Achieving Adaptivity Through Strategies in a Distributed Software Architecture

Claudia Raibulet[1], Luigi Ubezio[2], and William Gobbo[2]

[1] Università degli Studi di Milano-Bicocca, DISCo – Dipartimento di Informatica Sistemistica e Comunicazione, Viale Sarca 336, Edificio 14, 20126 Milan, Italy
raibulet@disco.unimib.it
[2] IT Independent Consultant,
Milan, Italy
ubezio@gmail.com, william.gobbo@hotmail.it

Abstract. Designing information systems which are able to modify their structure and behavior at runtime is a challenging task. This is due to various reasons mostly related to questions such as what should be changed, when should be changed, and how should be changed at runtime in order to maintain the functionalities of a system and, in the same time, to personalize these functionalities to the current user, services requests and situations, as well as to improve its performances. The systems which manage to address properly these aspects are considered adaptive. Our approach to design adaptive systems exploits strategies to implement the decisional support and to ensure an efficient modularity, reusability and evolvability of the architectural model. In this paper we describe the main types of the strategies defined in our solution, as well as how these strategies are exploited at run-time in the context of an actual case study in the financial domain.

1 Introduction

Adaptivity is one of the keywords related to the design of today's information systems. It addresses the modifications performed in a system during its execution. These modifications aim to improve the productivity and performance of a system, and to automate the configuration, re-configuration, control and management tasks. They may be translated into modifications of structural, behavioral or architectural components [5, 6, 8].

This paper presents our solution for the design of an Adaptive MANagement of Resources wIth Strategies (ARMANIS) in a service-oriented mobile-enabled distributed system. In this context, we focus on the definition of various types of strategies, which implement the decisional support playing a fundamental role in the process of achieving adaptivity. They exploit a late binding mechanism which enables us to combine structural and behavioral elements in order to obtain a personalized solution for each request based on its input information.

In previous work, the architectural model of our solution which exploits reflection to achieve adaptivity has been described [10]. In the same paper we have presented an implementation example based on a distributed peer-to-peer paradigm. The service-oriented features of ARMANIS have been described in [12]. We have validated the proposed model by applying it in a healthcare system [11]. Furthermore, we have adapted and implemented our solution for mobile devices [2]. In this paper, we aim to focus the attention on the strategies we consider to be fundamental to achieve adaptivity through ARMANIS. In [10, 14] we have introduced the concept of strategy and described its design and implementation issues through the Strategy and Composite design patterns. In this paper we describe the types of strategies to be considered and exploited at runtime by the various components of our architectural model: domain, system and connectivity. The definition of these strategies can be considered a further evolution of our approach, evolution mostly due to the integration of the wired (e.g., LAN) and wireless (e.g., Bluetooth, WI-FI) networks, the usage of the RFID (Radio Frequency Identification) [4, 15] technology for the localization of the system actors, and to the possibility to apply it in various application contexts.

The rest of the paper is organized as following. Section 2 introduces the case study considered to validate our solution. Section 3 describes our architectural model by focusing on its main concepts meaningful for this paper. The application of our approach in the context of the case study is dealt in Section 4. Discussions and further work are presented in Section 5.

2 Motivating Example

The motivating example is collocated in the context of a finance case study. The main idea behind this example is to provide support to the bank staff members to offer customers entering a bank agency personalized services based on customers' profile and account status, as well as on the current business and financial advertisings offered by the bank. Customers appreciate and agree easier to exploit new and personalized services when the bank staff members have the chance to explain them face to face the advantages of the advertisings, rather than when the bank contacts them through communication letters, emails or phone calls.

A prerequisite to achieve this goal is to reveal the presence of the customers entering the bank agency in a non-intrusive way. A possible scenario may be based on a RFID approach, which notifies the system whenever a person having a tag crosses an access point. We suppose that the bank customers have a credit card enriched with a tag, which is able to be excited by radio signals and to send back its own identifier. We have already developed a complex access control framework for the management of multi-services structured based on the notification of the customers' presence [16]. In this case study the access point, called choke point, is the agency entrance(s).

