The PALIO Framework for Adaptive Information Services

Constantine Stephanidis[1,2], Alexandros Paramythis[1], Vasilios Zarikas[1], Anthony Savidis[1]

[1]Institute of Computer Science
Foundation for Research and Technology-Hellas
Science & Technology Park of Crete
Heraklion, Crete, GR-71110, GREECE
Tel: +30-810-391741 - Fax: +30-810-391740, Email: cs@ics.forth.gr

[2]Department of Computer Science
University of Crete

Abstract

This chapter describes the PALIO (Personalised Access to Local Information and services for tourists) service framework, focusing on its extensive support for service adaptation. The PALIO framework is currently used in the development of location-aware information systems for tourists, and is capable of delivering fully adaptive information to a wide range of devices, including mobile ones. Its open and expandable architecture can integrate a variety of pre-existing and forthcoming services, and retrieve information from a collection of different databases. The framework supports dynamic adaptation both in the content and the presentation of information, according to user- and interaction context-characteristics. This chapter discusses how the aforementioned adaptation capabilities can support the creation of Multiple User Interfaces (MUIs) on the Web.

Keywords: service framework, adaptivity, Web-based information systems, context-based adaptation, multi-device user interfaces

1 Introduction

In recent years, the concept of adaptation has been investigated under the perspective of providing built-in accessibility and high interaction quality in applications and services in the emerging information society (Stephanidis, 2001a; Stephanidis, 2001b). 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).

In the context of this chapter, adaptation concerns the interactive behaviour of the User Interface (UI), as well as the content of applications and services. Adaptation implies the capability, on the part of the system, of capturing and representing 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 (Savidis and Stephanidis, 2001).

The PALIO project[1] addresses the issue of universal access to community-wide services, based on content and UI adaptation beyond desktop access. The main challenge of the PALIO project is the creation of an open system for the unconstrained access and retrieval of information (*i.e.* not limited by space, time, access technology, etc.). Under this scenario, mobile communication systems play an essential role, because they enable access to services from anywhere and at anytime. One important aspect of the PALIO system is the support for a wide range of communication technologies (mobile or wired) to facilitate access to services.

The PALIO project is mainly based on the results of three research projects that have preceded it: TIDE-ACCESS, ACTS-AVANTI, and ESPRIT-HIPS. In all of these projects, a primary research target has been the support for alternative incarnations of the interactive part of applications and services, according to user- and usage context- characteristics. As such, these projects are directly related to the concept of Multiple User Interfaces (MUIs), and have addressed several related aspects from both a methodological and an implementation point of view.

The ACCESS project[2] developed new technological solutions for supporting the concept of *User Interfaces for all* (*i.e.* universal accessibility of computer-based applications). It facilitated the development of UIs adaptable to individual user abilities, skills, requirements, and preferences. The project addressed the need for innovation in this field and proposed a new development methodology called Unified User Interface development. The project also proposed a set of tools enabling designers to deal with problems encountered while attempting to access technology in a consistent, systematic and unified manner (Stephanidis, 2001c, Savidis and Stephanidis, 2001).

The AVANTI project[3] addressed the interaction requirements of disabled individuals who were using Web-based multimedia applications and services. One of the main objectives of the work undertaken was the design and development of a UI that would provide equitable access and quality in use to all potential end users, including disabled and elderly people. This was achieved by employing adaptability and adaptivity techniques at both the content and the UI levels. A unique characteristic of the AVANTI project was that it addressed adaptation both at the client-side UI, through a dedicated, adaptive Web browser, and at the server side, through presentation and content adaptation (Stephanidis et al., 2001; Fink, Kobsa and Nill, 1998).

The HIPS project[4] was aimed at developing new interaction paradigms for navigating physical spaces. The objective of the project was to enrich a user's experience of a city by overlapping the

[1] Personalised Access to Local Information and services for tOurists (see Acknowledgments section).

[2] TIDE TP1001 - ACCESS "Development platform for unified ACCESS to enabling environments", funded by the European Commission (1994-1996). The partners of the ACCESS consortium are: CNR-IROE (Italy) - Prime Contractor; FORTH-ICS (Greece); University of Hertforshire (UK); University of Athens (Greece); NAWH (Finland); VTT (Finland); Hereward College (UK); RNIB (UK); Seleco (Italy); MA Systems & Control (UK); PIKOMED (Finland).

[3] ACTS - AC042 AVANTI "Adaptable and Adaptive Interaction in Multimedia Telecommunications Applications", funded by the European Commission (1995-1998). The partners of the ACTS-AVANTI consortium are: ALCATEL Italia, Siette division (Italy) - Prime Contractor; CNR-IROE (Italy); FORTH-ICS (Greece); GMD (Germany), VTT (Finland); University of Sienna (Italy), TECO Systems (Italy); STUDIO ADR (Italy); MA Systems and Control (UK).

[4] ESPRIT-25574 HIPS "Hyper-Interaction within Physical Spaces" funded by the European Commission (1997-2000). The partners of the HIPS consortium are: Universita degli Studi di Siena (Italy) - Prime Contractor; University of Edinburgh (UK); Alcatel Italia SpA (Italy); Istituto Trentino di Cultura (Italy); Cara Broadbent &

physical space with contextual and personalized information on the human environment. HIPS developed ALIAS (Adaptive Location aware Information Assistance for nomadic activitieS), a new interaction paradigm for navigation. ALIAS allowed people to simultaneously navigate both physical and related information space. The gap between the two was minimised by delivering contextualised and personalized information on the human environment through a multimodal presentation, according to the user's movements. ALIAS was implemented as a telecommunication system that took advantage of an extensive electronic database containing information about the particular place. Users interacted with the system using mobile palmtops with wireless connections or computers with wired connections. (Oppermann and Specht, 1998).

PALIO, building on the results of these projects, proposes a new software framework that supports the provision of tourist services in an integrated, open structure. It is capable of providing information from local databases in a personalised way. This framework is based on the concurrent adoption of the following concepts: (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 World Wide Web.

The framework presented above exhibits several features that bear direct relevance to the concept of MUIs. Specifically, the framework supports adaptation not only on the basis of user characteristics and interests, but also on the basis of the interaction context. The latter includes (amongst other things) the capabilities and features of the access terminals, and the user's current location. On this basis, the PALIO framework is capable of adapting the content and presentation of services for use on a wide range of devices, with particular emphasis on nomadic interaction from wireless network devices.

The rest of this chapter is structured as follows: Section 2 provides an overview of the PALIO system architecture and its adaptation infrastructure. Section 3 discusses the PALIO system under the Adaptive Hypermedia Systems perspective. Section 4 goes into more depth on those characteristics of the framework that are of particular interest regarding MUIs, and presents a brief example of the framework in operation. The chapter concludes with a summary and a brief overview of ongoing work.

2 The PALIO System Architecture

2.1 Overview

The *Augmented Virtual City Centre* (AVC) constitutes the core of the PALIO system. Users perceive the AVC as a system that groups together all information and services available in the

---

Jegher Associes (France); GMD (Germany); SINTEF (Norway). The project's website is: http://www.media.unisi.it/hips

city. It serves as an augmented, virtual facilitation point from which different types of information and services can be accessed. Context- and location- awareness, as well as the adaptation capabilities of the AVC, enable users to experience their interaction with services as *contextually grounded* dialogue. For example, the system always knows the user's location and can correctly infer what is near the user, without the user having to explicitly provide related information.

