2

Reference Architecture and Framework

M. Adorni, F. Arcelli, S. Bandini, L. Baresi, C. Batini, A. Bianchi, D. Bianchini, M. Brioschi, A. Caforio, A. Calì, P. Cappellari, C. Cappiello, T. Catarci, A. Corallo, V. De Antonellis, C. Franza, G. Giunta, A. Limonta, G. Lorenzo, P. Losi, A. Maurino, M. Melideo, D. Micucci, S. Modafferi, E. Mussi, L. Negri, C. Pandolfo, B. Pernici, P. Plebani, D. Ragazzi, C. Raibulet, M. Riva, N. Simeoni, C. Simone, G. Solazzo, F. Tisato, R. Torlone, G. Vizzari, and A. Zilli

2.1 Introduction

The goal of the MAIS system is to provide support for flexible and adaptive execution of applications in a distributed, multichannel, mobile information system. In such a system, a fundamental requirement is an ability to describe the continuously evolving execution environment and user characteristics. Service requests are therefore satisfied by considering both the request itself and its provisioning environment.

The first part of this chapter presents the general architecture of the MAIS system. The MAIS architecture allows us to define a set of "pluggable" modules which can be composed to provide adaptivity at different levels in the MAIS system. The main architectural components are introduced in Sect. 2.2; more details of the components are provided in the rest of the book.

The MAIS reference framework, illustrated in the second part of this chapter, provides the essential basis for all of the adaptive mechanisms that are illustrated in the book. The MAIS reference framework defines a common understanding of the elements of a mobile information system that are used to enable communication among the various modules of a MAIS system during information exchange and service provisioning. The reference framework is composed of a set of models: the functional model, the architectural model, and the context model. These are described in the second part of this chapter.

2.2 The MAIS Architecture

Four main architectural elements can be identified in the MAIS architecture (see Fig. 2.1):

- *front-end components*, which focus on supporting adaptive user interaction and on building adaptive Web applications;

- *back-end components*, which focus on adaptive service provisioning in a mobile information system, with variable context and provisioning channels;
- a *reflective architecture*, a common infrastructural element which provides information about the state of the architectural components and their related quality of service, as well as support for gaining information about the context and profile of the user;
- *infrastructure components*, which support adaptivity in mobile devices at the architectural and network levels, and provide basic data management functionality.

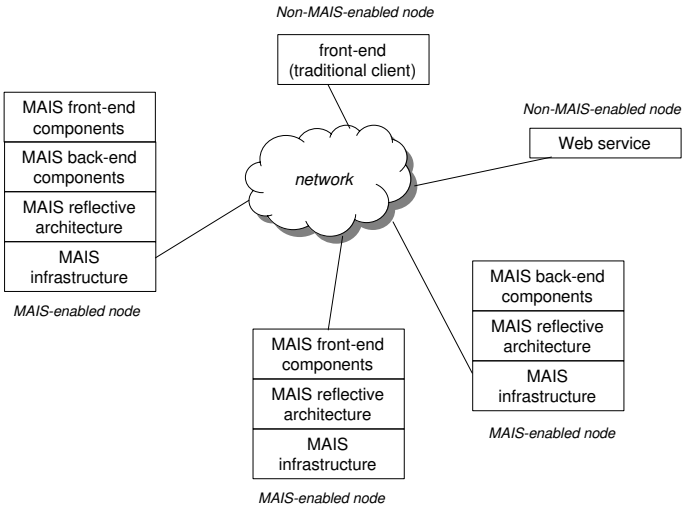

Fig. 2.1. MAIS components

When the architecture is deployed, not all of the components have to be MAIS-enabled. In fact, the front-end and back-end components of the system can either be all MAIS-enabled or a hybrid set of MAIS-enabled and traditional components. The use of MAIS-enabled front-end components provides us with front-end adaptivity, while MAIS-enabled back-end components provide us with flexible services. If we decide to adopt traditional components, we can integrate them at the front end in the form of traditional clients, and/or at the back end under the form of Web services. In general, however, the use of mixed components will cause the system to provide a lower degree of adaptivity.

2.2.1 MAIS Front-End Environment

In the MAIS front-end environment, we focus on two aspects of adaptivity (see Fig. 2.2):

- In the *mobile flexible deployment environment*, we study how adaptivity may be supported at the device and network levels. Core technologies are involved: we study mechanisms for adaptive transmission on mobile and automotive wireless networks, presented in detail in Chap. 5; very small database systems which can be deployed on mobile devices or smart cards, discussed in Chap. 6; and adaptive hardware architectures that can help to lower power consumption, illustrated in Chap. 7.
- In the *front-end adaptivity tools environment*, the focus is on tools that help us to design or support adaptive applications. We study how to design adaptive context-aware Web applications, and how to provide adaptation during interaction and in selecting and adapting contents, taking into account user preferences and profiles. This is done by focusing on issues related to the use of small interaction devices in mobile settings, as illustrated in Chap. 8, and on multimodal interaction for users with difficulties in interacting with the system, discussed in more detail in Chap. 11 for the case of the e-learning application domain.

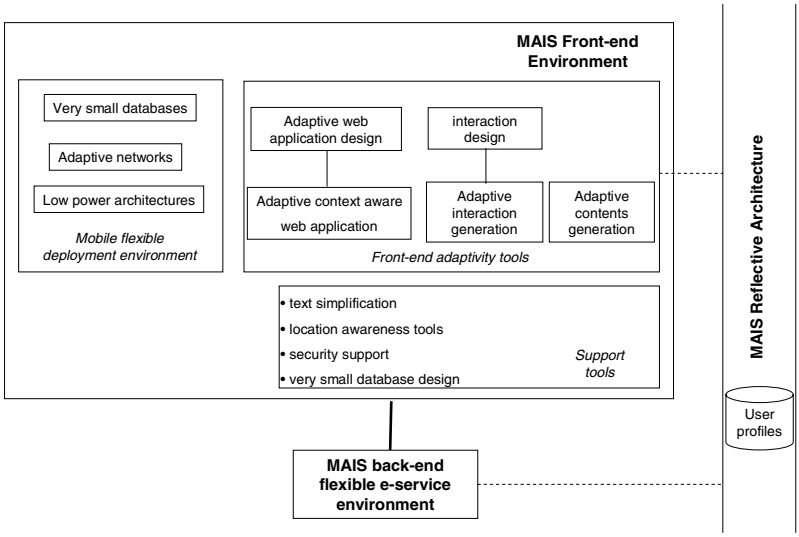

Fig. 2.2. MAIS front-end components

The MAIS front-end environment also contains a number of *support tools*, which are considered and illustrated in later chapters within a description of the components of the mobile flexible deployment environment and of the

front-end adaptivity tools. We shall mention here text simplification, which allows content to be dynamically modified to present textual information at the appropriate level of detail according to the user's needs; location awareness tools, which underlie many of the adaptivity functions, and provide users with tools to modify directly or indirectly their user profile and context information; security support, which is needed to guarantee that information is managed according to the required security level; and very-small-database design tools, to design the structure of databases on small devices taking into consideration both the user's requirements and the constraints posed by small devices.

