Context-aware Composition of *E*-Services [⋆]

L. Baresi[1], D. Bianchini[2], V. De Antonellis[2],
M.G. Fugini[1], B. Pernici[1], and P. Plebani[1]

[1] Politecnico di Milano – Department of Electronics and Information Science
`baresi|fugini|pernici|plebani@elet.polimi.it`
[2] Università di Brescia – Dipartimento di Elettronica per l'Automazione
`bianchini|deantonellis@ing.unibs.it`

Abstract. Composition of *e*-Services in a multichannel environment requires taking into account the constraints imposed by the context: user profiles, geographical locations, available channels, and usable devices. In this paper, we propose an approach for context-aware composition of *e*-Services based on an abstract description of both *e*-Services and context. *E*-Services are described in terms of functionality and quality of service. The context describes the channels that can be used to access *e*-Services. The paper proposes adaptation rules as the means to allow the composition and dynamically select *e*-Service channels according to the constraints posed by available architectures and application-level requirements. Composition and adaptation rules are exemplified on a simple case for emergency management.

1 Introduction

Nowadays, users are greatly interested in easily accessing a wide variety of services through different channels [8, 10]. This means that *e*-Services can be accessed using different devices (e.g., PCs, laptops, palmtops, cell-phones, TV sets), but also different network technologies and application protocols. For example, the same service could be delivered via call centers, the Web, or message-based systems like SMS. In particular, nomadic users want to access services from their current locations and with the best possible performance. As a consequence, the need emerges for the design of networks and services that are highly flexible and adaptive, capable of exploiting available resources in an optimal way and with different levels of quality of service.
The Italian MAIS (Multichannel Adaptive Information Systems) project [12] aims to create an enabling platform, a methodology, and design tools for the development of distributed information systems based on *e*-Services and with significant adaptability features.
Adaptive information systems span different research areas. For example, the simplest way towards adaptation is through transcoding [3]. As to profiles and context information, we can mention the WWRF (Wireless World Research

⋆ This work has been partially supported by the Italian MIUR FIRB MAIS (Multichannel Adaptative Information Systems) project.

Forum) [16], the Cameleon project [5], and the CC/PP (Composite Capabilities/Preferences Profile) initiative of the W3C *device independence working group* [15]. Gu et al. [9] present a precise model for quality of service and also propose rules to support the negotiation phase. But also, the UWA project studied and proposed *customization rules* [6] to describe and constrain the adaptability of ubiquitous Web applications. Finally, composition is addressed in [11], where they use DAML-S and coloured Petri nets to reason on *e*-Service composition.

The goal of this paper is to define a modeling framework for adaptive information systems based on *e*-Services. We aim at separating the aspects at application and technological levels and at providing a basis for formalizing contracts between *e*-Service providers and the user.

In the proposed modeling approach, *e*-Services are defined by means of two complementary perspectives: the request perspective, which provides an abstract model of the context where the *e*-Service is requested and executed, and the provisioning perspective, which models available *e*-Services in terms of their functional description and composition. Both perspectives consider proper characteristics of quality of service to allow *e*-Service negotiation in terms of user requests, channels used in the interaction, and constraints from the *e*-Service provider. These negotiation policies are coded in adaptation rules. The two perspectives and adaptation rules form the basis for context-aware composition of *e*-Services that is able to dynamically select *e*-Service channels according to the constraints posed by available architectures and application-level requirements. In particular, adaptation rules allow us to dynamically adapt the execution flow of our services –described as workflows – according to the characterization of the selected channel.

The paper is organized as follows. Section 2 shows the two modeling perspectives, while Section 3 discusses the application of the approach to a reference example. Section 4 and Section 5 address the specification of quality of service and adaptation rules, respectively. Finally, Section 6 introduces the MAIS platform and Section 7 concludes the paper.

2 *E*-Service model

In this paper we present two different modeling perspectives related to the *provisioning* and *request* of *e*-Services. The provisioning perspective specifies who provides the *e*-Service, what the *e*-Service does and how to invoke its functionality, according to the available quality of service. The request perspective specifies who requires the *e*-Service, i.e., the actor, who wants to have a certain level of quality for the required *e*-Service, has a particular user profile, and operates in a particular *context*.

In the MAIS project, we use UML (Unified Modeling Language) [13] as modeling language because of its expressiveness.

