ALIGNING SYSTEM ARCHITECTURES ON REQUIREMENTS OF MOBILE BUSINESS PROCESSES

ABSTRACT
The support of mobile workers with mobile IT solutions can create tremendous improvements in mobile business processes of a company. The main characteristic of such a mobile system is the ability to connect via a (mobile) network to a central server, e.g. in order to access customer data. This paper presents a detailed description of the four main software architectures for mobile client/server-based systems and their main characteristics. Beyond, typical business requirements in mobile environments like the location of use, data topicality, interaction requirements, synchronisation mechanisms and many more are mapped onto each of these architectures. The presented results can be used for discussing concurrent business needs as well as for deriving a mobile system architecture based on these needs.

KEY WORDS
mobile system architecture, business requirements, software architecture

1 Introduction

Since the availability of mobile broadband networks and the reduced costs for mobile devices the use of mobile applications has become an interesting opportunity in several fields. Companies with large divisions of mobile employees (e.g. service technicians, sales representatives, healthcare services) can use mobile applications to gain access to corporate applications and databases at the point of service (POS). Therewith better coordination of mobile employees, rapid task assignment, the avoidance of error-prone format conversion, instant access to customer data and many more becomes feasible [6], [9]. The architecture of a mobile system can range from always-online systems using browser-based clients to fat client systems synchronizing with a central server occasionally. Which software architecture fits best for a specific mobile task depends basically on business needs. The frequency of movement, the probability of network availability, requirements for data topicality, update mechanisms, synchronisation procedures and many more play a crucial role. Beyond, the costs for the development of a mobile system depend strongly from the chosen type of architecture. As these issues are of particular relevance in the decision process of software architects and IT project managers, this paper explains the main types of mobile system architectures with their advantages and disadvantages, typical business requirements for mobile systems and a matching scheme in order to identify the suitable architecture for a given set of business requirements.

This paper is organized as follows: Section 2 gives an overview about related work. Section 3 introduces four main types of architectures for mobile systems, showing their structure with component diagrams and explaining their main characteristics as well as their advantages and disadvantages. In section 4, typical business requirements for mobile systems are given and interdependencies between the shown system architectures are described. Section 5 draws a conclusion.

2 Related Work

The changes for the discipline of software engineering when developing systems for mobile environments are discussed in [10]. The authors state that ”mobility represents a total meltdown of all stability assumptions [...] associated with distributed computing”. A comprehensive overview of software engineering for mobile systems is given, regarding issues like models, algorithms, applications and middleware to solve in the future. Our paper addresses the some of these issues. In [11] an architectural model that identifies the components representing the essential aspects of a mobile agent system is described. The interaction design for mobile information systems is subject of [3]. The authors developed a platform that supports the rapid prototyping of multi-channel, multi-modal, context-aware applications and describe how it was used to develop a tourist information system. A lot of work has been done regarding system architectures and other technical aspects of mobile system. An example for this work is [5], where a three-layer software architecture for distributed and mobile collaboration is presented. [12] presents an approach for the modelling and performance evaluation of mobile multimedia systems using generalized stochastic petri nets. The author focuses on verifying the optimal performance achievable under some QoS constraints in a given setting of design parameters. In [7] an approach for modelling software architectures for mobile distributed computing is presented. The modelling method aims at the verification of the correctness of both the functional and non-functional properties of the resulting mobile system. The authors of [2] report on a project that aims for providing a system architecture simplifying the task of implementing mobile applications with adaptive behaviour. Temporal, spatial and personal mobility are considered in this approach. A

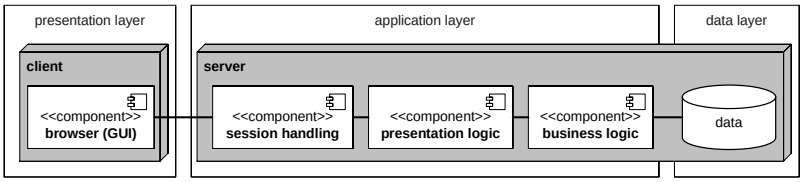

Figure 1. Web-based always-online architecture

comparison of different architectures for mobile systems is given in [8], where the client/server model, the agent-based client/server model and the mobile agent model is considered. In [13] is discussed, how the approach of service-oriented architectures can be applied within the development of mobile systems. It is shown, how to compose mobile services and how to apply these services in the system architecture. The work described in [4] is an essential basis for this paper. The authors give an overview about different types of software architectures for mobile systems, based on an analysis of different types of user and device mobility.

