A Generic Adaptation Framework for Web-based Hypermedia Systems

***Alexandros Paramythis*[1]*, Constantine Stephanidis*[1,2]**

Institute of Computer Science, Foundation for Research and Technology – Hellas
Science and Technology Park of Crete
Heraklion, Crete, GR-71110, Greece
email: cs@ics.forth.gr

Department of Computer Science, University of Crete, Greece

Abstract: This chapter introduces a framework intended for facilitating the implementation of Web-based Adaptive Hypermedia Systems. The framework is orthogonal to Web "serving" approaches, and poses only minimal requirements in that direction. As such, it can be easily integrated into existing, non-adaptive Web-publishing solutions. This chapter presents in detail several aspects of the framework, and provides an overview of its application in the European Commission funded IST-1999-20656 PALIO project ("Personalised Access to Local Information and Services for Tourists"). Furthermore, it discusses some of the lessons learnt from our work on the framework thus far, as well as what we consider the most likely directions of future work in the area.

Keywords: adaptive hypermedia, adaptation framework, adaptation techniques, XML, declarative adaptation actions

1 Introduction

Adaptation characterises software products that automatically configure their parameters according to the given attributes of individual users (e.g., mental / motor / sensory characteristics, requirements and preferences), and to the particular context of use (e.g., hardware and software platform, environment of use). Adaptive software systems have been considered in a wide range of research efforts. The relevant literature offers a wealth of examples illustrating tools for constructing adaptive interaction (e.g., Brusilovsky et al., 1998; Horvitz et al., 1998; Kobsa and Pohl, 1995), and case studies in which adaptive interface technology has improved, or has the potential to improve, the usability of an interactive system (e.g., Dieterich et al., 1993; Benyon 1997).

Adaptive Hypermedia Systems (AHS for short), in particular, are a relatively new area, which has drawn considerable attention since the advent of the Web. There exist today numerous AHS, in various applications domains, with a great variety of capabilities (see, e.g., Ardissono and Goy, 1999; Balabanovic and Shoham, 1997; Brusilovsky et al., 1998; Henze, 2001; Oppermann and Specht, 1998; Kobsa, 2001). Major categories of AHS include educational hypermedia, on-line information systems, on-line help systems, information retrieval systems, and institutional hypermedia.

This chapter presents a generic framework for the development of adaptive Web-based hypermedia systems and services. Adaptation, in this context, implies the capability, on the part of the system, to capture and represent knowledge concerning alternative instantiations suitable for different users, contexts, purposes, etc., as well as for reasoning about those alternatives to arrive at adaptation decisions. Furthermore, adaptation implies the capability of assembling, coherently presenting, and managing at run-time, the appropriate alternatives for the current user, purpose and context of use.

In the context of this chapter, the term "framework" is used to refer to an architectural design describing the components of the system and the way they interact (Campbell et al., 1997). The confines of an architectural framework for software systems are perhaps best described as per (Jacobson et al., 1997):

> "The software architecture, first of all, defines a structure. Software components have to fit into some kind of design. […] Second, the architecture defines the interfaces between components. It defines the patterns by which information is passed back and forth through these interfaces."

The presented framework comprises both implemented components and specifications (in the form of programmatic interfaces and associated semantic "contracts") of how core- and external- components interact to attain adaptive system behaviour. The framework has been implemented in Java, and comprises concrete classes, which implement the functionality of the core components, as well as abstract classes and interfaces, which are used when integrating external components with the framework.

The main characteristics of the framework can be summarised as follows: support for declarative (vs. programmatic) specification of adaptive system behaviour; composition of adaptive hypermedia techniques from lower-level adaptation actions; inherent support for different approaches to representing and evaluating user- and context- models, as well as adaptation logic itself; domain-independence, coupled with provisions for capturing the semantics and specificities of individual application domains; and, finally, orthogonal applicability to any document-centric hypermedia system with XML-compliant output.

The framework under discussion was employed in the development of the PALIO tourist information system. The European Commission-funded IST-1999-20656 PALIO project ("Personalised Access to Local Information and Services for Tourists", see "Acknowledgements" section) addressed the issue of Universal Access to community-wide services, based on content and user interface adaptation beyond desktop access. The presented framework was used to enable adaptive system behaviour at the interaction and content levels, on the basis of user- and context- characteristics (including terminal device capabilities, user location, etc.) The evaluation of the resulting information system by end users provided very positive feedback with respect to the system's adaptive features.

The rest of this chapter is structured as follows. The next section, "Generic Adaptation Framework", introduces the framework itself. The presentation commences with an account of the motivation behind the framework and a brief overview of related work (2.1, "Background and Related Work"). Section 2.2, "Framework Overview" contains a high-level view of the main premises of the framework, and its relation to current Web publishing schemes. The core of the framework is described in section 2.3, "Adaptation Engine". Following that, *adaptation actions*, one of the cornerstone concepts of the framework, is discussed in detail (2.4, "Adaptation Actions"). The discussion covers both the types of adaptation actions that the framework currently supports, and, at a more theoretical level, the relationship between adaptation actions and adaptive hypermedia techniques in the literature. Section 2.5, "Adaptation Decision-Making", addresses another major aspect of adaptive systems, namely, deciding upon the need for, and the type of, adaptation. Subsections discuss the framework's support for alternative approaches to decision-making, and the default rule-based implementation. Section 2.6, "Integrating Modelling Components" presents the way in which the framework abstracts over different dynamic and static system models in the context of user / interaction, decision making and model updating. Section 3, "Applying the Framework in PALIO", provides an overview of our experiences in using the adaptation framework to enable adaptive behaviours in the PALIO tourist information systems. Section 4, "Conclusions and Future Work", concludes the chapter with the valuable insight gained along the way, and an outline of what we consider likely directions of future work in the area.

2 Generic Adaptation Framework

2.1 Background and Related Work

The main goals in the development of the presented adaptation framework were:

- Support a wide range of adaptive hypermedia techniques, in a domain-independent way.
- Provide constructs that facilitate the declarative specification of adaptive behaviour.
- Achieve, at the architectural level, "orthogonality" with existing Web-publishing approaches, so that the framework can be easily integrated into existing non-adaptive systems and services.
- Enable the clear separation of adaptation components, so that their implementation can be varied independently.

These goals are discussed briefly below and contrasted against related work in the field.

To start with, the term "adaptive hypermedia techniques" (Brusilovsky, 1996) is used to refer to modifications which can be adaptively applied to hypermedia documents and which can be synthesised to arrive at higher-level adaptation *method*, such as "additional explanations" and "global guidance" (Brusilovsky, 1996). Figure 1 presents a partial classification of "server-side" adaptation techniques. The figure has been based on the classification introduced in (Brusilovsky, 1996) and refined in (Brusilovsky, 2001) and contains the techniques that should be directly supported by the framework. Techniques not shown include *adaptive multimedia adaptation* (a sub-category of *adaptive presentation*), *natural language adaptation* (a sub-category of *adaptive text presentation*), and *map adaptation* (a sub-category of *adaptive navigation support*).