The main building blocks of the AVC are depicted in Figure 1, and can be broadly categorised as follows:

- 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 is the framework upon which other services are built. It provides the generic building blocks required to compose services. Examples include the maintenance of the service state control, the creation of basic information retrieval mechanisms (through which service-specific modules can communicate with, and retrieve information from, various distributed information sources/servers in PALIO), etc. Seen from a different perspective, the SCC acts as a central server that supports multi-user access to integrated, primary information and services; appropriately adapted to the user, the context of use, the access terminal and the telecommunications infrastructure.

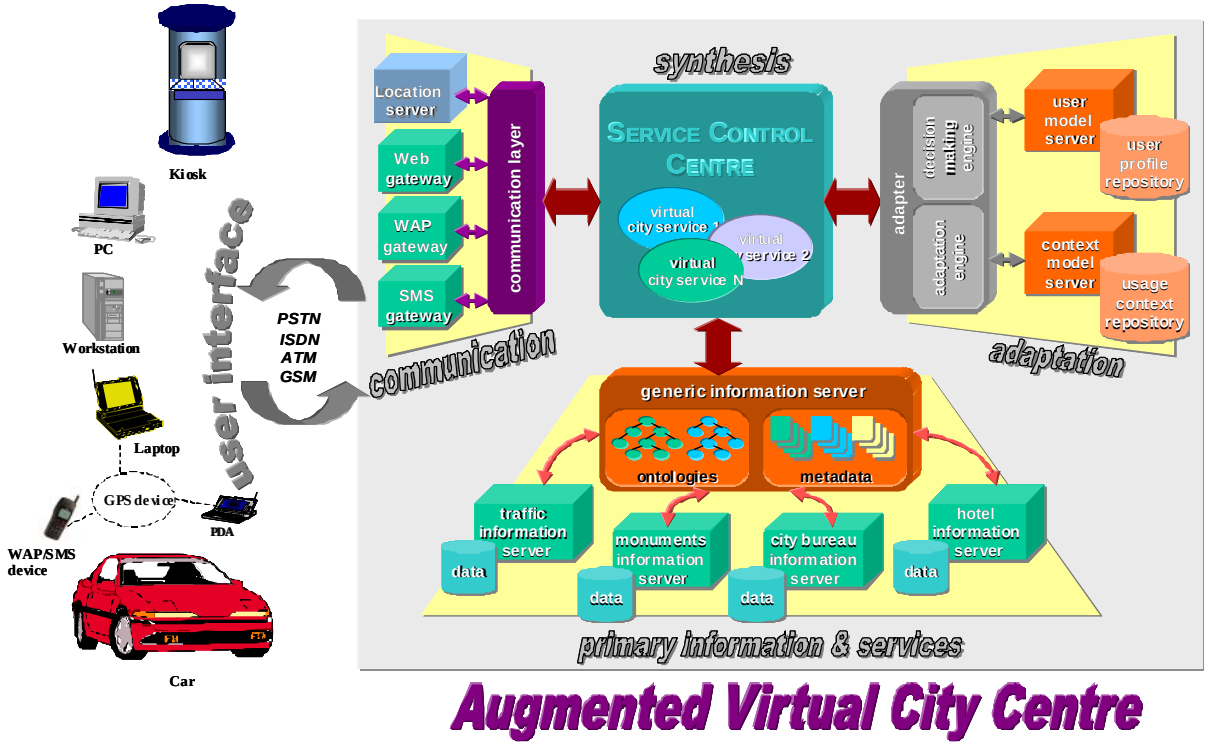

Figure 1: Overall architecture of the PALIO system.

- The **User Communication Layer** (CL)[5] 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). Specifically, the CL transforms incoming communication from the user into a common format, so that the rest of the system does not need to handle the peculiarities of the underlying communication networks and protocols. Symmetrically, the CL transforms information expressed in the aforementioned common format into a format appropriate for transmission and presentation on the user's terminal. In addition to the above, information regarding the capabilities and characteristics of the access terminal propagates across the PALIO system. This information is used to adapt the content and presentation of data transmitted to the user, so that it is appropriate for the user's terminal (*e.g.* in terms of media, modalities and bandwidth consumption).
- The **Generic Information Server** (IS) integrates and manages existing information and services (which are distributed over the network). In this respect, the IS 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 Service Control Centre. Secondly, it communicates directly with the distributed servers that contain the respective data, or realise the services. The existing information and services that are being used in PALIO are termed *primary*, in the sense that they already exist and constitute the building

[5] 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.

blocks for the PALIO services. The PALIO (virtual city) services, on the other hand, are synthesised on top of the primary ones and reside within the SCC.

- The **Adaptation Infrastructure** is responsible for content- and interface- adaptation in the PALIO System. Its major building blocks are the Adapter, the User Model Server and the Context Model Server. These are described in more detail in the next section.

From the user's point of view, a PALIO service is an application that can get information on an area of interest. From the developer's point of view, a PALIO service is a collection of dynamically generated and static template files, expressed in an XML-based, device-independent language, which are used to generate information pages.

A PALIO service is implemented using eXtensible Server Pages (XSPs). XSPs are template files written using an XML-based language and processed by the Cocoon[6] publishing framework (used as the ground platform in the implementation of the SCC) to generate information pages that are delivered to the users in a format supported by their terminal devices. If, for example, a user is using an HTML browser, then the information pages are delivered to that user as HTML pages, while if the user is using WAP, then the same information is delivered as WML stacks.

A PALIO XSP page may consist of: (a) static content expressed in XHTML, an XML-compatible version of HTML 4.01; (b) custom XML used to generate dynamic content, including data retrieval queries needed to generate dynamic IS content; (c) custom XML tags used to specify which parts of the generated information page should be adapted for a particular user.

In brief, services in PALIO are collections of:

- Pages containing: static content expressed in XHTML, dynamic content expressed in the PALIO content language, information retrieval queries expressed in the PALIO query and ontology languages, and embedded adaptation rules.
- External files containing adaptation logic and actions (including files that express arbitrary document transformations in XSLT format).
- Configuration files specifying the mappings between adaptation logic and service pages.
- Other service configuration files, including the *site map* (a term used by Cocoon to refer to mappings between request patterns and actual service pages).

An alternative view of the PALIO architecture is presented in Figure 2. Therein, one can better observe the interconnections of the various components of the framework, as well as their communication protocols. Furthermore, the figure shows the relation between the Cocoon and PALIO frameworks.

[6] 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 at the project's homepage, at: http://xml.apache.org/cocoon/index.html.