3 Types of Software Architectures for Mobile Systems

When analyzing the communication behaviour of a mobile application, its architecture is of particular relevance. According to [4], mainly four different types of mobile client/server-based systems can be distinguished. First, a complete offline architecture could be used where (nearly) no communication via a network occurs. As we focus mainly on network issues, we do not consider this type of architecture. Second, an offline architecture is conceivable, where the mobile user synchronizes the mobile application occasionally with a central server. Third, a hybrid architecture could combine the advantages of an offline and an online architecture: If a network is available the mobile application communicates online with a central server, if the network is unavailable, the mobile application works offline and can be synchronized later with the central server. Fourth, with an always online architecture, the mobile application would communicate with a central server exclusively. This architecture is typical for web-based systems. In the following, we present a component-based view onto these architectures and explain their main characteristics.

3.1 Web-based Always-online Architecture

Within this architecture (see Fig. 1), the client works always online via a (mobile) network. The presentation layer is completely realized at the client side, where only a browser component is needed to realize the graphical user interface (GUI). The application layer is located server-side and contains a session handling component as well as components for presentation and business logic. The data layer contains the databases at the server side.

The main advantage of this architecture is that only a browser is needed at the client side. Thus, the systems architecture allows the cooperation with a wide range of client systems independently from the client' operating systems or other client-side conditions. Furthermore, as all the data and the logic is located server-side, no update or synchronization mechanisms are needed. All clients can work on the same central data base, using the recent data as well as the recent presentation and business logic. The effort for the administration of such a systems is very small compared to other system architectures.

The main disadvantage of this solution is that always a (mobile) network is needed, otherwise the client is unusable. When using a mobile network, the coverage in certain areas might not be given. Beyond, mobile networks usually offer just a small bandwidth causing a probably unsatisfying performance of the client. Furthermore, browser-based applications are limited in terms of user interface design to the facilities of HTML, which is quite less than users know from fat client applications. The simultaneous connection of a large number of clients to the central server also causes high requirements for the central server regarding its performance as well as its operational availability.

3.2 Rich Client Always-online Architecture

Within this architecture (see Fig. 2), one disadvantage of the web-based always-online architecture is addressed: Using a rich client at the client side, the limitations of the user interface design to the facilities of HTML can be overcome, as rich clients offer the full variety of interaction elements for the interface design. Beyond, rich clients offer the advantage of moving presentation logic to the client and therewith offering performance and costs improvements through a reduction of data traffic via the mobile network. The presentation layer is completely realized on the client, containing a GUI and a presentation logic component. Additionally, the client contains some elements of the application layer, i.e. an update component for the presentation logic as well as a session handling component. Both components have an equivalent at the server side, where also a component for the business logic is located (application layer). The data layer is completely realized at the server side containing templates for the presentation logic as well as other databases.

The main advantages of this solution are the full user interface design capabilities and the reduced data traffic,

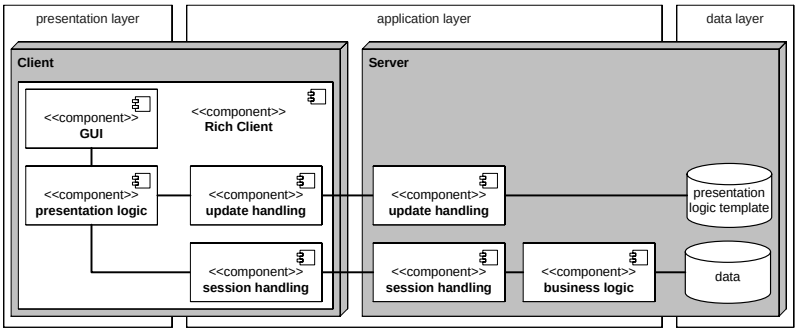

Figure 2. Rich Client Always-online Architecture

still having a thin client. Furthermore, all clients can work on the same central data base, using the recent data as well as the recent presentation and business logic. The effort for the administration of such a system is small compared to other system architectures (except the web-based always-online architecture). Furthermore, with rich clients partly asynchronous communication becomes feasible, producing an improved user interaction. Short-time network disconnection can be intercepted through the client-side session component. The disadvantages are the same as in the Web-based Always-online Architecture. Additionally, components for the session and update handling need to be developed, update mechanisms are needed.