A general scenario consists in the following steps. The system reveals the presence of the customers through their enhanced credit cards. Furthermore, the system looks for the customer's account and identifies the services he can be interested in. Based on the commercial importance of the customer for the bank, various staff members are notified of his presence (e.g., the cashiers, an account consultant, the director of the agency). This information may be displayed on various devices used by the bank staff members (e.g., desktop monitors, wall monitors, PDAs, mobile phones) and in various modes based on their current location and the activities they are performing.

We underline that the actual users of this system are the staff members of the bank agency. The system provides them support to experience a new way of interacting with the bank customers.

Related to this case study we present the following two significant scenarios.

Scenario 1: The bank staff members are notified when a customer enters a bank through a message on their working device. In this scenario, we consider an ordinary customer who interacts only with one of the cashiers at work.

Through the credit card, the system reveals the presence of the customer and identifies the customer's name and bank account number. Based on the customer's profile, the system identifies if there are any advertisings which should be communicated to the customer. All this information is displayed on the desktops used by the cashiers.

Scenario 2: An important customer enters the bank. In this case the director of the bank agency is notified. The way he is notified depends on his location (e.g., office room, meeting room, out of the bank agency) and of the device he has available (e.g., working terminal, wall monitor, PDA, mobile phone). If the director is not present in the agency, the vice-director is notified.

3 ARMANIS' Architectural Model

Figure 1 shows the main components of our architectural model:

- the Domain Manager, which provides a Graphical User Interface (GUI), a Request Execution Protocol (REP) module, and the knowledge specific to the current application domain; the GUI shows to the users the domain services offered by the software system; the REP module defines the steps for the execution of services' requests in the context of the application domain;
- the System Manager, which controls the services provided by the hardware and software components of a system (e.g., print, display); these services are independent of the application domain and depend only on the system's architecture; system services are exploited by the domain services;
- the Connectivity Manager, which supervises the communication with other ARMANIS-enabled nodes; this manager deals with various types of networks both wired (e.g., LAN) and wireless (e.g., WI-FI, Bluetooth).

As shown in Figure 1, the three managers are related to the three main types of knowledge available in every software system. We consider them separately to ensure the modularity of the solution, its maintenance, as well as the reusability as much as possible of the design and implementation components. Each of these managers consists of three main elements: knowledge, services and strategies. The knowledge represents the information owned by each manager. The services represent the functionalities they offer to the users or other components of a system. The strategies implement the mechanisms through which runtime adaptation is achieved.

In the remaining of this section attention is focused on the description of those elements of our architectural model which are independent of any application domain. Hence, we describe the main concepts used by the Request Execution Protocol module to indicate how services are executed. Furthermore, the strategies defined by the system and connectivity modules are introduced.

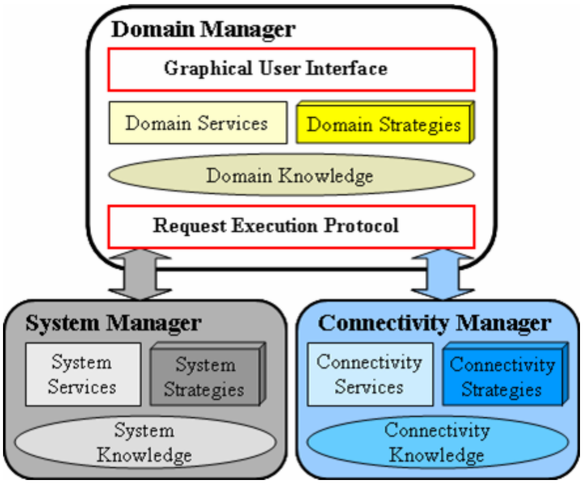


Fig. 1. ARMANIS' Architectural Model

3.1 The Request Execution Protocol Model

The Request Execution Protocol module describes formally the behavior of our approach related to the execution of services requests. The main elements of this module are shown in Figure 2. A *ServiceRequestType* defines the *States* of a request and the *Transitions* from one state to another. Each *ServiceRequest* instance corresponds to a *ServiceRequestType*. For each service request, the Checker element verifies what type it belongs to and assigns it the identified type. If there is no type matching, the request cannot be addressed.