An example of front-end adaptivity in the tourism application domain is the ability to provide access to services close to the tourist, and personalizing the selection to his/her preferences: for example, for a vegetarian a list of restaurants in the area would be sorted listing vegetarian restaurants first. Interaction can also vary: the list can contain very little detail if it is displayed on a small device with a limited screen size, while it could be richer in detail on a desktop; vocal interaction may be supported, for example when the user is driving and he/she has difficulties in looking at a small screen. User profiles may be stored in small smart cards tailored to various applications, for example for obtaining access to health services in a tourist area where the user is not resident when he/she needs special care.

2.2.2 MAIS Back-End Environment

The MAIS back-end environment focuses on adaptive service provisioning (Fig. 2.3): various services may be selected to provide a given functionality, or their adaptation might be required, according to the service requests from the front-end environment and their context of invocation. The focus of the adaptivity is a flexible process orchestration, where the services invoked in the process are selected in a flexible way, allowing the invocation of abstract services which are executed by selecting, negotiating, and optimizing service quality in order to provide flexible, high-quality services.

Such adaptive service provisioning is provided by the flexible service invocation and orchestration environment, illustrated in Chap. 3. As in service oriented architectures, services are selected from an extended service registry (the MAIS Service Registry), in which service descriptions are enriched with functional and quality-of-service descriptions. A knowledge-based recommendation environment provides support for analyzing user requests so as to be able to recommend the most appropriate services for a given user on the basis of his/her preferences and characteristics; the components of the knowledge-based recommendation environment are presented in Chap. 10. In Chapter 9, the design of back-end flexible services is discussed, proposing a methodology for the development of services for mobile information systems.

There are examples of adaptive service provisioning in the tourism domain linked both to service selection and to the execution of new added-value services composed of several services. Service selection can retrieve services

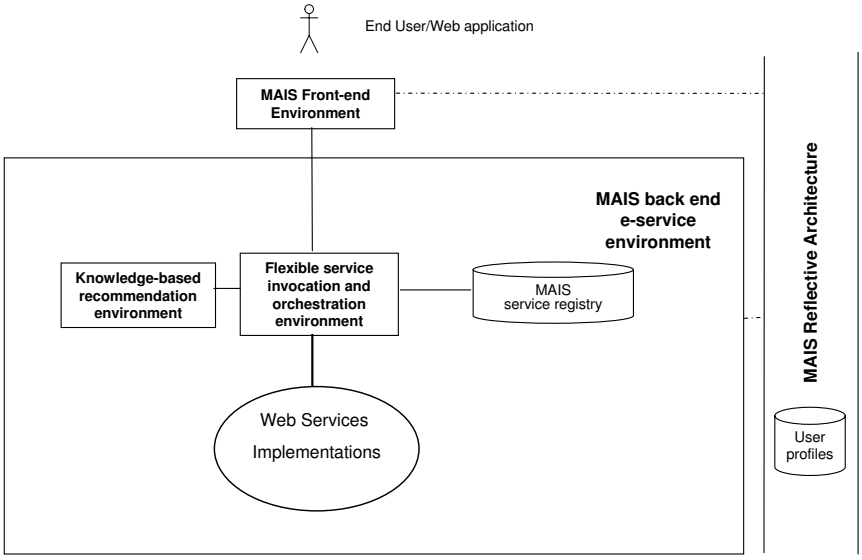

Fig. 2.3. MAIS back-end components

relevant to a user in a given category according to his/her profile and preferences: for example, if an accommodation service is requested by a student visiting a town on a low budget, the selection of services will include youth hostels and lower-category hotels, as well as bed and breakfast accommodation. Composed services may include the preparation of a travel plan and the reservation and payment of several services. On the basis of a common general travel preparation schema, the selection of services can take into consideration individual preferences, previous choices, local constraints such as the time and location of the service, and global constraints such as timing constraints. The services offered may vary according to the modality of interaction of the user: for example, if the user is traveling and using a small device for planning his/her trip, he/she might receive electronic tickets, whereas more traditional delivery services might be used to deliver tickets if the user is planning his/her trip from home instead.

2.3 The MAIS Framework

The MAIS reference framework provides a common understanding of mobile information systems and their components which is used by all modules of a MAIS system. The framework is rather general and can be adopted as an underlying basis for providing a common conceptual structure for all types of mobile information systems, independent of the MAIS architecture illustrated

in the previous section. The MAIS reference framework is composed of the following models:

- the *functional model*, for describing services and related content, according to the double perspective of service providers and service requestors;
- the *architectural model*, for describing device components used to access services, the behavior of each component, and how they interact with each other;
- the *context model*, for describing the context in which services are required and used, and the specification of user profiles and provisioning channels, in terms of devices, networks, network interfaces, and application protocols used to access services.

2.3.1 Functional Model: E-Service General Model

In service oriented architectures (SOAs) [17], a service is seen as a set of operations offered by one party to another: service providers create and hold the services, while service requestors first look for the best service in a service registry according to specified matching criteria, and then invoke the service directly from the provider.

We consider two modeling perspectives related to the provisioning and request of services (see [45]):

- the *service provisioning model* specifies the providers of the service, what the service does, and how to invoke its functionalities, according to the available or negotiable quality of service;
- the *service request model* specifies the invocation context of the service requestor the requested provisioning channel, and quality of service requirements.

The goal of these two models is to provide the general framework for adaptive service invocation.

