A Pattern-Based Approach to Mobile Information Systems Conceptual Architecture

Walter A. Risi and Gustavo Rossi

LIFIA, Universidad Nacional de La Plata, Argentina
{walter,gustavo}@@lifia.info.unlp.edu.ar

Abstract. As mobile access to information becomes a common trend in most commercial and corporate information systems, new software engineering challenges arise. While the most visible problems in mobile systems are the communication infrastructure and technical constraints of portable devices, there are also conceptual architecture issues that must be properly addressed.
This paper describes our work in progress towards categorizing and analyzing conceptual architectures for mobile information systems. We have defined the basics of a domain-specific architectural pattern catalog for this domain, and mined several patterns from existing, well-known systems. Our approach does not concentrate on the technical issues, but rather on identifying those architectures that better fit business goals regarding mobility.
Our goals are twofold. First, we aim to build a catalog of architectural solutions proven successful in industrial practice. On the other hand, we want to define a high-level, common-language to describe mobile architectural solutions.

1 Introduction

As mobile access to information becomes a common trend in most commercial and corporate information systems, new software engineering challenges arise. While the most visible problems in mobile systems are the communication infrastructure and technical constraints of portable devices (e.g. mobile phones and PDAs), there are also architectural issues that must be properly addressed at the conceptual level.

Current research in software artifacts regarding mobility concentrates primarily on fine-grained technological issues, such as wireless networking [14] and constrained hardware [13]. Unfortunately mobile information systems are seldom stable (because of technological improvements or changes in the underlying business model) and must be continuously upgraded or improved. Though it is widely known the importance of emphasizing design issues in applications that evolve fast, little attention has being paid to analyzing the conceptual issues that must be addressed when building a mobility-enabled information system

A. Olivé et al. (Eds.): ER 2002 Ws, LNCS 2784, pp. 159–172, 2003.

architecture. Only recently some design problems related with mobile software have been addressed in the literature [10].

Despite the hype around the technological issues, we claim that the architectural issues to address are even more critical. The design of a mobile information system architecture has challenges that simply go beyond the purely technological restrictions. For example, a mobile information system may require customizing its service for different types of users (mobile or not). On the other hand, a mobile information system may need to give mobile service to very different type of mobile users (e.g. wired and wireless).

This paper describes our work in progress towards categorizing and analyzing conceptual architectures for mobile information systems. We have defined the basics of a domain-specific architectural pattern catalog for this domain, and mined several patterns from existing, well-known systems. Our approach does not concentrate on the technical issues, but rather on identifying those architectures that better fit business goals regarding mobility.

We are building a catalog of architectural solutions proven successful in industrial practice. This catalog should be the basis of a high-level, common language to describe mobile architectural solutions. This language can be used during the process of conceptual modeling of this kind of applications, both to provide specificity to a general purpose modeling language (such as UML) and to face modeling and design problems at a higher abstraction level by reusing successful solutions.

This paper is organized as follows. In Sect. 2, we briefly review some of the concepts underlying this paper - such as software architecture and patterns. In Sect. 3, we present the general motivation for our approach. In Sect. 4, we present a preliminary classification of our patterns. In Sect. 5, we present some of the patterns we mined. Sect. 6 presents shows how our approach relates to the actual process of building and analyzing applications . Finally, in Sect. 7 we draw some conclusions and describe our lines for future work.

2 Software Architecture and Patterns

In recent years, there has been some discussion regarding the software architecture definition. In this paper, we use the term software architecture to refer to a structural plan that describes the elements of the system, how the interact together, and how these interactions fit the business goals. In other words, we adopt the *software blueprint* view of architecture [12,9]. We believe this is the most suitable view for today's industrial practice.

Though there are differences among the existing approaches to architecture, most of them agree in separating architectural concerns into several views [12]. This paper concentrates on what has been called the *conceptual view* [9]. This view describes the architecture in terms of the high-level computational elements that comprise the solution. In this view, the architect organizes the domain-specific computational elements (components) and the interactions between them (connectors), building a solution that meets the business require-

ments. For example, an architect designing a compiler may produce an architecture consisting of several components (e.g. parser, lexer), connected to each other by specific connectors (e.g. direct procedure calls).