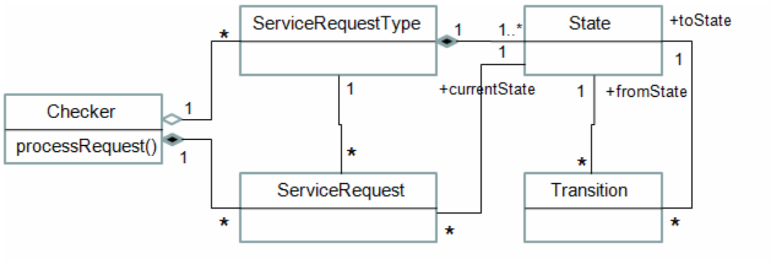


Fig. 2. Elements of the Request Execution Protocol

The definition of service execution protocols is based on:

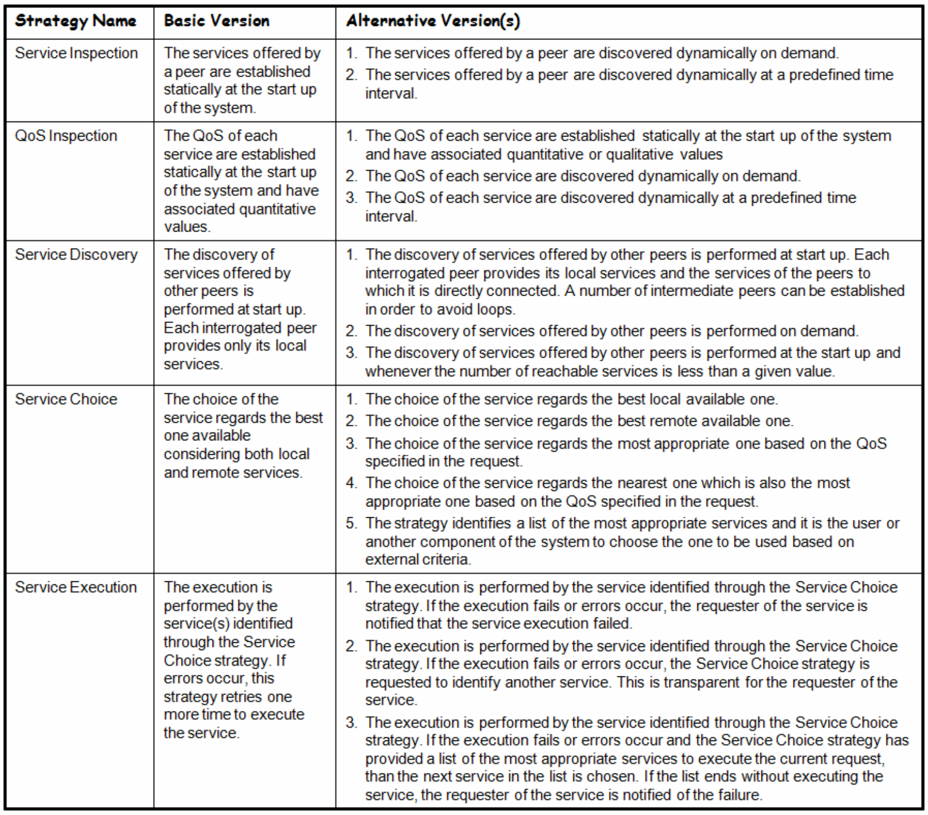
- an XML description of the requests;
- a sequence of steps expressed in terms of a finite state machine chart; this sequence guarantees the order of the steps to execute a request.

In ARMANIS all the information exchanged among peers is document centric.

3.2 System Strategies

The System Manager performs five main tasks: the inspection of the system services offered by the local peer, the inspection of the qualities of services (QoS) of the system services offered by the local peer, the discovery of system services on remote peers, the choice of the most appropriate service for a request (considering local and remote services) and the execution of services. Thus, it defines five types of strategies one for each of these tasks. These strategies are described in Table 1. Each strategy may have one or more alternatives.