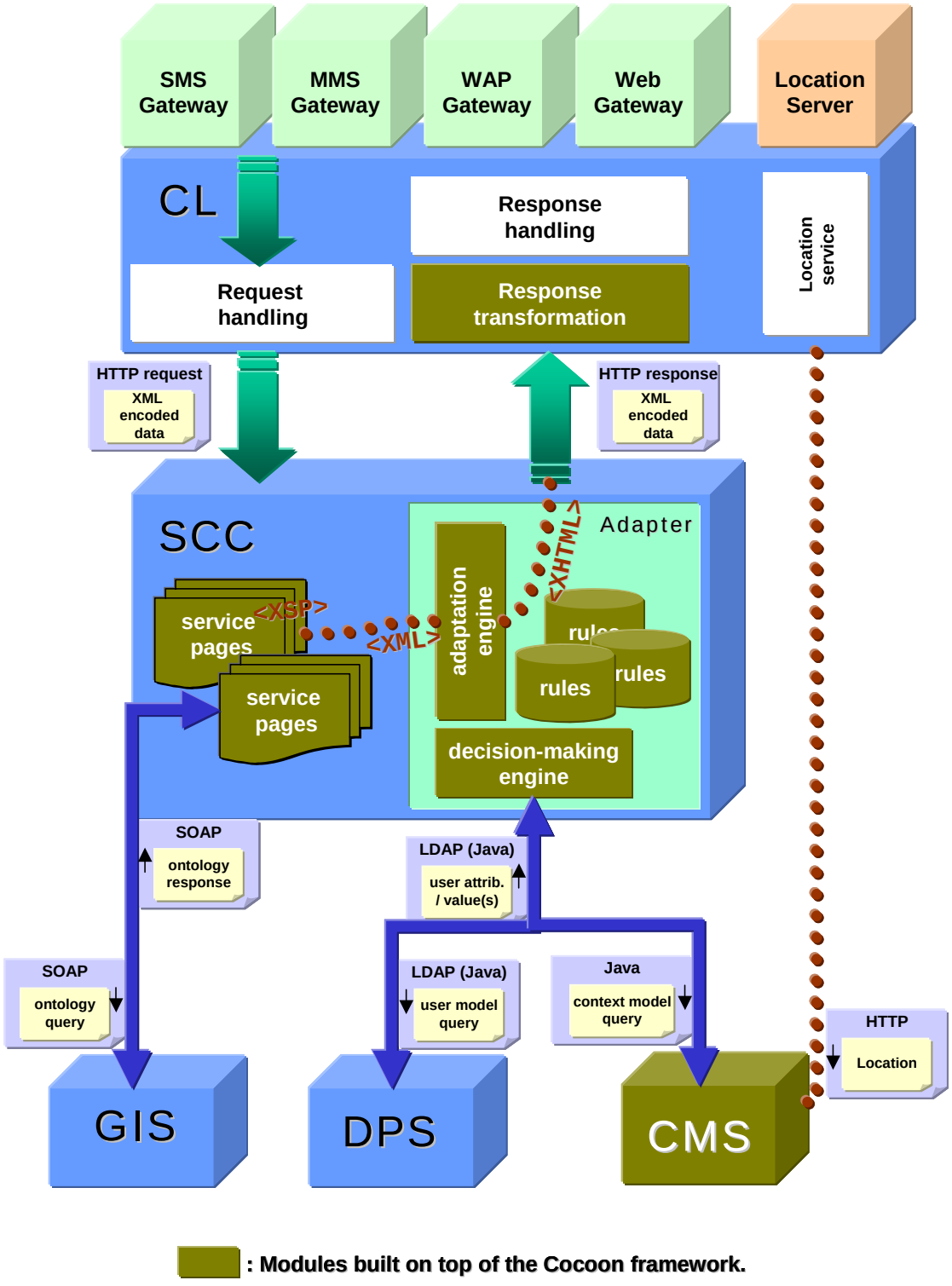

Figure 2: Components and communication protocols in the PALIO framework.

2.2 The PALIO Adaptation Infrastructure

In the PALIO system, the Adaptation Infrastructure is responsible for content and presentation adaptation. As already mentioned, its major components include the User Modelling Server, Context Modelling Server, and Adapter.

In PALIO, user modelling is carried out by *humanIt's*[7] **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's current version of the DPS, the principal part of a user model is devoted to representing users' interests and preferences. This part's structure is compliant with the information ontology, providing PALIO with a domain taxonomy. This domain taxonomy is mirrored in the DPS-hosted system model (see below).

User models also incorporate information derived from group modelling, by providing the following distinct probability estimates: *individual probability*, an assumption about a user's interests, derived solely from the user's interaction history (including information explicitly provided by the user); *predicted probability*, a prediction about a user's interests based on a set of similar users, which is dynamically computed according to known and inferred user characteristics, preferences, etc.; and *normalised probability*, which compares an individual's interests with those of the whole user population.

The DPS *usage* model is a persistent storage space for all of the DPS's usage-related data. It is comprised of interaction data communicated by the PALIO SCC (which monitors user activity within individual services) and information related to processing these data in user modelling components. These data are subsequently used to infer users' interests in specific items in PALIO's information ontology and/or domain taxonomy.

The system model encompasses information about the domain that is relevant for all user-modelling components of the DPS. The most important example of system model contents is the domain taxonomy.

In contrast, the *service* model contains information that is relevant for single components only. Specifically, the service model contains information that is required for establishing communication between the DPS core and its user-modelling components.

The **Context Modelling Server** (CMS), as its name suggests, maintains information about the usage context. A usage context is defined, in PALIO, to include all information relating to an interactive episode that is not directly related to an individual user. PALIO follows the definition of context given in (Dey & Abowd, 2000), but diverges in that users engaged in direct interaction with a system are considered (and modelled) separately from other dimensions of context. Along these lines, a context model may contain information such as: characteristics of the access terminal (including capabilities, supported mark-up language, etc.), characteristics of the network connection, current date and time, etc.

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

In addition, the CMS also maintains information about: (a) the user's current location (which is communicated to the CMS by the Location Server, in the case of GSM-based localisation, or the access device itself, in the case of GPS), and (b) information related to *push* services that users are subscribed to. It should be noted that in order to collect information about the current context of use, the CMS communicates, directly or indirectly, with several other components of the PALIO system. These other components are the *primary carriers* of the information. These first-level data collected by the CMS then undergo further analysis, with the intention of identifying and characterising the current context of use. Like the DPS, the CMS responds to queries made by the Adapter regarding the context, and relays notifications to the Adapter about important modifications (which may trigger specific adaptations) to the current context of use.

One of the innovative characteristics of the PALIO CMS is its ability to make context information available at different levels of abstraction. For example, the current time is available in a fully qualified form, but also as a day period constant (*e.g.* morning); the target device can be described in general terms (*e.g.* for a simple WAP terminal: tiny screen device, graphics not supported, links supported), etc. These abstraction capabilities also characterise aspects of the usage context that relate to the user's location. For instance, it is possible to identify the user's current location by its geographical longitude and latitude, but also through the type of building the user may be in (*e.g.* a museum), the characteristics of the environment (*e.g.* noisy), and so on. The adaptation logic (that is based on these usage context abstractions) has the advantage that it is general enough to be applied in several related contexts. This makes it possible to define adaptation logic that addresses specific, semantically unambiguous characteristics of the usage context, in addition to addressing the context as a whole. Section 4.1 below discusses this issue in more detail.

The third major component of the adaptation infrastructure is the **Adapter**, which is the basic adaptation component of the system. It integrates information concerning the user, the context of use, the access environment and the interaction history, and adapts the information content and presentation accordingly. Adaptations are performed on the basis of the following parameters: user interests (when available in the DPS or established from the ongoing interaction), user characteristics (when available in the DPS), user behaviour during interaction (provided by the DPS, or derived from ongoing interaction), type of telecommunication technology and terminal (provided by the CMS), location of the user in the city (provided by the CMS), etc.