The concept of *architectural patterns* emerged several years ago [15] and it is indeed similar to the widely spread notion of design patterns [7]. They share the intent of capturing, recording and conveying design experience. A pattern allows describing a recurrent design problem, together with its solution in an abstract way, such that the solution can be applied to different (though similar) problems. They provide a design vocabulary for the domain to which they are applied. However, while design patterns are used at the so-called micro-architectural level in which the design components are classes, objects and the interaction mechanisms are messages, architectural patterns deal with a higher-level view of the system in which primitives are components and connectors.

3 Why Mobile Architecture Patterns?

The definition of a mobile information system architecture is nowadays rather ad-hoc by current software engineering standards. The software architect relies on existing approaches to general software architecture, well-known solutions to fine-grained issues (e.g. underlying technology) and commercial solutions. However, the architect's toolbox lacks a catalog of vendor-independent architectural solutions. Indeed, most approaches consist in buying a predefined solution without a previous architectural analysis of the underlying problem.

While choosing commercial solutions is in general a cost-effective approach, performing an high-level architectural analysis of both the problem and the candidate solutions can help choose the best-fit solution for the problem. Instead of choosing an architecture based on vendor's claims about it, an architect could define a candidate architecture, analyze available architectures, and then decide on buying or building a solution.

Recognizing this need, we have defined a general strategy towards the classification and analysis of mobile architectures proven successful in industrial practice. Our strategy consists in the definition, maintenance and usage of a *pattern catalog*, providing a handbook for the architect facing this kind of engineering task. As mentioned before, these patterns help to improve the conceptual vocabulary of the designer, thus providing meaningful abstractions during conceptual modeling and architectural design.

4 Pattern Classification

Mobile devices are highly constrained in nature. Distinct devices have very different types of limitations, and where one device is strong another may be very weak. However, a mobile information system may have to consider the availability of the service to those very different devices. On the other hand, mobile users are not interested in exactly the same kind of information that standard users are. The classification we describe in the following was defined to reflect the

different dimensions of these problems in the light of the already mined patterns – that is, following a bottom-up approach . We believe that this classification may evolve and even grow as new patterns are mined.

Physical Accessing. This category addresses the issues about how to physically access an information source from a mobile client, depending on the constraints imposed by requirements and device restrictions. Hardware limitations have a strong impact on the way mobile devices access to information.

For example, a mobile phone has better accessing facilities than a standalone PDA (i.e. without a modem) depending on a synchronization to a desktop machine to retrieve information. A PDA with a dial-up modem has better accessing facilities than a standalone PDA, but less autonomy than a wireless-enabled PDA. However, a dial-up connection is likely to be available in almost every city of the world, while today's wireless network areas are reduced.

While some alternatives may be preferable to others in certain contexts, the designer of the information system architecture may have to provide a solution that works both for a high-end autonomous clients and also for a client very limited in terms of information accessing. A common reason for this is that most adequate devices are also the most expensive ones, and the majority of users have devices that stand in the middle-ground, or even in the low-end of the spectrum.

Logical Addressing. This category addresses the issues that arise when a mobile client has to address a space of information sources. While a desktop client of an information system has a context suitable for freely navigating through the space of information sources, the choice of a mobile client is generally more reduced. A mobile client may be capable of processing only a subset of the whole universe of information – e.g. a subset of the web –, and even if capable, may only be interested in seeing a subset of it.

For example, most PDA devices have specialized browsers to access information from the web. However, most of these browsers do not work well with any page, but only with a subset that are encoded in such a way that they are suitable for mobile devices – e.g. smaller screens and a limited HTML syntax. A mobile user may not want to see the whole universe of information sources, but rather the subset that its suitable for its device and its needs – e.g. a set of mobile channels, suitable for PDA accessing.

Constrained devices are weaker and more specialized than mainstream devices. The choice of available information sources that the device can access is generally a subset of the whole universe of choices, not only because of hardware limitations but also because of the specialized interests of the mobile user . A mobile client in these conditions should not have to deal with the complexity of addressing the whole universe of information sources, but rather the subset in which it is interested.

Customization. This category provides architectural patterns that address the problems of customizing the information sources themselves. Again, the constrained nature of mobile devices poses physical and conceptual restrictions to the information that can be deployed in such a device. A mobile device may

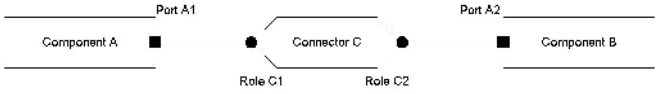

Fig. 1. Using the UML to Describe Conceptual Architecture Concepts

not be capable of interpreting any information source encoding. Even if so, the mobile device user may not be interested in the same view of the information that a desktop client is.