Table 1. System Strategies



Strategy Name	**Basic Version**	**Alternative Version(s)**
Service Inspection	The services offered by a peer are established statically at the start up of the system.	1. The services offered by a peer are discovered dynamically on demand. 2. The services offered by a peer are discovered dynamically at a predefined time interval.
QoS Inspection	The QoS of each service are established statically at the start up of the system and have associated quantitative values.	1. The QoS of each service are established statically at the start up of the system and have associated quantitative or qualitative values 2. The QoS of each service are discovered dynamically on demand. 3. The QoS of each service are discovered dynamically at a predefined time interval.
Service Discovery	The discovery of services offered by other peers is performed at start up. Each interrogated peer provides only its local services.	1. The discovery of services offered by other peers is performed at start up. Each interrogated peer provides its local services and the services of the peers to which it is directly connected. A number of intermediate peers can be established in order to avoid loops. 2. The discovery of services offered by other peers is performed on demand. 3. The discovery of services offered by other peers is performed at the start up and whenever the number of reachable services is less than a given value.
Service Choice	The choice of the service regards the best one available considering both local and remote services.	1. The choice of the service regards the best local available one. 2. The choice of the service regards the best remote available one. 3. The choice of the service regards the most appropriate one based on the QoS specified in the request. 4. The choice of the service regards the nearest one which is also the most appropriate one based on the QoS specified in the request. 5. The strategy identifies a list of the most appropriate services and it is the user or another component of the system to choose the one to be used based on external criteria.
Service Execution	The execution is performed by the service(s) identified through the Service Choice strategy. If errors occur, this strategy retries one more time to execute the service.	1. The execution is performed by the service identified through the Service Choice strategy. If the execution fails or errors occur, the requester of the service is notified that the service execution failed. 2. The execution is performed by the service identified through the Service Choice strategy. If the execution fails or errors occur, the Service Choice strategy is requested to identify another service. This is transparent for the requester of the service. 3. The execution is performed by the service identified through the Service Choice strategy. If the execution fails or errors occur and the Service Choice strategy has provided a list of the most appropriate services to execute the current request, than the next service in the list is chosen. If the list ends without executing the service, the requester of the service is notified of the failure.

The service inspection strategies are exploited to identify which are the services offered by a peer. The services are described through a common ontology. The basic version of this strategy considers that services are static, meaning that no services can be added or removed at run-time. Such a strategy is applicable in most of the cases. For example, the services offered by a server or a mobile phone hardly change at runtime. However, there are cases in which new types of services are added or a new type of network is available, thus alternatives to this strategy considering this aspect have been also defined.

Similar strategies have been inserted for the identification of the QoS associated to each available service. Furthermore, QoS may have both qualitative and qualitative values. For more information about the mapping between high-level QoS and low-level QoS see [13].

The service discovery strategies are defined to inspect the services offered by other peers in the distributed system. Two approaches can be defined for this type of strategy. Through the first, only the peers directly connected to the requester one are interrogated. The second gives the possibility to specify how many intermediate peers should be interrogated. For example, establishing the intermediary peers at 1, peer P1 sees the S1.1, S1.2, S2.1, S2.2, S3.1, and S3.2 services provided by the P1, P2 and P3 peers (see Figure 3). In this case P1 cannot reach the services provided by peer P4.

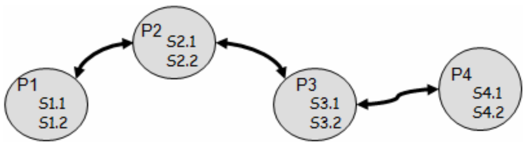


Fig. 3. Connection among peers

The service choice strategies decide which system service is used to execute the current request. In our approach, this choice is performed based on the QoS and the location of services [10]. To each service having the type of the requested one is assigned a score which indicates how close the QoS and its location are with respect to those requested. The lower the score is, the better it suits to execute the service. This strategy may provide also a list of the most suitable services which can fulfil the request. This option is very useful when the user wants to choose himself the service, or when other components of a system choose the most appropriate service. In the case of execution failures, having a two or more services which can execute the request avoids the overheads needed to identify other candidates.

3.3 Connectivity Strategies

The Connectivity Manager performs three main tasks: the discovery of other ARMANIS-enabled peers, the management of the connection among peers, and the management of the disconnection of peers. Thus, it defines three types of strategies one for each of these tasks (see Table 2).

The strategy to discover ARMANIS-enabled peers depends on the type of the network. In a highly dynamic system, the discovery of peers is done permanently, while in an almost static network discovery it is done at the start-up and/or on demand. An intermediary approach considers that the discovery of other reachable peers is done when the number of the identified or connected peers is less than a minimum number.