The Adapter is comprised of two main modules, the **Decision Making Engine** (DME) and the **Adaptation Engine** (AE). The DME is responsible for deciding upon the need for adaptations, based on: (a) the information available about the user, the context of use, the access terminal, etc., and (b) a knowledge base that interrelates this information with adaptations (*i.e.* the adaptation logic). Combining the two, the DME makes decisions about the most appropriate adaptation for any particular setting and user/technology combination addressed by the project.

The AE instantiates the decisions communicated to it by the DME. The DMS and AE are kept as two distinct entities in order to decouple the adaptation decision logic from the adaptation implementation. In our view, this approach allows for a high level of flexibility. New types of adaptations can be introduced into the system very easily. At the same time, the rationale for arriving at an adaptation decision and the functional steps required to carry it out can be expressed and modified separately.

3 PALIO as an Adaptive Hypermedia System

Adaptive Hypermedia Systems (AH systems, or AHSs) are a relatively new area of research in adaptive systems. They have drawn considerable attention since the advent of the World Wide Web (which is an easily accessible, widely deployed hypermedia system). Today, there exist numerous AH systems in a variety of application domains, with a great variety of capabilities (see Ardissono and Goy, 1999; Balabanovic and Shoham, 1997; Brusilovsky et al., 1998; Henze, 2001; Schwab et al., 2000; Kobsa, 2001; etc.). While a full review of all AH systems related to PALIO is beyond the scope of this chapter, this section will discuss the most important characteristics of PALIO and attempt to position it within the general space of AHSs.

Major categories of AH systems include: educational hypermedia, on-line information systems, on-line help systems, information retrieval systems, and institutional hypermedia. A closer inspection of the main categories gives rise to further classification, based on the goal or scope of the system's adaptation, the methods used to achieve that goal, etc.[9] These are some of the sub-categories:

- On-line information systems can be further categorised into: classic on-line information systems, electronic encyclopaedias, information kiosks, virtual museums, handheld guides, e-commerce systems, and performance support systems.
- Information retrieval (IR) systems can be categorised into: search-oriented adaptive IR hypermedia systems (classic IR in Web context, with search filters), browsing-oriented adaptive IR hypermedia systems (adaptive guidance, adaptive annotation, adaptive recommendation), adaptive bookmark systems (systems for managing personalised views), and information services (search services, filtering services).

The PALIO framework is not constrained to any one of the above categories of AH systems. Being a comprehensive adaptation framework, PALIO can be used to develop AH services and systems in several of these categories. For instance, the PALIO framework could be employed in the development of an adaptive educational system equally well as in the development of an on-line help system. However, PALIO is more targeted towards facilitating the development of on-line information systems (and subcategories therein), and would require significant extensions to be rendered suitable for implementing *all* aspects of adaptive IR systems.

The rest of this section provides a brief overview of the adaptation tools available in PALIO. The discussion will be split into three parts. The first part will discuss the type of information that is taken into account when adaptations are decided upon. The second part will discuss the facilities available to expressing decisions in a computable form. Finally, the third part will address the types of adaptation actions that can be effected by the system.

[9] The detailed description of the different types of AHS is beyond the scope of this chapter. Interested readers are referred to the excellent reviews, (Brusilovsky, 1996) and (Brusilovsky, 2001), which have long served as reference points in the area.

3.1 Adaptation Determinants

The term *adaptation determinant* refers to any piece of information that is explicitly represented in the system, and which can serve as input for adaptation logic. Seen from a different perspective, adaptation determinants are the facts known by the system, which can be used to decide upon the need for, and the appropriate type of, adaptation at any point in time. A partial taxonomy of typical adaptation determinants is depicted in Diagram 1 (adapted from (Brusilovsky, 1996)). Determinants which appear, in the diagram, in normal text are modelled in PALIO, while those that appear in *italics* are not currently supported. In general, three main categories of determinants can be identified: information about users themselves; information about groups to which users may belong; and, information about the environment of execution[10].

An important fact that is already evident from the taxonomy is that information about the group to which a user belongs can be used alongside (or, even better, in combination with) information about the individual user. This approach is actively pursued in PALIO, where, for instance, the interest of a particular person in a specific type of information can be inferred (with some degree of probability), from the system's knowledge about the related interests of other members in the user's group. Naturally, this approach cannot be extended to all user-related information: user traits (affecting a person's personality, cognitive aptitudes, learning style, etc.), for instance, cannot be inferred from group modelling.

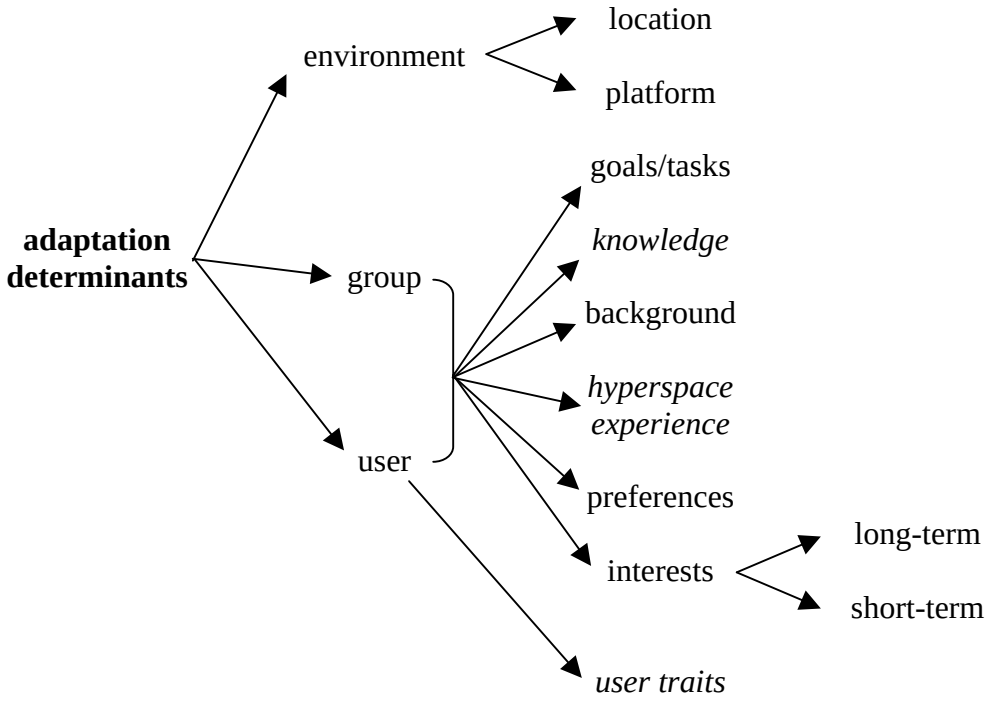

Diagram 1: Adaptation determinants.

[10] The term *usage context* (or *context of use*, often abbreviated to *context*) is used throughout this chapter to denote all factors characterising an interactive situation that are external to the user (including the user's geographical location). This encompasses the space of factors denoted by *environment* in the taxonomy.

A second important point concerns the fact that knowledge about the user is not always sufficient to identify user needs. One should also consider the more general context in which the user interacts (including the user's location, the characteristics of the access terminal/device, etc.). In fact, with respect to context-related factors, users with varying characteristics can be expected to have similar or identical needs. As already discussed, the PALIO framework takes full account of this fact, and incorporates a Context Modelling Component, which undertakes the monitoring of observable context-related parameters. These are available to the adaptation designer to be used as explicit adaptation determinants.