For example, while most PDA devices have web-browsers, the screen limitations makes some web pages impossible to be viewed on such devices. Some browsers even do not support the complete HTML syntax, and therefore cannot access arbitrary pages on the web. Finally, a mobile user accessing a web site may be interested only in information relevant to its mobile context - e.g. a businessman accessing a financial company web site is more interested in finding market information rather than the history of the company.

A mobile user may be interested in the same information source that a non-user user is interested in, but not exactly in the same view of that source. Mobile devices may not be powerful enough to process the information source – e.g. for limitations in parsing the information format. Even if so, mobile users are interested in a view of the information source that is relevant to their mobile context.

5 Pattern Catalog

In this section we present some of the patterns we have mined in the categories mentioned in the previous section. We use a format similar to the one used in [7], though slightly customized to better fit the higher-level nature of architectural patterns. We have also included references to commercial implementations of each pattern, so that the practicing architect can use it to evaluate existing alternatives to the implementation of the pattern. Actually, the list is reduced in this paper for space reasons – and included in the *Known Uses* section – but shows the general intent of serving as a suggestion list. Again for space reasons, we have only included the *Related Patterns* section in those patterns for which we consider it more valuable.

Our pattern format includes an *UML* [8] diagram to explain the structure of the solution. While the UML is not particularly oriented to architecture description, there are several approaches to the usage of the UML for software architectures. We have deliberately chosen the one defined in [9] for being both simple and expressive. Note that we have actually simplified the mentioned notation for simplicity reasons, and instead we compliment our diagrams with textual explanations.

The diagram depicted in Fig. 1 shows an example of the notation we adopted. The diagram shows a component connected to a connector. A *component* is an

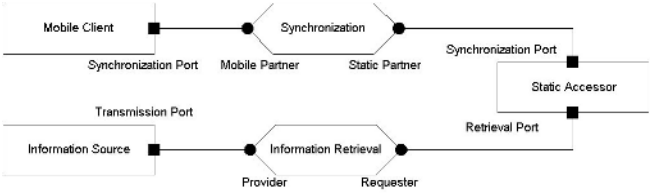

Fig. 2. Static Accessor

architectural entity that represents a conceptual unit of computation. The component exposes its functionality through one or more ports. The component can interact with other components through connectors, which represent a communication mechanism. A *connector* connects two or more components, through their ports. These elements are actually *stereotyped classes* - for more information about the usage of these stereotypes in this context, refer to [9].

5.1 Physical Accessing Pattern: Static Accessor

Intent. Provide a mobile client with the possibility of accessing an information source, when the client itself is not capable of accessing the information source on its own.
Problem. You have to deploy information into a constrained device that does not have enough capabilities to gather information on its own (e.g. a PDA without a modem, and without wireless capabilities). It may happen that the most common configuration of such a device does not provide such facilities (e.g. most Palm and PocketPC devices do not provide a modem in its basic configuration). It may also be the case that the targeted line of devices does not have such facilities at all.
Solution. Make the mobile client dependent of a more powerful, non mobile device. Provide a way so that the dependable client can be fed from the static device – the static accessor. The responsibility of interacting with the data source belongs to the static device, and the dependent client takes the information through the static accessor. See Fig. 2.
Participants. The *Mobile Client* is the mobile device interested in the information available from the information sources. The *Static Accessor* is a non-mobile device capable of retrieving information from the sources. The *Synchronization* connector is a mechanism that allows the mobile client to get the retrieved information from the static accessor. The *Information Retrieval* connector is a mechanism that allows the static accessor to get information from the information source. Finally, the *Information Source* is the information source in which the mobile client is interested.
Collaborations. Whenever the Mobile Client requires information from the Information Source, the Synchronization mechanism is activated. The latter can be activated from the Mobile Client side, or from the Static Accessor side (e.g.