Service Provisioning Model

Figure 2.4 presents the main elements of the service provisioning model. An *eService*, described by means of a name that identifies it, a short textual description, and a service category (such as, "commercial service" or "information service"), may be provided through one or more *channels*. Each service is provided by a *ServiceProvider* and possesses a *FunctionalDescription* that describes its operational aspects. Services can be composed in a recursive manner to create *CompositeEServices*, which are described through workflows (using an extension of WS-BPEL [133] as described in Chap. 3, referred to as BPEL in the following), where component services are connected by means of control constructs.

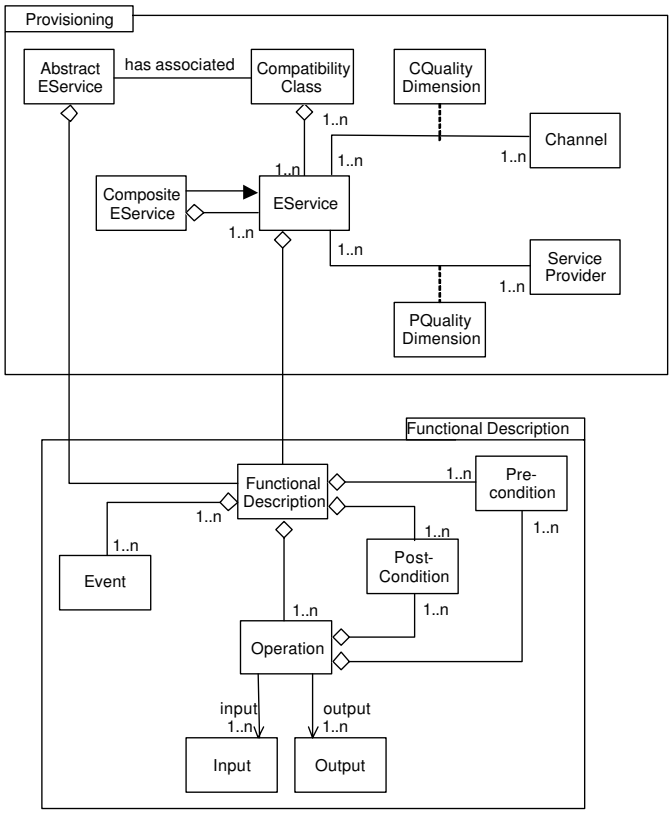

Fig. 2.4. MAIS service provisioning model [45]

The *FunctionalDescription* expresses what the service requires from the users or the agents that invoke it, and what, in turn, it provides them with. This is done by means of a set of operations, events, preconditions and post-conditions. Each operation has a name and a short textual description of what the operation does. In particular, it provides information as to whether the service requires one or more inputs, gives back one or more outputs, and it is associated with a set of preconditions and/or postconditions, that predicate on these Inputs and/or Outputs. Preconditions and postconditions can also be defined on the whole service. Inputs and outputs are described by a name and a value type. Pre- and post-conditions are logical expressions that must be verified before and after the execution of the operation or the whole service. Events are used to model external actions that are asynchronous with respect to the normal flow of the service. Each event has a name that identifies it

and a type, such as "temporal event", "data event", and so on: for instance, a temporal event is a timeout that occurs during the execution of an operation.

The service provider is described by a name and standard address information. An association class *PQualityDimension* expresses the quality parameters guaranteed by the service provider (PQoS), for example data reliability, provisioning time, service availability, and service price.

To pursue adaptivity, a service can be provided through one or more channels; an association class *CQualityDimension* represents the quality of the service with respect to the channel used (CQoS). Several parameters to express quality of service can exist, such as response time, channel availability, usability, accessibility, integrity, bandwidth, reliability, and price. Channels are modeled as conceptual channels, according to the context model described in Sect. 2.3.3. Services are grouped into *compatibility classes* for substitutability purposes. A compatibility class is associated with an abstract service, that represents the service required in a process execution in terms of the functionalities that it provides. A compatibility class group reunites, on the basis of a predefined similarity criterion based on a comparison of functional descriptions, services that are able to substitute for each other in satisfying the abstract service considered. A service can belong to more than one compatibility class at the same time. The mechanisms for classifying services into compatibility classes and for their selection according to a set of requested characteristics are described in Chap. 3.

Table 2.1. Example of service description

General description	**Name**	FlyAll Flight reservation
	Description	This service allows booking a flight
	URI	http://www.FlyAll.it/FlyALL.wsdl
	Provided By	<name>CCII FlyAll</name> <phone>+39 123 456 789</phone> <email>info@flyall.com</email> <physicalAddress>Via Roma 33, Milano, Italy</physicalAddress> ...
Functional attributes	**Operation**	FlightReservation
	Input list	NumberOfPersons, DateOfArrival, DateOfDeparture, FlightNumber, CreditCard
	Output list	BookingReceipt, ...
	Preconditions	CreditCardIsValid, ...
	Postconditions	ChargedAccount, ...
Extra functional attributes	**Service Category**	Travel
	Compatibility Class	FlightReservationAndPayment
	Channels	www, email, call center, SMS
	PQoS parameters	Security, Accuracy, Cost, Stability, ...
	Additional parameters	EconomyClass, BusinessClass

An example of a service description is shown in Tab. 2.1. The service provisioning model provides the basis for the MAIS Service Description Language and the flexible service provisioning described in Chap. 3, and for the service knowledge management illustrated in Chap. 10.

Service Request Model

Figure 2.5 presents the elements that define the service request model.

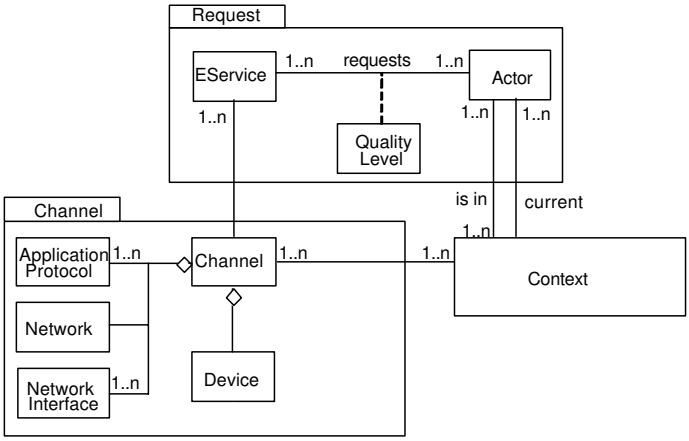

Fig. 2.5. MAIS service request model [45]

The actor class models the user (or software application) looking for services. The actor requires one or more services with a given quality level. The quality level requirements could be response time, availability, security, conformity to standards, or service price.