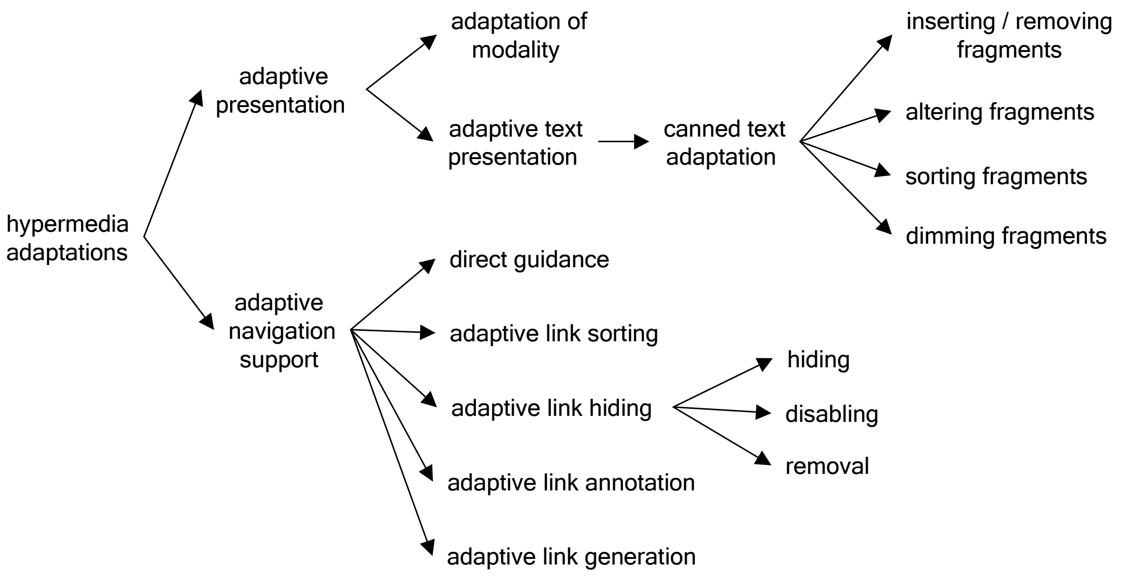


Figure 1: Classification of adaptation techniques supported by the framework, as per (Brusilovsky, 1996) and (Brusilovsky, 2001)

There exist today several Web-based AHS, which address a number of these techniques in a more or less domain-independent manner. A representative example in this category is the AHA! system (De Bra & Stash, 2002; De Bra et al., 2002b). In its second generation, AHA! has become a rather comprehensive system, including also integrated authoring tools. Although this greatly facilitates the creation of new adaptive systems, there is at least one significant problem with the approach taken: the AHS is rather “monolithic” and cannot be easily integrated with pre-existing, non-adaptive systems; rather the AHS *is* the Web-publishing system. This, along with a specific approach to achieving adaptations (based on domain concepts and their relationships (De Bra et al., 2002a)) seriously impedes the applicability of the system outside its original domain of adaptive course provision.

A different approach is represented by the KnowledgeTree framework (Brusilovsky and Nijhaven, 2002). Whereas the majority of AHS are designed to exist as stand alone systems, KnowledgeTree has been designed to source adaptive content and functionality externally, not encapsulating them into a monolithic core. KnowledgeTree is specifically intended to facilitate interoperation and reuse at the level of distributed, reusable learning activities (with the emphasis being on learning activities, as opposed to learning objects). To this extent, KnowledgeTree goes into the realm of run-time communication and interoperation standards, seeking to standardize the ways in which different specialized subsystems supporting aspects of the (adaptive) learning process can communicate and exchange information that would allow them to be aggregated into a “whole”. Although KnowledgeTree is explicitly targeted towards adaptive learning environments, its main concepts can easily be generalised across application domains. However, KnowledgeTree imposes a specific portal-oriented structure to the Web-publishing system, which may inappropriate in certain scenarios.

The concept of declarative specification of adaptation logic is not new. Several AHS (including, for example, AHA!) make it possible to specify the adaptive behaviour of the system through relations between domain concepts and actions to be taken when specific conditions are met. However, the decomposition of adaptations applied into more basic building blocks that can be reused individually has only been recently addressed. The body of work that is perhaps closest to the framework presented herein, in this respect, is the “LAG” model (Cristea & Calvi, 2003). This is a three three-layer model and classification method for adaptive techniques comprising the following levels: direct adaptation rules, adaptation language and adaptation strategies. The model is aimed at standardizing adaptation techniques at the different levels and, thus, enable the exchange of adaptive techniques between different applications. It also aims to help the authors of adaptive hypermedia by giving them higher-level “handlers” of low-level adaptation techniques (Cristea & Calvi, 2003). Although the objectives of the two approaches are quite similar, there exist fundamental differences in the way they are achieved. In LAG, for instance, the lowest layer addresses adaptation techniques as functions that map the current state of the AHS and its models to a subsequent (adapted) state. The middle layer comprises the adaptation rules, and the third layer addressed adaptation “strategies”, as these relate to the user’s information processing characteristics and cognitive styles. To contrast this with the approach taken in the presented framework, please refer mainly to sections 2.4, “Adaptation Actions” and 2.5, “Adaptation Decision-Making”.

2.2 Framework Overview

One of the main premises of the developed framework is that it follows a *document-centric* approach, and is geared towards *XML-based document representation*. This necessitates that either the documents “served” by a Web-based system are “expressed” in an XML-based language, or that they can be easily converted into such a representation. The framework does not assume that this is the final step in the document processing cycle. It only requires that, *at some stage of this cycle*, documents be represented in XML; the end result could be, for example, a simple text file, PDF document, etc.

The process of adapting documents requires the cooperation of at least two different types of components, namely the *decision-making component*, and the *adaptation engine*. The former is responsible deciding upon adaptations to be performed. The latter is responsible for applying adaptation decisions, expressed through adaptation actions. Adaptation decisions, in turn, typically require access to the adaptation models (e.g., user model, context model, domain / application model), which are encapsulated by the *modelling components*. Communication with the modelling components is also necessary in the case of models that are updated dynamically. In this case, the communication concerns the exchange of interaction data that will be used as “evidence” towards the dynamic updates.

Communication between the components that make up the framework is done through a set of well-defined programmatic interfaces, intended to enable the decoupling of the components and facilitate their replacement with alternative implementations. This is also the case for accessing and manipulating information available from sources “external” to the framework, such as user modelling servers and domain-specific information servers.