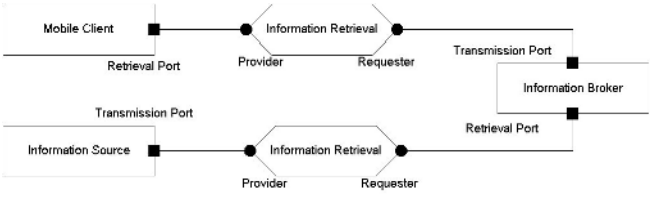

Fig. 3. Information Broker

when the static accessor discovers that there's new information available from the Information Source). The Static Accessor communicates with the Information Source through an Information Retrieval mechanism, getting the information in which the Mobile Client is interested.
Consequences. Any portable device gets the possibility of interacting with an information system as long as a suitable accessor is built. The autonomy of the client is reduced, since the information retrieval process requires the presence of the static accessor. The mobile client part is simplified, but having to provide a suitable accessor may involve additional work for the developers.
Known Uses. The mobile information system AvantGo [2] uses Palm OS's HotSync [3] technology to transfer information from information sources (AvantGo channels) to the dependable clients (Palm OS handhelds) through a static accessor (a desktop machine). AcroClip [1] for Casio OS handhelds also uses this pattern.

5.2 Logical Addressing Pattern: Information Broker

Intent. Take control of a mobile client's addressing capabilities. Have the address space of available information sources preprocessed by an intermediary.
Problem. You have to allow the mobile user to retrieve information from an arbitrary or predefined set of information sources (e.g. web sites). You want to keep the mobile client simple (e.g. due to hardware restrictions), and therefore you need to reduce the information addressing capability of the mobile device. It may also be the case that you want to prevent mobile clients to access arbitrary information sources – e.g. for corporate security reasons. In that case, you need an intermediary capable of assuming the responsibility of addressing the whole universe of information sources – e.g. the whole web – and presenting a transformed universe to the mobile client – e.g. a set of corporate channels.
Solution. Provide a broker capable of interacting with the whole universe of information, presenting the mobile client with a transformed (possibly reduced) view of that universe, where that view is suitable for mobile access. See Fig. 3.
Participants. The *Mobile Client* is the mobile device interested in the information available from the information sources. The *Information Retrieval* connector is a communication mechanism that allows information transfer between components. The two appearances of the information retrieval connectors in the

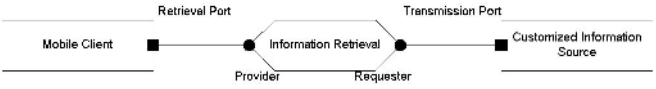

Fig. 4. Custom Authoring

diagram does not imply that both appearances are identical. The *Information Broker* is software component that can address information sources and retrieve data that can be delivered to the clients. Finally, the *Information Source* is the information source in which the mobile client is interested.
Collaborations. The Mobile Client gets information from the Information Broker through an Information Retrieval mechanism. The Information Broker retrieves information from a set of available Information Sources using another Information Retrieval mechanism
Consequences. The mobile client addressing system may be simplified, but having to provide a suitable broker may involve additional work for the developers. The broker provider has control on the information that the mobile clients access.
Known Uses. AvantGo [2] centralizes the access to a growing set of channels, where AvantGo's M-Commerce server acts as a broker between the information sources and the actual mobile clients. The AcroClip [1] service also offers access through channels, controlled by a proprietary broker.

5.3 Customization Pattern: Custom Authoring

Intent. Provide the mobile client with information suitable for its direct manipulation, so that no expensive customization is required.
Problem. You have to deploy information from information sources (e.g. web sites) in a constrained device that does not support the format in which those sources encode their information (e.g. Palm's m100 handheld, for which a full-blown web browser is not available).
Solution. For each information source of interest, provide a customized version of that source, so that the format is exactly the one suitable for the client. See Fig. 4.
Participants. The *Mobile Client* is the mobile device interested in the information available from the information sources. The *Customized Source* is an information source that represents information in a format that can be handled directly by the mobile client. Finally, the *Information Retrieval* connector is a mechanism that allows the mobile client to get information from the information source.
Collaborations. The Mobile Client retrieves information directly from the Information Source, through the Information Retrieval mechanism. Since the Information Source is already customized for the Mobile Client, no additional processing is necessary.