2.1 *E*-Service provisioning

Figure 1 shows the components used to model an *e*-Service with respect to the e-*Service provisioning perspective.*
An *EService* is described by means of a *name* that identifies it, a short textual *description*, a service *category* (such as, for example, *commercial service* or *information service*), and an aggregation of three types of elements:

- a *Channel*, on which the *e*-Service is provided; to pursue adaptativity, an *e*-Service can be provided through one or more channels and an association class *CQualityDimension* represents the quality of the *e*-Service with respect to the channel used; there can exist several parameters to express the quality of service, for instance: response time, channel availability, usability, accessibility, integrity, bandwidth, reliability, and price;

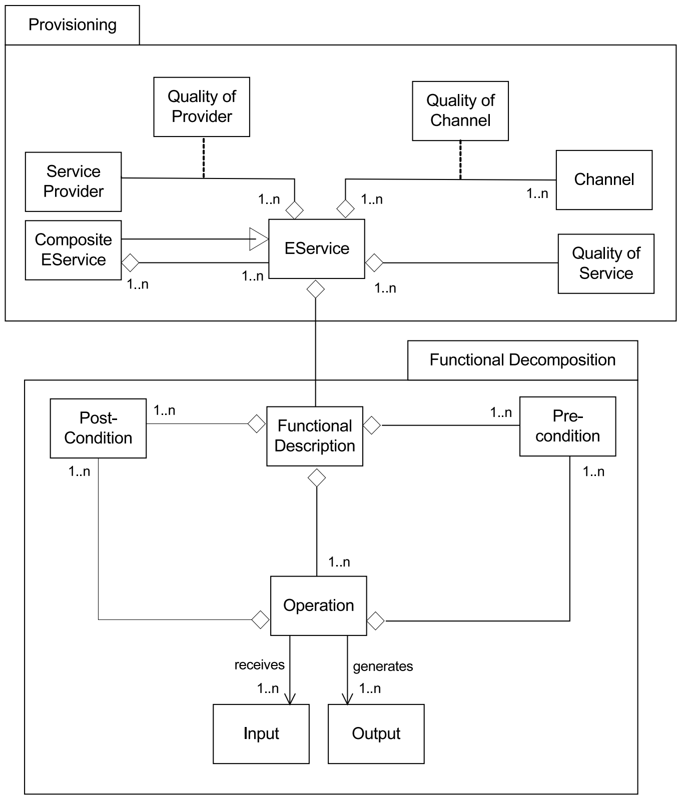

Fig. 1. Service provisioning perspective

- one or more *ServiceProviders*, each of them described by a *name* and standard address information; an association class *PQualityDimension* expresses the quality parameters guaranteed by the service provider: for example, data reliability, provisioning time, service availability, price;
- a *FunctionalDescription* that describes the operational aspects of the *e*-Service.

The *FunctionalDescription* – inspired by WSDL [4] – is a set of *Operations*, with a *name* and a short textual *description* about "what the operation does":

- each operation requires one or more *Inputs* and gives back one or more *Outputs*; input and output parameters are described by a *name* and a *value*;
- each operation is associated with a set of *PreConditions* and *PostConditions* that predicate on the *Inputs* and *Outputs* of the operation; the former must be verified before the execution of the operation, while the latter must be satisfied after the execution; pre- and post-conditions can also be defined on whole services;
- external *Events* are used to model actions that are asynchronous with respect to the normal flow of the *e*-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; events must be managed through appropriate rules.

E-Services can be composed in a recursive manner to create *CompositeEServices* (similarly to BPEL4WS [7]). Composite *e*-Services are described through workflows, where component *e*-Services are connected by means of control constructs [14].
Moreover, *e*-Services are grouped into *CompatibilityClasses* for substitutability purposes. A compatibility class is associated with an *AbstractEService*, that is the *e*-Service required in a process execution expressed in terms of the functionality it provides. A compatibility class groups, on the basis of predefined "similarity" criteria performed through comparison between functional descriptions [1], *e*-Services that are able to substitute each other in satisfying the considered abstract service. When an *e*-Service during the execution of some tasks is not available anymore, it can be automatically substituted by another *e*-Service that belongs to the same compatibility class and that offers at least the same functionality. An *e*-Service can belong to more than one compatibility class at the same time.

2.2 *E*-Service request

Figure 2 presents the elements that define our *service request perspective.*
An *Actor* issues requests for *EServices*, already defined in Section 2.1, with a given *Quality Level.* In a simplified setting, *Quality Level* indicators could be: speed, availability, security, conformity to standards, and price. More thorough descriptions should be used in real cases. The actual quality of service with which