3.2 Decisions on the Basis of Adaptation Determinants

The primary decision engine implemented for PALIO is rule-based. Although first order logic rule engines can easily be plugged into the framework (the latter having been specifically designed to allow for that), it was decided that, to facilitate the wide adoption of the framework, a simpler and more accessible approach was in order. Along these lines, a new rule language was created, borrowing from control structures that were commonly supported in functional programming languages. The premise was that such structures were much more familiar to designers of adaptive systems, while at the same time, they afforded lower degrees of complexity when it came to understanding the interrelations and dependencies between distinct pieces of adaptation logic. This approach was first applied in the AVANTI project with very good results (Stephanidis et al., 2001).

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 system. Adaptation rules expressed in such a rule language may be defined either in external files or embedded in the document to be adapted. Rules embedded in pages are constrained in that they can only be applied within the specific document in which they reside, and therefore are not reusable.

Rules external to documents 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, by referring to it via the name of the enclosing rule-set. The possible values of the scope attribute are *global*, *service* and *local*. These denote that the rules defined in a rule-set may apply to all of the pages served by PALIO, to pages belonging to a specific service, or to specific pages indicated through configuration files, respectively. The use of names and scopes is complementary to the use of the service configuration files, and is intended to add more flexibility to the definition of relationships between rule-sets and service documents.

Every rule, whether internal or external to the document, has the following attributes: (a) the rule *name*, which is an identifier for the rule; (b) the rule *class*, which is optional and may be used as an alternative way for grouping rules; (c) the rule *stage*, which defines whether the rule should be applied during the first, or the second adaptation phase; and (d) the rule *priority*, which provides a partial ordering scheme for rule evaluation and application and may take the values of *high*, *medium* or *low*. The stage property defines whether the adaptation rule should be applied

before or after querying the IS (if this is required to process the document). Adaptations performed during the first phase (before querying the IS) are either unconcerned with IS-ontology data, or adapt (*i.e.* apply modifications to) IS queries. Rules applied during the second phase (after querying the IS) *are* concerned with IS-ontology data.

The framework currently supports three types of rules: *if-then-else* rules, *switch-case-default* rules, and *prologue-actions-epilogue* rules. If-then-else rules are the simplest type of conditional rules; they bind sets of adaptation actions with the truth-value of a conditional expression. Following the example made by functional languages, an if-then-else rule in PALIO is composed of a *condition* (containing an expression based on adaptation determinants), a *then* part (containing the adaptation actions to be taken whenever the condition is satisfied), and an optional *else* part (containing the actions to be taken when the condition fails).

Switch-case-default rules can be used to relate multiple values (outcomes of run-time expression evaluation) to sets of adaptation actions. In this case, adaptation actions are executed if the value of a *variant* (expression) is equal to a *value*, or within the *range* of values, specified as a selection *case*. The switch-case-default rule construct supports the definition of multiple cases and requires the additional definition of an (optionally empty) set of *default* adaptation actions to be performed if none of the defined cases apply.

Prologue-actions-epilogue rules are intended mainly for the definition of unconditional rules. In other words, the specific rule construct is provided to support the definition of (sets of) actions to be performed at a particular stage in the sequence of adaptations. Although the construct can be simulated with an if-then-else construct (where the condition is always true), the prologue-actions-epilogue structure (through *action-sets*) provides an explicit separation between preparatory actions, the adaptations themselves, and clean-up actions. This separation allows for better rule structuring and improves the maintainability of the rule definition. A very common use of the *prologue* and *epilogue* parts is the creation/assignment and retraction of variables that are used by the adaptation actions (*e.g.* in order to determine the point of application of an action).

The conditional parts of first two rule constructs presented above, as well as the definition of variants and variant ranges in the case of the switch-case-default construct, are composed of expressions. These consist, as one would expect, of operands and operators applied on the operands. The main supported operand types in PALIO are: *string*, *number*, *boolean*, *date*, *location* and *null*. The *string* type is used for character sequences. The *number* type is used for all kinds of numerals including integers, floating point numbers, etc. The *boolean* type is used to express truth values and can only assume the values *true* or *false*. The *date* type is used for absolute, relative and recurring dates and temporal ranges. The *location* type is used to express absolute and relative geographical locations. Finally, the *null* type is a utility type with restricted use; authors may use it to determine whether a variable exists in the scope of a particular data source, and whether its type has been set. The operators supported by the PALIO framework can be classified into the following main categories: comparison operators (>, >=, <, <=, <>), mathematical operators (+, -, *, /, %), logical operators (*and*, *or*, *not*), string operators (*concatenate*, *contains*, and *substring*), date-specific operators (*get-year*, *get-month*, etc.), and location-specific operators (*near* and *distance*).

3.3 Adaptation Actions

The PALIO framework supports the following adaptation action categories: inserting document fragments, removing document fragments, replacing document elements/fragments, sorting document fragments, setting and removing element attributes, selecting among alternative document fragments, and applying arbitrary document transformations expressed in XSLT. In addition to the above basic adaptation actions, the PALIO framework also supports the manipulation of variables (*i.e.* storing and retrieving values from different run-time scopes).

Adaptation actions can be (and usually are) defined in external files, but can also be embedded within documents. The *target* or *reference* element of an adaptation action, which can also be a dummy element acting as a placeholder for the action's target, needs to be identified. When the adaptation action is specified externally, such an identification can be done using: (a) the element's *class* attribute, which is better suited for the identification of multiple insertion points in a document, as several elements can share the same *class* attribute; (b) the element's *id* attribute, which is better suited for the identification of single reference points in the document– the *id* attribute of any element should be unique within a document; (c) an XPath expression, to be used whenever more complex constraints need to be expressed with respect to the insertion point. When the adaptation action is embedded in a document, the default reference element is the rule (context) node itself.

In the following descriptions, constraints related to maintaining XML document validity are omitted for brevity, but should be intuitively understood in most cases (*e.g.* when adding an element as a parent to another element, or changing the value of an attribute).

Inserting Document Elements/Fragments

This adaptation action enables authors to insert new elements or document fragments into the document. For the insertion to be performed, the author needs to specify the reference element (or elements) at which the insertion will take place, as well as the relative position of the new element/fragment with respect to the reference element. The relative position of the inserted element/fragment with respect to reference element can be: (a) immediately *before* the reference element; (b) immediately *after* the reference element; (c) *inside* the reference element, as its *first child*; (d) *inside* the reference element, as its *last child*; (e) *outside* the reference element, as its *parent*.

Removing Document Elements/Fragments

This adaptation action enables authors to remove individual or multiple elements, or document fragments contained within elements, from the document. For the removal to be performed, the author needs to specify the element (or elements) to be removed, as well as whether any child elements should also be removed. The identification of the element to be removed can be done using the same approach as in the case of insertion, with the following important exception: when the action is part of a rule embedded in a document, authors cannot make use of the "current context node" idiom, because then the removal would affect the rule node itself, which is impossible. When removing an element, authors have the option to *remove* or *retain* the element's children.

Replacing Document Elements/Fragments

This adaptation action enables authors to replace single or multiple elements in the document, or document fragments contained within elements. The element(s) to be removed, along with any child elements that should also be removed, and the element(s) that are to be inserted instead, need to be specified by the author. This action can be treated, from a semantic point of view, as a removal action combined with a subsequent insertion action. This action improves on the aforementioned combination by making it possible to easily transfer the child elements of the element(s) being removed to the element(s) being inserted.