The opening of connections to remote ARMANIS-enabled peers depends on the type of the application domain. If it is based on a high communication and collaboration among peers, then the connection is established whenever a peer is discovered. Otherwise, it is more efficient to establish a connection on demand. Furthermore, connections may be opened when the number of the connected peers or the number of the responses received from the connected peers is less than a specific value.

Table 2. Connectivity Strategies

Strategy Name	**Basic Version**	**Alternative Version(s)**
Peer Discovery	The discovery of peers starts when the system is initialized and never ends.	1. The discovery of peers is performed at the start up of the system. 2. The discovery of peers is performed at the start up of the system and it is done whenever the number of the discovered peers is less then a given value. 3. The discovery of peers is performed at the start up of the system and it is done whenever the number of the connected peers is less then a given value. 4. The discovery of peers is performed on demand.
Peer Connection	The connection to a remote peer is performed whenever a peer is discovered.	1. The connection to remote peers is performed at the start up of the system when peers are discovered. 2. The connection to a remote peer is performed on demand and the peer is chosen among the ones identified during the start up of the system. 3. The connection to remote peers is performed at the start up of the system and it is done whenever the number of the connected peers is less then a given value. 4. The connection to remote peers is performed at the start up of the system and it is done whenever the number of the successful responses to a service request is less then a given number.
Peer Disconnection	The disconnection of a peer is performed when a communication exception is caught.	1. The disconnection of a peer is performed during the discovery process when a peer is no more reachable.

The disconnection of peers is done whenever a communication exception is caught due to the fact that a peer does not respond before a specific time limit. An alternative is to disconnect peers during the discovery process when peers become unavailable.

4 ARMANIS Applied to a Case Study

The domain knowledge for the case study introduced in Section 2 is composed of information related to customers, staff members and the financial and business services offered by the bank. Information related to the customers includes personal data, account number and conditions, and contracted services. The information of the staff members are related to their personal data, qualification and role(s), and access rights. In addition, for each staff member there are indicated alert channels which are related to the type of devices he can use for professional activities and the modality of notification. For example, a cashier uses always a desktop computer. He receives alerts of different importance in different ways: through a simple pop-up alert, a modal pop-up requiring user intervention to be closed, or a full screen message. A consultant may use a desktop or a laptop. When he uses the desktop, his location is fixed, while when using the laptop he may change his location. The director of the bank agency may use a desktop, a laptop, a PDA or a mobile phone.

The system knowledge is composed of all the devices (e.g., monitors, servers, desktops, laptops, PDAs) available in the bank agency, the services they provide (e.g., display, print, photocopy) and their related QoS (e.g., resolution, dimension, number of printed pages per minute). Each device with computational characteristics is considered an ARMANIS peer. For example, a printer is not considered a peer. It is connected to a printer server or to a desktop and the last devices are those which provide the printing service.

Due to the fact that this case study is not a very dynamic one, the service and QoS inspections are performed on demand when an upgrade is done. The service inspection is also performed on demand. The strategies related to the service choice are different for different services. For example, in the case of choosing a printer, version number 4 (see Table 1) is adopted: the nearest one which is the most appropriate one based on the QoS specified in the request (e.g., A3 format and color printer). In the case of the display service, version number 1 is chosen: the best local available one.

For the execution of services, the basic version of the strategy is used in the context of this case study.

The connectivity knowledge is related to the topology of the bank agency network. The discovery of peers is done on demand. For example, it is performed when a staff member enters or leaves the bank with one of the devices through which he performs professional tasks. The presence detector identifies the staff member who transits an access point, and this event is the trigger for the execution of the peer discovery strategy.

The basic version of the peer connection strategy is used in this case study. For the disconnection strategy version 1 has been adopted.

Figure 4 shows the main steps performed by the system when a customer enters the bank agency. It shows the key points where strategies are used.

