Building applications for mobile devices that will support business processes involving SAP

Master’s Thesis, November 2013
Building applications for mobile devices that will support business processes involving SAP

Submitted in the partial fulfillment of
The requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK INFORMATION ARCHITECTURE

by

IJsbrand van Rijn
Born in Leiden, the Netherlands
Building applications for mobile devices that will support business processes involving SAP

Author: IJsbrand van Rijn
Student ID: 1308262
Email: I.VanRijn@student.TUDelft.NL

Abstract

Keywords: mobile application, development, SAP, process modeling, iOS, design guidelines, OData

To support employees in carrying out their business processes, companies often use Enterprise Resource Planning (ERP) systems to support these processes, and simultaneously control these processes. Meanwhile mobile devices are more and more embedded in people's life and way of working. Building applications for these mobile devices that will support employees to execute their assigned business processes will become an important factor. One of the biggest suppliers of ERP systems is SAP and this thesis focuses on mobile applications that support processes that have SAPs ERP system involved. SAP offers already a range of standard mobile applications that only supports certain activities, but this thesis focuses on applications that support one or multiple processes involved with SAP. To support building these mobile applications, this thesis focuses on the different steps of building these applications: Choosing a software development methodology, modeling the to-be supported processes, unlocking the data and functionality of the backend SAP systems and the implementation of the mobile application. For each of these steps a set of design guidelines is provided to help developers to make the right choices.

Graduation Committee:
Prof. Dr. Ir. G.J. Houben, Faculty EEMCS¹, TU Delft
Dr. Ir. A.J.H. Hidders, Faculty EEMCS, TU Delft
Drs. K.M. van der Put, Senior Manager, Scheer Management BV
Dr.ir. M.F.W.H.A. Janssen, Faculty TPM, TU Delft

¹ Electrical Engineering, Mathematics and Computer Science
Preface

After more than seven years of studying at the University of Technology in Delft, this thesis is the final product of the Master of Science training for the field of Computer Science with a special focus on Information Architecture.

A lot has happened since September 2006 when I entered the education at the TU Delft for the first time. I have made friends, failed some exams but luckily passed them all at some point, and done some great projects. Now it is time to say goodbye to this forming period of my life, and enter the period of working.

There are a few people I want to thank, but foremost I want to thank God for his guidance and support during my entire education. I also want to thank my parents for always supporting me (financially and mentally), and my wife for always believing in me, even though it was not always that easy to see the end.

For educating me I want to thank all the teachers I have had during my education, and some in particular: Dr. Ir. A.J.H. Hidders for guiding me through my graduation as supervisor at the TU Delft, and both Prof. Dr. Ir. G.J. Houben and Dr.ir. M.F.W.H.A. Janssen for being part of the Thesis committee. Besides them I want to thank the TU Delft for educating me and making me who I am now.

Besides the TU Delft I want to thank Software AG and Scheer Management BV for granting me an internship in which I could execute a project to base this thesis on. A few people I want to thank in person: Kim van der Put, for being my supervisor during my internships, Gerrit Achterberg, for supporting me during the project of building a demo application and the jolly welcome every morning and Mirko Scheepers, Gerard van Gent and Marcel Beelen for reviewing (parts of) this thesis. I also want to thank Ferry Boogaards and Diederik van Duuren for allowing me my internship first at Software AG, and later at Scheer Management BV.

What you now rest to read is the remaining part of this thesis, and I hope you will enjoy reading it. And for everyone still learning, I want to quote Albert Einstein:

"The important thing is not to stop questioning. Curiosity has its own reason for existing."

IJsbrand van Rijn
Katwijk, The Netherlands
November 2013
# Contents

Abstract .......................................................................................................................... v
Preface ............................................................................................................................ vii
Contents .......................................................................................................................... ix
List of Figures ................................................................................................................ x
List of Tables .................................................................................................................. xi
Part I: Preliminaries ........................................................................................................ 1
  1 Introduction ............................................................................................................... 3
  2 Using existing methodologies ................................................................................. 11
Part 2: Creating the foundation ..................................................................................... 27
  3 Modeling processes ................................................................................................. 29
  4 Opening up the back-end (SAP) .............................................................................. 39
Part 3: Implementing the application ............................................................................ 55
  5 The mobile application ............................................................................................. 57
Part 4: The result: Guidelines ....................................................................................... 69
  6 Conclusions .............................................................................................................. 71

References ...................................................................................................................... 81
Acronyms ........................................................................................................................ 85
Appendix A: Quotation Process ..................................................................................... 87
Appendix B: Simple EPC ............................................................................................... 89
Appendix C: OData Format ........................................................................................... 91
List of Figures