Manipulating Element Attributes

This adaptation action enables authors to set the value of specific element attributes, or remove element attributes entirely. To set an attribute value, the author needs to specify the element(s) affected, as well as the name of the attribute to be set and the value that it will assume. To remove an element attribute entirely, the author only needs to specify the element(s) affected, and the name of the attribute to be removed. When setting an attribute value, authors can use either static values, or (more typically) dynamically calculated/retrieved values (relating to the current user/session, context of use, etc.).

Selecting Among Alternative Document Elements/Fragments

This adaptation action facilitates the selection of fragment(s) among alternative fragments declared within the document. The use and selection of alternative content fragments requires authors to first introduce, and identify, the alternative (but not necessarily mutually exclusive) fragments into the document. Second, the author must select one or more of them through the respective adaptation actions. All alternatives that are not explicitly selected are automatically removed from the document. The selection of alternatives can take place: (a) by selecting exactly one of the alternatives and discarding the rest; or (b) by selecting zero or more of the alternatives and discarding the rest. The first case is a special case of the second. However, it is provided as a separate adaptation action in PALIO for reasons of authoring convenience.

Sorting Document Elements/Fragments

This adaptation action facilitates a common technique (that is difficult to define and implement with basic building blocks) in Adaptive Hypermedia Systems: the sorting of document fragments, links, etc. As in the case of the definition and selection of alternatives, sorting adaptations require two steps of specification on the part of the author. First, the author provides a list of document elements/fragments that are to be sorted within the document. Second, a respective sorting adaptation action needs to be authored and introduced into a rule (typically external to the document).

The specification of elements/fragments adheres to the following principles: there exists a generic container termed *list*, which holds the items (elements/fragments) to be sorted (note that the container has been called a *list* to benefit from author familiarity with the container concept and not because the container provides any facilities for traversing its contents); the container holds *items*, each of which must provide a *name* attribute (used to relate the item, directly or indirectly, to a value that will be used subsequently for sorting); the container may also hold additional supporting content that accompanies the items to be sorted.

The specification of each of the items to be sorted is accomplished through the item element and requires only the identification of a name for the item, which will be used during sorting to associate the item with a concrete value. The specification of sorting behaviour includes two additional optional components: the identification of the maximum number of items to be retained, and the identification of a threshold value that acts as a cut-off point for items. The sorting order to be applied to the items is defined as an attribute of the sort adaptation action.

Applying Arbitrary XSL Transformations

This type of adaptation action is intended to cover any complex cases that cannot be addressed through the use of the basic adaptation actions listed above. Specifically, this adaptation only defines that an external XSLT file is to be applied to the document, providing the URL of that file as an attribute. The PALIO adaptation framework then loads the file and its application to the (in-memory) representation of the document. The current implementation of the framework contains a number of ready-made XSLT files, which address common non-trivial adaptations, such as the transformation of nested lists into tables and vice versa.

4 PALIO in the Context of MUIs

4.1 PALIO as a WebUI

Following the definitions found in Chapter 1 of this book, “Multiple User Interfaces: Definitions, Challenges and Research Opportunities”, PALIO would be classified as a framework for building WebUIs. This means that the framework is specifically intended to support the development of on-line, server-side information services characterised by their capability to dynamically adapt on the basis of user- and context-characteristics. PALIO goes far beyond *transcoding* approaches (Maglio & Barrett, 2000) to transform content and interface for presentation on different devices. In fact, this type of transformation is the last (and, in some

ways, the least significant) adaptation step that takes place before documents are served to a user. The rest of this section is devoted to pointing out and discussing the capabilities of the PALIO framework in the context of MUIs and their associated principles.

4.1.1 Device- and Platform-Independence

One of the highly innovative characteristics of PALIO is the extensive support it offers for creating applications and services that are truly independent from the target device and computing platform. As the benefits of such independence are well explained in other chapters of this book, we will focus on the ways in which this independence is achieved within the PALIO framework.

The first level of support can be clearly traced to the use of a single format for describing services in a document-oriented fashion–with each document encapsulating portions of the system's interactive functionality. Service developers do not need to concern themselves with the exact characteristics of the access terminals or network connections. Instead, they are free to focus on providing rich service functionality and conveying that in a flexible, XML-based form. This serves as an *abstraction* that transcends the device- and platform- evels, although it may, when necessary, make provisions for them. As an example of the latter, authors can indicate which portions of a document go together naturally. These sections can then be used to sub-divide the document, possibly to split it into cards for presentation on a WML-enabled device.

The second level of support concerns the encoding and representation of user actions within the framework. As already discussed, the CL component of the architecture is responsible for converting all incoming user requests (originally expressed in various application-level communication protocols) into a unified representation. This common representation is then propagated into the rest of the framework, thus shielding run-time components and (more importantly) service developers from the specifics of the platforms and communication protocols. Although this approach is clearly oriented toward a request-response system, it has been successfully deployed to support settings as diverse as traditional Web-based interaction and mobile-phone SMS-based interaction. Furthermore, the particular model has been employed in the realisation of *push* services, where there is no immediate action on the part of the user, but rather a pre-existing request for receiving particular types of information as these become available.

4.1.2 Device- and Platform-Awareness

The fact that service authors/developers can construct services in a device- and platform-independent manner does not imply that they cannot take specific devices and platforms into account at different stages of the authoring process. As already discussed, the framework offers support for dynamically adapting the system's output to fit the characteristics of the access terminal. More interestingly, this support goes far beyond the mapping or transformation of documents to the desired output format.

When discussing the CMS component of the architecture (see section 2.2), we mentioned that the context characteristics are available at various levels of abstraction. This implies that the content and presentation of documents can be adapted to abstract characteristics of the device-platform combination, without binding adaptations to any specific device or platform. Achieving this

involves the employment of generic adaptation rules, which are aware of a subset of the characteristics of devices or supported markup language, and are applicable over a multitude of device-platform combinations.

The power of this approach may be better exemplified in the following scenario. A service author wishes to apply different navigation and presentation strategies/schemes, based on the size of the access terminal's screen and its capability to support links. A decision is made to: (a) differentiate between three screen sizes: normal (*e.g.* PC monitor), small (*e.g.* PDA screen) and tiny (*e.g.* mobile phone screen); and (b) distinguish between support for direct hyperlinks (*e.g.* normal hyperlinks in an HTML document), support for indirect links (*e.g.* for SMS-based interaction, which asks the user to send a particular message to a specified number), and no support for links altogether. The PALIO framework inherently supports these distinctions and enables the author to employ adaptation rules that explicitly address them individually or together (*e.g.* creating a navigation scheme for small-screen devices with support for direct links). Effectively, this enables authors to automatically reuse these rules in settings or combinations that were not known or anticipated at the time of authoring. Without such support, one would need to explicitly address every single output platform in isolation. Additionally, new or unknown platforms would require the author to repeat the process all over again.

As an example, consider the case of the various versions of the WML markup language and their forthcoming complement, XHTML Mobile Profile 1.1[11]. All these languages and versions show several differences between each other–differences that are profound in some cases. However, they are all intended for deployment on similar, resource-constrained devices. If one were to address each existing combination of every markup language/version and screen-size combination, one would be faced with a daunting task. Following the PALIO approach, supporting a new or updated markup language would only require the appropriate revision of the lowest-level transformations undertaken by the CL component (mapping from the uniform internal document representation to the final output format). In other words, one can define a specific set of transformations for WML 1.3 compliant browsers and a different set for XHTML Mobile Profile browsers. However, *both* sets of transformations could be applied to the same document, which would have already been adapted for presentation on small-screen devices with support for direct links.