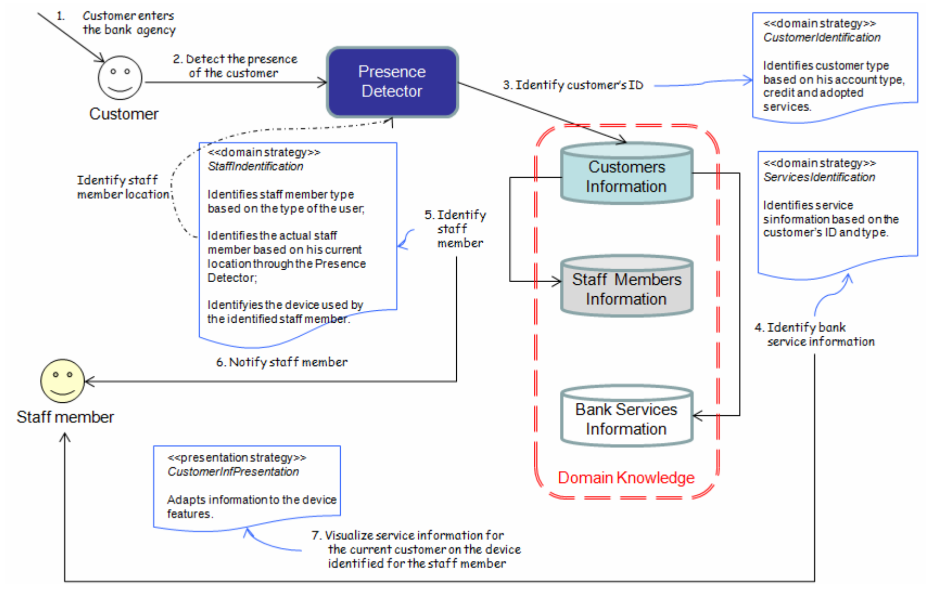

1. Customer enters the bank agency
2. Detect the presence of the customer
Customer
Presence Detector
3. Identify customer's ID
<<domain strategy>> CustomerIdentification
Identifies customer type based on his account type, credit and adopted services.
Identify staff member location
<<domain strategy>> StaffIndentification
Identifies staff member type based on the type of the user;
Identifies the actual staff member based on his current location through the Presence Detector;
Identifyies the device used by the identified staff member.
5. Identify staff member
Customers Information
Staff Members Information
Bank Services Information
Domain Knowledge
<<domain strategy>> ServicesIdentification
Identifies service sinformation based on the customer's ID and type.
4. Identify bank service information
6. Notify staff member
Staff member
<<presentation strategy>> CustomerInfPresentation
Adapts information to the device features.
7. Visualize service information for the current customer on the device identified for the staff member

Fig. 4. The Main Steps for the Finance Case Study related to the Notification of a Customer's Presence to the Staff Members

In the following, we describe how ARMANIS is exploited in the scenarios introduced in Section 2.

Scenario 1: In this scenario only domain knowledge is used. Adaptivity is performed in the context of the domain knowledge. The system tries to extract the best matching between the customer's profile and the advertisings offered by the bank.

The customer's presence is revealed by the presence detector module: based on the credit card identification tag, the system is able to identify and access the customer's information. We have designed a domain specific strategy called *CustomerIdentification* (see Figure 4) to categorize the customer based on his account profile and exploited services. Such a strategy depends on several factors which may be independent of the account information (e.g., when the customer is an important business or political person) or dependent on the account information (e.g., the customer has a significant amount of non-invested money). This strategy may change depending on internal (e.g., the current objectives of the bank) or external factors (e.g., financial crisis) and, consequently, consider one customer as important even if previously he was considered an ordinary one, or vice-versa.

Once the customer has been identified and categorized, two further activities are performed (see Figure 4 – step 4 and 5). These activities may be fulfilled concurrently. One is related to the identification of the services which may be proposed to the customer. This step exploits a domain strategy called *ServicesIdentification* (see Figure 4). Actually, this strategy may be composed of several strategies which are combined in order to provide the best solution for the current client. They consider the available funds of the customer, the services he uses, the services he is not using yet, as well as the current advertisings of the bank. The other activity regards the identification of the alert channels related to the staff members to be notified of the presence of the customer based on the last's type. This task is performed by the domain strategy called *StaffIdentification* (see Figure 4). In this scenario, this step consists in the identification of the cashiers at work and present at their working stations.