When a service is provided, the information that is presented to the user can vary according to the context. We assume that an actor, during interaction with a system, can work in more than one context, but only one of these is his current context at a given instant. According to the context model described in Sect. 2.3.3, the context is associated with the possible provisioning channels described at the logical level in terms of devices, application protocols, the network interface, and the network. Other information defined in the context model, such as location and user profile, is also relevant for providing adaptive service invocation, as discussed in greater detail in Chap. 3 for back-end adaptivity and in Chap. 8 for front-end adaptivity.

2.3.2 Architectural Model

The architectural model has the objective of providing a representation of the application and of the service execution environment.

Figure 2.6 shows the MAIS architectural model presented in [25, 249]. The physical layer contains the physical objects of a system. The technological layer (i.e., operating systems, network services and other basic software components) renders the system objects visible via platform-dependent mechanisms. The *base reflective layer* defines basic reflective classes, allowing quality of service (QoS) [112, 181] to be observed and controlled in a platform- and standard-independent way. According to the principle of separation of concerns, such classes do not embed any policy/strategy. The *Extended Reflective Layer* embeds strategies which may provide an improved level of QoS. It enables the definition of composite components and, in particular, channels as aggregations of devices, network interfaces, network services, and application protocols as described in Sect. 2.3.3. Furthermore, it defines mechanisms for the management of events.

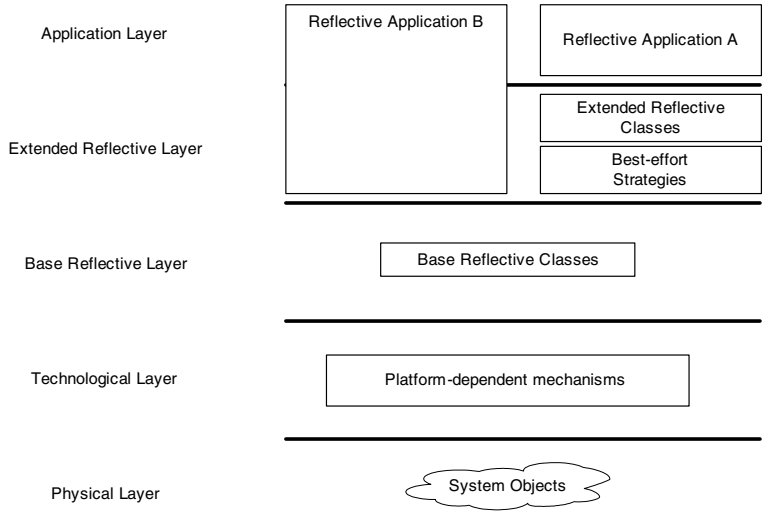

Fig. 2.6. MAIS reflective architecture

The *application layer* consists of higher-level software components that exploit the platform-independent functionalities provided by the lower layers. Such components will be referred to in the following as *applications*. In this perspective, the components that implement the MAIS framework are viewed as applications by the underlying architectural layers. In general, applications perform their job by exploiting domain-specific knowledge and by manipulating the system objects via functional, nonreflective features (not highlighted

in Fig. 2.6). *Reflective* applications exploit both domain and reflective knowledge, i.e., knowledge about the internal architecture of the system and about the QoS of its components. They either rely on best-effort strategies (reflective application A) or implement domain-specific strategies on top of the base reflective layer (reflective application B). The base and extended reflective layers provide the same interface to the application layer. In the next section, attention will be focused on the base reflective layer. A complete description of all MAIS reflective classes is available and can be downloaded from the MAIS Web site. The reflective mechanisms for propagating knowledge about system objects are illustrated in Chap. 4.

Base Reflective Layer

In the MAIS reflective architecture, the QoS features of system objects (computational components, devices, network components, and so on) are represented by reflective objects (*R_Objects*), i.e., instances of reflective classes (*R_Classes*).

The main reflective classes defined in this layer are presented in Fig. 2.7. A *FunctionalObject* models the functional, nonreflective aspects of system objects as exposed by the technological layers which implement the functional aspects of system components. *R_Object* is the superclass of any reflective class. Its methods allow the QoS of an *R_Object* to be observed and controlled. QoS is modeled by specific classes: since QoS features depend strongly on the application domain, their definition can be changed without affecting the definition of the base reflective classes. The reflection mechanisms are modeled by the association that exists between *R_Object* and *FunctionalObject*s. The five major specializations of *R_Object* are the following:

- *R_Component* models computational components whose QoS features are significant for the current application domain;
- *R_Node* models network components which route information flows through *R_NetworkServices*;
- *R_NetworkLink* models the QoS of an actual connection between an *R_Node* and an *R_NetworkService*;
- *R_Flow* represents an instance of a service provided by an *R_NetworkService*, that is, the data flow between two nodes.

Quality-of-Service Representation

Representation of quality-of-service information is an essential aspect of adaptive systems. A discussion of quality dimensions is presented in Sect. 2.3.4 and Appendix A. In this section, we illustrate how quality-of-service information is represented in the reflective architecture.

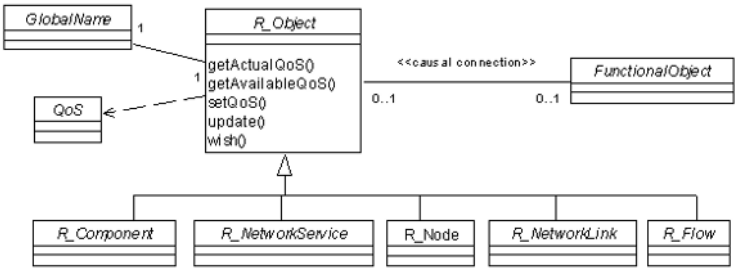

Fig. 2.7. MAIS Base Reflective Layer elements

The *QoS* class (see Fig. 2.8) is the superclass of any quality-of-service feature. As QoS depends strongly on the application domain, it has been considered useful to define a general scheme that establishes how QoS can be defined, rather than provide a comprehensive list of QoS features. In this way, the same system component may exhibit different QoS features depending on the application domain. The QoS class has been specialized in *ComponentQoS*, which models the QoS of computational components, and *NetworkQoS*, which models the QoS of network components. Figure 2.8 shows further QoS specializations, including some concrete examples of QoS.