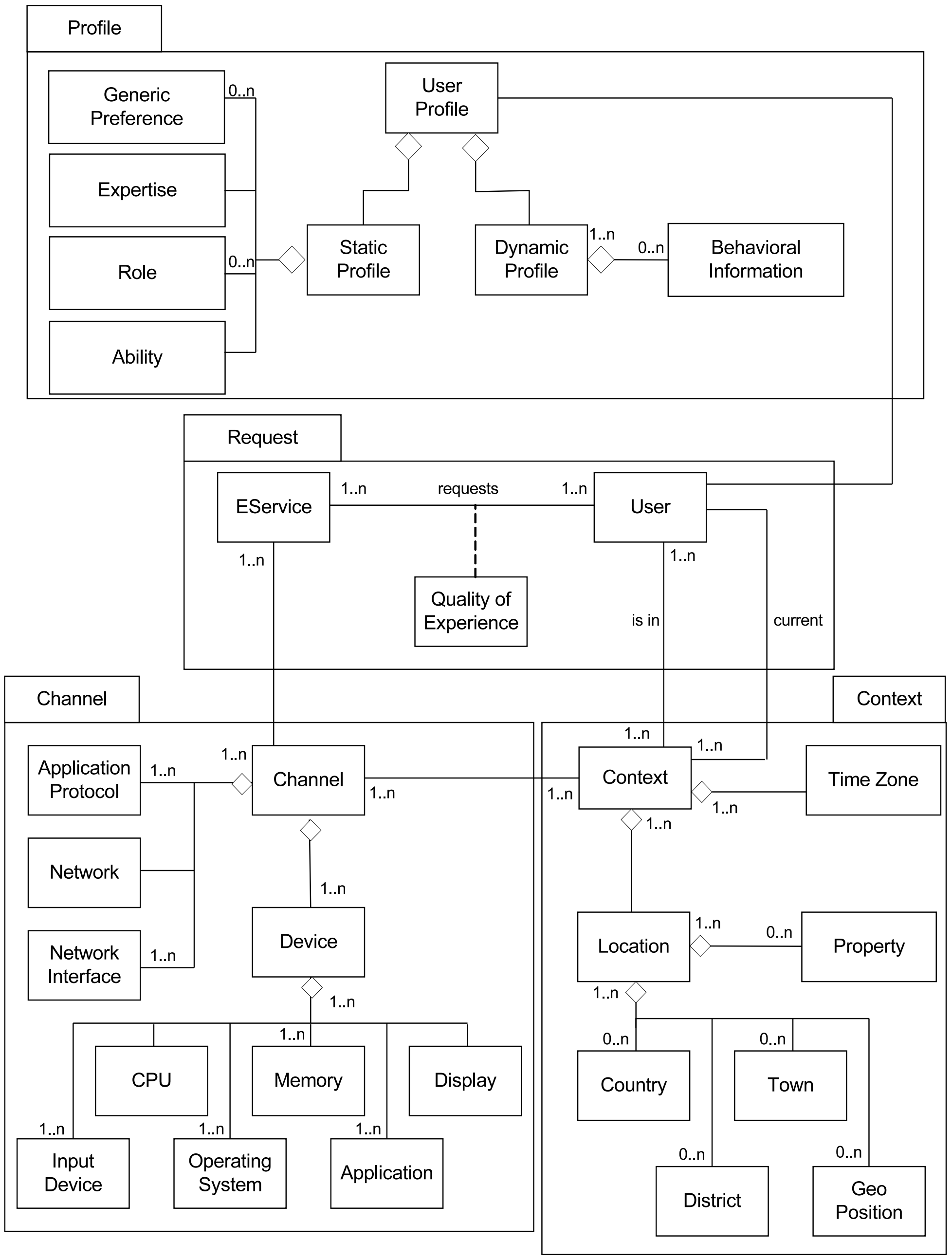

Fig. 2. Service request perspective

the service is supplied comes from the negotiation of this *Quality Level* with that supplied by the provider (i.e., *PQuality Dimension* in Figure 1).

Each *Actor* has a *Profile*, defined as a set of *User Preferences.* Since *User Preference* is abstract, the *Profile* can contain a *Role*, which identifies the role played by the actor while using the application, its *Expertise* on the application, and a set of *Generic Preferences* to add application-specific characterizations to the profile. The hypothesis is that *Role* or *Expertise* define the minimum profile, which can always be enriched with further information: *Generic Preference* has a *name* and *value* to let the designer render and "quantify" any property. The additional requirement that a *Profile* must have at most one *Role* and one *Expertise* is not shown here, but can be easily added by means of an OCL constraint[3].

We assume that an *Actor is in* some *Contexts* at the same time, but only one of these is his *current Context.* The latter identifies the channels that can be used to supply required services, while the former define the back-up options, that is, the set of available *Contexts* – and thus channels – in which the user can be moved to recover from problems with his current channel.

A *Context* is characterized by a *Location* and a *Time Zone. Location* can be zero or more *Geo Positions*, i.e., latitudes and longitudes, *Districts*, e.g., special-interest areas, *Towns*, and *Countries.* Moreover, a *Location* can be associated with a set of *Properties*: a general-purpose mechanism to add further information to the context description. For instance, we could use a *Property* to specify weather conditions. *Time Zone* describes the *Context* with respect to its time information, i.e., its offset from Greenwich mean time, and the daylight saving time.