3.3 Rich Client Hybrid Architecture

The two architectures described before suffer from one main disadvantage: If no mobile network is available, the application is not usable for the mobile worker. The rich client hybrid architecture addresses this issue (see Fig. 3). The aim of this architecture is to keep the client as thin as possible but to assure that the application works also in the emergency when no mobile network is available. The client consists of a GUI and a presentation logic component (presentation layer). Additionally, a business logic component as well as session handling, update handling and data synchronization components are located at the client side (application layer). Also a database is needed at the client (data layer). The business logic, session handling, update handling and synchronization components are also needed at the server side. The server-side data layer consists of templates for business and presentation logic and other databases. In the normal case the application works always-online, using business logic and data from the server side (like in the rich client always-online scenario). In case of loosing the network connection, the application would use the equivalent components at the client. Thus, a synchronization mechanism is needed.

The main advantage of this architecture is the ability to work always-online but having also the capability to work offline when no network is available. Through the use of a rich client, the full user interface design capabilities are available and reduced data traffic can be achieved. When establishing a network connection, all clients can work on the same central data base, using the recent data as well as the recent presentation and business logic. Furthermore, with rich clients partly asynchronous communication becomes feasible, producing an improved user interaction. Short-time network disconnection can be intercepted through the client-side session component. The disadvantages of this solution are mainly the same as in the two architectures described before. Additionally, when working offline, a synchronization mechanism needs to transfer data stored at the client, resolving possible conflicts with the server-side data base. In this architecture, the client is a thick client as many components needed to realize different functionality. All these components need to be developed and updated regularly, which causes a high administration effort.

3.4 Fat Client Offline Architecture

The fat client offline architecture is quite similar to the rich client hybrid architecture except that no online connection is provided (see Fig. 4). The mobile worker uses the application always offline, the client represents the whole application. The data stored on the client is occasionally synchronized with a central server, e.g. via a stationary network at the mobile workers office. At these points, also the components for business and application logic need to be synchronized. The main advantage of this architecture is, that through the use of a fat client, the full user interface design capabilities are available. As no network connection is needed, the application can be used very flexible in nearly every environmental situation. No costs or performance problems due to the restrictions of mobile networks occur. The main disadvantage of this solution is, that components for the update handling need to be developed and deployed regularly. A synchronization mechanism needs to transfer the data stored at the client, resolving possible conflicts with the central server. The synchronization intervals are completely influenced by the user, the data and component topicality can not be assured.

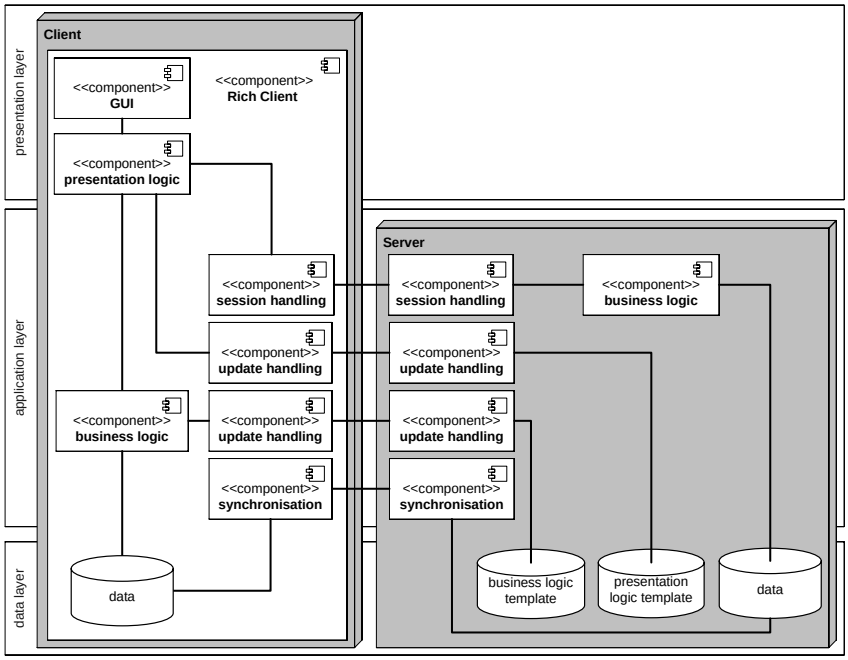

Figure 3. Rich client hybrid architecture

3.5 Architecture Comparison

The web-based always-online architecture is obviously a very small and light-weight architecture, causing relatively small effort for its development and administration. But, it has also some significant shortcomings in connectivity and usability issues. The other three architectures aim on the improvement of these shortcomings. But, as more as they improve connectivity and usability, the effort for development and administration grows rapidly. The main task for IT project managers and software architects is, to decide how much effort for the development and the administration of such a solution is justifiable for the given business needs. The following section shows some of the typical business requirements for mobile solutions and explains how the appropriate system architecture can be deduced.