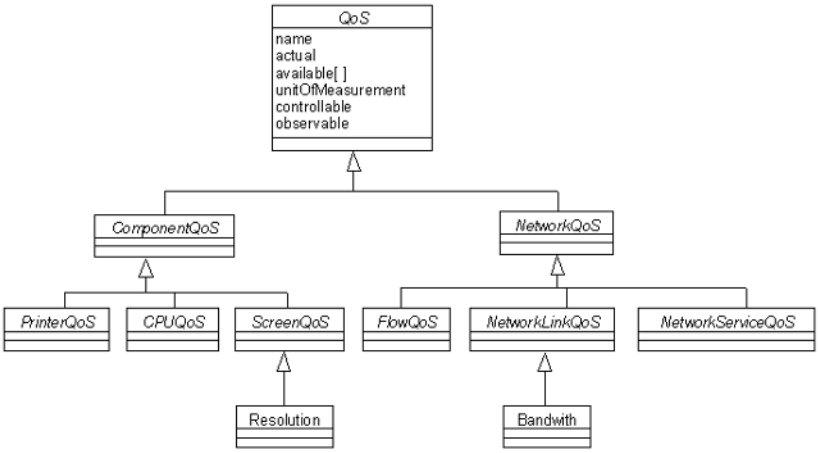

Fig. 2.8. MAIS *QoS* class

An Example: Computational Components

As an example, we present a representation of computational components (see Fig. 2.9). For instance, this structure can be used to represent a laptop, with its elementary components and their quality characteristics, such as its weight, screen size, network interfaces (e.g., a wireless access interface), associated devices (e.g., a satellite positioning system), and location.

R_Component models computational components, i.e., pieces of hardware or software, whose QoS features are significant. An *R_Component* is hosted by one computing node. An *R_ElementaryComponent* models base objects, which, in terms of QoS features, are viewed as black boxes, for instance a monitor. An *R_CompositeComponent* is an aggregate of subcomponents, for instance a laptop. It exhibits its own QoS features (e.g., the weight of a laptop). Moreover, according to the Composite Design pattern [175], it exhibits composite QoS features which depend on the features of its subcomponents (e.g., the screen resolution of the laptop). How composite QoS features can be obtained is specified by suitable strategies in the extended reflective layer.

An *R_Component* is characterized by a *Location*, which models the physical location of a computational component. *Location* is an abstract class that can be specialized in terms of absolute, relative, or geographical coordinates, etc. For instance, to locate the personal computers in a building, a building plan may be used.

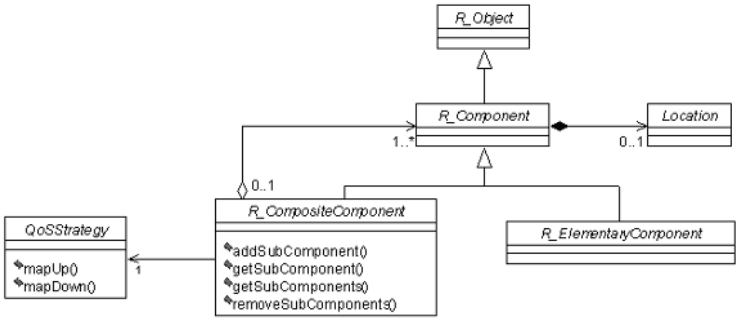

Fig. 2.9. Representation of MAIS computational components

2.3.3 Context Model

A widely and uniquely accepted definition of the concept of context is not available in the literature, but the term is usually adopted to indicate a set of attributes that characterize the capabilities of the access mechanism, the preferences of the user, and other aspects of the context in which information

and services are delivered [379]. These may include the access device (even in the presence of strong heterogeneity of the devices), the quality of service of the network, the preferences of the user, the location, the time, the language, and so on. Therefore, adaptiveness is usually concerned with several independent coordinates. The context is any information that can be used to characterize the situation of a person, a place, or an object that is considered relevant to the interaction between a user and an application, including the user and application themselves.

The MAIS context model, shown in Fig. 2.10, defines the context as an aggregation of four different main groups of properties:

- a *channel* that defines the channel used by the context; it identifies both the physical device and the connection used to access the application;
- a *location and a time* description that identify where the user is located while interacting with the application;
- a *user profile* to further characterize who is interacting with the application;
- a set of *activities* that describe relevant information about the ongoing activities in which the user is involved.

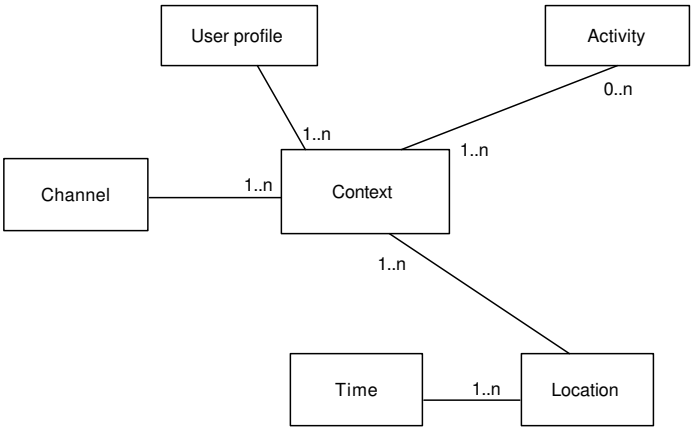

Fig. 2.10. MAIS context model

All the context elements are described as classes and are represented in the reflective-architecture model described in Sect. 2.3.2, which is extensible with respect to base reflective classes. In the following, we present a general description of the four components of the MAIS context model.

Channel Model

In MAIS, channels are represented at two levels of abstraction in order to enable system designers to focus on the correct level of detail.

In particular, a *conceptual distribution channel* is described according to the elements which are considered significant from the user's or application's point of view; in the *logical distribution channel* the channel is represented in greater detail. However, an abstraction from strictly technological aspects or products and from aspects of the implementation remains.