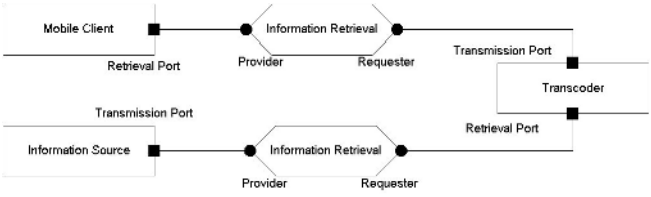

Fig. 5. Transcoder

Consequences. The information is specially designed for the mobile clients in mind, so there is no possibility that it does not suit perfectly its clients. The content provider has additional work, since it has to provide a customized version of its information sources for each of the mobile clients in mind. Updating the information can be a heavy task, if there are several customized versions to be maintained.
Known Uses. AvantGo [2] provides an infrastructure so that clients provide their own customized versions of the information they provide.

5.4 Customization Pattern: Transcoder

Intent. Automate the process of adapting information from arbitrary sources for mobile clients' manipulation.
Problem. You have to deploy information from information sources (e.g. web sites) in a constrained device that does not support the format in which those sources encode their information (e.g. Palm's m100 handheld, for which a full-blown web browser is not available). The cost of making customized versions of the information sources is too high (e.g. because the number of information sources is too big, or because the number of different targeted devices is high).
Solution. Create a component able to convert data from arbitrary information sources to a format suitable for the potential clients. See Fig. 5.
Participants. The *Mobile Client* is the mobile device interested in the information available from the information sources. The *Information Retrieval* connector is a communication mechanism that allows information transfer between components. The two appearances of the information retrieval connectors in the diagram does not imply that both appearances are identical. The *Transcoder* is software component that can translate information represented in an particular format to a format that can be handled by the mobile client. Finally, the *Information Source* is the information source in which the mobile client is interested.
Collaborations. The Mobile Client receives information from the Trans-coder using an Information Retrieval mechanism. The Transcoder receives information in its original format from the Information Source using another Information Retrieval mechanism.
Consequences. The information provider has no additional work in the customization of the information for the mobile clients. The transcoder automat-

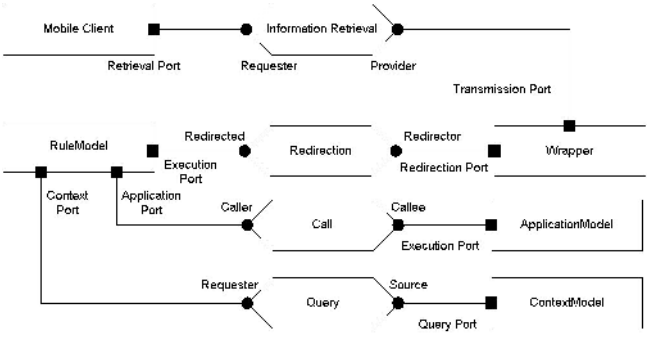

Fig. 6. Customized Behavior

ically handles this task. Building the transcoder can be a difficult task. There maybe several mobile clients in mind, with significant differences that the transcoder must support. The parsing of the original information source can be difficult in itself.

Known Uses. PumaTech [4] provides a service that allows accessing to standard web pages via Palm OS devices. PumaTech's solution relies on conversion heuristics that convert pages to version suitable for constrained hardware, in a way that semantics are preserved. IBM also provides a transcoder-based solution as part of its *Websphere* family of products [5]. AcroClip [1] has built-it transcoders for several web sites, and also allows users to define their own.

5.5 Customization Pattern: Customized Behavior

Intent. Provide different application response according to the mobile client.

Problem. In many situations you need to adapt an applications functionality to the kind of interface device (a web browser, a mobile appliance) in a dynamic way. Suppose that you are building an Internet application for selling products in the style of *amazon.com*. You may want to give special offers to users of some mobile devices: for example you have just signed an agreement with a cellular phone company and you want to induce customers to use that device. Unfortunately this business rule may change often; the discount rate may change, the condition on products that the customer must buy may vary, etc. If you tie the business rules either to the application model or to the components standing for the interface device you may have maintenance problems

Solution. Decouple the business rules from the application model and these from a context model that describes the current users state (device, network connection, etc). See Fig. 6.

Participants. The *Mobile Client* is the mobile device interested in the information available from the information sources. The *Application Model* contains the base application objects and behaviors. It should be built in

such a way of being independent of different adaptation policies (for example to different mobile devices, types of network connections, etc) The *Rule Model* contains the logic of customization; by interacting with the context model it triggers corresponding application behaviors. The *Context Model* represents the actual user profile, including the information on the current mobile device The *Wrapper* redirects client requests to the application model, to the customization model

Collaborations. For each client request that must be customized (for example according to the mobile device) the wrapper mechanism redirects the request to the Rule Model. Rules interact with both the context model (they usually represent conditions) and the application model triggering the corresponding actions. The result of the rules execution is returned to the client.