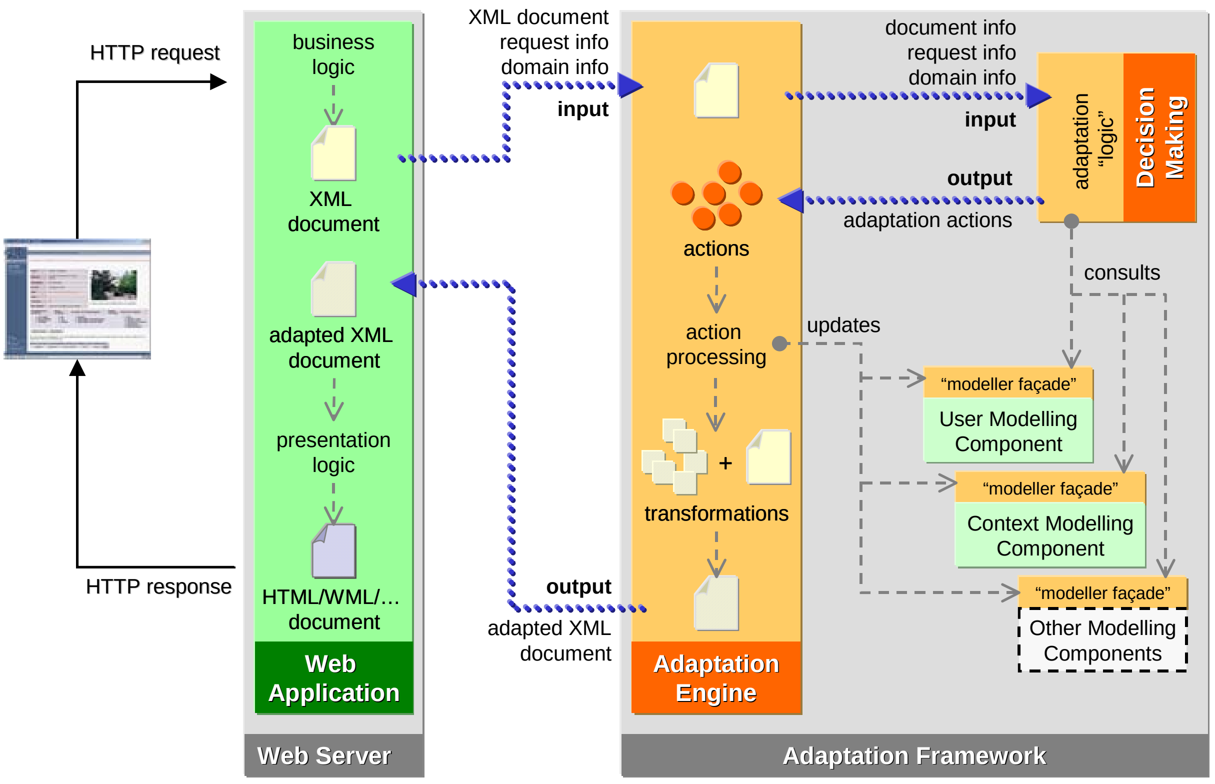


Figure 2: A high level view of the adaptation process, showing framework components and their interactions.

Figure 2 depicts a high-level view of the framework components and their interactions. Furthermore, it shows the adaptation process, as this is implemented by the framework components, and relates it to the document processing cycle typical of current Web publishing architectures. The next section describes the figure in detail and outlines the responsibilities of the framework’s main component, the adaptation engine.

2.3 Adaptation Engine

The responsibilities of the adaptation engine include the invocation of the decision-making component and the realisation of adaptation decisions (expressed through adaptation actions). The rest of this section outlines the adaptation process supported by the framework (and depicted in Figure 2) and the role of the adaptation engine within that process.

The left part of Figure 2 shows an abstract view of a typical Web-based system or service: a request is received from the client; business logic is executed as a result of the request; a document is selected / assembled / generated / etc., as the basis for the response; in some cases, the basic document may then be further processed (e.g., to add navigation information, render it to an appropriate output format, etc.); the final document constitutes the system's response to the user's request.

Integration of the presented framework (right part of Figure 2) in an existing Web-based system or service involves primarily the invocation of the adaptation engine, at any point in the document processing cycle. In Java-based systems, the adaptation engine can be invoked through a Java class, which acts as a "gateway" into the framework. For non-Java based systems, or when distribution is desirable, there is the possibility to set up the engine as a remotely accessible Web service.

When the engine is invoked, the following are passed as parameters: the XML document / fragment itself; information about the request that resulted in the generation of the document / fragment; optionally, domain-specific information, which can be utilised by the decision-making component (see below).

The adaptation engine communicates with the *decision-making component*, which is responsible for determining what type of adaptations are to be performed to the document / fragment. The information sent to the decision-making component include: information about the document itself; information about the request (as above); and, domain-specific information (as above). The decision-making component, in turn, consults the various dynamic and static models that relate to adaptation, in order to decide upon the necessity for, and type of, adaptations to be applied. The result of the decision-making process is a set of *adaptation actions* that are communicated back to the adaptation engine. (Section 2.5, "Adaptation Decision-Making" and 2.6, "Integrating Modelling Components", provide additional details on the decision-making component and the modelling components respectively).

The adaptation actions are interpreted by the adaptation engine and result into either: (a) XSLT[1] transformations to be applied to the document at hand, or (b) "update" actions which directly or indirectly modify the contents of dynamic models. "Update" actions are communicated to their corresponding modelling components. XSLT transformations are applied directly to the XML document. The transformed document is the output of the adaptation process and is sent back to the invoker. (Section 2.4, "Adaptation Actions" provides an overview of supported actions).

It should be noted that, although in the figure the steps described above are depicted as occurring only once per cycle, there is no such restriction in the framework. In fact, when the underlying system creates responses through the composition of fragments, it is necessary that

[1] http://www.w3.org/TR/xslt

the steps be repeated for each of the fragments (and possibly for the whole as well). Furthermore, fragment-based document composition is not the only scenario that requires repetition of the steps. For example, in PALIO (see section 3, “Applying the Framework in PALIO”) the adaptation cycle is repeated twice: adaptations in the first cycle, or “stage” are intended to adapt information queries embedded in documents, while adaptations in the second “stage” are intended to adapt the results of the queries and the rest of the document.

Additional features of the framework, which are not discussed in further detail in this chapter, include the following. The framework is capable of “pushing” information to the user, as a result of adaptation decisions made at run-time, and outside the scope of the request-response cycle. Specifically, the framework supports: retrieval of user- and context-related information through channels other than the browser-server connection (e.g., user location); explicit triggering of the decision-making component to ensure that adaptation logic is evaluated against the new information artefact; and, immediate execution of specialised adaptation actions that may be the outcome of the above process. These actions are apparently focused on the generation / selection of documents that are subsequently communicated to the user. This capacity of the framework has been used in the context of the PALIO information systems to adaptively send information to users (through SMS) on the basis of the user’s current location and the users’ inferred interest in venues, tourist landmarks, public facilities, etc., in their vicinity.

2.4 Adaptation Actions

Adaptation actions are the means for adaptively transforming documents (or fragments of documents). They are translated internally by the framework’s *adaptation engine* into XSLT transformations that are subsequently applied to the document / fragment. It is exactly this role of the adaptation engine that renders it a focal point of the framework’s implementation.

In order to support the declarative specification of adaptations to be performed, an XML-based language has been developed. The language circumscribes the actions that can be performed, as well as any additional information required by the adaptation engine to successfully carry out these actions. There is no restriction as to the number or type of actions that can be performed on an individual document or fragment. However, since actions are applied sequentially and may effect significant modifications to a document, it may be necessary to handle dependencies between them. The framework makes the fundamental assumption that such dependencies are external to the actions themselves and are, thus, expressed and handled separately. In other words, if actions need to be sequenced to achieve the desired adaptation effect, the adaptation engine assumes that such sequencing has already been applied by the decision-making. In addition to explicit sequencing, the framework supports the assignment of rule priorities to enforce the correct handling of action dependencies (see section 2.5, “Adaptation Decision-Making” for details).

