8

Methods and Tools for the Development of Adaptive Applications

R. Torlone, T. Barbieri, E. Bertini, A. Bianchi, M. Billi, D. Bolchini, S. Bruna, L. Burzagli, A. Calì, T. Catarci, S. Ceri, F. Daniel, R. De Virgilio, F. Facca, F. Gabbanini, S. Gabrielli, G. Giunta, P. Graziani, S. Kimani, M. Legnani, L. Mainetti, M. Matera, E. Palchetti, D. Presenza, G. Santucci, L. Sbattella, and N. Simeoni

8.1 Introduction

The number and the spread of nontraditional devices able to provide access to the Web *everywhere* and *anytime* are increasing day by day. These devices include not only cellular phones, PDAs, and terminals for disabled people, but also new kinds of devices, possibly embedded into objects such as household appliances or vehicle dashboards. The characteristics of the various devices are so different that the issues related to delivering information and services on the Web involve not only presentational aspects, but also structural and navigational aspects. As an example consider a cellular phone: its limited computing capabilities require that information be filtered and organized as a collection of atomic units whose dimensions depend closely on specific features of the device.

It turns out that a novel and fundamental requirement in this scenario is the system's ability to adapt and personalize content delivery according to the *context* in which the client accesses the system. As has been observed in Sect. 2.3.3, context information usually involves several independent coordinates: the access device (even in the presence of strong heterogeneity of devices), the quality of service of the network, the user's preferences, the location, the time, the language, and so on.

In this chapter, we present models, methods, and techniques for the design and development of Web-based information systems that are adaptive with respect to the various coordinates of the context. Several research teams contributed to the results presented in this chapter, and the overall presentation is organized as follows. In Sect. 8.2, we illustrate a general design methodology for the development of adaptive information systems. In Sect. 8.3, we discuss the design and implementation of tools supporting adaptive interaction with Web information systems. Finally, in Sect. 8.4, we illustrate methods to evaluate the usability and accessibility of adaptive systems.

8.2 Design Methodologies for Multichannel Adaptive Information Systems

More and more users are asking for services and content that are highly tailored to their devices and, more generally, to their specific contexts of interaction. Accordingly, the development of suitable information systems presents new requirements to application designers. In order to help designers cope with such demands, we shall provide some guidelines to support them in the development of *multichannel*, *adaptive* Web information systems. As shown in Fig. 8.1, these guidelines cover several aspects of design and development, and have been combined to form a methodological framework.

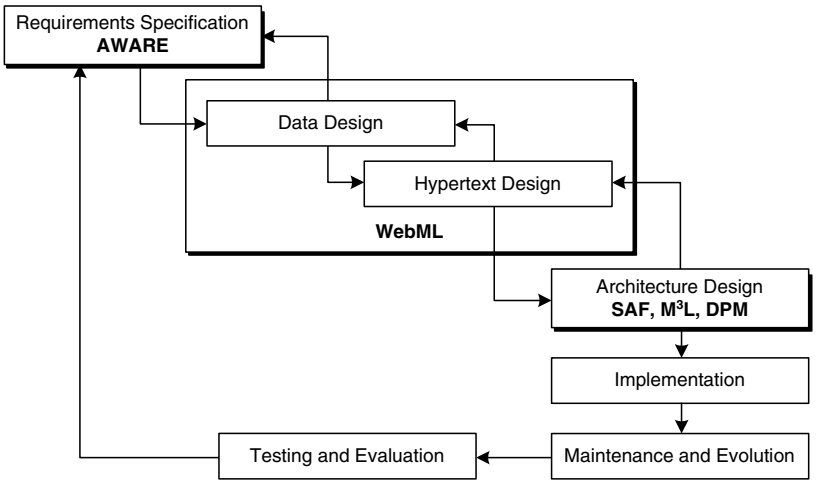

Fig. 8.1. Software life cycle, with special focus on adaptive Web applications

The framework includes a number of coordinated activities as follows:

- Requirements for Web applications are formalized by means of the *AWARE* method (Sect. 8.2.1) and serve as input to the subsequent design tasks.
- Data and hypertext design is based on the *WebML* method, extended appropriately to adaptive applications (Sect. 8.2.2).
- Sections 8.2.3 and 8.2.4, finally, put the methodological results on a technological basis, by outlining the multichannel and multimodal deployment architectures and implementations developed, namely *SAF*, M^3L, and *DPM*.

8.2.1 Modeling the Requirements of Multichannel Applications

AWARE (Analysis of Web Application REquirements) is a requirements engineering model which recognizes the central role of all the relevant stakeholders

in a project and their goals in eliciting, analyzing, and specifying requirements for an interactive application, as in traditional goal-based requirements engineering approaches [22, 136, 277]. Goal-oriented requirements engineering assumes that the "why" of the stakeholders' requirements has to be sought and documented, in order to highlight and keep track of the reasons behind the requirements and the design decisions. Lack of space prevents us from fully illustrating the methodology (for which we refer to [77]). Here we recall the key concepts of AWARE, inviting the reader to grasp the essence of the method.

Understanding Stakeholders, Users and Their Goals

Stakeholders are those people who have an interest in the success of an application or may have knowledge relevant to it and visions related to its success. The stakeholders include, of course, the clients who fund the development of the application, but may also include other company representatives, marketing managers, and sponsors, as well as decision makers, opinion makers, or domain and content experts external to the organization. Some of these stakeholders have personalized goals with respect to the application to be built (see the examples in Fig. 8.2), in the sense that they have a direct interest in its successful deployment and use (e.g., the client or her/his representatives). Others may not have goals but can project their *visions* on the application: thanks to their knowledge or expertise in the field, they can share their perspective and opinion on the project (e.g about the content, the technology, the communication strategy, the users, their needs, and so on).

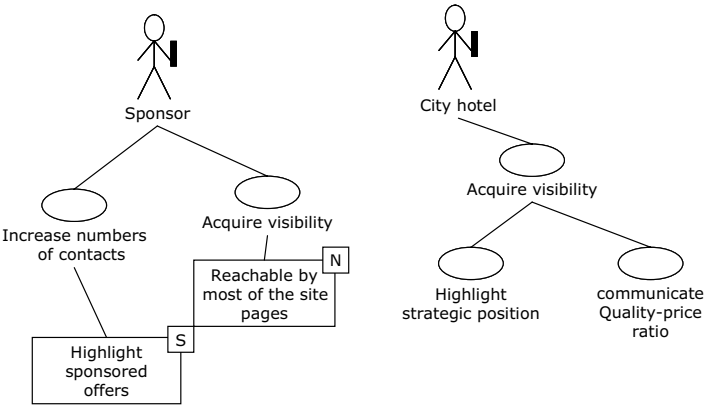

Fig. 8.2. Example of the main goals of stakeholders

The *users* are an important category of stakeholders. They can be described in terms of the *personal characteristics* of archetypal visitors (also called "person"). Personal characteristics are chosen along any dimension that analysts

consider relevant to the design. For an application related to tourism in a national park, the relevant persons may be first-time users, experienced users, children, parents, people between 15 and 18 years old, people over 30 years old, people with fast connections, people with slow connections, people who are not familiar with the technology, people with visual disabilities, hearing-impaired individuals, students, foreign tourists, first-time-visitor tourists, etc. Persons may be defined along any orthogonal dimension (e.g., site knowledge, family relationship, level of disability, age, domain expertise, or occupation) that contains user characteristics. A *user profile* aggregates a meaningful set of multidimensional characteristics, which tentatively describe a potential visitor to the application.

The requirements analysis should reason carefully about the *user's goals*, which should be plausible motivations for visiting the application, or the objectives of their interaction. The user's goals may vary in granularity from low-level, specific information seeking ("find the opening hours of the park on day X"), called functional goals, to higher-level, open-ended, ill-defined needs or expectations ("decide whether the city is worth visiting"), called soft goals [76].

Goal identification should also allow one to define the overall purpose of combining different communication channels, the definition and selection of the types of channel (mobile phone, PDA, interactive TV, kiosk, Website, etc.), the role played by each channel in the communication strategy, and the specific goals envisioned for each channel.

Analyzing Goals and Using Scenarios

AWARE adopts a refinement process to pass from all the stakeholders' (including the users') high-level goals to subgoals and, eventually, to application requirements. The raw material gathered during elicitation may consist of an unstructured mix of very high-level goals, pieces of design, examples of other sites, design ideas and sketches, design decisions, and detailed requirements. This first set of raw material must somehow be organized in order to be usable by analysts, and fed into design. The analysis of such material may be guided by the following lines of inquiry:

- *What does a given high-level goal mean? How can it be clarified?* Often, in fact, the goals of the users and of the main stakeholders are too vague, abstract, or generic, posing obstacles to devising operational indications for the designers. For example, if a goal of a main stakeholder is to "attract new tourists", analysts should inquire "What does it mean specifically?" To make tourists stay longer in the territory? To bring new people to the territory? From which countries?
- *What are the possible (realistic) ways to satisfy a high-level goal?* Analysts should elicit possible subgoals which may contribute to accomplish the long-term goal. For example, for the goal "How can we convince people

to stay longer in the surroundings of the city?", a possible subgoal might be to "highlight a variety of tours and sequences of visits to attractions lasting one week or more" or "explore tourism retention strategies useful for achieving the high-level goal".

Refinement involves a decision-making process which is crucial for the definition of the communication strategy that will be implemented in the application. This is the activity in which analysts make the most important strategic decisions about the application. The refinement process also applies to user goals. Goal refinement for user goals should ask: "How might a user with this profile want to accomplish her/his goal?" For example, the user goal "planning a visit to city X" may be decomposed into a number of subgoals, such as "know what are the *mustsees* of the city", "decide on a suitable hotel where to stay", or "see interesting hotspots near the hotel".

To facilitate the elicitation and refinement process, user scenarios may complement goal analysis. Scenarios are commonly recognized as powerful drivers for goal-based approaches. Scenarios may take the form of narrative descriptions (also defined as "stories about use") of circumstances in which the application is used by a user with a specific profile. The essential ingredients of a scenario are a user profile and a user goal. When a plausible story which combines such a profile and such a goal is constructed, a scenario emerges which describes a success story of use of the application. Scenarios can assist analysts in discovering new requirements, exemplifying goals, revealing new goals, and facilitating the communication of the requirements to the stakeholders.