Consequences. The application model remains stable when we add new business rules related with mobile devices or new devices. The rule model may get monolithic if you add many flat rules.

Known Uses. In the *UWA (Ubiquitous Web Applications)* project [10] rules are separated from the application and the context model is itself composed of user, device and network sub-models. In the *Munich Reference Architecture* for adaptive hypermedia applications [11] a similar approach is used. WebML [6] also uses a variant of this pattern.

Related Patterns. The Transcoder pattern is similar to the Customized Behavior in that it provides an automated way of presenting different views of the information to the mobile clients. However, while the Transcoder focuses mainly on the *looks* of the information, the Customized Behavior pattern focuses on the *guts* of the information generation process. Therefore the Transcoder acts like a facade over a system, being an information-unaware solution. In the Customized Behavior pattern, the rules and context which comprise the customization mechanism are aware of the kind of information the system provides.

6 From Patterns to Applications

In this section we show how the patterns we presented can help in both describing a commercial solution and defining a solution to fit a business goal. We present three examples on the usage of patterns, that go from representing commercial architectures to potential usages of these patterns in an enterprise context.

6.1 Representing AvantGo's Dynamic Mobility Model

The *Dynamic Mobility Model* (DMM) [2] is the solution suggested by AvantGo to provide mobile access to web-based information systems.

In Fig. 7 we show the conceptual architecture of the DMM. The *Static Accessor* pattern is used to allow non-wireless devices to access the system. An *Information Broker* provides the mobile clients with a reduced version of the internet, based on *channels*. Finally, the *Custom Authoring* pattern is used to provide information encoding in a format directly suitable for mobile clients.

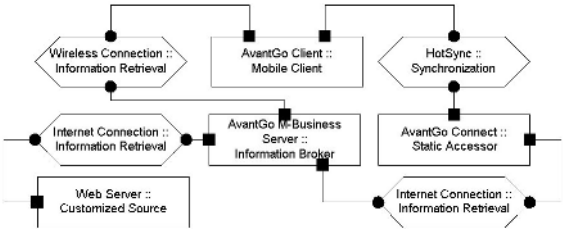

Fig. 7. AvantGo's Dynamic Mobility Model

Having a conceptual description of a technological infrastructure such as AvantGo enables architects to analyze the solution in a completely objective way. Given an actual problem, the architect can define the best-fit architecture and then compare it to AvantGo's to see if it really fits the business goals.

6.2 Customized Mobile Access to a Corporate Database

A software company wants to provide access to its corporate database to both their vendor force and also its customers. For its vendor force, it wants to provide updated information about their products, solutions and prices. Instead, for their customers, it wants to provide information about special offers and customized support. The first problem the architect faces is providing a customized view of its information database for the different users that may be interested in that information.

Since the vendor force needs updated information, the company decides to provide its vendors with wireless-enabled, high-end PDA devices. However, the customers have a variety of mobile devices, most of them not-wireless enabled. Thus, the second problem the architect faces is providing both wired and wireless access to the information. Also, the information encoding may be different for the high-end vendor devices, compared to the possibly-low-end customer devices.

Following the approach mentioned in section 3, the architect browses the architectural pattern catalog and selects those patterns that can help in solving the problem at hand. The resulting conceptual architecture is depicted in Fig. 8.

The proposed architecture is a high-level view of the solution. The *Static Accessor* pattern is used to provide wired access to the information source. The *Customized Behavior* pattern is used to provide different views on the same information source - the corporate database. Once the architecture is depicted, the architect can then discuss about building or buying a solution, depending on the available schedule and expense.