As already mentioned, adaptation actions are applied by the framework on XML-based documents and fragments. Therefore, it is often the case that authors need to specify the element (or elements), on which, or in relation to which, the actions are applied. These will be referred to henceforth as the “reference” elements of the action. Reference elements can be specified in the following ways (common to all adaptation actions): using the element’s *tag name;* using an arbitrarily specified attribute, or set of attributes, of an element (e.g., “class”, “id” for XHTML-based documents); using an XPath[2]-style “selector” expression.

[2] http://www.w3.org/TR/xpath

The basis for deciding which actions need to be inherently supported (as opposed to actions that can be “composed” from more basic ones) has been: (a) an analysis of possible low-level adaptations that can be performed in interactive systems (see, e.g., (Dieterich et al., 1993)), and (b) an analysis of known hypermedia techniques in the literature, with respect to the effort required for synthesising them from basic actions (e.g., “dimming” would be generally rather straightforward, whereas “sorting” might not even be realistically practical to compose).

Currently, the framework supports the following document-oriented adaptation action categories (see (Stephanidis et al., 2004) for a more extensive description): inserting document elements / fragments; removing document elements / fragments; replacing document elements / fragments; sorting document elements / fragments; setting and removing element attributes; selecting among alternative document elements / fragments; applying arbitrary document transformations expressed in XSLT. Section 2.4.1, “Synthesising Adaptation Techniques” discusses how these basic actions can be composed into higher-level adaptation techniques

Further to the above document-oriented actions, adaptation decisions may also comprise model-update actions. These, as their suggests, are intended to effect direct or indirect modifications in the system’s dynamic models. These actions are expressed as manipulations of scoped variables and are further described in Sections 2.4.2, “Model-update actions”, and 2.6, “Integrating Modelling Components”

2.4.1 Synthesising Adaptation Techniques

Table 1: Synthesizing Adaptive Hypermedia Techniques (Paramythis et al., 2003b)

	Atomic Adaptation Actions in the Adaptation Framework / Adaptive Hypermedia Techniques	Inserting document elements / fragments	Removing document elements / fragments	Replacing document elements / fragments	Manipulating element attributes	Selecting among alternative document elements / fragments	Sorting document elements / fragments
Adaptive Navigation	Link annotation	✓			✓		
	Link hiding		✓		✓		
	Link disabling		✓	✓	✓		
	Link removal		✓	✓			
	Link sorting						✓
	Link generation	✓		✓			
	Direct Guidance	✓	✓	✓		✓	
Adaptive Presentation	Adaptation of modality					✓	
	Inserting fragments	✓					
	Removing fragments		✓				
	Altering fragments	✓	✓	✓	✓		
	Sorting fragments						✓
	Dimming fragments				✓		

One of the desiderata for the framework under discussion was to provide adequate support for all the techniques in Figure 1, in a way that allows for future refinements and extensions. As already mentioned, adaptation actions are intended as lower-level building blocks that can be used in isolation or in combination to synthesise higher-level adaptation techniques. Table 1 presents the adaptive hypermedia techniques that are directly supported by the framework, associating them with the adaptation actions that can be used (or are required) to implement them (Paramythis et al., 2003b). Note that, although there is a distinction in the domain of the techniques (i.e., whether they address presentation or navigation), as per the classification depicted in Figure 1, such a distinction is not present in the adaptation actions themselves. As these are at a lower level of abstraction, they are necessary in both domains.

2.4.2 Model-update actions

Model-update actions are defined, in the framework, as model variable manipulations. In more detail, the framework supports the concept of model *variables*, variable *namespaces*, and variable *manipulation*. A "variable" can be any piece of information that resides in an external modelling component (e.g., the probability that the user is interested in a particular tourist venue is a user model variable that can be accessed from the user modelling server). The concept of variables has been apparently borrowed from programming constructs. Our goal has been to provide adaptation designers / authors with a familiar metaphor that would enable them to think uniformly about, while abstracting over, information interchange with modelling components.

We considered variable manipulation an appropriate metaphor, as it has explicit and intuitively clear affordances for: (a) storage of values; (b) retrieval of values; (c) application of operations on values; and (d) composition of types. Integrated modelling components are required to support at least the storage and retrieval operations (see section 2.6, "Integrating Modelling Components" for details). The semantics of each may vary depending on the model and modelling approach and are not strictly defined in the context of the framework.

Variable namespaces are an optional extension to the concept of variables. They are intended, on the one hand, to ensure uniqueness of variable references where potential problems might exist, and, on the other hand, to structure access to the "global" variable space that would be created if all modelling components pooled their variables together. As namespaces are optional, the framework does not assume their presence.

2.5 Adaptation Decision-Making

The adaptation engine communicates with the decision-making component through two complementary programmatic interfaces. The first of the interfaces covers the retrieval of adaptation decisions from the decision-making component by the adaptation engine. The second addresses the propagation of decisions from the decision-making component to the engine and can only be supported under certain conditions. Due to space considerations, the rest of this section focuses exclusively on the first mode of communication.

When sending a request to the decision-making component, the adaptation engine sends along information about (see also Figure 2): the document to be adapted; the user request that resulted in the generation of the document; domain-specific information sent to the framework by the non-adaptive part of the system. The engine expects as a response a set of adaptation actions to be applied to the document at hand (the set may, of course, be empty). The way in which adaptation logic is represented and evaluated within the decision-making component can vary widely and is "opaque" to the framework.

The framework is accompanied by a default implementation of a decision-making component, which is rule-based. To facilitate the wide adoption of the framework, a new, simple rule language was created, borrowing from control structures that are commonly supported in functional programming languages. An XML binding was developed for the aforementioned rule language, while a rule interpreter and a corresponding rule engine supported the run-time operation of the component. A similar approach was first applied in the AVANTI project with very good results (Stephanidis et al., 2001).

Rules are organised into rule-sets; typically, each rule-set resides in a different file, but this is a matter of organisation, rather than a constraint imposed by the framework. Rule-sets have a specific *name* and *scope*. The name may be used to refer to the rule-set within configuration files and can be used, for example, to enable / disable a whole set of rules. The possible values of the scope attribute are not pre-defined and are intended to facilitate rule organisation in an implementation-specific way. For example, in PALIO three scopes existed (*global*, *service* and *local*), reflecting the basic units of functionality organisation in the system.

Every rule has the following attributes: (a) *name*: an identifier for the rule; (b) *class*: optionally used as an alternative way for grouping rules; (c) *stage*: optionally used to specify that rules are to be evaluated within named adaptation cycles; (d) *priority*: can optionally be used to provides a partial ordering scheme for rule evaluation and application.

The framework currently supports three types of rule constructs: *if-then-else* rules, *switch-case-default* rules, and *prologue-actions-epilogue* rules. The semantics of If-then-else and Switch-case-default rules are quite close to their counterparts in programming languages and are omitted for the sake of brevity. The prologue-actions-epilogue construct is intended mainly for the definition of unconditional rules. It supports the definition of (sets of) actions to be performed at a particular stage in the sequence of adaptations. The concept of *action-sets* is used to provide an explicit separation between "preparatory" actions, the adaptations themselves, and "clean-up" actions. It is argued that this separation allows for better rule structuring and improves the maintainability of the rule definition.