Defining and Organizing Requirements

For each channel, which includes not only the type of device but also the characteristics of the context of use (see Chap. 2), the requirements for content-intensive interactive applications are expressed in natural language and their level of detail is negotiated between analysts and the design team. The requirements are not aimed at capturing all the functionality of the application, but only at those crucial features needed by designers to shape the user experience and by stakeholders to agree on initial specifications. To organize the requirements set and to facilitate the subsequent design activity, AWARE classifies the requirements according to the aspects of the design for which they have an implication. The AWARE requirement taxonomy includes (among other things) the following dimensions [77]: *Content*, *Structure of Content*, *Access Paths*, *Navigation*, *Presentation*, *Operation*, *Accessibility* and *Adaptation*.

The requirements give coarse-grained, semi-structured indications to designers. We propose a model, named *IDM (Interactive Dialogue Model)*, as an *innovative conceptual tool* to facilitate the transition between requirements and detailed application design. After a first (incomplete and provisional) set of requirements, designers may need to give a coherent shape to the user experience in terms of possible *dialogues that the user may be engaged in*, to

properly support the scenarios envisioned during the requirements analysis. In this dialogue-based perspective, the in-the-large structure of the application takes the shape of a dialogue generator, and the user may activate one or more dialogues within a limited range of possibilities.

IDM enables us to communicate, document, and take decisions about the following concerns: What is the overall content? What is the overall organization of the content? How can the user access the content and browse through the various pieces? What are the operations/transactions available to the user?

Through a few simple, intuitive primitives (based on theories about dialogue, and linguistic theories), IDM enables designers to define the overall patterns of the communication and interaction dialogue before digging into details that depend on technical issues. IDM also embeds methodological support for anticipating the description of context-aware services at requirements/design time, such as the impact of the user's location on the dynamics of the dialogue, and the provision of high-level rules describing the behavior of the application in response to changes in the user context. As described below, IDM schemas can also be easily mapped onto lower-level design languages such as WebML.

8.2.2 Design of Multichannel Adaptive Web-Based Applications

The design of the front-end of the application leverages a conceptual-modeling approach based on the adoption of WebML (Web Modeling Language) [110]. WebML is a visual language for specifying the content structure of a Web application and the organization and presentation of contents in one or more hypertexts.

The design process starts with the specification of a data schema, expressing the organization of the content of the Web application. The *WebML data model* uses Entity–Relationship primitives. The *WebML hypertext model* then allows one to describe how content, previously specified in the data schema, is published in the application hypertext. The overall structure of the hypertext is defined in terms of *site views*, *areas*, *pages*, and *content units*. A *site view* is a hypertext, designed to address a specific set of requirements. Several site views can be defined on top of the same data schema, for serving the needs of different user communities, or for arranging the composition of pages to meet the requirements of different access devices such as PDAs, smart phones, and similar appliances. A site view is composed of *areas*, which are the main sections of the hypertext and comprise, recursively, other subareas or pages. *Pages* are the actual containers of information delivered to the user; they are made of *content units*, which are the elementary pieces of information extracted from the data sources by means of queries, and published within pages.

Content units and pages are interconnected by *links* to constitute site views. Links can connect units in a variety of configurations, yielding to

composite navigation patterns.[1] Besides representing user navigation, links between units also specify the transportation of some information that the destination unit uses for selecting the data instances to be displayed.

Some WebML units also support the specification of content management operations. They allow the creating, deleting or modifying of an instance of an entity (through the `create`, `delete` and `modify` units respectively), or the adding or dropping of a relationship between two instances (through the `connect` and `disconnect` units, respectively). Recently, WebML has also been extended to model invocations of Web services; in this context, application data can be derived from external data sources as well [85]. For a more complete presentation of WebML and its visual notation, the reader is referred to [110].

IDM	WebML - Data	WebML - Hypertext
Topic	Entity	/
Dialogue Schema	/	Site View
Dialogue Act	Attributes/Entity/Relations	<Cluster of> Pages/Areas
Introductory Act	Access Schema	Content Unit/Index Chain
Group Strategy	/	Navigation Pattern
Transition Act	Interconnection Schema	Navigation Pattern
Transition Strategy	/	Navigation Pattern
Operation Act	/	Content Management Pattern
Structural Strategy	/	Navigation Pattern

Fig. 8.3. Mapping of IDM onto WebML primitives. The "/" symbol indicates the lack of corresponding primitives in one of the two WebML models

With respect to the requirements engineering process described earlier, Fig. 8.3 shows a possible mapping between IDM concepts and WebML primitives. Such a mapping supports the translation of requirements into conceptual design.

WebML and Context-Awareness

The overall design process for context-aware applications can follow the activity flow typically used for conventional Web applications. However, some new issues must be considered in modeling and exploiting the context, in order to achieve adaptive behavior.

During data design, the user and context requirements can be translated into three different subschemas complementing the application data (see Fig. 8.4):

- The *user sub-schema*, which clusters data about users and their access rights to application data. In particular, the entity `User` provides a basic profile of the application's users, the entity `Group` allows access rights for group of users to be managed, and the entity `Site View` allows users

[1] See [110] for a complete list of WebML navigation patterns.

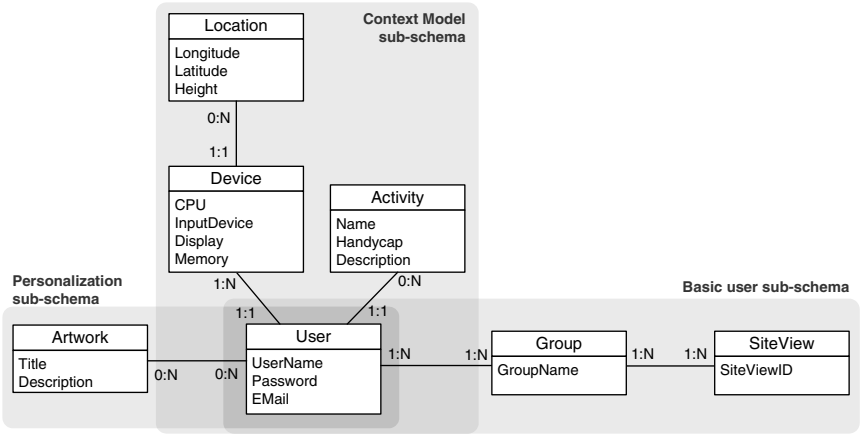

Fig. 8.4. Three subschemas representing context data

(and user groups) to be associated with views over the application's data source. In the case of adaptive context-aware applications, users may require different interaction and navigation structures, according to the varying properties of the context.

- The *personalization subschema*, which consists of entities from the application data, associated with the `User` by means of relationships expressing user preferences for some entity instances. In general, relationships defined between the entity `User` and any other entity of the application data support the personalization of the content of the entity with respect to the identity of the user. For example, the relationship between the entities `Artwork` and `User` in Fig. 8.4 allows the selection and the presentation to the user of the artworks s/he likes most.
- The *context sub-schema*, which includes entities such as `Device`, `Location` and `Activity`, that describe particular properties of the context that are considered by the application in order to provide adaptivity. Context entities are connected to the entity `User` to associate each user with her/his (personal) context.

Such a context representation is consistent with the MAIS context model (see Chap. 2) and slightly extends it to take into account some requirements specific to WebML (e.g., site view specification).

During hypertext design, adaptive functional requirements are considered to augment the application's front end with reactive mechanisms. As illustrated in Fig. 8.5, our basic assumption is that context-awareness is a property to be associated only with some *pages* of an application, and not necessarily the application as a whole. Location-aware applications, for example, adapt "core" contents to the position of a user, but typical "access pages" (including

links to the main application areas) might not be affected by the context of use.

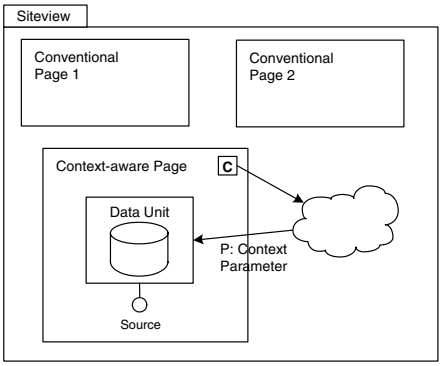

Fig. 8.5. Coarse hypertext schema highlighting context-aware pages. Context-aware pages are labeled with a `C` and are associated with a context cloud

We therefore tag adaptive pages with a `C`-label (standing for "Context-aware") to distinguish them from conventional pages. This label indicates that some *adaptivity actions* must be associated with the page. During application execution, such actions must be evaluated prior to the computation of the page, since they can serve to customize the page content or to modify the predefined navigation flow. As shown in Fig. 8.5, adaptivity actions are clustered within a *context cloud*. The cloud is external to the page, and the adaptivity actions that it clusters are kept separate from the page specification. The aim is to highlight the two different logics derived from the roles played by pages and context clouds: while the former act as providers of content and services, the latter act as modifiers of such content and services.

In order to continuously evaluate the state of the context and the executing page's adaptivity actions, the `C`-pages must be provided with autonomous intervention capabilities. In the absence of dedicated *push* mechanisms,[2] such capabilities can be achieved by periodically *refreshing* the viewed page and giving the adaptive logic of the application the possibility to intervene in the application itself before rendering the actual response. Where no *push* mechanisms are available, this *polling* mechanism provides a valuable "simulation" of the required active behavior. Such mechanism could also be managed by a middleware architecture, such as the MAIS reflective architecture (see Chap. 4).

[2] The standard HTTP protocol underlying most of today's Web applications implements a strict *pull* paradigm.

Specifying Adaptivity Actions

The design of *context clouds* assumes a central role for context-aware applications. Context clouds are associated with the page by means of a directed arrow, i.e., a link, exiting the C-label. This link ensures communication between the page logic and the cloud logic, since it can transport parameters derived from the content of the page, which may be useful for computing the actions specified within the cloud. Vice versa, a link from the cloud to the page can transport parameters or, in general, values computed by the adaptivity actions, which might affect the adaptivity of page contents with respect to a new context.