4.1.3 Uniformity and Cross-Platform Consistency

Beyond enabling the creation of services that are aware of the characteristics and capabilities of different devices and platforms, the principles discussed in the previous section also have a major impact on attaining the goals of interface uniformity and consistency. Specifically, using the described mechanisms and approach, the navigation schemes and presentation templates employed in PALIO are bound to outgoing documents by means of adaptation. As a result, ensuring consistency across platforms can be addressed as a service-level issue, or even seen from the much broader perspective of the service's container, or service centre. Furthermore, since presentation and content can be adapted independently in PALIO, achieving consistency in the interface does not in any way affect the generation of the service's information content.

[11] For more information on WML and XHTML Mobile Profile, refer to the Web site of the Open Mobile Alliance: http://www.openmobilealliance.org/

In more practical terms, the two service centres that were developed to demonstrate the PALIO project followed the concept of *gradually downsizing* a common presentation and navigation theme. This involves: (a) the creation of multiple *key templates* intended for major categories of output device configurations (these templates are comprised of both presentation and navigation elements and were derived from a common visual and interactive design); (b) the authoring of adaptation rules which further modify the templates on the basis of device characteristics within each of the categories. Each of the key templates, as well as the adaptations applied to them, created a step-wise simplification of the system's navigation and presentation scheme, by observing a set of principles (*i.e.* what functionality should be available through the interface) and their coupling to the characteristics of the device at hand (i.e., how users perceive and interact with that functionality on a device / platform with given characteristics).

Using the case of inter-service navigation as an example, the "gradual downsizing" approach is demonstrated by the following adaptation rules: when the access terminal has a normal-size screen and direct-link support, access to services other than the "current" one is effected through a "menu" on the left side of the document (containing hyperlinks to each of the services); for terminals with small-size screens, direct-link support and high-bandwidth network connections, the services' menu is moved to the end of the page; for set-ups similar to the latter, but with low-bandwidth connections, the services' menu resides in its own, separate "page" and a link pointing to that page replaces the menu; etc. The screenshots in section 4.2 illustrate two examples of the approach described here.

4.1.4 General Context Awareness

The characteristics of devices and network connections are not the only dimensions of context that are taken into account in PALIO. Of at least equal importance are a user's location and the time. We have already discussed PALIO's support for context-based adaptation, as well as how certain characteristics of context can be abstracted over, or inferred from raw sensor data. Here we will focus on how this enables the delivery of services that can support a user in different contexts of use, or conversely, different users in the same or similar contexts of use.

The premise of context-based, or context-aware, adaptation is that by identifying distinctive characteristics of the context within which a user is interacting it becomes possible to better understand and address the needs of the user. A basic assumption behind that premise is that context can define, or at least influence, a user's needs in terms of interaction and information seeking behaviour, in a way that is independent from the particular characteristics of any given user.

Residing on the server side, the PALIO framework has limited access to context-related information. Specifically, the primary sources of raw context information are: (a) the access terminal's browsing application[12] (from which one can typically identify the type of device, markup language supported, and network connection at hand); (b) the user's current location with a granularity dependent upon the technology used (*i.e.* GPS- or GSM-based positioning);

[12] Here, the term "browsing" does not necessarily refer to hypermedia browsing. Instead, a user's "browsing application" is understood to be that part of the device's software environment which enables a user to access the PALIO information systems.

and (c) the current date and time (in principle it is possible to identify the user's date and time, if this is different from the system's, through the browsing application).

A first level of abstraction over context data is derived, as already discussed in the preceding sections, by generically describing characteristics of the context, instead of specific instance values. For example: specific times are categorised in *zones* (*e.g.* morning), specific dates are associated with a season (*e.g.* summer), specific screen sizes are classified as tiny, small, normal and large, etc. Where possible, the same is done for the user's location. For example, specific geographical locations (expressed in terms of latitude and longitude) are associated with landmarks, buildings, etc.

This augmented representation of the user's interaction context is subsequently used to infer further characteristics of the context that are not available through actual sensors. For example, the noise level within a building is a potentially important context attribute, which is not available as a primary source of context information in PALIO. In some cases it is possible to infer the amount of noise (with varying levels of certainty) from semantic information about the user's location, possibly coupled with information about the time of day and season of year. Consider, for instance, the case of a library building: it is a rather safe assumption that the level of ambient noise in this environment is quite low throughout the day and year. Conversely, tourists frequent the archaeological museum in the city of Heraklion, Crete, primarily during summer months and in the mornings and afternoons. It would be a logical assumption (although not as certain an assumption as in the previous example) that during those times the level of noise in the museum is higher than during the rest of the year.

All explicitly derived and inferred context characteristics are made available through the CMS component of the framework as adaptation determinants. This means that service authors can create adaptation logic that is exclusively context-oriented, or opt to include context parameters in adaptations rules that related the characteristics of users with those of the context in which they are immersed. The examples of adaptation on the basis of device- or platform-characteristics in the previous section belong to the first category. An example belonging in the second category would be the recommendation of restaurants to a user on the basis of the user's culinary preferences, coupled with the user's current location (*i.e.* which restaurants, close to the user, might be to his liking). A further example can be found in the section 4.2 below.

4.1.5 User Awareness

Although not at the centre of MUI research, user-oriented adaptation plays a very significant role in the PALIO framework. The intent of adaptation in this respect is to tailor the system to the particular abilities, skills, requirements and preferences of the individual user. A large part of adaptations in this category are directly intended to tailor the service (*i.e.* the information content) to the needs of the individual user. However, another dimension (arguably more relevant to MUIs) is the adaptation of presentation with the objective of tailoring the interface to accommodate user requirements and preferences.

The number of adaptations possible in this direction is quite large. One could even envisage using rudimentary demographic data (*e.g.* a user's age group) to perform substantial adaptations in the UI (*e.g.* by using different colour or graphics themes for different age groups). Out of this

large number of possibilities, PALIO is focusing on interface adaptations that are intended to facilitate *universal access* (Stephanidis, 2001b) of the services. Specifically, users that register with a PALIO service centre can identify that they require a specialised interface due to a disability. Disabilities explicitly addressed by PALIO at this point in time include blindness, colour blindness, low vision, and motor impairments. Elements of the interface that are adapted to ensure high-quality interaction for these categories of users include the overall organisation of the interface, the manipulation of colour schemes and graphics (including replacing graphics with equivalent content), the introduction of specialised navigation facilities, etc.

Furthermore, since user profiles are stored on the server, users can enjoy the accessibility afforded by such specialised adaptations over the full range of devices supported by the system. Additionally, the PALIO framework is capable of selecting among alternative presentation modalities (when these are available), adopting the one that is most suitable for the user and device combination. For example, if the description of a venue is available in both text and audio formats and the user is blind, and the user's device is capable of rendering audio files, PALIO can compose a response that incorporates the audio description, rather than the textual one.

4.2 A Brief Example

To better illustrate the capabilities of the PALIO framework and its relation to MUIs, this section presents an example from the PALIO information system installed for demonstration and testing purposes in the city of Siena, Italy. For brevity, the example will not delve into technical details; instead, it will focus on the results of applying the PALIO framework on the following two fictional interaction scenarios.