After these two activities are fulfilled the system notifies the identified cashiers of the presence of the customer and displays the financial services extracted to be proposed to the customer in an appropriate way decided by the *CustomerInfNotification* strategy (see Figure 4).

The states of the system to perform these activities related to the notification of a customer's presence to the staff members are described in Table 3.

The staff members request further information while interacting with the customers.

At the end of the meeting, the staff members update the information related to the customers with the communications made to the customers and the obtained results in order to avoid repetitions in the future.

Table 3. Activities and states in the REP module for the notification of the customer's presence to the staff members

State Nr.	Activity Name	State Name
1.	BusyWaitPresenceNotification	CustomerPresenceNotification
2.	IdentifyCustomer	CustomerIdentification
3.	ExtractCustomerInformation	CustomerInformation
4.	IdentifyCustomerType	CustomerType
5.	IdentifyStaffMember	StaffMemberIdentification
6.	IdentifyAdvertisings	AdvertisingsIdentification
7.	NotifyStaffMember	StaffMemberNotification

Scenario 2: In the second scenario all types of knowledge are exploited. Besides the adaptivity issues related to the domain knowledge common with the first scenario, in this case adaptivity exploits also the system and connectivity parts of the system.

When an important customer enters the bank agency the director is notified. The domain manager identifies the devices registered for the director. The detection module identifies the current location of the director. Actually, this activity is performed only if the director is present in the agency, otherwise the system is already aware that he is out of the agency. This is due to the disconnection strategy of the connectivity manager which is notified by the presence detector module when the director leaves the agency and the domain specific strategy which manages the presence of the staff members at work.

If the director is in his office, the notification arrives on his desktop through a modal pop-up message. In all the other locations inside the agency, the notification arrives on the closest device (i.e., reachable peer identified by the service choice strategy of the system manager) where more information may be displayed at once.

If the director is not in the bank agency, then the *StaffIndentification* strategy (see Figure 4) requires the notification of the vice-director (or the next staff member in the hierarchy at work) and the notification of the director on his mobile phone.

4.1 Implementation Notes

The current implementation of our approach considers three types of peers: front-end, central and service. Front-end peers run applications which are used by the staff members. Applications are written in C# (for desktops) and J2ME (for mobile devices). This type of peers communicates through XML-RPC protocol with the peers storing domain information and offering services. The central peers are used to store information related to the staff members and to the customers and to offer domain specific services. In addition, they store information about the passage of customers through choke points (e.g., customers entering or leaving a bank agency). Applications running on central peers are written in the Java language due to its portability feature which allows them to be independent of any operating system. The service peers process signal information revealing the presence of the customers, which are further sent and stored on the central peers. Applications on service peers are written in C++ to have better performances in signals analysis.

The presence recognition system is composed of two antennas: one inside and one outside the agency. An antenna reads the tags carried by the customers and sends a signal to the system. Each signal is filtered in order to reduce the redundancy of information and has associated to a timestamp. In this way the system is able to infer if a customer is entering or leaving a bank agency. For more technical information on the customers' presence recognition see [16].

5 Conclusions and Further Work

Adaptivity is gaining more and more the attention of the academic and industrial worlds. This affirmation is sustained by the increasing number of events (e.g., conferences, workshops) and publications (e.g., ACM TAAS journal, books, special issues) having adaptivity as central topic, as well as by the various projects (e.g., Odyssey [9], ReMMoC (A Reflective Middleware to support Mobile Client Interoperability) [7], MobiPADS (Mobile Platform for Actively Deployable Service) [3], CARISMA (Context-Aware Reflective mIddleware System for Mobile Applications) [1]).

In this paper we have presented the main aspects of our architectural model for adaptive distributed systems focusing attention on the design of various types of strategies exploited to implement decisions at runtime. Three main types of strategies have been introduced. Strategies similar to the system services (service and QoS inspection, service discovery, service choice and service execution) have been defined also for domain services. Further work will be related to (1) the description of these strategies through a formal approach such as the one used in the context of the Rainbow project [5] and (2) the extension of the current set of defined strategies while considering other case studies.