In order to support the specification of adaptivity actions in the context cloud, WebML has been extended through some new visual constructs that refer to a number of dimensions:

1. *Acquisition and management of context data.* These actions may consist of:
   - Acquisition of fresh context data, provided by means of device- or client-side-generated URL parameters. A new `Get URL Parameter` unit has been introduced to support the retrieval of parameters generated at the client side and communicated to the application by appending "parameter–value" pairs to an HTTP request's query string. Once fresh context parameters have been retrieved, the values previously stored in the data source are replaced accordingly. This operation is specified by means of WebML content management units.
   - Acquisition of context data from the context model. The execution of adaptivity actions may require the retrieval of context data already stored in the application data source, without requiring any visualization. A `Get Data` unit has therefore been introduced. Similarly to WebML content units, it specifies the retrieval of values from the data source, according to a selector condition. Differently from content units, it does not publish the retrieved values in a page.
2. *Condition evaluation.* The execution of some actions may depend on the evaluation of some conditions. The pattern that recurs most often consists of evaluating whether the context has changed, and hence triggering some adaptivity actions. The evaluation of conditions is specified through two control structures, represented by the `If` and `Switch` operation units that have been recently proposed for extending WebML for workflow modeling [84].
3. *Page content adaptivity.* Parameters produced by context data acquisition actions and by condition evaluation can be used for page computation. They are sent back to the page by means of a link exiting the context cloud and going to the page. The result is the display of a page where the content is *filtered* with respect to the current context.

4. *Navigation adaptivity.* The effect of condition evaluation within the context cloud can be an automatic, i.e., context-triggered navigation causing redirection to a different page. The specification of context-triggered navigation just requires a link exiting the context cloud to be connected to pages other than the cloud's source page.
5. *Adaptivity of the whole hypertext structure.* In order to deal with coarse-grained adaptivity requirements, for example due to the user changing his/her device, role, and/or activity within a multichannel, mobile environment, a switch to a different site view might be needed. Therefore, a `Change Site View` unit has been introduced, which takes the identifiers of a target site view and of a target page as input. In order to support "contextual" switching, the input link also transports parameters characterizing the current state of interaction, such as those representing selections made by the user, session data (e.g., the object identifiers of the user and group), and parameters characterizing the current context, retrieved through the data acquisition cycle performed most recently.
6. *Adaptivity of presentation properties.* In order to support more fine-grained adjustments to the application's appearance, a `Change Style` unit has been introduced, for representing run-time modification of the presentation properties of the style sheet coding.

An Example of Context-Aware Design with WebML

In the MAIS project, we have experimented both with the methodology and with the extension of the model by means of a prototype application providing context-aware tourist information [247]. Figure 8.6 shows a fragment of the application design, illustrating some of the extensions presented above.[3]

Starting from the *Sight Details* page, the schema states that, on the first automatically triggered access to the page, the context cloud is accessed and the operations included in it are performed,[4] Hence, the user's *Latitude* and *Longitude* are retrieved from the request parameters by two `Get Url Parameter` units. The retrieved values are used by the `Get Data` unit *Get Sight* to identify a suitable *Sight* for the current user's position. Then, the object identifier (OID) retrieved by the `Get Data` unit is checked by the `If` unit. If the OID value obtained is not null, the corresponding *Sight* is shown in the *Sight Details* page (*content* adaptation); otherwise, the previously retrieved *Latitude* and *Longitude* are used to get a city map from an external Web service. The invocation of an appropriate service could be managed by the *Concrete Service Invoker*, as outlined in Chap. 3. The user is then redirected to the *City Map* page, and the perceived effect is that of an automatically

[3] For more details about the visual notation of the WebML extensions for context-awareness, the reader is referred to [109].

[4] For the sake of simplicity, the cloud is not explicitly represented here; however, it consists of five operation units positioned outside the two pages.

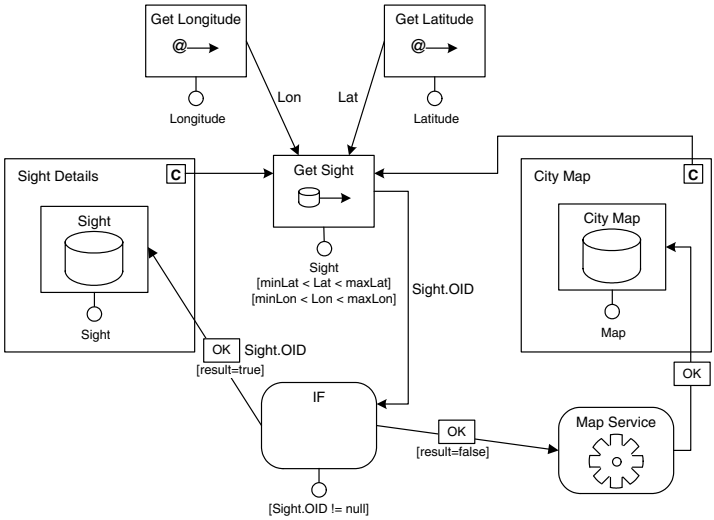

Fig. 8.6. A schema exemplifying some of the WebML extensions introduced for context-awareness

performed *navigation action*. Figure 8.6 also models the *City Map* page as a context-aware page sharing its context cloud with the *Sight Details* page. Therefore, as soon as an automatic refresh of the *City Map* page occurs, the shared context cloud is triggered again and the application is adapted to the user's new position.

8.2.3 Multimodal Deployment of Adaptive Applications

To support the deployment, execution, and delivery of adaptive, multimodal applications, designed using the methodology proposed in Sect. 8.2.2, two frameworks have been developed: the first is focused on adaptive, context-aware applications, and the second deals with multimodal delivery of hypertexts. The two frameworks will be described below.

Context-Awareness

A framework called *SAF* (Situation Aware Framework) has been developed to allow the simple design, delivery, and execution of context-aware Web applications. The kinds of adaptation supported by SAF are:

- *layout*: adaptation of the disposition of objects in the page space;
- *presentation*: adaptation of color scheme, font type, and font size;
- *entity instance selection*: selection of a specific instance of an entity;
- *attribute selection*: selection of the attributes of an entity to be shown.

In SAF, situation awareness is managed through a declarative approach. An SAF situation–aware service can be modeled with WebML, using properties that express declaratively the adaptation behavior that the platform must perform. In SAF, the adaptation behavior of the platform can be determined using two approaches:

- *Explicit*: the service designer must explicitly declare the kind of adaptation desired using properties or rules, as described above.
- *Implicit*: this kind of adaptation is performed automatically by the framework according to a set of predefined rules. If the user is running, for example, the layout can be switched to one column mode and the font size can be set to the largest size available. The rules managing this process are defined in the framework.

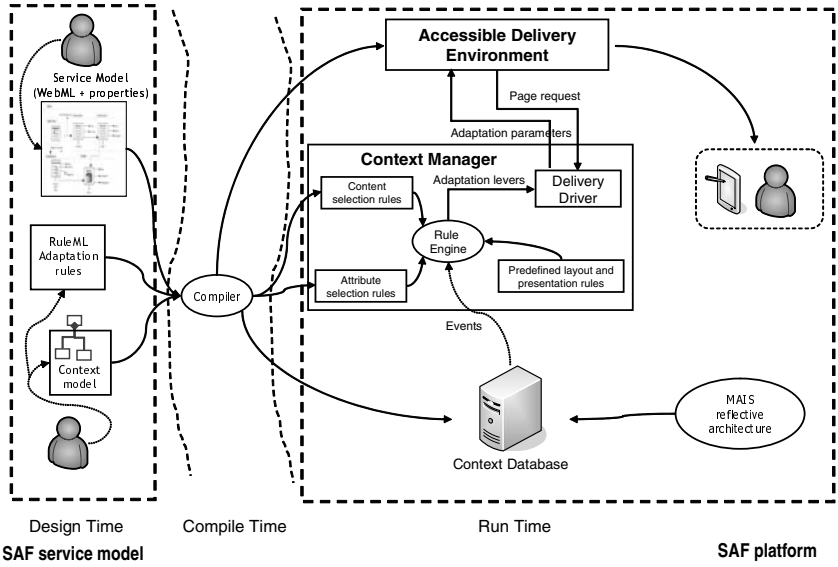

Fig. 8.7. The SAF architecture

Figure 8.7 shows a brief description of the architecture of the framework and of its components. The left-hand side of the figure describes the design-time components:

- The *context manager* must design the context model, and define the set of properties (and values) used to describe the environment.
- The *service designer* creates the service description using the extended WebML model and explicitly defines the adaptation behavior of the system.

The right-hand side of the figure shows the run-time components. The context-aware module shown is responsible for taking the decisions about the adaptation actions to be performed. According to rules that have been defined, the status of the context, and the events that have occurred, the rule engine decides which adaptation actions to perform and uses a driver to set up the delivery environment configuration.

The M^{3}L Multimodal Framework

The M^{3}L framework is designed to deliver multimodal content using, both for input and for output, vocal and visual interaction modes. These modes has been chosen considering the capabilities and characteristics of the devices currently available on the market. The underlying design model of the M^{3}L multimodal framework is based on an extension of the WebML language. To make a multimodal-enabled WebML service, additional information is needed. For every component of the page, it is necessary to describe the interaction modes to be used, both for input and for output. A WebML design is enriched with properties associated with the various WebML units. These properties are translated into M^{3}L using specific attributes.

The M^{3}L multimodal framework is able to manage different devices at the same time synchronizing them to offer a coherent view of the same service on the various channels supported. The user feels that he/she is interacting with a single integrated service even if information is transmitted and delivered through several physically different channels. The proposed solution allows services to be written no more than once, through the use of a multimodal-specific markup language (which gives us the name M^{3}L).

M^{3}L is defined as a set of XHTML modules [378]: the "multimodal" (used to structure the M^{3}L document) and "M^{3}L Forms" (specific to user input) modules. The two modules, together with the XHTML framework, constitute the M^{3}L language. M^{3}L conforms to the XHTML Host language specification.

The two modules define new elements and attributes used to manage input (especially for data collected by form) and output synchronization. Attributes are defined to allow the developer to choose the best interaction mode for output data. These attributes allow service developers to select a preferred (or compulsory) delivery or input mode. The *out* attribute allows one to specify which modes can be used to deliver the content of an element to the user. This attribute is made available to any tag that contains information that must be presented to the user. The *mode* attribute, on the other hand, specifies the modes that a user can use to input data. It is associated with form fields and may have three possible values: "text" to indicate that the user can use a keyboard, "voice" to indicate that the user may use her/his voice, and "all" to say that every known input mode may be used. If, for example, the tag <p> has its *out* attribute set to "visual", the text contained will be delivered only through the visual mode (via a screen, for example).