6.3 Mobile Access to a Commercial Website

A small company runs an entertainment web portal, which provides information about latest movies and movie theaters. The company believes that mobile infor-

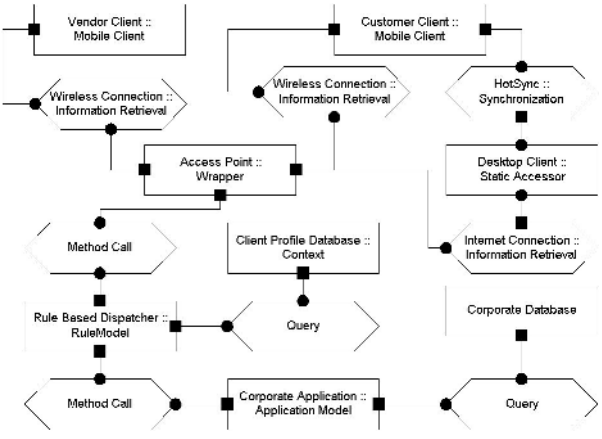

Fig. 8. Architecting Customized Mobile Access to a Corporate Database

mation access will provide it with an interesting advantage over its competitors. Moreover, since the company already has a website running, it wants to avoid having to author a parallel website for mobile access. Also, since the company has a small staff, management is heavily concerned about doubling the website maintenance effort.

The software architect browses the architectural pattern catalog and sees that a *Transcoder* based solution can solve both the development time effort and the staffing problems. The architect takes the suggested commercial component list from the pattern description and browses the web for commercial solutions, comparing prices and making a report about both the architecture and the prices the commercial tools involves. The architect also discards building a transcoder in-house for the development effort it involves. However, the architect also considers a *Custom Authoring* solution and calculates the approximate development and maintenance costs associated to it. The latter solution was also considered, since the commercial component providing access to customized sources has a significantly lower price than the transcoder-based solutions. The architect elaborates a report afterwards, and presents it to management for its cost/benefit evaluation.

7 Conclusions and Future Work

The current work on mobile information system architectures is mainly focused on the technological aspect of this kind of systems. While there is considerable research in approaches for implementing systems using a certain wireless technique or other, there are no studies about how the mobility aspect conceptually relates to an information system.

This paper presented an approach to describing, categorizing and analyzing conceptual architectures for mobile access to information systems. Our approach is based on the architectural pattern concept. We presented several patterns that represent common mobility configurations existing in current industrial practice.

Abstracting existing configuration as architectural patterns provides the designer with a catalog of architectures. The practitioner can then choose the architectural structures that best fit the business goals, without having to rely on vendor-specific solutions or terminology.

We are currently working on mining other architectural solutions that occur in existing mobile systems. Our intention is to grow a complete and solid architectural pattern language for providing mobile access to information systems. We are also working on defining an architectural methodology for the better usage of these patterns. Our main intention is to integrate these patterns into existing modeling tools and/or languages, in order to provide the architect with a toolbox of primitives so he can describe his mobile applications in a higher-level of abstraction.

References

1. AcroClip. `http://www.pcsync.de`.
2. AvantGo M-Commerce Server. `http://www.avantgo.com`.
3. HotSync Technology. `http://www.palm.com`.
4. PumaTech BrowseIt Server. `http://www.pumatech.com`.
5. Transcoding Publisher. `http://www.websphere.ibm.com`.
6. WebML. `http://www.webml.org`.
7. R.J. Erich Gamma, Richard Helm, and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison Wesley Professional Computing Series, 1994.
8. O.M. Group. *UML (version 1.3)*. OMG, June 1999.
9. C. Hofmeister, R. Nord, and D. Soni. *Applied Software Architecture*. Addison Wesley, 2000.
10. G. Kappel, W. Retschitzegger, and W. Schwinger. Modeling Ubiquitous Web Applications: The WUML Approach. In *International Workshop on Data Semantics in Web Information Systems (DASWIS)-2001), co-located with 20th International Conference on Conceptual Modeling (ER2001), Proceedings*, November 2001.
11. N. Koch and M. Wirsing. The Munich Reference Model for Adaptive Hypermedia Applications. To be presented at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, 2002.
12. P.B. Kruchten. The 4+1 View Model of Architecture. *IEEE Software*, 12(6):42–50, Nov. 1995.
13. J. Noble, C. Weir, and D. Bibby. *Small Memory Software: Patterns for Systems with Limited Memory*. Addison-Wesley, 2000.
14. S. Schaefer, S. Marney, T. Willis, and K. Parkkinen. OOPSLA Workshop on Architectural Patterns for Wireless Computing. 2001.
15. M. Shaw. Patterns for Software Architectures. In J. Coplien and D. Schmidt, editors, *Pattern Languages of Program Design*, chapter 24, pages 453–462. Addison-Wesley, 1995.