The following is a simplified example taken from PALIO, and specifically from an adaptation rule-set that controlled adaptations related to the detailed presentation of accommodation venues. The example makes use of some of the facilities of the prologue-actions-epilogue construct, and demonstrates two subsequent steps:

(a) First, the most interesting accommodation facility for the current user is identified. This is done by retrieving from the user-modelling server the probabilities associated with the user's interest in any of the facility categories found under "*accommodation.facilities. establishment.public*", and selecting among them the one with the highest probability (note also that a threshold value is set of 0.6 to ensure that the rule is applied only when there is sufficient evidence for a user's interests).
(b) Occurrences of the identified facility (if any) are emphasized for easier recognition and faster access by the user.

```
<!-- The tag namespace "adapt" has been ommitted for clarity -->

<ruleset name="accomodation-details" scope="service">

  <rule name="get-most-interesting-facility" stage="second" priority="medium">
    <action-set>
      <actions>
        <bind name="user.most_interesting_facility" src="Temporary"
              type="string">
          <get-max threshold="0.6" return="@name">
            <variables
                 from="user.interests.accommodation.facilities.establishment.
                       public" src="DPS"/>
          </get-max>
        </bind>
      </actions>
    </action-set>
  </rule>

  <rule name="emphasize-most-interesting-facility" stage="second"
        priority="medium">
    <if>
      <condition>
        <ne>
          <variable name="user.most_interesting_facility"
                    src="Temporary" type="string"/>
          <constant type="null" value="null"/>
        </ne>
      </condition>
      <then>
        <set-attribute-parameterised>
          <select-element>
             <select-element-type>
               //accomodation/facility
             </select-element-type>
             <select-element-attribute>type</select-element-attribute>
             <select-element-attribute-value>
               <variable name="user.most_interesting_facility"
                         src="Temporary" type="string"/>
             </select-element-attribute-value>
          </select-element>
          <attribute-name>
             style
          </attribute-name>
          <attribute-value>
             emphasized
          </attribute-value>
        </set-attribute-parameterised>
      </then>
    </if>
  </rule>

</ruleset>
```

2.6 Integrating Modelling Components

External modelling components are integrated with the framework through a set of programmatic interfaces referred to, collectively, as the “modeller façade”. These interfaces support the querying of the models’ contents, as well as their manipulation (for dynamic models). There are also provisions for propagating information about modifications in dynamic models to registered observers (the decision-making component being the primary such observer).

One of the primary tasks of a modeller façade implementation (henceforth referred to as “model adapter”) is to propagate evidence derived from monitoring users’ interaction with the system (or changing context parameters, etc.) to the external modelling component. Although not explicitly depicted in Figure 2, this is a vital part of the adaptation process; without monitoring information and dynamic updates in the respective models, there is no way to achieve adaptive system behaviour (although adaptability would still be possible).

The evidence communicated to a model adapter may follow the concept of variables and namespaces (discussed earlier), although this is not required. The framework makes no assumptions about the approach used to derive, or infer model attributes from the evidence communicated. Nevertheless, it does assume that models can be queried, and it does expect the return values of such queries to be either among a number of “basic” types, or to be composites which contain other composites or basic types. This limits somewhat the spectrum of modelling approaches that can be employed, precluding specifically ones that follow a "one-step" approach to adaptation (e.g., solutions that employ neural networks-based intelligence). It is argued, however, that this is not a significant limitation of the framework; modern approaches to adaptation require that models be "transparent" (Höök et al., 1996), or "scrutable" (Kay, 1995), also leaves out such "opaque" modelling approaches.

As mentioned above, the contents of models can be "queried" and accessed using the predefined programmatic interfaces of model adapters. The simplest form of a query is the retrieval of the values of specific, named model attributes. Supporting this minimum retrieval functionality is a requirement for integration with the framework.

The extended querying capabilities that are defined by the framework are also quite simple, and typically consist of applying common set operators and functions on sets of data. The reason for keeping requirements at such low levels has been to allow for the possibility of supporting them in the model adapter implementation, if they are not inherently supported by the modelling component / server. As in the case of sending evidence, variables (and namespaces) can be used in accessing model attributes, although their use is not required.

Two main modelling components were integrated in PALIO in this fashion: a user-modelling server, and a context-modelling server (see 3.2, "The PALIO System Architecture").