1st scenario: An English-speaking wheelchair-bound user is in Piazza del Campo (Siena, Italy) in the morning. She is interested in sightseeing and prefers visiting monuments and museums. She is accessing the system from her palmtop, which she rented from the Siena Tourist Bureau and which is fitted with a GPS unit. She accesses the **City Guide** service and asks for recommendations about what she could see or do next.

2nd scenario: An Italian-speaking able-bodied user is also in Piazza del Campo (Siena, Italy) around noon. He hasn't used the system before and, therefore, there is no information about what he might prefer. He is accessing the system from his mobile through WAP. He also accesses the **City Guide** service and asks for recommendations of what he could do next.

The result of the first user's interaction with the PALIO system is shown in Figure 3a. Relevant characteristics include: (a) the presentation language is English; (b) the front-end is tailored for a small-screen terminal capable of colour and graphics; (c) the system's recommendations are in accordance with the user's preferences; (d) recommended sites are in the immediate vicinity of the user; and (e) accessibility information is provided immediately (at the first level), since this information will impact on whether the user can visit the place or not.

The result of the second user's interaction with the PALIO system is shown in Figure 3b. Relevant characteristics include: (a) the presentation language is Italian; (b) the front-end is tailored for tiny-screen terminals without assumptions made about colour and graphics; (c) the system's recommendations are derived from preferences for the user's group–asterisks next to each recommendation indicate other users' collective assessments of the venue recommended; (d) recommendations include a wider range of activities (*e.g.* sightseeing and eating); (e) one type of activity (eating) is relevant to temporal context (it's noon); (f) recommended sites are in the general vicinity of the user; and (g) accessibility information is not provided immediately (at the first level).

5 Summary and On-Going Work

This chapter has presented the PALIO service framework, with a focus on its adaptation infrastructure. PALIO constitutes a substantial extension over previous efforts toward universal access, since it introduces and explicitly accounts for novel types of adaptation and new

(a) GPS-enabled PDA

(b) WAP terminal

Figure 3: Output on different devices from the PALIO Information System in Siena, Italy.

interactive platforms beyond the desktop. Accordingly, it pursues an architectural model of interaction that is expected to be widely applicable in service sectors other than tourism.

From the perspective of MUIs, PALIO offers an innovative approach to the development of user- and context-aware WebUIs. The framework supports virtually all types of hypermedia adaptation techniques, thus enabling the easy creation of on-line systems. These systems maintain high levels of consistency over different platforms, while at the same time, they allow developers to make the best use of available resources. PALIO provides support for the strict separation of content and presentation, allowing each to be manipulated and adapted independently, thus making it possible to cater for the needs of non-traditional user groups (such as disabled users).

For service creators, PALIO is a powerful and easy-to-use development system for many reasons. First, the notions of *user model* and *context model* (including user location) are seamlessly and transparently supported through their incorporation in the PALIO adaptation rules language. Second, adherence to web standards, as well as the use of XML, XSLT and other technologies developed by the W3C, allows web developers to easily move from other web development systems to PALIO. Finally, PALIO supports automatic content creation for different user terminal device types, by automatically transforming content format.

The PALIO system can be easily expanded to include new sources of information and it can be ported to different application domains. This is possible because PALIO abstracts the information sources it uses through the use of a domain-specific ontology. Adding new information sources will require the incorporation of the new information databases to the GIS and the modification of the PALIO information ontology. Furthermore, using the Cocoon publishing framework as the base of the PALIO system and Java as the development language means that PALIO can run on any server platform and that it can support a wide range of adaptation using standard web publishing technology.

The PALIO framework and developed information systems and demonstration services are under evaluation as of the time of this writing. Preliminary evaluation results are positive, both from the perspective of end-user experiences, and from the perspective of service development.

Ongoing work is targeting the provision of better support for *push* services and the incorporation of support for standardised device profiles. In particular, we are currently working on more intelligent push services that can take the full interaction context and previous interaction history into account when deciding if and what type of information should be sent to a user (based on possibly uncertain knowledge). In parallel, we are working to incorporate support for Composite Capability/Preference Profiles (CC/PP)[13], and for UAProf[14], in order to attain a more standardised way to describe, communicate and retrieve device capabilities.

[13] For more information on CC/PP refer to: http://www.w3.org/Mobile/CCPP/
[14] For more information on UAProf refer to: http://www.openmobilealliance.org/documents.asp

6 Acknowledgements

The PALIO project (IST-1999-20656) is partly funded by the Information Society Technologies Programme of the European Commission – DG Information Society. The partners in the PALIO consortium are: 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, Napoleon Maou, and Margherita Antona.

7 References

Ardissono, L. and Goy, A. (1999) Tailoring the interaction with users of electronic shops, in *Proceedings of 7th International Conference on User Modeling, UM99* (ed. J. Kay), Springer, Wien, pp. 35-44.

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

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

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

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

Dey A. K., and Abowd, G. D. (2000) Towards a better understanding of context and context awareness, in *Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness, affiliated with the CHI 2000 Conference on Human Factors in Computer Systems,* ACM Press, New York.

Fink, J., Kobsa, A. and Nill, A. (1998) Adaptable and adaptive information provision for all users, including disabled and elderly people. *The New Review of Hypermedia and Multimedia,* 4, 163-188.

Henze, N. (2001) Open Adaptive Hypermedia: An approach to adaptive information presentation on the Web, in *Proceedings of the 1st International Conference on Universal Access in Human-Computer Interaction (UAHCI 2001), held jointly with HCI International 2001* (ed C. Stephanidis), Lawrence Erlbaum Associates, Mahwah, NJ, pp. 818 - 821.

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

Maglio, P. P. and Barrett, R. (2000). Intermediaries personalize information streams. *Communications of the ACM,* 43(8), 96-101.

Oppermann, R., Specht, M. (1998) Adaptive Support for a Mobile Museum Guide, in *IMC '98 - Workshop "Interactive Application of Mobile Computing"*, Rostock, November 1998. On-line available at: http://www.egd.igd.fhg.de/~imc98/Proceedings/imc98-SessionMA3-2.pdf

Savidis, A., and Stephanidis C. (2001) The Unified User Interface Software Architecture, in *User Interfaces for All – Concepts, Methods and Tools* (ed C. Stephanidis). Lawrence Erlbaum Associates, Mahwah, NJ (ISBN 0-8058-2967-9), pp. 389-415.

Schwab, Pohl, W., and Koychev, I. (2000) Learning to recommend from positive evidence, in *Proceedings of 2000 Int. Conf. on Intelligent User Interfaces,* ACM Press, New York, pp. 241-247.

Stephanidis, C. (2001a) Adaptive techniques for Universal Access. *User Modelling and User Adapted Interaction International Journal,* 11 (1-2): 159-179.

Stephanidis, C. (2001b) User Interfaces for All: New perspectives into Human-Computer Interaction, in *User Interfaces for All – Concepts, Methods and Tools* (ed C. Stephanidis). Lawrence Erlbaum Associates, Mahwah, NJ (ISBN 0-8058-2967-9), pp. 3-17.

Stephanidis, C. (2001c) The concept of Unified User Interfaces, in *User Interfaces for All – Concepts, Methods and Tools* (ed C. Stephanidis). Lawrence Erlbaum Associates, Mahwah, NJ (ISBN 0-8058-2967-9), pp. 371-388.

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