4 Architectural Implications of Typical Business Requirements

In the following, typical business requirements for mobile applications are described and evaluated regarding their effects on the system architecture. The results are shown in Table 1. If an architecture is thoroughly suitable to fulfil a given business requirement, it is marked with '+', if it is suitable with some restrictions it is marked with '0'. If an architecture is not suitable for a given business requirement, this is indicated by '–'.

- **Point of Service.** The typical location of the POS is of particular relevance for deriving a suitable system architecture. If the application is used mostly in a stationary environment, all kind of architectures are conceivable. If the application is used mobile in urban environments, mobile networks will probably be available most of the time. Thus, the web-based as well as the rich client always-online architecture will be suitable in most of the cases, the other two architectures will work of course in every situation. If the mobile application is used in rural environments, the availability of a mobile network can frequently not be assured. In these cases, only the hybrid rich client architecture as well as the fat client offline architecture is conceivable.

- **Data Topicality.** The requirements for data topicality have also a main influence on the system architecture. If the mobile worker needs always real-time data topicality, only the always-online architectures can be considered. The rich client hybrid architecture can be used, if the requirements for data topicality are not so strict (e.g. data from the previous day are sufficiant). The fat client offline architecture is only conceivable, if the requirements for data topicality are not critical (e.g. a couple of days or weeks).

- **Source Code Redundancy.** The architecture of a mobile system influences also the source code redundancy. If the business logic of an application is used in different contexts (e.g. client and server side), the business logic component often needs to be developed twice because of different system requirements. Each later change in the business logic component also needs to be implemented twice. If the complete avoidance of source code redundancy is demanded

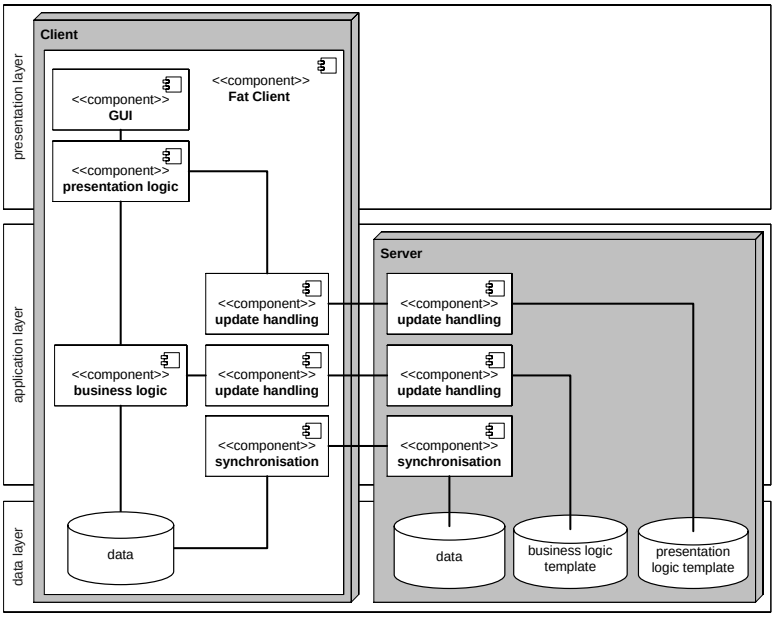

Figure 4. Fat client offline architecture

(single source, single instance), only the web-based always-online architecture is conceivable. If a small source code redundancy is accepted (single source, multiple instances), the always-online rich client architecture should be chosen. If no restrictions regarding the source code redundancy exists, the hybrid rich client architecture and the fat client offline architecture are feasible.

- **Software Distribution.** The distribution of new releases for mobile applications often causes a high effort. To avoid this, the web-based always-online architecture should be chosen, as within this architecture the update process can be conducted for a single server instance of an application. If an online update process is accepted, the always-online and the hybrid rich client architecture can be used. As the update process for fat client offline systems usually requires a high amount of data to be transferred, often only offline updates (e.g. via CD, DVD) are possible, causing very high costs and organizational effort.

- **User Interface Design and Interaction Techniques.** Both the rich client and the fat client architecture offer the full range of elements for designing the user interface. If the application is not to complex and only simple user interactions are needed (feasible with HTML), the web-based always-online architecture can be used.

- **Security Issues.** Within mobile applications often confidential data is processes and transferred. Thus, a couple of security mechanisms are needed. From the administrator's point of view, a centralized security management would be desirable. This would be feasible with both the always-online architectures. The hybrid rich client architecture as well as the fat client offline architecture requires also security effort at the client side.