The user *requests* a *Service* on a given *Channel.* As soon as this *Channel* becomes unavailable, the *Actor* can switch to a new *Channel* among those associated with his *current Context.* The choice is even wider since the *Actor* can change *Context* and thus the set of available channels.

A *Channel* is characterized by one *Device*, one *Network*, one or more *Network Interfaces*, and one or more *Application Protocols.* A *Device* comprises one *CPU*, one or more *Input Devices*, one *Display*, and one or more *Memories.* These hardware components are described through the usual characteristics: for instance, a *Display* is defined by means of its size and number of colors. Besides specifying the hardware on board, we also define the software with which it is equipped: one ore more *Operating Systems* and one or more *Applications. Network* defines the network as a *name* and a *bandwith*, *Network Interfaces* define how the *Device* can connect to the *Network* (e.g., Ethernet, WI-FI), and *Application Protocols* specify the application protocols that can be used: for instance, HTTP, SOAP, SMTP.

[3] OCL (Object Constraint Language) is the constraint language supplied with UML [13].

3 Reference example

The MAIS project defines sample domains to test multichannel adaptive applications. In this section, we present an example in the *emergency management* domain, where we have to deal with interventions for territorial emergencies due to hydro-geological events or sanitary emergencies (e.g., due to car accidents). In particular, we consider the *collection and diffusion of information at fixed stations* to deliver on-line traffic information to travelers, policemen, firemen, highway operators, and tele-control centers.

This activity implies rapidity of service provisioning and urgency. We assume that provisioning sites are fixed stations spread on wide areas and endowed with small-sized computing facilities (e.g., PCs). Data collection is managed by operators and users on the territory via mobile stations, smart phones, GSM, and radio systems. The fixed stations coordinate the information flow from and to mobile stations, operate data processing, such as check and cross analysis of received data, and distribute information back to reachable mobile users, and also to the other fixed stations.

Given the distributed nature of the domain, GSM/SMS and mobile channels play a key role. The strategy, in this context, is to maximize the availability of information on these channels and minimize the delay in redistributing collected critical information.

The scenario considers that fixed stations act as coordinators and mobile stations (e.g., travelers as well as police cars) collect data on local traffic and forward them to the closest, and reachable, fixed station. Fixed stations receive these data, check them for accuracy and timeliness, and process them with weather information and statistical data available on site.

Considering composed *e*-Services modeled as workflows, whose steps are operations or *e*-Services, the composed service *Collection and diffusion of information at fixed stations* can be modeled as the *cooperation* of two services (Figure 3): an *e*-Service to collect data, which is executed on mobile stations, and an *e*-Service to process and re-transmit data, which is executed on fixed stations.

Both services are modeled as workflows and include *channel switches.* The cooperation between the two services is modeled through message exchange with conditions. Each service starts executing as soon as it receives a *start* event. Service requests are processed within the given context, but if a message is urgent and the current channel is not available, the workflow schema specifies a channel switch. Urgency (i.e., priority) is modeled through a condition in the workflow. We specify explicitly those channel switches that are visible to the user and may involve a change in the process composition logic. As shown in the example, service/channel switches can occur basically because of:

1. e-*Service characteristics*: these are inherent to the workflow, hence statically identifiable in the design phase. An example is the condition of no close stations available for data transmission on Internet-based channels. In this case, the service can be suspended; alternatively, if the message is urgent, it