We have described how our solution is used in the context of a case study. This case study is under development and we plan to extend it with further adaptive strategies and improve the already existent once. For example, when a person enters the bank and he is not a customer then the staff members may be interested in convincing the person to become a customer. Or, when the director is notified that an important customer is present in the bank agency, we will consider in the adaptation process also the activity the director is currently performing in order to avoid disturbing him from another important task. We plan to address also customers with financial problems having important debts. The system identifies and proposes the best solution for this type of customers to overcome their financial problems. In this scenario besides the cashiers also one of the account consultants is notified.

Further work will be related to performance evaluations considering additional case studies. For example, we aim to consider also case studies similar to the personalized tourist guides inside a museum or in a city in order to address a wider range of issues related to the design of strategies for adaptivity.

References

1. Capra, L., Emmerich, W., Mascolo, C.: CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications. IEEE Transactions on Software Engineering 29(10), 929–945 (2003)
2. Ceriani, S., Raibulet, C., Ubezio, L.: A Java Mobile-Enabled Environment to Access Adaptive Services. In: Proceedings of the 5th Principles and Practice of Programming in Java Conference, pp. 249–254. ACM Press, Lisbon (2007)
3. Chan, A.T.S., Chuang, S.N.: MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing. IEEE Transactions on Software Engineering 29(12), 1072–1085 (2003)
4. Finkenzeller, K.: The RFID Handbook – Fundamentals and Applications in Contactless Smart Cards and Identification. Wiley & Sons LTD, Swadlincote (2003)
5. Garlan, D., Cheng, S.W., Huang, A.-C., Schmerl, B., Steenkiste, P.: Rainbow: Architecture-based Self-Adaptation with Reusable Infrastructure. IEEE Computer 37(10), 46–54 (2004)
6. Gorton, I., Liu, Y., Trivedi, N.: An extensible and lightweight architecture for adaptive server applications. Software – Practice and Experience Journal (2007)

7. Grace, P., Blair, G.S., Samuel, S.: ReMMoC: A reflective Middleware to Support Mobile Client Interoperability. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 1170–1187. Springer, Heidelberg (2003)
8. McKinley, P.K., Sadjadi, S.M., Kasten, E.P., Cheng, B.H.C.: Composing Adaptive Software. Computer 37(7), 56–64 (2004)
9. Noble, B.: System Support for Mobile, Adaptive Applications. IEEE Personal Communications, 44–49 (2000)
10. Raibulet, C., Arcelli, F., Mussino, S., Riva, M., Tisato, F., Ubezio, L.: Components in an Adaptive and QoS-based Architecture. In: Proceedings of the ICSE 2006 Workshop on Software Engineering for Adaptive and Self-Managing Systems, pp. 65–71. IEEE Press, Los Alamitos (2006)
11. Raibulet, C., Ubezio, L., Mussino, S.: An Adaptive Resource Management Approach for a Healthcare System. In: Proceedings of the 19th International Conference on Software Engineering & Knowledge Engineering, Boston, Massachusetts, USA, pp. 286–291 (2007)
12. Raibulet, C., Arcelli, F., Mussino, S.: Exploiting Reflection to Design and Manage Services for an Adaptive Resource Management System. In: Proceedings of the IEEE International Conference on Service Systems and Service Management, pp. 1363–1368. IEEE Press, Los Alamitos (2006)
13. Raibulet, C., Arcelli, F., Mussino, S.: Mapping the QoS of Services on the QoS of the Systems' Resources in an Adaptive Resource Management System. In: Proceedings of the 2006 IEEE International Conference on Services Computing, pp. 529–530. IEEE Computer Society Press, Los Alamitos (2006)
14. Raibulet, C., Ubezio, L., Gobbo, W.: Leveraging on Strategies to Achieve Adaptivity in a Distributed Architecture. In: Proceedings of the 7th Workshop on Adaptive and Reflective Middleware (2008)
15. Song, J., Kim, H.: The RFID Middleware System Supporting Context-Aware Access Control Service. In: Proceedings of the 8th International Conference on Advances Communication Technology, vol. 1, pp. 863–867. IEEE Press, Los Alamitos (2006)
16. Ubezio, L., Valle, E., Raibulet, C.: Management of Multi-Services Structures through an Access Control Framework. In: Kaschek, R., et al. (eds.) UNISCON 2008. LNBIP 5, pp. 519–530. Springer, Heidelberg (2008)