5 Conclusion

In this paper we presented the four basic software architectures conceivable for mobile systems. The decision, which architecture the most suitable in a given project situation is, depends on the business requirements on the one hand side as well as on the restrictions for the development and administration effort on the other hand side. Both aspects have strong interdependencies. The higher the business requirements are, especially in terms of connectivity and usability, the higher the costs for the development and the administration of the solution are. Considering this, no general recommendation for the system architecture of a mobile system can be given. It is rather necessary to asses single aspects of business requirements like the exact needs for connectivity, redundancy and update preferences and many more. For each single aspect the best system architecture can be derived, but often the result will vary over all considered aspects. Then, a compromise need to be found for the given situation. The above given business requirements and their relation to the developed system architectures can help to support this process.

6 Acknowledgements

The Chair of Applied Telematics/e-Business is endowed by Deutsche Telekom AG. The results presented in this paper

	always-online web-based	always-online rich client	hybrid rich client	offline fat client
point of service				
office	+	+	+	+
urban areas	0	0	+	+
rurally areas	–	–	+	+
data topicality				
real–time data required	+	+	–	–
relatively recent data required	+	+	+	–
other topicality required	+	+	+	+
source code redundancy				
large redundancy accpeted	+	+	+	+
small redundancy accepted	+	+	–	–
no redundancy accepted	+	–	–	–
software distribution				
offline distribution accepted	+	+	+	+
online distribution accepted	+	+	+	0
distribution not accepted	+	–	–	–
user interface design and interaction techniques				
should be extensive	–	+	+	+
can be restricted to HTML	+	+	+	+
security issues				
decentral organisation accepted	+	+	+	+
central organisation demanded	+	+	0	–

Table 1. Business requirements and their architectural implications

were partly developed within a research project in cooperation with the company inverso GmbH [1].

References

[1] http://www.inverso.de.

[2] I. Augustin, A. C. Yamin, J. L. V. Barbosa, and C. F. R. Geyer. Isam, a software architecture for adaptive and distributed mobile applications. In *Proceedings of ISCC '02*, page 333, Washington, DC, USA, 2002. IEEE Computer Society.

[3] R. Belotti, C. Decurtins, M. C. Norrie, B. Signer, and L. Vukelja. Experimental platform for mobile information systems. In *Proceedings of MobiCom '05*, pages 258–269, New York, NY, USA, 2005. ACM Press.

[4] M. Book, V. Gruhn, M. Hülder, and C. Schäfer. A methodology for deriving the architectural implications of different degrees of mobility in information systems. In M. H. Fujita, editor, *New Trends in Software Methodologies, Tools and Techniques*, pages 281–292. IOS Press, 2005.

[5] S. Dustdar and H. Gall. Architectural concerns in distributed and mobile collaborative systems. *Journal on System Architecture*, 49(10-11):457–473, 2003.

[6] V. Gruhn, A. Köhler, and R. Klawes. Modeling and analysis of mobile service processes by example of the housing industry. In W. M. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management*, pages 1–16. Springer LNCS 3649, 2005.

[7] I. Issarny, F. Tartanoglu, J. Liu, and F. Sailhan. Software architecture for mobile distributed computing. In *Proceedings of WICSA '04*, page 201, Washington, DC, USA, 2004. IEEE Computer Society.

[8] Z. Kan, J. Luo, and J. Hu. The design of a software technology architecture for mobile computing. In *Proceedings of HPC '00*, page 762, Washington, DC, USA, 2000. IEEE Computer Society.

[9] F. F.-H. Nah, K. Siau, and H. Sheng. The value of mobile applications: a utility company study. *Communications of the ACM*, 48(2):85–90, 2005.

[10] G.-C. Roman, G. P. Picco, and A. L. Murphy. Software engineering for mobility: a roadmap. In *Proceedings of ICSE '00*, pages 241–258, New York, NY, USA, 2000. ACM Press.

[11] M. Schoeman and E. Cloete. Architectural components for the efficient design of mobile agent systems. In *Proceedings of SAICSIT '03*, pages 48–58. South African Institute for Computer Scientists and Information Technologists, 2003.

[12] T. Tsang. Modelling and performance evaluation of mobile multimedia systems using qos-gspn. *Wireless Networks*, 9(6):575–584, 2003.

[13] D. van Thanh and I. Jorstad. A service-oriented architecture framework for mobile services. In *Proceedings of AICT-SAPIR-ELETE '05*, pages 65–70, Washington, DC, USA, 2005. IEEE Computer Society.