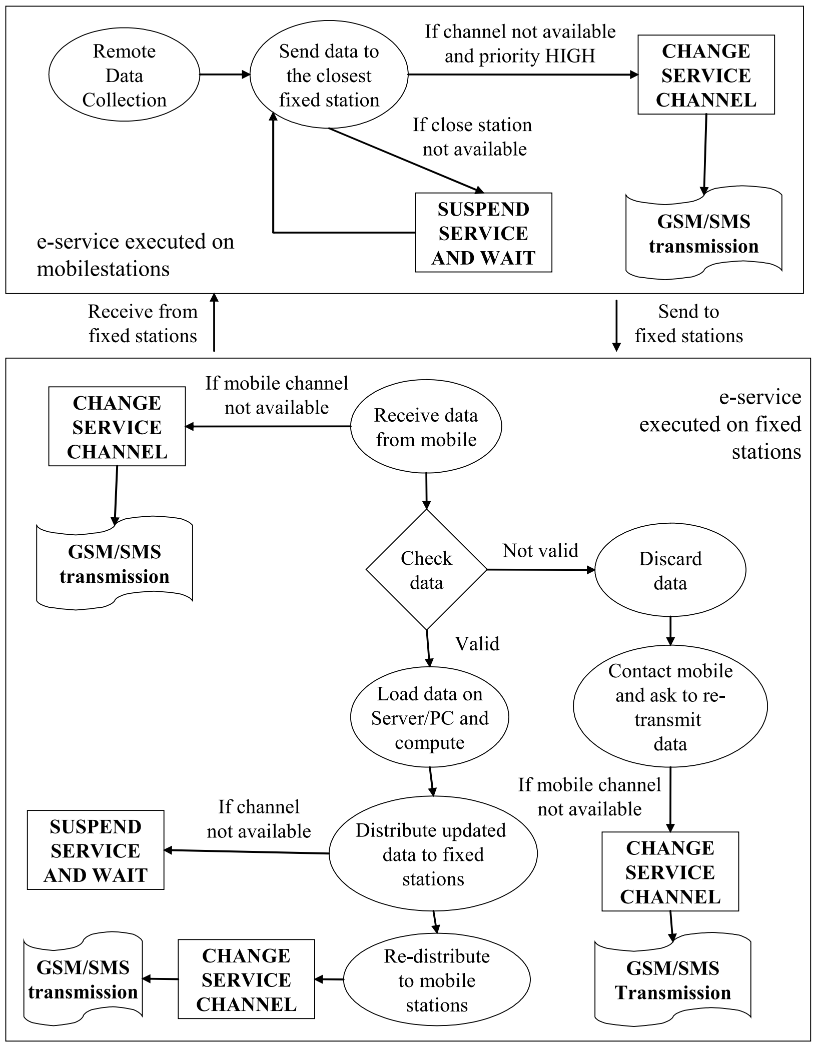
Remote Data Collection
Send data to the closest fixed station
If channel not available and priority HIGH
CHANGE SERVICE CHANNEL
If close station not available
SUSPEND SERVICE AND WAIT
GSM/SMS transmission
e-service executed on mobilestations
Receive from fixed stations
Send to fixed stations
If mobile channel not available
CHANGE SERVICE CHANNEL
Receive data from mobile
e-service executed on fixed stations
GSM/SMS transmission
Check data
Not valid
Discard data
Valid
Load data on Server/PC and compute
Contact mobile and ask to re-transmit data
If mobile channel not available
If channel not available
SUSPEND SERVICE AND WAIT
Distribute updated data to fixed stations
CHANGE SERVICE CHANNEL
GSM/SMS transmission
CHANGE SERVICE CHANNEL
Re-distribute to mobile stations
GSM/SMS Transmission

Fig. 3. Collection and diffusion of information

can be simplified and sent on a different channel (e.g., SMS). It is a design choice to decide the conditions that can lead to adaptation actions.

2. *Channel status*: these are parameters that depend on provisioning and can vary over time. For instance, system workload, speed, bandwidth, and price – our *PQualityDimensions* in Figure 1 – can make the *e*-Service be deviated on a different channel during its execution. At design time, we must foresee both important events, to capture significant changes of the current channel, and the ways to react to these changes.

Summarizing, besides conditions, the schema for a composed service must be augmented with *adaptation rules* to specify how to adjust the execution flow at run-time. These rules, presented in Section 5, predicate in terms of parameters bound to the quality of service and available context. Adaptation actions include: suspend, alternatives in the workflow, channel switch, and request for channel upgrade.

4 QoS specification

The reference example highlights situations in which channel availability influences process execution. In this paper, we focus on adaptivity related to quality of service parameters. As described above, we take in account two perspectives: *e*-Service request and *e*-Service provisioning. In the same way we define the quality of service following such two standpoints. We define a set of basic *quality dimensions* that characterize the service being provided on a given channel and requests for quality levels in service requests. Quality of service dimensions are grouped in quality of service parameters related to the provider (*PQuality-Dimension*) and quality of service parameters related to the provisioning on a given channel by a given provider (*CQualityDimension*) as shown in Figure 1. Consistently to the models presented in Section 2, we can represent such characteristics using XML:

```
<Service name="sendData">
  <Provider name="provider1">
    ...
    <PQualityDimensions>
      <ProvisioningTime unit="s">10</ProvisioningTime>
      <ServiceAvailability hoursperday=24 daysperweek=7>90\%</ServiceAvailability>
      <Price currency="USD">100</Price>
    </PQualityDimensions>
  </Provider>

  <Channels>
    <Channel name="PDA/HTTP/GPRS">
      <Device>PDA</Device>
      <ApplicationProtocol>HTTP</ApplicationProtocol>
      <Network>GPRS</Network>
      <NetworkInterface>ModemGPRS</NetworkInterface>
      <CQualityDimensions>
        <Bandwidth unit="Kbps" min=56.6 max=113.2/>
        <ChannelAvailability hoursperday=24 daysperweek=7>99\%</ChannelAvailability>
        <Price model="flatrate" currency="USD">100</Price>
      </CQualityDimensions>
    </Channel>
    <Channel name="Phone/HTTP/UMTS">
      <Device>Phone</Device>
      <ApplicationProtocol>HTTP</ApplicationProtocol>
      <Network>UMTS</Network>
      <NetworkInterface>UMTSModem</NetworkInterface>
      <CQualityDimensions>
        <Bandwidth unit="Kbps" min=50 max=1024/>
        <ChannelAvailability hoursperday=24 daysperweek=7>99\%</ChannelAvailability>
        <Price model="per minutes" currency="USD">0.30</Price>
      </CQualityDimensions>
    </Channel>

    <Channel name="PDA/WAP/GPRS">
      <Device>PDA</Device>
      <ApplicationProtocol>WAP</ApplicationProtocol>
      <Network>GPRS</Network>
      <NetworkInterface>ModemGPRS</NetworkInterface>
      <CQualityDimensions>
        <Bandwidth unit="Kbps" min=28.3 max=56.6/>
        <ChannelAvailability hoursperday=24 daysperweek=7>96\%</ChannelAvailability>
        <Price model="per minutes" currency="USD">0.30</Price>
      </CQualityDimensions>
    </Channel>
  </Channels>
</Service>
```