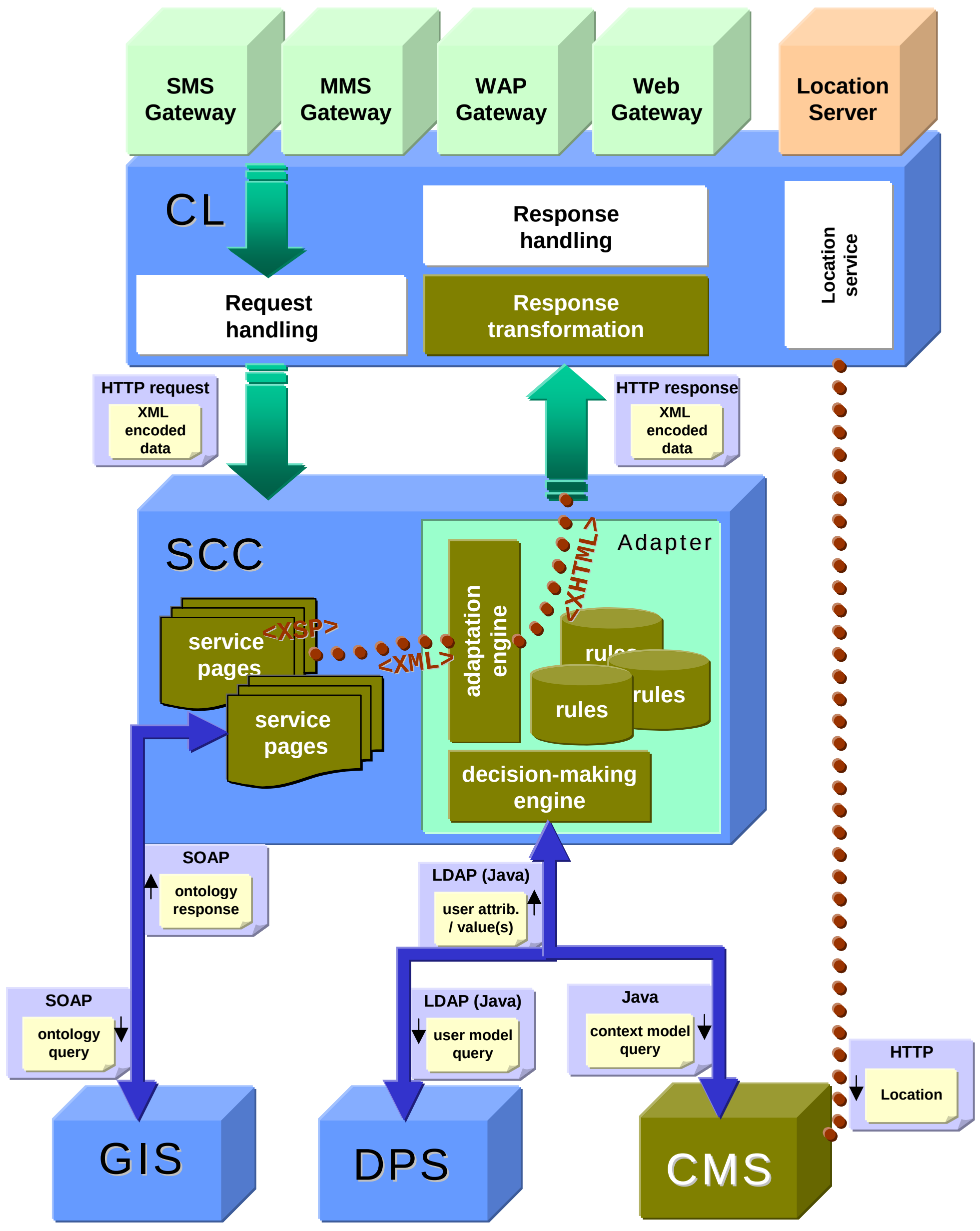
3 Applying the Framework in PALIO

3.1 The PALIO project

PALIO (see "Acknowledgements" section) was a European Commission funded research project that addressed the issue of Universal Access to community-wide services, based on content- and interface- level adaptation, beyond desktop access. The main challenge of the PALIO project was the creation of an open system for the unconstrained access and retrieval of information (i.e., not limited by space, time, access technology, etc.). One important aspect of the PALIO system was the support for a wide range of communication technologies (mobile or wired) to facilitate access to services. In summary, the most important characteristics of PALIO were: (a) integration of wireless and wired telecommunication technologies to offer services through both fixed terminals in public places and mobile personal terminals (e.g. mobile phones, PDAs, laptops); (b) location awareness to allow the dynamic modification of information presented (according to user position); (c) adaptation of the contents to automatically provide different presentations depending on user requirements, needs and preferences; (d) scalability of the information to different communication technologies and terminals; (e) interoperability between different service providers in both the envisaged wireless network and the Web.

3.2 The PALIO System Architecture

The main components and communication channels in the PALIO system are depicted in Figure 3 (Stephanidis et al., 2004). The *Service Control Centre* (SCC) is the central component of the PALIO system. It serves as the access point and the runtime platform for the system's information services. The SCC acts as a central server that supports multi-user access to integrated, primary information and services.



🟫 : **Modules built on top of the Cocoon framework.**

Figure 3: Components and communication channels in the PALIO framework. (Stephanidis et al., 2004)

The *User Communication Layer* (CL)[3] encapsulates the individual communication servers (Web gateway, WAP gateway, SMS gateway, etc.) and provides transparent communication independent of the server characteristics. This component unifies and abstracts the different communication protocols (*e.g.* WAP, HTTP) and terminal platforms (*e.g.* mobile phone, PC, Internet kiosk). To achieve this, the CL transforms incoming communication into a common "input" format. Symmetrically, it transforms information expressed in a common "output" format into one appropriate for transmission and presentation on the user's terminal. Additionally, it propagates information regarding the capabilities and characteristics of the access terminal into the PALIO system.

[3] The term "layer" is used in the PALIO project for historical reasons; the CL is not a layer in the sense of layered software architecture, but rather a component.

The *Generic Information Server* (GIS) integrates and manages existing information and services (which are distributed over the network). In this respect, the GIS acts as a two-way facilitator. Firstly, it assembles appropriate content and data models (in the form of an information ontology and its associated metadata), upon which it acts as a mediator for the retrieval of information and the utilisation of existing services by the SCC. Secondly, it communicates directly with the distributed servers that contain the respective data, or realise the services.

Adaptation support comes, apparently, from the framework described thus far in this chapter. The user modelling server used in PALIO was *humanIt's*[4] *Dynamic Personalisation Server* (DPS). The DPS maintains four models: a *user* model, a *usage* model, a *system* model, and a *service* model. In general, user models consist of a part dedicated to users' interests and preferences, as well as a demographic part. In PALIO, the principal part of a user model was devoted to representing users' interests and preferences. This part's structure was compliant with the information ontology, providing PALIO with a domain taxonomy. This domain taxonomy was mirrored in the DPS-hosted system model.

Usage context modelling in PALIO is undertaken by a purposely-developed *Context Modelling Server* (CMS). A usage context is defined to include all information relating to an interactive episode that is not directly related to an individual user. A context model may, therefore, contain information such as: characteristics of the access terminal, characteristics of the network connection, current date and time, etc. In addition to these, the PALIO CMS also maintained information about: (a) the user's current location, and (b) information related to *push* services that users subscribed to.

The PALIO system was implemented on top of the Cocoon[5] publishing framework. This was used as the ground platform in the implementation of the SCC, to generate information pages that were delivered to the users in a format supported by their terminal devices. The adaptation framework was used in PALIO to support several types of adaptation at the service level (Paramythis et al., 2003a). Examples include:

- Users are adaptively assisted by the system in retrieving information in accordance to their requirements and preferences, through: form simplification and pre-filling; query augmentation; filtering and sorting of query results; selection of "levels of detail"; selection of pieces of information to include in "pages"; etc.
- Users can receive recommendations on items that might be of interest to them, on the basis of their individual profiles / models, but also drawing from collective group experience and "opinions".
- User requests for information or services can be automatically augmented with location, through the employment of intuitive geographical concepts (e.g., "near").
- Personalised, location-based services are supported (e.g., recommending places that fall within the users' interests and are in their current vicinity).
- A wide variety of access devices and modalities are supported through the automatic adaptation of presentation and content to account for device capabilities and network characteristics, coupled with the framework's support for transforming to / from many formats and markup languages.

[4] http://www.humanit.de/

[5] Apache Cocoon is an XML publishing framework that facilitates the usage of XML and XSLT technologies for server applications. Designed around pipelined SAX processing, to benefit performance and scalability, Cocoon offers a flexible environment based on the separation of concerns between content, logic and style–the so-called *pyramid model of web contracts*. More information can be obtained from the project's homepage, at: http://cocoon.apache.org/.

The potential of the presented adaptation framework is perhaps better illustrated by the way in which the existing PALIO information systems address the issue of accessibility (Emiliani et al., 2003). Specifically, accessibility is addressed at two complementary levels: presentation and content. Presentation-oriented adaptations are targeted at ensuring that the interactive front-end of the services is accessible and usable by different categories of disabled people. In this respect, PALIO explicitly accounts for blindness, colour blindness, low vision, and motor impairments that may affect a user's interaction with the system. At the content level, services can automatically provide users with seldom sought after information that is, however, of particular relevance to their disability (e.g., the accessibility of a venue to wheelchair-bound persons). Such information is also utilised internally by the system to tailor the services themselves to individual user requirements.

4 Conclusions and Future Work

4.1 Lessons Learnt

The employment of the adaptation framework in PALIO has provided valuable feedback in at least three directions: (a) orthogonality of the adaptation framework to the underlying Web "serving" architecture; (b) authoring support for designing / defining adaptations, and (c) user reactions to the adaptive facilities of the system.