Figure 8.8 displays the main components of the architecture of the M^{3}L framework. The *multimodal integrator* is the core of the multimodal framework. It manages the overall operation logic of the system and integrates the inputs coming from the various connected channels and modes. The integrator determines the outputs to be sent to the user and manages the synchronization between the channels.

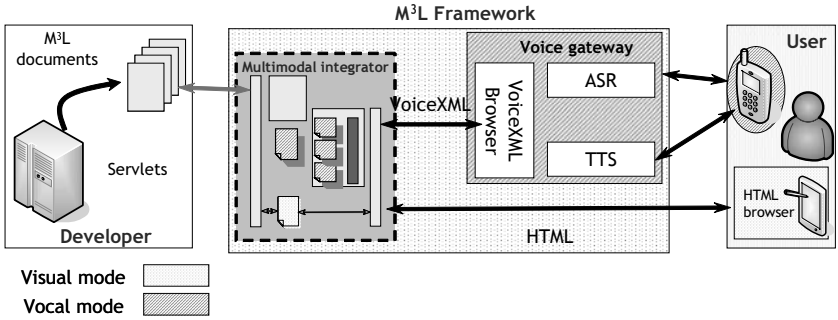

Fig. 8.8. M^{3}L Framework architecture

The *M^{3}L repository* is the container for the multimodal services and content. The *voice server* is the component that allows vocal communication between the user and the service. It receives VoiceXML documents generated by the multimodal integrator for communication with phones, interprets them, and manages the vocal interaction with the user. A TTS (text-to-speech) unit is used to generate the speech provided to the user, and an ASR (Automatic Speech Recognition) unit is used to manage the user's speech and to collect input data. The voice server enables the vocal interaction, allowing the transmission of voice over ordinary PSTN or GSM networks. As an alternative, it is possible to send the VoiceXML file directly to the client, as long as he/she has a suitable vocal browser.

8.2.4 Multichannel Delivery Environment

Dynamic Presentation Manager

The Dynamic Presentation Manager (DPM) is a software module for adaptive presentation of information depending on the delivery environment. The adaptation is based on the current operative context. At run-time, when the user gets to a particular page, it will be possible to personalize and customize the presentation of the information, in a way that depends on the current context in which the user is involved. In this way, a context-aware application [328] is generated. In MAIS, the DPM is used for adapting pages designed with WebML and generated through the WebML code generator. Therefore,

the architectural pattern used in our approach is based on the MVC (Model View Controller) design pattern, where the DPM module is located within the View layer, providing us with the advantages obtained by using this pattern.

A more specific architecture that underlines the functional components of the DPM module is shown in Fig. 8.9. Input data, in the form of *situational data* and *application data*, represent a new, large set of data called *context data*. In our approach, the context is based on three main entities: the user profile (*subject*), hardware and software device features (*tool*), and application data (*object*).

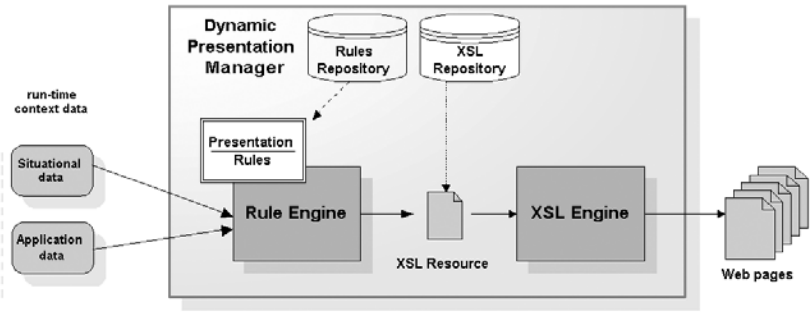

Fig. 8.9. DPM architectural schema

An example of situational might be *screen resolution* = 1024 × 768, *battery level* = high, *memory amount* = 512 MB, and *CPU power* = 1GHz; another possible example might be *marital status* = married, *educational qualification* = degree, and *mother tongue* = English. Application data are produced from the business logic and depend on the specific application domain. This data can be adapted, in terms of content, by other external modules using the information contained within the user profiles. The DPM adaptive presentation does not include the adaptation of content but involves only "look and feel" and layout aspects.

The Rule Engine component uses some rules called *presentation rules* to determine the appropriate XSL (eXtensible Stylesheet Language) [387] file to be passed to the XSL Engine component. Presentation rules are written using the JESS (Java Expert Shell System) language [322]. They are structured as *condition* → *action*. In our work, the term *condition* is replaced by a particular context instance and the term *action* is replaced by the selection of an XSL file. To select the appropriate XSL file, the presentation rules use the situational data and application data provided as input.

The XSL Engine component contains an XSL transformer, which performs a stylesheet transformation using the XSL file selected by the Rule Engine and the application data, serialized in a convenient form. The final result of this transformation is an adapted page that will be presented to the final user,

in an appropriate markup language (i.e., (X)HTML or WML). The Rules Repository is the source of the presentation rules, and the XSL Repository is the source of the XSL files.

An Example of Adaptive Presentation

We have implemented a simple prototype in the MAIS project to validate our multichannel adaptive approach. At design time, using a tourist scenario [247], we developed the data model, the hypertext model, and the presentation model by means of WebML [110]. Starting from this set of models, we then generated the page code with the WebML code generator. During the page generation, exploiting a functionality of the generator itself, we added a set of properties to the JSP page source code in order that these properties would be caught by the DPM to generate the adaptive presentation. At the end, we integrated the DPM with the WebML run-time platform in order to deliver information on several different channels. We accessed the system from three different devices: from a PDA, from a PC, and from a mobile phone, as shown in Fig. 8.10. We implemented presentation rules to fit the tourist scenario, and using an XSL file for each channel/device, we produced the following results:

- replacement of widgets;
- resizing of the page fonts,
- adjustment of the page layout.

Fig. 8.10. Multidevice access from various devices: PDA, PC, and mobile phone

Widget replacement is important for improving the presentation on various channels and for devices with limited hardware resources (i.e., mobile phones in this case). The effects of this replacement are, for example, that images on devices with small screen dimensions are replaced by textual items, and GUI elements such as combo boxes, buttons, and window menus are replaced by lists of items.

Page font resizing is used to present information with an increased font size. In our work, this type of resizing is used to present more easily readable information either in the case of disabled users or to aid the user when in movement. Adjustment of the page layout is used to emphasize aspects of presentation and customization.

To obtain the results described above, we used several context instances structured in terms of the device configuration, such as screen resolution; the protocol being applied; the physical state of the device, such as battery level; and user activity (i.e., user in movement or not).

8.3 Adaptive Interaction in Web Information Systems

In this section, we focus on adaptive tools for Web-based information systems and illustrate some architectures, methods, and techniques that can be used to realize an adaptive interaction.

We start by describing how the interaction with the user can be modeled. We then illustrate, in Sect. 8.3.2, a general architecture for a context-aware adaptation tool capable of implementing the desired interaction in a flexible way. Finally, in Sect. 8.3.3, we present an effective matching technique that can be used in content selection to meet user needs. The focus of this section is on the definition of adaptive interaction, starting from the requirements for specific scenarios, possibly modeled as indicated in Sect. 8.2.1.

8.3.1 Modeling User Interaction

It is our belief that, to deal with the many different implementations that a single application must support, it is fundamental to have a single abstract model able to define user interaction. Having an abstract interface permits us, in fact, to decouple the activity of defining the service dialogue from the activity of implementing the service for several different contexts.

Formalizing the Interaction

The foundation of our design method for interactions is an abstract model capable of describing user system interaction by describing basic activities whose composition will produce a simple but effective Internet-based application. As a consequence, we model the information that is exchanged between

the user and the system. Moreover, our interaction modeling foresees different ways of presenting the same information, in order to adapt it to several physical means and/or channels, such as mice, touch screens, or audio. Using this approach, the designer is provided with a formalism to specify the information content of each presentation and the connection between the various parts, in order to indicate the behavior of the application, that is, how the system evolves as the user interacts with it. Our proposal consists of two main parts:

- a set of *abstract interaction units* (AIUs), to be used as building blocks for the abstract definition of the interface;
- the UML *activity diagram*, which is the formalism for connecting the AIUs that make up the interface.

A set of AIUs has been produced by analyzing the user interfaces that are actually used to model standard Web services. Starting from specific interaction elements, we have grouped them into higher-level units on the basis of functional similarity. Such units express the key interactive features that the specific elements of each group have in common. The challenge is to collect a small set of atomic units that can describe an interaction, abstract enough to be completely unrelated to the particular device on which the interface will be realized, but expressive enough to let designers model complex services. Our effort has produced a small set of AIUs, described below.

The UML *activity diagram* is basically a statechart diagram in which each state represents an activity and each transition is triggered by the end of this activity.

The Set of AIUs

We foresee two main interaction activities: *browsing*, i.e., just observing something produced by the system, and *inputting*, i.e., providing the system with some information. A browsing activity may return values to the system; for example, a point may be returned while browsing an image. An inputting activity may be based on two different strategies: filling in fields with free text or choosing from several predefined choices.

According to these strategies, we foresee a basic set of AIUs: `BrowseImage`, `InteractImage`, `BrowseText`, `BrowseMessage`, `BrowseTable`, `InteractTable`, `FillList`, `SelectChoice`, and `SelectMultipleChoice`. Each AIU is characterized by a signature that defines the input that it expects and the output that it returns. We describe the `BrowseTable` AIU below, to provide an example of their structure. This AIU allows browsing a relational table and has the following signature:

```
BrowseTable(TableId,TableDescription,ListOfBrowsingCommands,
Mode) : {NULL, elemOfListOfBrowsingCommands}
```

The `TableDescription` is a parameter with two components: `TableName`, which is used as a title during the presentation of the table, and `TableSummary`,

which is a text description that can be used when the video channel is not available or is disturbed, or as an alternative when capability of the device of displaying a large table is very poor. The `ListOfBrowsingCommands` is a set of commands oriented towards server-side table manipulation (e.g., moving quickly to a tuple); such commands do not allow any state other than the one hosting the AIU to be reached (i.e., they correspond to self-transition). The `Mode` parameter has three values: (i) Full: the table is presented to the user without any omissions; (ii) Manual: the system allows client-side modification of the table structure, and (iii) Automatic: the table is reduced on the basis of the user profile and other parameters (on the server-side, through selection and projection operations).