In this case, we consider service *sendData*, which sends data to the fixed stations. The provider assures its provisioning with a `ProvisioningTime` equal to at most 10 seconds, `ServiceAvailability` equal to 90% 24 hours 7 days a week, and a price of 100 USD. We also suppose that it can be used through three wireless channels: a PDA which uses (i) HTTP or (ii) WAP over GPRS and (ii) a SmartPhone which communicates via HTTP over UMTS. These three channels provide different performances and availability degrees described by `Bandwidth` and `Availability`.
From a user perspective, the quality of service can be defined as a set of *quality levels* that depend on either the basic quality dimensions presented above or dimensions that capture the status of the system. In the first case, the actor that uses the service knows the quality features and is able to negotiate the more appropriate quality level. In the second case the system informs the actor that some aspects of the system can be observed. For example, let us consider the following dimensions:

```
<QualityLevels>
  <Dimension name="Speed">
     <Level name="very high">
       <Bandwidth>
         <Condition type="greaterThan" unit="Kbps">512</Condition>
       </Bandwidth>
       <ProvisioningTime>
         <Condition type="lessThan" unit="s">2</Condition>
       </ProvisioningTime>
     </Level>
     <Level name="high">
       <Bandwidth>
         <Condition type="between" unit="Kbps">
           <min>60</min><max>512</max>
         </Condition>
       </Bandwidth>
       <ProvisioningTime>
```

```
        <Condition type="between" unit="s">
          <min>2</min><max>4</max>
        </Condition>
      </ProvisioningTime>
    </Level>
    ...
  </Dimension>

  <Dimension name="Availability">
    <Level name="very high">
      <ChannelAvailability>
        <Condition type="greaterThan" dimension="hourperday">10</Condition>
        <Condition type="greaterThan" dimension="dayperweek">6</Condition>
        <Condition type="greaterThan" dimension="percent">99</Condition>
      </ChannelAvailability>
    </Level>
  </Dimension>
</QualityLevels>
```

The actor can negotiate the `Speed` and `Availability` of the service. Thus, after the negotiation phase, the actor selects, for each *offered* quality dimension, the required level. During the execution of the process which involves *e*-Service *sendData*, the execution system has to ensure that the quality does not assume a rate lower than the negotiated level.

5 Adaptation rules

In an adaptive context, *e*-Service composition has to handle critical situations. Due to *e*-Service failures or insufficient quality of service, the execution could consider service changes or reconfigurations. The first case has already been studied in [1]: We analyze the functional aspects of the *e*-Service to discover a similar one that can substitute the former. *E*-Service substitutability and similarity are tackled by considering the compatibility classes introduced above.
The second case is analyzed in this paper. Service provisioning still occurs, but using a different channel, that is, by changing one of its components: device, network, application protocol, or network interface. Additional *e*-Services might be invoked to handle aspects related to the reconfiguration.
To disclose those critical situations that require service reconfiguration, we suppose that a system can monitor the execution and reveal any modification about the quality dimensions considered during the negotiation phase. Thus, following the event-condition-action paradigm [2], we define *adaptation rules* to specify how the service reconfiguration has to occur. Such rules are associated with the workflow schema.
In our reference example, we consider a critical situation that refers to a communication breakdown due to a service or channel unavailability while the service is sending data to a fixed station using a SmartPhone over UMTS. In particular, since priority is high, if the service becomes unavailable, the service itself will must be changed. For instance, we could select a different service provider. After a negotiation phase, we would decide a new quality level. On the contrary, if the channel fails, the system can decide to change the communication channel by selecting a new transmission protocol like GPRS. In our approach, we define events to notify significant changes on quality dimensions. For instance:

```
<Events>
  <Event type="ChangeSpeedEvent">
    <Conditions>
      <Predicate dimension="Speed" condition="lessthan">
        <value>high</value>
      </Predicate>
    </Conditions>
    <Actions>
      <Action type="changeDevice"/>
      <Action type="eventActivation">ChangeDeviceEvent</action>
    </Actions>
  </Event>

  <Event type="ServiceNotAccessibleEvent">
    <Conditions>
      <Predicate dimension="Availability" condition="true">
      </Predicate>
    </Conditions>
    <Actions>
      <Action type="changeChannel"/>
      <Action type="eventActivation">ChangeChannelEvent</action>
    </Actions>
  </Event>

  <Event type="ChangeChannelEvent">
      <Action type="notification">The channel is changed</action>
  </Event>
<Events>
```