Regarding the first of the above aspects, namely orthogonality of the framework to the Web "serving" architecture, we found that our goal was largely achieved, as we were able to easily integrate with the Cocoon-based, service-oriented approach at the core of PALIO. The main challenge encountered was the propagation of monitoring information to the appropriate modelling servers. The difficulties involved were: (a) the need to synchronize between the application domain model (i.e., information ontology), the user interests' representation and the "evidence" sent by the system for monitoring; and, (b) the limited guidance from previous results in the literature as to how monitoring should approach potentially "mixed initiative" requests (i.e., requests that may have the explicit or implicit result of adaptations introduced in a document as part of a prior processing cycle).

The second of the above issues may require some additional explanation. Specifically, a question repeatedly encountered was how to differentiate between evidence resulting from direct user behaviour, versus evidence indirectly resulting from adaptations that were introduced by the system. Although differentiating between the two was possible, the framework did not provide adequate support for generalising the relevant behaviour, resulting in repetitive work on the part of the adaptation designers / authors.

Another category of findings concerned the level of authoring support required for designing / defining adaptations. In PALIO, adaptations were designed and implemented by a multidisciplinary group bringing together expertise in Human-Computer Interaction, Adaptation Theory, Web-based development, XML- and XSL- based document publishing, etc. Furthermore, the adaptation rules and actions were all written "by hand", using only the support offered by XML schema aware editors.

The synthesis of the team, as well as the shifting focus from design towards implementation, made apparent the need for multi-level design and authoring support tools for adaptation. Specifically, informal inquiries between team members identified the following as the activities most demanding of authoring support: (a) design of adaptation logic (whether rule-based or otherwise); (b) automated dependency checking between adaptation actions (i.e., whether modifying the sequence between actions modifies the end result and how); and, (c) authoring abstractions at the level of hypermedia techniques, in addition to the level of actions. Although authoring tools for adaptive systems exist today, they are, in most cases, domain-specific.

Finally, a few words are in order in terms of user feedback. The evaluation activities carried out in the context of PALIO did not employ an evaluation framework that can provide direct and explicit results in relation to adaptation (Paramythis et al., 2001). However, the structure of the evaluation, as well as the explicit usage issues that it set out to explore, made it possible to indirectly draw some adaptation-related conclusions. These, in summary, were that users (who were not aware of the adaptive facilities of the system): appreciated very much the system's capability to present disability-related information and do so in an accessible way; found that the system's recommendations were very relevant to their interests in most occasions; and, had a very positive attitude towards location-orientation and location-sensitivity in the provided tourist services (PALIO Consortium, 2003). Although these results do not suffice to judge the comparative merits of the adaptation framework with respect to alternative approaches, they are definitely strong indicators of the fact that useful adaptation behaviour can be implemented using it.

4.2 Future Directions

Starting from activities targeted on refining the framework itself, our current and planned work addresses the following topics:

- Investigation of additional adaptation actions as candidates for inclusion in the framework's default action set. Considered actions are at higher levels than the ones already supported. The main criterion in our work is the added value that each of these actions may bring to the implementation of typical Web-based adaptation scenarios.
- Experimentation with alternative approaches to modelling and decision-making. We are currently working on providing alternative decision making components as part of the default framework facilities, which will employ Bayesian networks, and tools from formal decision theory to enable some measure of adaptation "introspection" by the system.
- Implementation of monitoring support facilities. Specifically, we are currently designing a new component for the framework, which will assist authors in specifying what gets monitored, as well as when and how it does get monitored.
- Development of a range of consistency checking tools. These include most importantly tools for: verifying references to domain model entities, as these are implicitly referenced within other system models (e.g., the user model); and, simplifying and supporting the task of discovering dependencies between actions.

At a more general level, it is our goal that the framework spearheads a new perspective in the design and implementation of adaptive hypermedia systems, characterised by:

- The support for XML-based, declarative specification of adaptive behaviour, as opposed to more programmatic approaches.
- The concept of adaptation actions as a common and well-defined "vocabulary" in the specification of adaptive behaviour, alongside with the concept of synthesising higher-level adaptive behaviour from simpler building blocks.
- The separation of adaptation logic from the hypermedia elements to which it refers.
- The possibility to easily integrate and interchange alternative adaptation technologies and techniques.

Adoption of (parts of) the framework has the potential to "standardise" portions of the implementation of adaptive systems. Such standardisation would result in increased levels of reuse, both at the level of dedicated software components, and at the level of common adaptive behaviours across systems (for example, presentation-level adaptation that transform XHTML output to ensure accessibility by sight-impaired users).

Furthermore, we consider the framework to be a step in the direction of enabling the development of sophisticated authoring tools for adaptation, which will enable non-specialists to design and implement adaptive behaviours. Such authoring tools are currently encountered only in specialised domains (e.g., authoring of adaptive on-line courses, see (Brusilovsky, 2003)), while they are lacking at more general levels. They are, in our opinion, one of the most important prerequisites for bringing adaptation to the mainstream.

5 Acknowledgements

The IST-1999-20656 PALIO project (1999-2002) was partly funded by the Information Society Technologies Programme of the European Commission – DG Information Society. The partners in the PALIO consortium were: ASSIOMA S.p.A. (Italy) – Prime Contractor; CNR-IROE (Italy); Comune di Firenze (Italy); FORTH-ICS (Greece); GMD (Germany); Telecom Italia Mobile S.p.A. (Italy); University of Sienna (Italy); Comune di Siena (Italy); MA Systems and Control Ltd (UK); FORTHnet (Greece).

The authors would also like to explicitly acknowledge the contributions from the following members of the Human-Computer Interaction Laboratory of FORTH-ICS: Chrisoula Alexandraki, Ioannis Segkos and Napoleon Maou.

6 References

Ardissono, L., & Goy, A. (1999). Tailoring the interaction with users of electronic shops. In J. Kay (Ed.), Proceedings of the 7th International Conference on User Modeling (UM99), pp. 35-44. Wien: SpringerWienNewYork.

Balabanovic, M., & Shoham, Y. (1997). Fab: content-based collaborative recommendation. *Communications of the ACM,* 40(3), 66-72.

Benyon, D.R. (1997). Intelligent *Interface Technology to Improve Human-Computer Interaction* (Tutorial). In HCI International 1997, San Francisco, USA.

Brusilovsky, P. (1996) Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction,* 6(2-3), 87-129.

Brusilovsky, P., Kobsa, A. & Vassileva, J. (eds.) (1998). *Adaptive Hypertext and Hypermedia*. Dordrecht: Kluwer Academic Publishers.

Brusilovsky, P. (2001) Adaptive hypermedia. *User Modeling and User Adapted Interaction, Ten Year Anniversary Issue*, 11(1/2), 87-110.