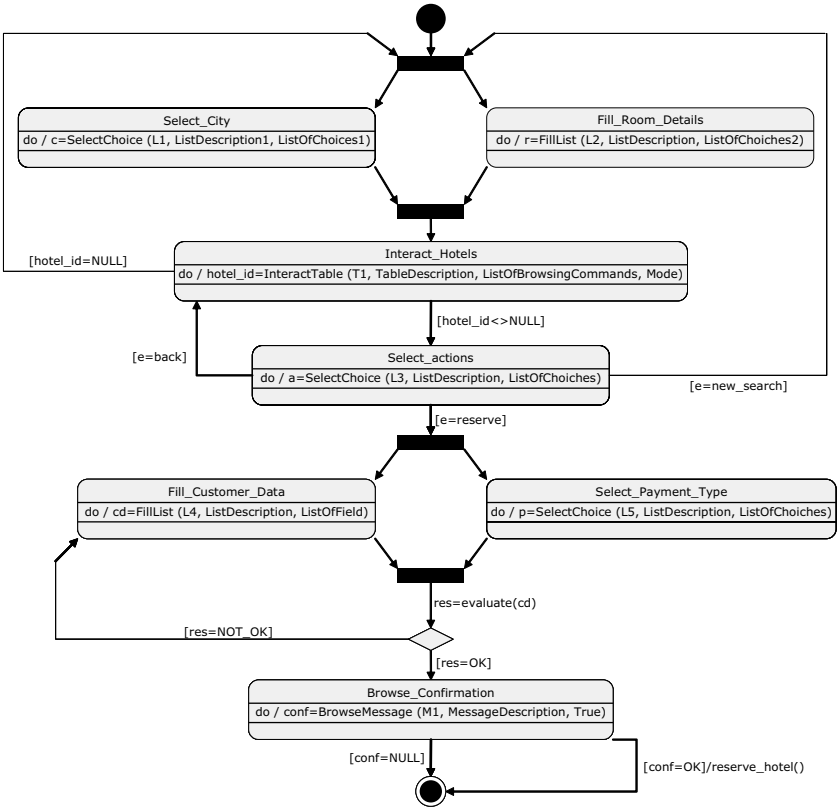

Fig. 8.11. Activity diagram and AIUs to model a hotel reservation service

AIUs at Work

A UML activity diagram is used to compose the AIUs and define the service. Each activity state can contain one AIU and models the user's activity with a specific interaction unit. A transition from one state to the next is triggered by the user interacting with this specific unit, and each transition corresponds to a computation performed on the server. Some interaction units can also appear in parallel, through a fork construct. This takes into account the common situation in which a single presentation contains more than one AIU at the same time, and the case where we are modeling a task that involves interactions that do not have a predefined sequential ordering.

In order to clarify the use of this model, we shall provide an example describing how a simple service for reserving a hotel room can be modeled. We refer to Fig. 8.11, which depicts an activity diagram filled with the specific AIUs utilized to model such a service.

The user starts by inputting data about the city he/she wants to search for and some details about the period for which he/she wants to make a reservation. Since these are two separate tasks they are modeled with two separate AIUs. The city specification is a `SelectChoice` AIU, the details specification is a `FillList` AIU, for which the user is requested to input data about the reservation period. The order of these two tasks is irrelevant, so they are connected by a fork construct. The final implementation could present these tasks in either order. Alternatively, if the selected device has enough screen space, they could be presented in a single unified view.

As the user sends input data, the system passes to the next activity, modeled as an `InteractTable` AIU. The result of the query ("search for a hotel") is, in fact, a set of objects (i.e., hotels), each with a predefined set of attributes. The system, depending on to the capabilities of the device, could choose to present only a certain set of attributes and to replace the presentation of the others with a link pointing to further information.

When the user selects a certain hotel, the system moves to the next activity, the selection of an action to perform on the hotel. The transition from the `Interact_Hotels` activity to the `Select_Action` activity involves parameter passing. When the user selects an object from a table, through the `InteractTable` AIU, the system sends an output parameter, used in the subsequent tasks.

In the `Select_Action` activity, the user is requested to select, from the following list, an action to be performed: reserve the hotel, start a new search, or return to the previous result. This is modeled with a `SelectChoice` AIU which, depending on the device connected, could be realized in various ways: buttons, links, a menu, etc.

The system can proceed to three different activities according to the selection that has been made: (i) return to the starting point, if the user chooses to perform a new search; (ii) go back to the previous result; or (iii) proceed with the reservation task, if she selects to reserve the hotel. If he/she chooses

to proceed with the reservation, the system steps forward to a new fork, below which two concurrent activities take place: the specification of the customer data and the selection of the kind of payment he/she wants to use (e.g., credit card). As in the case of search parameters, we have concurrent AIUs. This means that they can be implemented in a parallel or sequential order. The `Fill_Customer_Data` activity is modeled as a `FillList` AIU because it is intended to accept data directly from the user. The `Select_Payment_Type` activity is modeled as a `SelectChoice` AIU because it is intended to present a predefined list of payment methods, from which the user can select his/her preferred one.

This simple example shows how the composition of abstract interaction units can be done in order to model a service. After this phase, we need an adaptation tool capable of translating this model into a final implementation. In the next subsection, we discuss the architecture and features of such adaptation tool.

8.3.2 Context-Aware Adaptation Tools

As has been observed in Chap. 1, in the case of a data intensive Web information system, it is useful to consider its three main components separately: the content (that is, the data to be published), the presentation (that is, the layout of the pages), and the navigation (that is, the hypertext structure of the Web site). Since the adaptation process should operate on all these components, it turns out that a possible architecture of a system supporting adaptive interaction is that shown in Fig. 8.12. This includes:

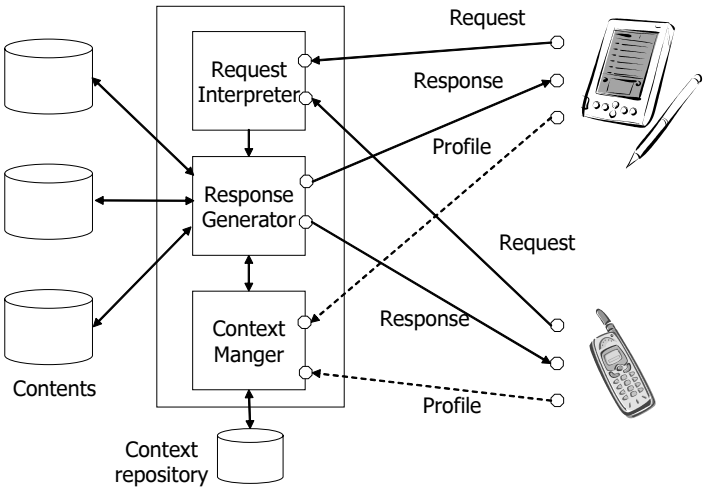

Fig. 8.12. A general reference architecture for an interaction manager

- a *request interpreter*, capable of translating a specific user request (a page or a specific object) into a query over the underlying data;
- a *response generator*, capable of generating all the components of a response to be delivered over the Web (that is, content, structure, and layout) that satisfies the given request and is appropriate to the client profile;
- a *context manager*, capable of obtaining and managing a description of the client's context (usually called the user profile) and of supporting the Response Generator in the execution of its task.

This simple scheme can be easily extended to a situation in which several levels are present between the content and the users, as shown in Fig. 8.13. On the client side, we can have several components (which we call proxy clients) capable of managing the adaptation requirements of a family of contexts with common characteristics. A proxy client should exhibit the features of the proposed scheme, with the difference that a request is actually translated into an appropriate request to the subsequent level according to a "generic" context suitable for all members of the family. On the server side, a proxy server manages several requests from various proxy clients, selects the appropriate content for the given context, and generates the response to be delivered. The proxy client receives the response and can perform a further adaptation taking into account the specific context of the final client. In this way, adaptation is distributed throughout the various levels.

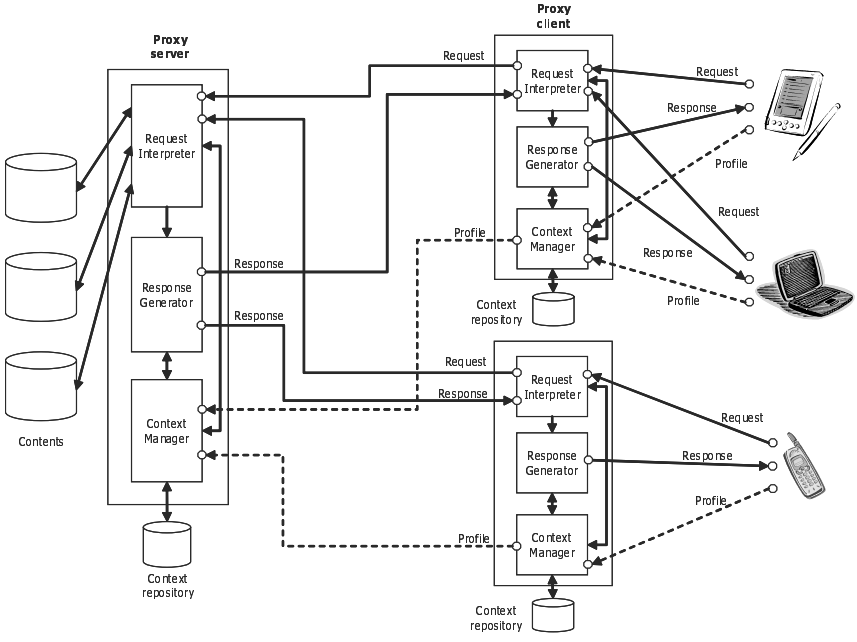

Fig. 8.13. A distributed architecture for the interaction manager

The fundamental component of this architectural scheme is the context manager, which should be able to:

- (dynamically) capture and classify (possibly heterogeneous) incoming client profiles, making use of a local repository of context information;
- coordinate the various (and possibly conflicting) adaptation requirements for a given profile;
- send to the response generator some *adaptation specifications* for all the levels of the response (content, navigation, and presentation).