Event `ServiceNotAccessible` occurs when the communication breakdowns. In such a situation, if `Availability` is true, we impose a channel switch, because we need to find another means to deliver the service. If `Availability` is false, we must switch to another service (this is not shown in the rule). The two rules are applied to an excerpt of our running example in Figure 4 (the *e*-Service on mobile stations). `ChangeSpeed` can be triggered when sending data to the closest fixed station (because the speed drops down drastically), while `ServiceNotAccessible` is triggered when we discover that the channel is not available, but we want to send our data with high priority.
To maintain a consistent environment, every service reconfiguration generates an event to inform all active services and the user about the new status.

6 MAIS Architecture

The e-service composition and selection approach presented in the previous sections is based on the adaptive architecture that we are developing in the MAIS project. This architecture assumes adaptivity at various levels, as shown in Figure 5.
As we discussed in Section 2, services may be accessed by different types of devices, provided using different technological channels, and networks may present different levels of quality of service. All information about networks, devices, and service providers is supplied as *Context description* used by the *Context manager*. The architecture is reflective to provide functionality to access context information, and to control channel configurations, according to suitable parameters.

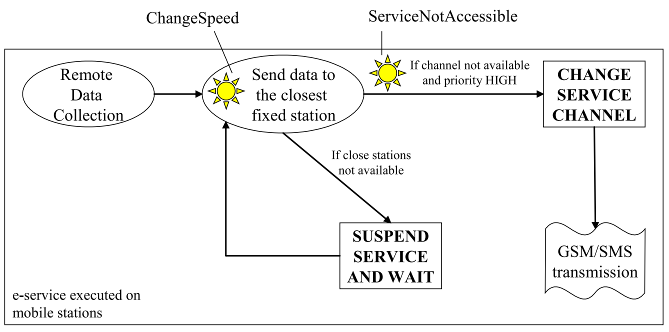
ChangeSpeed
ServiceNotAccessible
Remote Data Collection
Send data to the closest fixed station
If channel not available and priority HIGH
CHANGE SERVICE CHANNEL
If close stations not available
SUSPEND SERVICE AND WAIT
GSM/SMS transmission
e-service executed on mobile stations