Conceptual Distribution Channel

From a conceptual point of view, a channel is defined as a way to gain access to a given class of information services or devices. At this level, a channel is defined by focusing on specific general characteristics which are relevant to service access. For instance, the Web channel (or www channel) is characterized by the use of the HTTP protocol, whereas a mobile phone channel is characterized by the use of mobile devices such as hand-held devices and PDAs. The channels defined in the model include the Web, mobile phone (voice), mobile phone (SMS), call center, email, and mobile device channels. In the definition of conceptual channels, the detailed characteristics of the provisioning channel are not represented, and only the most prominent characteristics are considered. These definitions of channels are used when a general description of the channel is needed, for example when specifying the requirements of applications. The details of the channel are present instead in the model of the logical distribution channel, which provides the model used by the adaptive mechanisms in the MAIS platform.

Logical Distribution Channel

From the logical point of view, a distribution channel is composed of (Fig. 2.11):

- the *device* that the user possesses;
- the *network interface* through which the device is connected to the network;
- the *network* used to transfer information;
- the *application protocols* used by services.

Each of the elements that make up the logical distribution channel is characterized by a number of attributes (for example, the screen resolution of the device or the network topology). Each attribute is associated with a value, which can be numeric, as for example in the case of the weight of a device, a set of numbers (possibly continuous), or a mathematical function, for example the graph function describing the network topology.

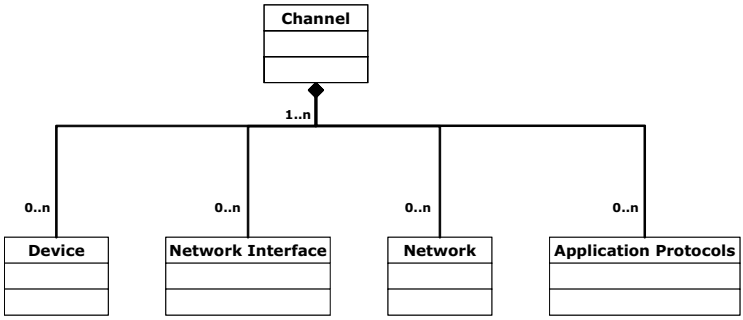

Fig. 2.11. MAIS logical-channel model

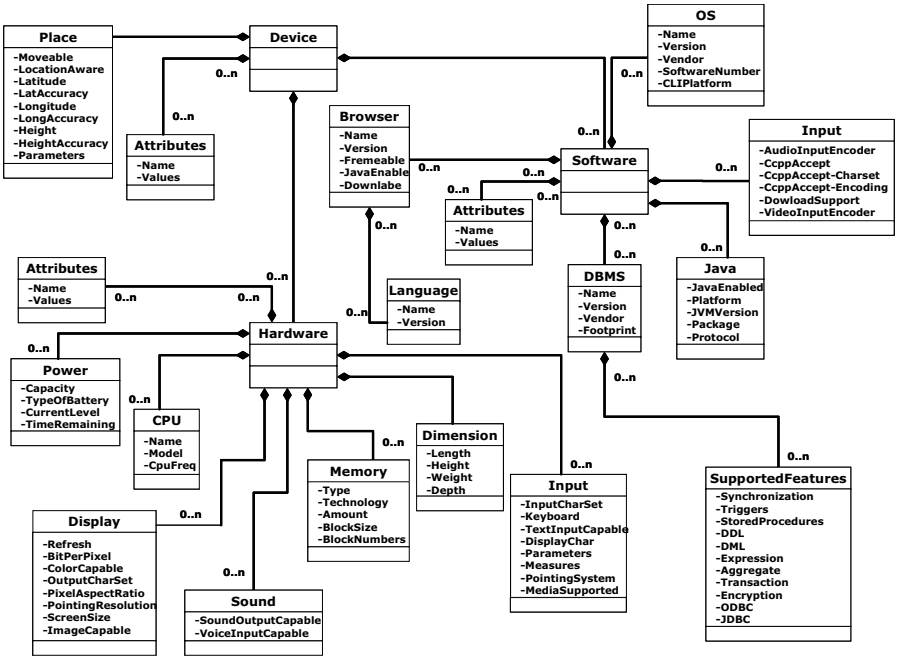

Fig. 2.12. MAIS device model

Figures 2.12–2.15 model the elements that characterize a distribution channel.

We can illustrate the logical-channel model better by examining some relevant attributes for all of the four elements defined above, by considering a typical banking information system. The first element describing a distribution channel is the device on which the end-user interacts. In the domain of financial applications we consider the following relevant attributes for this component: the screen resolution and the number of colors, which are pertinent when the information system wants to send users graphical informa-