To guarantee the flexibility of the overall system, this component should be extensible, in the sense that the various activities should be carried out for different types of profiles and according to orthogonal dimensions of adaptation, possibly not fixed in advance.

In Fig. 8.14 a possible architecture for the context manager that can meet these requirements is shown. The basic component of this module is the profile interpreter, which should be able to get and identify possibly heterogeneous profiles (e.g., CC/PP, XML, or HTTP headers) and translate them into a uniform representation. Such profile representations are taken as input by a series of modules, one for each adaptation dimension (e.g., the device characteristics, the user preferences, or the location).

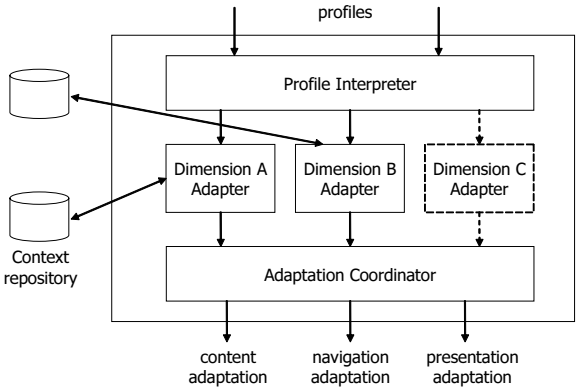

Fig. 8.14. An extensible context manager

The main task of these modules is to generate a uniform set of adaptation specifications, to be sent to the response generator, that satisfy one dimension's specific requirements. This work can be supported by a specific data repository, in which predefined or previously generated profiles and corresponding specifications are collected. In the next subsection, we show a possible implementation of an adaptation module for the user dimension.

Since each module can generate different and possibly conflicting specifications, coordination is needed to provide an integrated set of specifications

that take into account the various adaptation requirements and can be sent effectively to the response generator module. The adaptation coordinator is devoted to the execution of this task.

It is important to note that, owing to the uniformity of the representations and techniques used by the various adaptation modules, this scheme can be extended in a natural way: a new adaptation module can easily be added to satisfy the requirements of adaptation according to a previously unpredicted coordinate.

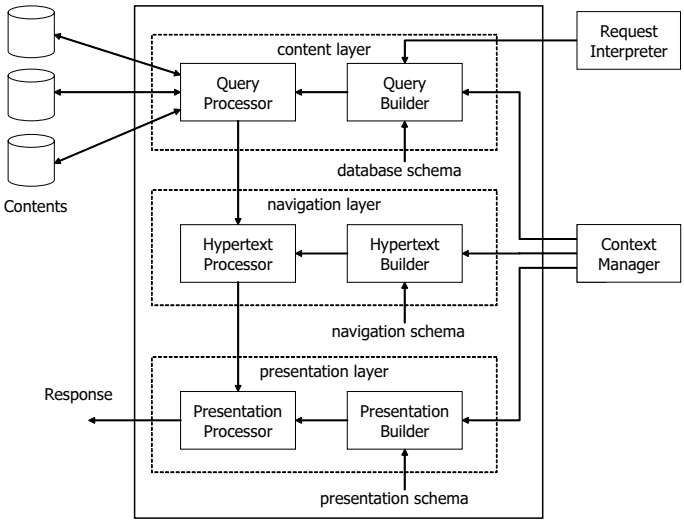

Fig. 8.15. A response generator

The response generator is composed of three modules (Fig. 8.15), one for each level of the response to be delivered over the Web. The first module combines the query returned by the request interpreter with the adaptation specification given by the context manager and generates a query to be executed by a query processor (possibly external to the system). The second module operates over the navigation scheme of the Web site (e.g., by splitting pages or adding links) to satisfy the adaptation requirements, as specified by the context manager. Finally, the third module is in charge of taking the specifications of the adaptation related to the presentation and implementing them with an appropriate style sheet, possibly using the presentation adaptation techniques described in the next subsection.

8.3.3 Adaptive Presentation by Matching User Profiles

In this subsection we focus on content adaptation according to the context of the user, namely presenting the information in such a way that the infor-

mation that is presented first is that which best fits the user profile. This of course requires that the user specifies a demand profile, and each piece of information is associated with a supply profile. A "piece of information" will be, in the context of the example discussed below, a tourist event; the approach, however, is applicable to several domains. Adapting the information content is also relevant in relation to the user interface; in the case of mobile devices, where only small pieces of information can be displayed at a time, presenting the relevant information first helps in reducing the navigation time (and in general the interaction time).

The problem of matching the demand and supply of goods or services arises in several fields, including real estate agencies, recruitment agencies, dating agencies, and advertising in general. The problem of matchmaking consists, in general, in matching a set of demand profiles to a set of supply profiles; the match should be the *best possible* one, since perfect matches are unlikely to be possible. Different approaches to matchmaking can be found in literature [187, 226, 347, 358, 374], based on bipartite graph matching, vector-based techniques, and record matching in databases, among other things.

With respect to other known approaches, the formalism that we adopt here for representing demand and supply profiles, first presented in [97], differs in the fact that it allows both the specification of incomplete profiles, and reasoning about profiles in the presence of conflicting and missing information; this formalism is borrowed from the field of artificial intelligence, and in particular from description logics. In our approach, a demand profile and a supply profile are considered, and they are "adjusted" so that the supply profile fully satisfies the demand profile. A *penalty* is associated with each adjustment; the overall penalty is the sum of all penalties generated during the matchmaking process and the larger the penalty, the less the demand and supply match. If the supply fully satisfies the demand, the penalty is zero.

The matchmaking technique described above has been incorporated into a prototype whose task is to recommend tourist events to users through the use of matchmaking. The matchmaking comes into play when the user accesses the system to retrieve data about tourist events that he/she is likely to be interested in. Our system presents the events sorted according to the penalty returned in the match to the demand profile, so that the events that should be of interest to the user are presented first. This is important when the user is using a small-screen and small-keyboard device: in fact, in this case a reduction in the number of steps required to navigate to the relevant information is desirable [60].

Several recommendation and matchmaking systems are known [51, 264, 269]; IM3 adopts the approach cited above [97] for modeling and reasoning about profiles, and it is accessible through mobile devices, in particular smart phones running J2ME. Users can store their demand profile in a centralized server, which describes the kind of tourist events they are interested in; supply profiles, each describing a tourist event, are also published by users.

Besides the obvious advantage of allowing users to access information about tourist events while carrying just a cellphone, there is another important reason for allowing access to the system through mobile terminals: our system offers the possibility to connect to local servers via Bluetooth. These local servers offer information only about events happening nearby, thus offering a location-based service.

Matchmaking Techniques in IM3

In our case, the problem is, given a demand profile and a set of supply profiles, to sort the supply profiles according to their capability to satisfy the demand profile.

When we consider a (supply) profile of an event, and a (demand) profile of a user, specifying the kind of events he/she is interested in, we face two different problems: (i) *inconsistencies*, where some of the characteristics of the event are in conflict with the requirements in the demand profile, and (ii) *missing information*, where the profile of the event does not have information about some of the characteristics specified in the demand profile.

In our prototype, we adopt techniques that are capable of dealing with the above problems by use of a clear, formal approach, in particular by using an approach based on *description logics* [34]. We adopt a special description logic, tailored to our needs: such a formalism is able to (i) represent quantitative information, and (ii) deal with conflicting and incomplete information by using suitable reasoning services. The IM3 system manipulates profiles that are descriptions of tourist events; each event supplier gives a description of the event, while each user specifies the kind of events he/she is interested in.

All of the matchmaking techniques in our prototype have to take into account the relationships that exist between concepts. For instance, if a user is not interested at all in sports events, and we have a supply profile describing a football match, the system needs to know that football is a sport. In order to allow the system to reason properly about concepts, a *domain ontology* is used in the server (and, when necessary, in the clients) to represent inclusion and disjunction relations among concepts.

Architecture of the System

The core of the whole system is the server. It stores supply and demand profiles, and runs the matchmaking algorithm on them. The server uses Java Server Pages technology and runs a MySQL relatioal database management system to store all the profiles. A Java listener also runs on the server, accepting connections via sockets from client software running on mobile clients. Periodically, the server computes the penalty between all demand/supply profile pairs; in this way, when a query is issued to the system, the answer can be computed quickly and the results presented to the user, ordered according to the penalty.

Local servers carry out the same job as the main server, but with two significant differences: (i) each of them stores only events located in the area where the local server is based; events are retrieved in a suitable way from the main server, according to their location (which is incorporated in the profile) and periodically update;, and (ii) owing to the local nature of the information stored in them, the local servers are only accessible by mobile clients via Bluetooth (and therefore from a distance of less than 100 m), so that users have information about events taking place nearby.

The central server is accessible on the Web by means of an ordinary browser. Mobile clients have the possibility of accessing the central server via the Web through a browser capable of processing XHTML Mobile Profile markup. Alternatively, they can run a Java midlet on their J2ME virtual machine that is able to connect to the central server, the local servers, or other mobile clients running the same software. A mobile client can store a demand profile (assumed to be that of the owner) and a limited number of supply profiles.

8.4 Usability and Accessibility of Adaptive Applications

The context of multichannel information systems poses new questions and challenges in the area of usability and accessibility. To assure effective design and evaluation of mobile interactive systems, the set of standard procedures, methods, and guidelines needs refinement to include new aspects such as mobility, context, and the limitations of the user interface. The following three subsections address aspects of this matter.

8.4.1 Guidelines and Principles for Accessibility and Usability

In the current development of information systems and services growing attention is being paid to the user, both to enable a larger number of people to take advantage of opportunities and facilities which they can put into practice, and to make the use of those opportunities and facilities easier. Since the aim of information systems is to improve our lives, focusing on user preferences and adaptability of the system, adaptivity becomes of paramount importance. The basis of the development of such systems and services lies in the principles and techniques that make information systems capable of addressing people with different needs and preferences; among these principles and techniques, the crucial concept of accessibility emerges as a significant achievement. In such a complex context of use, accessibility principles intended to benefit disabled people can exercise a very helpful influence on nondisabled persons as well.

