8080/aha/Get?concept=tutorial.pages Search

Mail Home Radio My Netscape Search Shop Bookmarks

- tutorial
- readme
- welcome
- installation
- install
- environment
- ahacontext

Anonymous user (no email address) has read 21 pages and still has 13 pages to read - [list of read pages](#) - [pages still to be read](#)  
Changeable settings: [link colors](#) - [knowledge of AHA! 3.0 tutorial](#) - [password](#) - [Log off](#)

---

## The AHA! Page Format

Pages in AHA! applications are written in [XHTML](#), with or without an extension for special AHA! tags. Every AHA! page that uses the special tags must start with the following line:

```
<!DOCTYPE html SYSTEM "/aha/AHAstandard/xhtml-ahaext-1.dtd">
```

(where the /aha prefix is the *path* parameter in the [AHA! configuration](#)). The content of the page uses a default namespace of xhtml:

```
<html xmlns="http://www.w3.org/1999/xhtml">
...
</html>
```

The content of a page must strictly adhere to the XHTML or the XHTML+AHA standard. We have a short description of the [differences between HTML and XHTML](#). AHA! pages may contain four types of special elements: [headers and footers](#), [conditional fragments](#), [conditional objects](#) and [conditional links](#).

- [Headers and footers](#) are included using the `object` tag, for example:  

```
<object data="../header.xml" type="text/xml" />
```

The syntax of headers and of footers is the same. Headers and footers are optional.

- [Fragments](#) are conditionally included using the `if` tag. The content of a fragment must be a syntactically correct piece of XHTML+AHA code (possibly including other fragments), in which all the tags are properly nested.
- Content can also be included using [conditional objects](#). These are defined through the standard XHTML `object` tag, with a special type of `aha/text`. The advantages over conditional fragments are that no extension of XHTML is needed and that the same piece of content can be included in different pages without having to copy it.
- AHA! distinguishes three types of links:

Document: Done (4.246 secs)

# Information

1. The AHA! project is supported by the:
  - *NLnet Foundation*
  - *ADAPT Minerva Project 101144-CP-1-2002-NL-MINERVA-MPP*
2. Will be further developed thanks to the:
  - *EU FP6 Network of Excellence PROLEARN*
3. Information and prerelease versions:

**<http://aha.win.tue.nl/>**