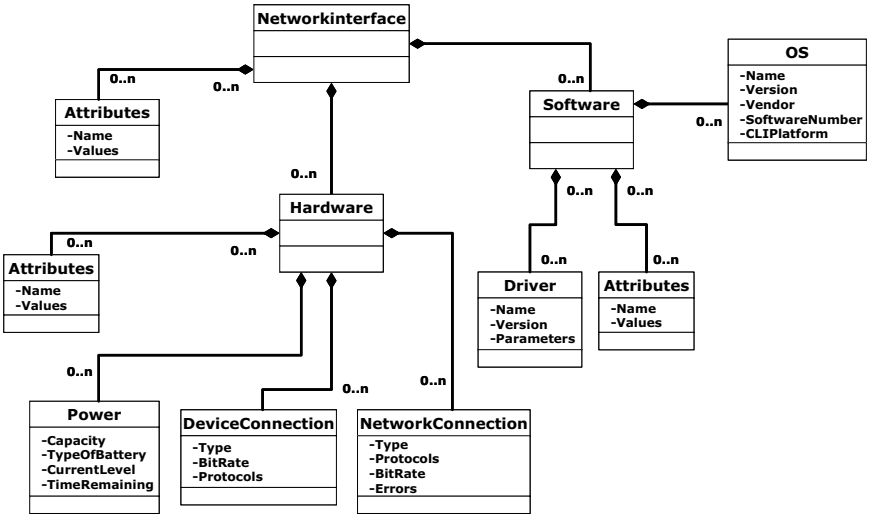

Fig. 2.13. MAIS network interface model

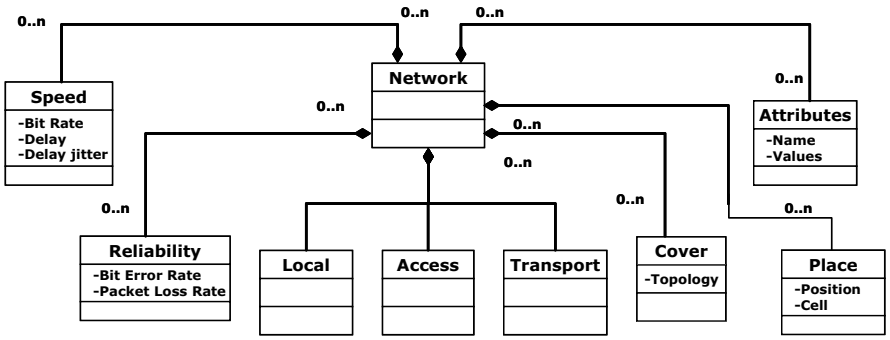

Fig. 2.14. MAIS network model

tion. Some other key attributes are audio support, describing the presence or absence of audio cards inside the device, and the input device used by customers. This attribute is relevant in defining the best interaction methods. For instance, an alphanumeric password is more complex to insert with a multimode numeric keyboard than with an alphanumeric one. The second component is the network interface, representing the connection between the devices and the transmission media. It is worth noticing that a device can access different transmission media by means of different interfaces. For example, a PC can access the Internet via a LAN through a network card or via POTS by means of an analogic modem. In the financial context, we consider that the transfer rate achievable by a specific interface is a relevant attribute. We also consider, as a distribution channel component, the set of application protocols that allow users to interact with the information system. We can

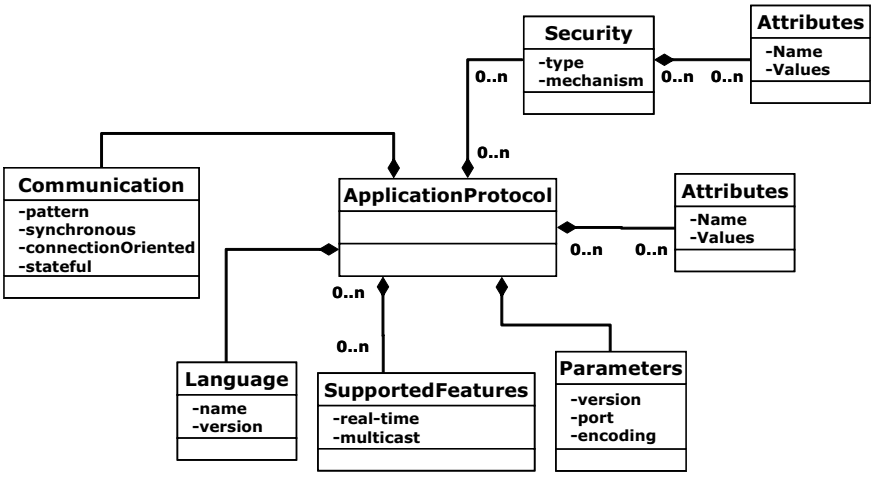

Fig. 2.15. MAIS application protocol model

identify two interesting attributes: the security support and the standardization of the application protocol, which is an important feature for reusing existing parts of applications.

Location and Time

The representation of the location may be domain-dependent. In general, for MAIS applications, modeling of geographical locations is needed in order to provide adaptive behavior at the application level.

A *location* is composed of three main parts (see Fig. 2.16): the *Political Geography* location, the *Physical Geography* location, and the *Architectural Position*. The Political Geography location describes the location in terms of *Country*, *State*, *County*, *City*, and *Street*, where each part is identified by a name. The Physical Geography location expresses the location using a *GeoPos* (geographical position), that is, a latitude and a longitude. Finally, the Architectural Position describes the location within a structure using a *Building* component, a *Floor* component, and a *Room* component.

Other types of locations may be needed for other applications. For instance, many applications are based on coordinates gathered by the Global Positioning System. In some other situations, an internal representation of buildings is needed, in terms of rooms and connections between them [101].

The location model can be extended for specific application domains using subclassing.

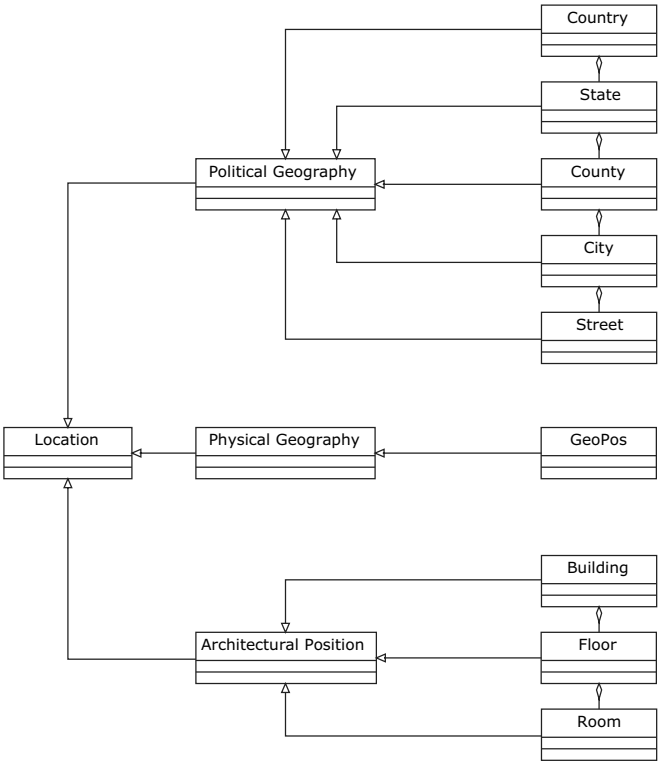

Fig. 2.16. Location model

User Profile

The context comprises a description of the user who is interacting with the application; this description is called the user profile. As shown in Fig. 2.17, in MAIS the user profile has three main components:

- *User information*, used to describe all the information about the user that is of interest for the system. It includes:
  - *personal information*, such as name, address, and age;
  - *expertise*, used to describe the user's expertise and knowledge where it can help in the interaction with the system and in the execution of an application;
  - *role*, used to represent the role played by the user in a well-defined social context; it may depend on the social or professional position of the user;
  - *activity*, used to collect information about the actions and the activities performed by the user during the interaction with the system.
- *User preferences*. As described in [250, 252], preferences can be:

  - *quantitative*, if they are defined by means of a scoring function that associates each object or service with a numeric score denoting the level of preference for the object of one user, or
  - *qualitative*, if they are expressed by means of a partial order that specifies a preference for one object over another one.
- The functional *abilities* of the user. These are described using the International Classification of Functioning, Disability and Health (ICF) standard [195]. The extensions to the context model implied by the ICF, which is described in detail in Appendix B, involve three areas: body functions (BF), activity and participation (AP), and environmental factors (ENV). Aspects related to body structures are not taken into account since they are sufficiently well described by body functions (within the restricted MAIS application domain). Body functions have to be considered because their attributes may convey useful information related to the user's capabilities (e.g., memory and attention capabilities have an impact on long, complex interaction sessions with the system). Some body functions attributes are considered as base attributes with respect to activity and participation (e.g., memory, attention, and emotional functions) while some others may be considered as less informative with respect to their activity and participation counterparts (e.g., the language attribute of the body function area). We have therefore defined three classes:
  - *BodyFunctions*, used to define physiological and psychological functions of a user who interacts with a device;
  - *ActivityParticipation*, used to define the ability to participate in the execution of a task;
  - *RelationalCapabilities*, used to define the role of the environment, particularly the set of relations, attitudes, and services useful for describing other people involved in cooperative activities.

  A detailed description of functional abilities is provided in Appendix B. These abilities not only allow the development of applications specifically targeted at people with disabilities, but also provide a rich model which can be employed to describe a variety of situations of user interaction: for instance, the inability of a driver to read the display of his/her portable computer can also be represented using the above-mentioned user characteristics.

The various components of the user profile can be either static or dynamic:

- A component is *static* if it does not change over time (e.g., some personal information about the user). Static information is usually set by the user by means of an appropriate form in the user profile manager.
- A *dynamic* component is subject to change over time (e.g., some preferences, the expertise level of the user or one of his/her activities) and therefore needs to be kept up to date. Usually, dynamic information is automatically collected and updated during user interaction by an appropriate system module.

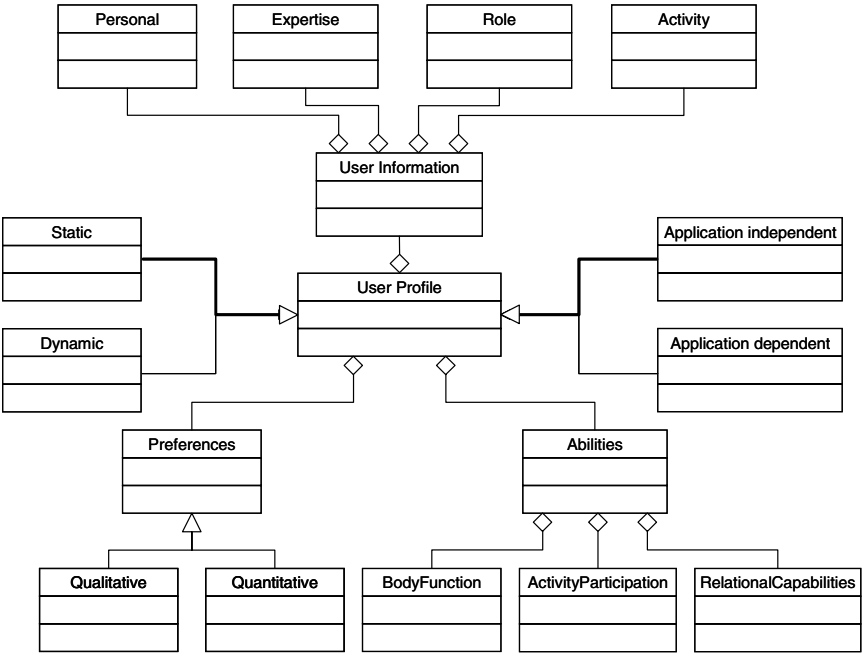

Fig. 2.17. MAIS profile model

Finally, a user component can be classified as *application-dependent* or *application-independent*. In some cases, generic user information is collected independently of the services that are made available. In other cases, only user information that is pertinent to a specific application domain is needed.

2.3.4 Quality of Service Dimensions

Quality of service is fundamental for adaptive applications in mobile information systems. Service provisioning and adaptive mechanisms at all levels are based on this element. As mentioned in the previous paragraphs, quality dimensions have been defined in all the models in the MAIS reference framework.

Quality dimensions can be *domain-dependent* or *domain-independent*. For instance, quality dimensions in the tourism domain may be the category of service of hotels, or the class of seats on flights (e.g., first class).

In this section, we shall briefly discuss general domain-independent quality dimensions, which are not directly dependent on any specific application domain. The MAIS reference framework provides a rich classification of quality-of-service dimensions that may be relevant in providing adaptive services to users. A detailed analysis of quality dimensions in the MAIS project has resulted in about 150 quality dimensions (the complete list of quality dimensions defined in MAIS is reported in Appendix A).

A *taxonomy of qualities* has been created, considering the different *levels in the quality model* that characterize the MAIS reference framework: the architectural level, the functional level, the context level, and the channel level.[1] Each level is characterized by a set of service quality dimensions that describe the service from the perspective of the specific level. For each dimension, we have provided a definition, and defined a metric and a measurement system. The details of each dimension defined are available in the technical documentation on the MAIS Web site.

Quality dimensions may be related to each other. This means that the trend of a variable may depend on the trends of other variables, which can belong to the same level or to different levels in the quality model. These relations are represented using *quality graphs* to model dependencies among qualities (see, for instance, Fig. 9.4). For example, the execution time depends both on channel-level dimensions (i.e., network dimensions) and on architectural-level dimensions (i.e., dimensions related to resources involved in the provider delivery architecture). Furthermore, the same quality dimension can be considered at more than one level. When this occurs, the dimension is associated with different meanings depending on the level considered. The relationships between quality dimensions may be taken into consideration in the service design process, as shown in Chap. 9., where dependencies among qualities are discussed.

Quality dimensions may be defined as *negotiable*, where they can be contracted between the user and the provider. The user can express preferences or define constraints on these dimensions in the service selection phase. The negotiable quality dimensions related to service provisioning include, for example, bandwidth, price, and response time. They are used in the service selection phase, in order to choose one from a number of functionally equivalent services, and in the provisioning phase, to achieve agreement on the desired quality level of the selected service. The negotiation process in flexible service provisioning is briefly discussed in Chap. 3.

[1] In this classification, the channel is considered separately from the context, given the relevance and number of quality dimensions associated with different elements of the channel.