Limitations of the Mobile Context

This complexity of the context of use derives from the increasing spread of information systems into our social life, as distributed services and mobile

devices tend to become ubiquitous, fostering the influence of the context much more than in the conventional desktop setting. The influence of the context implies a number of hardware and software limitations (listed briefly below) that must be taken into account, to assure accessible and usable systems and services. Devices have a small screen, support limited interaction (input), have limited bandwidth and high costs, with limited computational resources and limited availability (batteries), and have on a wide heterogeneity of operating systems and physical properties. In turn, the context is continuously changing, forcing small, focused interactions, while tasks tend to be fragmented, vaguely defined, and sometimes embedded in other activities. Supporting multitasking becomes difficult, and the context can cause limitations in hindrance of of the use of one or more channel (modalities).

Principles and Guidelines for Accessibility

Before discussing principles and guidelines, it is worth defining the difference between the two: a principle can be defined here as the most abstract design rule which can be applied in order to promote accessibility and usability, offering a way of understanding them in a more general sense. A guideline, in contrast, provides a direction for the design process, in both general and more concrete terms, in order to enhance the accessibility and usability of a system. It is to be noted that the more general the guideline, the more it resembles a principle; the more specific the guideline, the more it is suited to detailed design. As far as accessibility is concerned we must emphasize that it is a pervasive quality pertaining to every part of every system and service, since every single part of the delivery and enjoyment of a service, by means of one or more devices, must guarantee that a sufficient level of accessibility will be maintained to avoid compromising achieving of a result that is accessible overall. So the principles of accessibility developed over time following the progress of technology and the evolution of society, as an application of the principles of universal design, concern content as well as software, and, specifically, interfaces, and of course devices. This is in accordance with to the general principle of universal design: "the design of products and environments so that they are usable by all people, to the greatest extent possible, without the need for adaptation or specialized design".

Content and Interface Accessibility

User interfaces are a typical field where there is a possible limitation on accessibility. Adaptability of the interface to the user's needs, redundancy of information in the elements of the user interface, and the availability of more than one way to interact with a program or a service represent a range of solutions to the problem in this specific field. Guidelines and techniques to ensure the design of accessible user interfaces are available. The content of documents

and services is another field in which barriers can arise, with the result of the exclusion of some categories of users. As an example, multimedia content is related to specific sensory perceptual modalities so that persons with sensory limitations are discriminated against "normal" users. In this case, accessibility consists of using multimedia components to provide redundant information addressed to various sensory channels to ensure "equivalent" information for everybody; this is achieved by following specific guidelines. An enlightening example, concerning the Web field, is the set of internationally recognized guidelines produced by W3C, namely WAI (the Web Accessibility Initiative of the World Wide Web Consortium), which plays a critical role in making the Web accessible in general; however, these guidelines also have an impact on the mobile context. They explain (i) how to create accessible Web sites; (ii) how to design software that supports the production of accessible Web sites, such as authoring tools; and (iii) how to design accessible browsers and other user agents.

Device Accessibility

Devices represent another challenge for accessibility; in fact, in the everyday context of information systems, the increasing spread of distributed services, which are tending to become ubiquitous, fosters the utilization of a vast range of devices with extremely varying characteristics. The few examples that follow will show how far we are from a satisfactory degree of accessibility where devices are concerned. At present, general screen readers for PDAs and cellphones do not exist, so that these devices become inaccessible for most of their functions. Even if some operating systems allow the user to control devices via vocal commands, it is still not possible to browse the Web or perform a number of other important operations. Another relevant obstacle lies in stylus-based interaction, which cannot be adequately replaced by physical buttons: browsing without a pen is generally cumbersome and slow even if a screen reader is available. In contrast, when a PDA hosts an application which is capable of producing voice output and is explicitly designed to be used with buttons, interaction is possible even with very little training. In the case of persons with limited vision, the limited possibilities for high-contrast configurations with a PDA screen are an obstacle to using such devices. It is worth noting, however, that the screen size of a PDA would probably limit use of this device even in the presence of high-contrast configurations. On the other hand, when disabled persons are permitted to interact in an appropriate way, they often show substantial equivalence to nondisabled persons in device usage.

Usability Principles for Mobile Computing

Usability principles, in turn, play a primary role in designing systems and services that are effective, efficient, and satisfactory: the discussion that follows

is based on the idea of usability as defined by [148]. The principles can be classified into three main categories: learnability, flexibility, and robustness. Some of these principles acquire much more importance in the context of mobile computing than in the conventional desktop setting, for example responsiveness. On the other hand, and taking into account the challenges facing mobile computing, some other principles would should be employed cautiously in the ubiquitous setting, for example task migratability.

Learnability

Learnability refers to how easy it is to learn and remember functions and modalities provided by the system and can be seen as made up of: predictability, synthesizability, familiarity, generalizability, and consistency. A limited output is likely to increase the memory load, so that the system is less predictable. At the same time, these limitations tend to reduce the perception of internal changes that can be easily modified by contextual activities, making it difficult for the user to synthesize, while dialogues and the information architecture are simplified, producing an opposite, beneficial effect. *Familiarity* refers to the ability of a user to determine how to initiate interaction when the interface is encountered for the first time; in the mobile context it is actually very critical owing to the wide heterogeneity of operating systems and of the physical properties of devices, as is generalizability. *Consistency* of input/output, with respect to the meaning of actions in some conceptual model, can present possible consistency flaws owing to the habit of using the keyboard and to the various styles of signaling contextual events and information.

Flexibility

Flexibility refers to the extent to which the user and the system exchange information and control. It is made up of the following parts: dialogue initiative, multithreading, task migratability, substitutivity and customizability. *Dialogue initiative* is one of the principles that is most affected by the mobile context, in fact. While the usual suggestion is to minimize the preemptive dialogue of the system, in the mobile context it could be crucial, since the limited computational capabilities of devices can further limit complex activities. *Multithreading* is very limited in mobile devices owing to many factors; for example, the limited screen size and computational capability reduce the ability to run multiple applications at the same time, and switching between different application can be very cumbersome on a limited device. *Task migratability* refers to the ability to transfer control of tasks between the system and the user: for instance, a ubiquitous computing system can be used to run tasks that are mundane, routine, repetitive and obvious. *Substitutivity* is the extent to which an application allows equivalent input and output values to

be substituted one for the other: in the the case where input has to be explicitly specified, the application should make this as convenient/easy and as customizable as possible. For instance, the context could form the input to a particular task. *Customizability* refers to the ability of the user or the system to modify the user interface. For example, the presence or absence of some user interface objects and features could imply that the computing resources would be strained, or could even directly mean a higher bill for the user. Even in ubiquitous computing, situations can arise where the user runs across unfamiliar, complex, and intimidating technical (and contextual) choices and decisions are commonplace. Such situations provide great opportunities for the system to adjust the application on line with respect to the characteristics of the device in use.

Robustness

This refers to the level of support provided to the user for achieving and assessing goals successfully, and is made up of the following properties: observability, recoverability, and task conformance.

8.4.2 Heuristic Evaluation of Usability and Accessibility in Mobile Computing

Heuristic evaluation (HE) is a popular usability evaluation technique that permits one to evaluate user interfaces easily. It is a popular example of "discount usability", a set of usability methods that permit one to evaluate interactive systems by employing limited resources, and it can be easily introduced into various stages of the product life cycle, especially early in the development process, since it does not require either a functioning system or real end users. Its popularity, especially in industrial applications, is probably due to the fact that it is easy to learn and to implement. It requires three to five usability experts to inspect a user interface using a set of heuristics as a reference. Each expert goes over the functions of a system and uses the heuristics as a mnemonic guide to remind him/her where to look to find potential usability flaws. The results of the experts are then compared with a past test session to produce an integrated usability report. The use of HE in the context of mobile applications is promising, since all the benefits of the method are still valid in this new context and it can be easily used to evaluate mobile interfaces. The full applicability of it, however, is limited by the particular properties of mobile applications. In particular, the role of the context is prominent in this new environment, and a thorough evaluation must take into account the fact that many potential usability problems can arise from the specific context/situation in which the user and the application are immersed. The problem of introducing elements that take the context into account in the evaluation is not new. Many have argued that the context is an important

variable to consider when evaluating interactive systems; nonetheless, only recently, with the advent of mobile applications, it has acquired a major role. Moreover, since mobile devices, although they provide many benefits, impose new limits on the interface, the evaluator needs to take those limits into account. More precisely, the evaluator should take account of the fact that mobile devices suffer from a small screen, limited input, limited bandwidth, high costs, limited computational resources, limited availability (batteries), and wide heterogeneity. The explicit introduction of tools to deal with these limits during evaluation is thus necessary.

Background

The use of heuristic evaluation in nonstandard settings has attracted some interest in recent years. Mankoff et al. have proposed in [257] a revised HE method for evaluating ambient displays. The basic idea is to revise the standard set of heuristics by deleting those that do not apply to the specific context and by adding some new ones dealing with specific features of ambient displays. The approach is interesting in that it proposes the general method of revision of heuristics as a way to tailor and extend HE to nonstandard settings. A similar approach is followed in [36], where a revised set of heuristics and a development methodology for computer-supported cooperative work applications are proposed. Unfortunately, similar approaches in the context of mobile applications do not yet exist. Following a different trail of thought, others have investigated the use of HE in mobile computing with some enhancements to capture important details about the context. In [306], two variants of HE are compared: one in which HE is used in conjunction with some scenarios intended to capture contextual details, and another in which usability experts conduct a field study. The use of contextual cues is carefully analyzed in [218], in which laboratory studies and field studies are compared. The benefit of testing applications in the field seems to not pay for the increase in cost, time, and setup procedures.

Expert-Based Evaluation in Mobile Computing

The use of HE and, in a broader sense, all inspection-based evaluation techniques seems to be promising and still valid for mobile applications, but these techniques surely need some fine tuning. It is necessary to find a way to include contextual cues and aspects related to mobility, and it is also necessary to aid the evaluators in taking into account the particular limits on interaction in the case of mobile systems.