Brusilovsky, P. (2003) Developing Adaptive Education Hypermedia Systems: From Design Models to Authoring Tools. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Learning Technologies*. NL: Kluwer Academic Publishers. [On-line]. Available at: http://www.sis.pitt.edu/~peterb/papers/KluwerAuthBook.pdf

Brusilovsky, P., & Nijhavan, H. (2002). A Framework for Adaptive E-Learning Based on Distributed Re-usable Learning Activities. In Proceedings of the World Conference on E-Learning in Corp., Govt., Health., & Higher Ed. 2002(1), 154-161. AACE. [On-line]. Available at: http://www.aace.org/dl/index.cfm/fuseaction/View/paperID/9357.

Campbell, R.H., Islam, N., Johnson, R., Kougiouris, P., & Madany, P. (1991). Choices, Frameworks and Refinement. In L.-F. Cabrera, V. Russo & M. Shapiro (Eds.), *Proceedings of the 1991 International Workshop on Object Orientation in Operating Systems*, Palo Alto, California, USA, pp. 9-15. IEEE Computer Society Press, Los Alamitos, CA, USA.

Cristea, A., & Calvi, L. (2003). The Three Layers of Adaptation Granularity. In P. Brusilovsky, A. Corbett & F. de Rosis (Eds.), Proceedings of the 9th International Conference on User Modeling, UM 2003, Johnstown, PA, USA, June 22-26, 2003 (pp. 4-14). Lecture Notes in Computer Science, Volume 2702 / 2003. Berlin Heidelberg NewYork: Springer-Verlag.

De Bra, P., Aerts, A., and Rousseau, B. (2002a). "Concept Relationship Types for AHA! 2.0", Proceedings of World Conference on E-Learning in Corp., Govt., Health, & Higher Ed. 2002(1), 1386-1389, [online], AACE, http://www.aace.org/dl/index.cfm/fuseaction/View/paperID/9568

De Bra, P., Aerts, A., Smits, D., and Stash, N., (2002b). "AHA! Version 2.0, More Adaptation Flexibility for Authors", Proceedings of the AACE ELearn'2002 conference, October 2002, pp. 240-246.Dieterich, H., Malinowski, U., Kühme, T., & Schneider-Hufschmidt, M. (1993). State of the Art in Adaptive User Interfaces. In M. Schneider-Hufschmidt, T. Kühme, T., & U. Malinowski (Eds.), *Adaptive User Interfaces: Principles and Practice*, pp. 13-48. The Netherlands: North-Holland.

De Bra, P., & Stash, N. (2002). AHA! A General-Purpose Tool for Adaptive Websites. World Wide Web Conference, Poster Session, May 2002.

Emiliani, P.L., Paramythis, A., Burzagli, L., & Stephanidis, C. (2003). PALIO as an enabling platform for disabled and elderly people. In *C. Stephanidis, (Ed.), Universal Access in HCI: Inclusive Design in the Information Society - Volume 4 of the Proceedings of the 10th International Conference on Human-Computer Interaction (HCI International 2003)*, Crete, Greece, 22-27 June (pp. 547 - 551). Mahwah, New Jersey: Lawrence Erlbaum Associates (ISBN: 0-8058-4933-5)

Henze, N. (2001). Open Adaptive Hypermedia: An approach to adaptive information presentation on the Web. In C. Stephanidis (Ed.), *Universal Access in HCI: Towards an Information Society for All – Volume 3 of the Proceedings of the 9th International Conference on Human-Computer Interaction (HCI International 2001),* New Orleans, Louisiana, USA, 5-10 August (pp. 818-821). Mahwah, New Jersey: Lawrence Erlbaum Associates (ISBN: 0-8058-3609-8).

Höök, K., Karlgren, J., Waern, A., Dahlbaeck, N., Jansson, C.-G., Karlgen, K., & Lemaire, B. (1996). A Glass Box Approach to Adaptive Hypermedia. *User Modeling and User-Adapted Interaction, Special Issue on Adaptive Hypermedia,* 6(2-3), 157-184.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998). The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *the Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, pp. 256-265. San Francisco: Morgan Kaufmann.

Jacobson, I., Griss, M., & Johnson, P. (1997). Making the Reuse Business Work. *IEEE Computer*, 10, 36-42.

Kay, J. (1995). The um toolkti for cooperative user modelling. *User Modelling and User Adapted Interaction*. 4(3), 149-196.

Kobsa, A., & Pohl, W. (1995). The user modelling shell system BGP-MS. In: *User Modelling and User-adapted Interaction*, 4(2), 59-106.

Kobsa, A. (2001). Generic user modeling systems. *User Modeling and User Adapted Interaction,* 11(1-2), 49-63.

Oppermann, R., & Specht, M. (1998). Adaptive Support for a Mobile Museum Guide. *In the Proceedings of the IMC '98 - Workshop "Interactive Application of Mobile Computing*, Rostock, Germany. [On-line]. Available at: http://www.rostock.igd.fhg.de/veranstaltungen/workshops/imc98/Proceedings/imc98-SessionMA3-2.pdf

PALIO Consortium (2003). *PALIO field-trial - Results of the final evaluation.* IST-1999-20656 PALIO Project Deliverable D24.

Paramythis, A., Totter, A., & Stephanidis, C. (2001). A modular approach to the evaluation of Adaptive User Interfaces. In S. Weibelzahl, D. Chin & G. Weber (Eds.), *Proceedings of the*

Workshop on Empirical Evaluations of Adaptive Systems, held in the context of the 8th International Conference on User Modeling (UM'2001), Sonthofen, Germany, 13-17 July (pp. 9-24). Freiburg: Pedagogical University of Freiburg.

Paramythis, A., Alexandraki, C., Segkos, I., Maou, N., & Stephanidis, C. (2003a). Personalisable, context-aware services: the PALIO approach. In C. Stephanidis, (Ed.), *Universal Access in HCI: Inclusive Design in the Information Society - Volume 4 of the Proceedings of the 10th International Conference on Human-Computer Interaction (HCI International 2003)*, Crete, Greece, 22-27 June (pp. 582 - 586). Mahwah, New Jersey: Lawrence Erlbaum Associates (ISBN: 0-8058-4933-5).

Paramythis, A., Alexandraki, C., Segkos, I., Maou, N., & Stephanidis, C. (2003b). The PALIO framework for hypermedia adaptations. In C. Stephanidis, (Ed.), *Universal Access in HCI: Inclusive Design in the Information Society - Volume 4 of the Proceedings of the 10th International Conference on Human-Computer Interaction (HCI International 2003)*, Crete, Greece, 22-27 June (pp. 587 - 591). Mahwah, New Jersey: Lawrence Erlbaum Associates (ISBN: 0-8058-4933-5).

Stephanidis, C., Paramythis, A., Sfyrakis, M., & Savidis, A. (2001) A Case Study in Unified User Interface Development: The AVANTI Web Browser. In C. Stephanidis (Ed.), *User Interfaces for All - Concepts, Methods, and Tools* (pp. 525-568). Mahwah, NJ: Lawrence Erlbaum Associates (ISBN 0-8058-2967-9, 760 pages).

Stephanidis, C., Paramythis, A., Zarikas, V., & Savidis, A. (2004). The PALIO Framework for Adaptive Information Services. In A. Seffah & H. Javahery (Eds.), Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces (pp. 69-92). Chichester, UK: John Wiley & Sons, Ltd.