Fig. 4. Adaptation rules in our example

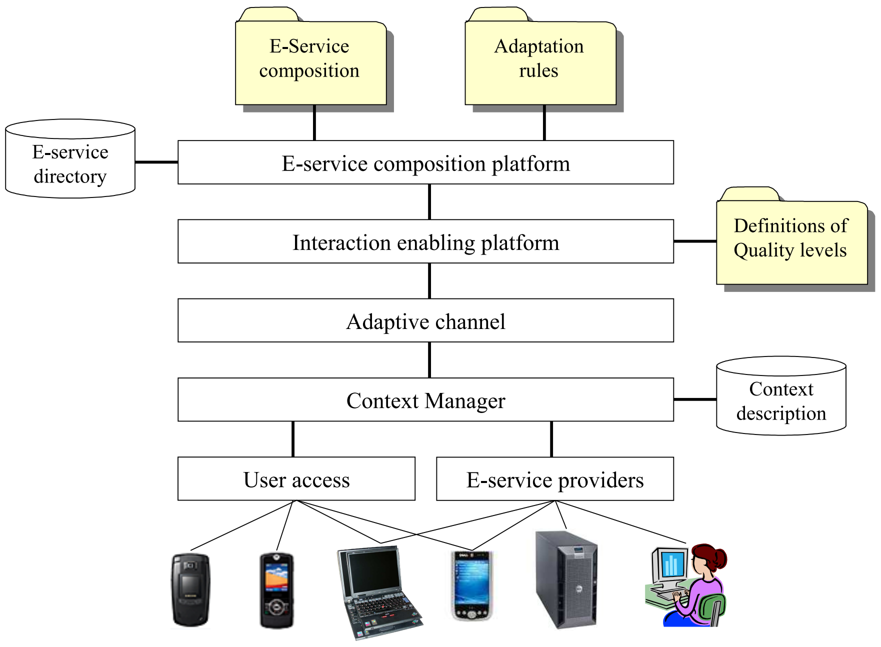

Fig. 5. MAIS architecture

The *Interaction enabling platform* uses the *Definitions of Quality levels* illustrated in Section 4 to transform quality of service requests at the E-*Service composition platform* into requests in technological terms to the *Adaptive channels* used by the application. Initially, such transformations are to be obtained by mapping application level rules into technological constraints; negotiation of parameters at different levels in the *Interaction enabling platform* will be considered at later phases in the project.
E-Services are stored in an E-*Service directory*, representing the characteristics of e-services in terms of provided functionality and possible channels for their provisioning. The *e*-Service directory provides functionality for retrieving *e*-Services to enable a dynamic composition of processes according to the characteristics of the user requests and context information. The directory being realized extends the *e*-Service registry developed in the VISPO project [14]. This registry stores *e*-Service information in terms of descriptors derived from WSDL abstract specifications and allows retrieval and adaptation of e-services to a given composition schema on the basis of a domain ontology.

7 Conclusions and future work

The paper analyzes context-aware composition of *e*-Services and discusses the MAIS approach for a flexible and channel-dependent composition of services. We propose to separate the characteristics of *e*-Service provisioning and user contexts, for *e*-Service requests, and we propose two separate models to describe them.
We also discuss *adaptation rules* for mapping user level requirements into technological constraints. These rules allow for the composition of *e*-Service specifications based on context information. In this way, we define flexible processes that are able to provide and access services which are adequate to the context available to the user.
The MAIS project is developing an adaptive approach that addresses both technology (e.g., networks, devices, micro-databases) and composition and enabling platforms to manage and control context information and to allow context-aware dynamic composition.

References

1. V. De Antonellis, M. Melchiori, and P. Plebani. An Approach to Web Service compatibility in cooperative processes. In *Proc. IEEE SAINT2003 of Int. Workshop on Services Oriented Computing: Models, Architectures and Application (SOC2003)*, Orlando, Florida, USA, 2003.
2. F. Casati, S. Ceri, S. Paraboschi, and G. Pozzi. *Database Support for Workflow Management: the WIDE Project*, chapter Active Rule Support. Kluwer Academics Publishers, 1999.
3. Y. Chen, W.Y. Ma, and H.J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proc. of the Twelfth Int. World Wide Web Conference (WWWW 2003)*, Budapest, Hungary, May 20-24 2003.

4. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web Services Description Language (WSDL) 1.1.* World Wide Web Consortium (W3C), March 2001. `http://www.w3.org/TR/2001/NOTE-wsdl-2001 0315`.
5. Cameleon Consortium. Cameleon web site. http://giove.cnuce.cnr.it/cameleon.html.
6. UWA Consortium. Uwa web site. http://www.uwaproject.org.
7. F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. *Business Process Execution Language for Web Services, version 1.0.* IBM, July 2002. `www.ibm.com/developerworks/library/ws-bpel/`.
8. A. Maurino et al. Studio delle caratteristiche dei canali (in italian). Technical report, MAIS Multichannel Adaptive Information Systems, 2003.
9. X. Gu, K. Nahrstedt, W. Yuan, D. Wichadakul, and D. Xu. An XML-based quality of service enabling language for the Web. Technical Report UIUCDCS-R-2001-2212, University of Illinois at Urbana-Champaign Computer Science Department, 2001.
10. J. Krogstie, K. Lyytinen, A. Opdahl, B. Pernici, K. Siau, and K. Smolander. Mobile information systemresearch challengers on the conceptual and logical level. *International Journal of Mobile Communication, special issue on Modeling Mobile Information Systems: Conceptual and Methodological issues*, 2003.
11. S. Narayanan and S.A. McIlraith. Simulation, verification and automated composition of web services. In *Proc. of the Eleventh Int. World Wide Web Conference (WWWW 2002)*, pages 77–88, 2002.
12. The MAIS Project Team. The MAIS Project. In *Proc. of the 4th International Conf. on Web Information Systems Engineering*, Rome, Italy, 2004.
13. Object Modeling Language (OMG). UML Unified Modeling Language - Version 1.5. `http://www.omg.org/technology/documents/formal/uml.html`, 2003.
14. The VISPO Project Home Page. `http://cube-si.elet.polimi.it/vispo/index.htm`.
15. W3C. Composite capabilities/preferences profile. http://www.w3.org/Mobile/CCPP/.
16. WWRF. Book of visions 2001. http://www.wireless-world-research.org/.