We see two broad classes of interventions: one possibility is to extend the exiting method with additional steps or tools, for example, written scenarios, contextual simulations, field tests, or video reviews [173], which permit the evaluator to perform a thorough analysis of the usability problems that can arise with a specific application. Another viable way is to refine the existing

set of heuristics/principles by introducing new aspects that explicitly deal with context, mobility, etc. The first approach has the benefit of eliciting new information by use of tools that directly explore contextual features, but this is done to the detriment of the original approach, and will eventually affect its simplicity. Extending the set of heuristics might have limited impact but it has the benefit of leaving the original method intact, probably preserving its simplicity. We have explored two methods that cover both classes. One method is based on the idea of supporting the evaluator with video data. We proposed and investigated this idea and obtained some interesting results. Video data provides evaluators with a more detailed understanding of the characteristics of users and the context of their interaction, leading to an improvement of the assessment in terms of the total number of flaws of the system detected. Another method that we have investigated provids a new set of guidelines that take into account specific aspects of mobile systems. The evaluator can use a map of common issues as a reference to inspect the user interface and find potential usability flaws. The map is based on a series of high-level issues that become more specialized as one goes deeper into the hierarchy. The basic classes of issues are the context, interaction with the device and the infrastructure, interaction with the application, cognitive issues, personalization, and social issues.

8.4.3 User Studies on Mobile Computing

While it is acknowledged that there are gains associated with mobile computing (such as ubiquity and portability), it is no secret that there are also pains (such as inherent device limitations, input/output challenges, and contextual factors). Usability evaluation is no exception in this respect. Usability evaluation has to come to terms with the ramifications of mobile computing, for instance:

- In this era, the need to take the real-world context into account has become more crucial than at any other time in the history of computing.
- In mobile settings, context-structured activities are based on a context that is more likely to change than in standard settings and often in complex and unexpected ways. Task-centric methods may not be directly applicable in evaluating mobile systems [5].
- The proliferation of systems and devices makes it difficult for the expert to know the limits and capabilities of the devices. Moreover, there are no solid models that describe the behavior of mobile applications, especially for those that include context sensing and preemptive behavior.
- The technology required to develop mobile systems is often cutting-edge technology. Developing a reliable and robust mobile system, therefore is not easy. In fact, most of the present effort is still at prototype level and is thus not robust [4, 5].

User-Based Methods in Mobile Computing

Although evaluation methods can generally be categorized into expert-based methods, model-based methods, and user-based methods, the work described here focuses on user-based methods in mobile computing. From a desktop-computing perspective, typical user-based methods include questionnaires, interviews, controlled experiments, observational methods, and physiological-monitoring methods [148]. Some of the conventional user-based evaluation methods can be applied to evaluate particular mobile applications. In other cases, these methods would need to be revised. There are also cases where it might be necessary to introduce evaluation methods that are unique to the mobile computing arena. While conventional methods such as interviews and questionnaires pose a challenge when one is targeting mobile applications, user-based tests tend to be even more challenging. There are various user-based techniques that can be employed in order to gain a richer understanding of the real-world setting. Such techniques tend to be nonconventional in traditional studies of human-computer interaction; they include ethnography, cultural probes, and contextual inquiry [4, 148]. Such methods can be used to complement the conventional user-based methods when it comes to evaluations involving mobile applications. Ethnographical methods concentrate on the everyday and routine/common aspects. They often require that the ethnographer be mobile. They also allow longitudinal studies. The foregoing is especially interesting considering that in mobile computing, the skills of the user often develop over some period of time. Ethnographical methods therefore tend to fit some aspects of mobile computing. Cultural probes and contextual inquiry also tend to be related to some aspects of mobile computing (such as the longitudinal and contextual aspects). Cultural probes are intended to uncover the emotional, uncommon, and spiritual. Although contextual inquiry resembles the ethnographical methods in the fact that it studies the user in context, it differs from the ethnographical approaches in that its "intention is to understand and to interpret the data gathered with the explicit aim of designing a new system" [148]. Ethnographical methods tend to be open-ended. A brief description of the various ways in which ethnographical methods can be applied in mobile computing is given below:

- Observing the users in the mobile computing setting as they interact with the system, without (or with) their knowledge [235].
- The user observes himself/herself and writes his/her observations down (regularly, e.g., daily in a diary [235]).
- Following the users around as they interact with the mobile application, with occasional interruptions in order to ask them relevant evaluation questions [145].
- Subjects involved in the evaluation have a pager that occasionally interrupts them with evaluation questions [145]. The method is referred to as a "beeper study". It tends to be less intrusive and the subjects may be more expressive.

- The system automatically (and remotely) logs user actions and activities so that a complete record of these can be analyzed to extract information about usage frequencies, errors, correlations, etc. This is in general not intrusive, but mature and standardized technology is still lacking, mainly because of device heterogeneity.

Another approach that could be used for evaluation in mobile computing is the Wizard-of-Oz technique; other simulation techniques, for example virtual reality could even be used. Such methods are especially appropriate when the mobile application is not fully complete [145]. However, the simulation should closely reflect the real context as much as possible, which is a nontrivial requirement.

It is also worth mentioning that we could also augment evaluation methods with video data. The idea is to use video representations of typical interactions happening in the real-world context as a way to support imagination and immersion in the real setting. These could be used as a support for methods for users (and even experts).

It is interesting to observe that researchers are deploying mobile devices into various real-world settings (e.g., libraries and museums) and setting up "living laboratories" by creating test beds for advanced research and development in mobile computing [4].

Some Parameters of User Studies in Mobile Computing

Designing user studies requires that various parameters be taken into consideration. In the following, a discussion of some of these parameters, from a mobile-computing point of view, is given.

Subjects

As in traditional evaluations, the subjects should be drawn from the user population. Regarding the number of subjects, using too few subjects may not provide reliable usability results whereas using too many subjects may not bring in any additional worthwhile results; the latter may in fact be a waste of resources. The debate about the minimum number of subjects for carrying out an evaluation test has been running for years.

Nielsen [283, 284, 285] and Virzi [376, 377] consider that five subjects are adequate to identify most usability problems with an application. However, several studies have challenged this finding on methodological and empirical grounds. For instance Spool and Schroeder [350] question five as the minimum number, as being too small for Web-based applications. Molich et al. [272, 273] observe too that it would take many more than five subjects to uncover all the usability problems with a Web-based product. Faulkner has carried out an evaluation on a Web-based product [160] and found that, on average, Nielsen's

suggestion was right. On the whole, however, her findings do indicate that a single usability test with five subjects is not sufficient.

Whatever line of argument is adopted and whatever the conclusion may be, several considerations are important: (i) there should be a clear definition of the user profile for mobile applications; (ii) more mature products or applications, which may have been subjected to various formative evaluations and corresponding improvements/refinements, may require more subjects; (iii) evaluations should be performed iteratively during the design process so that the application will ultimately be tested by a reasonably large number of subjects.

In the early stages of the life cycle, testing with a relatively small number of users (such as five to ten) for each user segment might be sufficient to identify most of the problems with navigation, and with the basic and overall design of the mobile application. Later on in the life cycle, quantitative tests can be performed. Such tests tend to include some significant or substantial statistical analysis. In such cases, a larger number of subjects is necessary.

What to Evaluate

In mobile computing, there are various aspects that can be evaluated. From the perspective of usability, some of the common aspects are task accuracy, navigation (e.g., browsing methods), input effort, efficiency, and the user/interaction experience. Such aspects are often assessed by collecting and analyzing performance measurements such as the time taken by the subject to complete a task successfully; the number of pages, screens, or steps the subject went through before completing a task; the number of tasks the subject completed successfully; the number of tasks the subject abandoned; the number of errors the subject makes before completing a task; and the number of times the subject asks for help before completing a task. Subjective measurements such as: spontaneous comments and ratings of ease of use. The test could be designed to use both independent and dependent variables. Typical independent variables in mobile computing include laboratory vs. real-world setting/field, real-world setting/field vs. simulated setting, and real application vs. simulated application. An experiment could have additions or variations within each of these variables. More examples of independent variables are definitely possible. Performance measures often serve as dependent variables.

Simulators and Emulators

While we are discussing the number of subjects to enlist for a usability evaluation test, it was noted that evaluation should start early in the life cycle, should be iterative and should incorporate various appropriate evaluation methods. Evaluation should thus commence well before the development of the mobile application starts. However, even when the time comes for the real application

to be developed, mobile computing often demands a lot of resources (some of which are not as easily available as in desktop application development). Various development environments/tools that enable the developer to realize some emulators or simulators exist. Despite this, emulators and simulators are not the real devices or applications and therefore should not be expected to provide exactly the same user experience as those real devices or applications. This is even more so the case in mobile computing, considering the key role of context. If the evaluator has to rely on simulators or emulators, one of the guidelines is to ensure that the simulator or emulator resembles the real application or device as much as possible.

Setting

User-based methods can be applied in the laboratory or in the field. In the laboratory specialist equipment is often available and the environment is relatively uninterrupted. However the laboratory lacks the real-world context of usage. In mobile computing, the impact of the context is crucial in assessing the usability of a mobile application. The context could involve aspects that are difficult to assess in the laboratory, such as interruptions and social interactions. On the other hand, the field environment often offers a natural environment in which the context is retained. However, the evaluator (and subjects) often have to reckon with real-life factors such as danger, distractions, and interruptions.

In previous usability research on mobile applications, a strong bias has been observed toward conducting laboratory evaluations instead of field studies, leading to a prevalent focus on the assessment of device functionality and thereby ignoring contextual issues affecting use [217]. While this trend could be attributed to some of the challenges that mobile computing presents for evaluation, on the basis of what is said in [148, 217], the trend could be due to the following specific factors:

- the difficulty of simulating mobile, real-world conditions of use in a laboratory;
- little is known or documented about the physical settings;
- the complexity and effort required for data collection and control of variables during field studies;
- some types of systems are more easily or better evaluated in the laboratory rather than in the field, for example safety-critical applications.

If the evaluator chooses to or has to re-create the real-world setting or environment (e.g., by use of virtual reality), he/she should ensure that the re-creation is an appropriate and good enough match to the real setting.

Usability Evaluation: Summary

Many user-based evaluation methods exist. For applicability to mobile computing, some of the conventional user-based evaluation methods might need to be revised or customized. In some cases, novel evaluation methods that are unique and relevant to mobile computing might be worthwhile. On the whole, no single individual evaluation method can truly identify/capture all the usability problems with a mobile application (or provide all of the information required about its usability). Moreover, the goals of the evaluation of specific mobile applications could vary, and therefore different approaches might be required and need to be integrated in order to realize such goals. The usability evaluation of particular mobile applications or systems therefore requires an integration/combination of various approaches.