Figure 1 SAP on Apple ........................................................................................................... 1
Figure 2 Core Business Scheer Management ........................................................................... 3
Figure 3 SAP GUI .................................................................................................................... 5
Figure 4 Action Design Research ............................................................................................ 7
Figure 5 The Waterfall Model (Boehm, 1988 page 64) .......................................................... 12
Figure 6 The Spiral Model (Boehm, 1988 page 64) ................................................................. 13
Figure 7 The 6 ASAP phases .................................................................................................. 15
Figure 8 ASAP Agile add-on .................................................................................................. 16
Figure 9 The followed project approach ................................................................................. 18
Figure 10 Scale of software development methodologies ......................................................... 23
Figure 11 BPMN, Gateway and OData .................................................................................... 27
Figure 12 A Simple UML 2.0 Activity Diagram .................................................................... 30
Figure 13 A Simple EPC Model .............................................................................................. 30
Figure 14 Standard EPC symbols ............................................................................................ 30
Figure 15 Some of the BPMN symbols .................................................................................. 31
Figure 16 ARIS ....................................................................................................................... 34
Figure 17 SAP CRM Best Practice C64 .................................................................................. 35
Figure 18 Create Quotation (Appendix A) ............................................................................. 36
Figure 19 SAP ERP Modules ................................................................................................. 39
Figure 20 The SAP NetWeaver Gateway ............................................................................... 40
Figure 21 The OData building stack ....................................................................................... 43
Figure 22 Connection between systems .................................................................................. 46
Figure 23 iOS Development Tools ......................................................................................... 55
Figure 24 Most important mobile platforms .......................................................................... 58
Figure 25 Smartphone (Galaxy S3) and tablet (the new iPad) ............................................... 59
Figure 26 Four ways of connecting to the backend .................................................................. 60
Figure 27 Guidelines (or Directions) ..................................................................................... 69
List of Tables

Table 1 Comparing methodologies .......................................................... 21
Table 2 Comparison between EPC and BPMN based upon several discussions .... 32
Table 3 Required Gateway components .................................................. 41
Table 4 Optional Components of SAP Gateway ...................................... 41
Table 5 The CRUD operations of the OData protocol ............................... 43
Table 6 Needed OData objects with their attributes .................................. 48
Table 7 Comparison of OData service development options ..................... 50
Table 8 Comparing development platforms .......................................... 62
Part I: Preliminaries

Figure 1 SAP on Apple
1 Introduction

In this chapter an overview is provided that describes the motivation for conducting this research. It furthermore describes the research question and sub questions that are derived from this motivation, and it gives an overview of the way in which the research has been executed. The last section of this chapter provides an overview for the remaining parts of the document.

1.1 Scheer Management

Scheer Management is a fairly new company on the market of business process management, because it was founded in 2010. It was founded by a group of experienced consultants, and its name was taken from one of the co-founders, prof. dr. dr. August-Wilhelm Scheer, the same man that founded IDS Scheer in 1984. The companies’ headquarters is established in Saarbrucken. The vision of Scheer Management is to support companies in their business process management and come up with innovative solutions. One of the services Scheer Management offers to its customers is SAP consultancy. SAP is a large software package provided by SAP AG and it is depicted in the next section.

Figure 2 Core Business Scheer Management

1.2 SAP

Systems Applications and Products in Data Processing (SAP) is one the largest software companies in the world. In the Software Top 100 list of the biggest software companies in the world SAP barely missed the podium and finished fourth with a revenue of more than 12,000 million dollars, and currently over more than 55,000 employees in more than 130 countries.
**History**

5 former IBM engineers founded SAP in 1972. The company released its first ERP software in 1973, the R/1. In 1979, the newer version of this software was launched, R/2. From 1992 until 1995 the company released several versions of the R/3 release of this software\(^2\,^3\). This is still the latest versioning of the ERP system, although it is regularly maintained and updated.

**Products**

SAP offers a broad spectrum of applications under its company name. Its most common product is SAP's Enterprise Resource Planning (ERP) software. According to the report of (Panorama Consulting Solutions 2012) SAP has a 22% market share, which is the highest of all the ERP vendors. It main competitors are Oracle and Microsoft Dynamics, but there are a lot of other ERP vendors out there. Because SAP has a big market share, the focus of this thesis is only on the ERP software of SAP.

ERP software is used to support business processes within a company. Standard ERP software supports all kind of business processes, but besides the standard ERP software SAP also has other software that support business processes executed in certain areas of companies. Examples are Customer Relationship Management (CRM) software and Supplier Relationship Management (SRM) software.

1.3 **Problem Definition**

According to Forrester\(^4\) around 2014 the total number of mobile devices with web access will be greater than the total number of PC with web access. Also currently the focus of applications for mobile devices is shifting from the consumer market to the business market. This is illustrated by the fact that Apple has launched a B2B-appstore, SAP has launched mobile apps and Oracle is marketing mobile applications as well.

Combining these predictions with its own innovative character, skill of process thinking and SAP experience Scheer Management sees an opportunity to improve its service offered to customers. Building mobile applications that not simply show data from SAP software but also offer functionality from SAP software will show customers that Scheer Management:

- Has excellent business knowledge
- Has a significant experience with process innovation
- Has a good understanding of SAP
- Has an innovative character

To interact with the SAP software, SAP offers several options. The most used option is the SAP GUI, the interface that can be installed on a computer to connect with the servers on which the SAP software is running. This GUI provides the user all the possible functionality and data views of the SAP system, but looks outdated and not very intuitive. There are also 2 different Web-based GUIs, the SAP NetWeaver Portal and the SAP NetWeaver Business Client. They resemble the standard GUI, although there are more opportunities to adjust these GUIs, and in the latter case add extra functionality.

![Figure 3 SAP GUI](image)

The differences between using a mobile application and the standard SAP GUIs are:

- **Interface**: Mobile applications are expected to have intuitive interfaces which users can understand in just a couple of seconds. In contrast, the SAP GUIs are beside outdated also overwhelming because they offer almost all the functionality of the SAP Software (depending on the user rights).

- **Connection**: Mobile applications are expected to be fast (although depending on the available data network such as Wi-Fi or HSPDA). Because the standard SAP GUIs run on computers, most of the time there is a good connection, but this is different for mobile devices. Furthermore the SAP GUIs are directly linked with the servers running the SAP software. Mobile applications have to be connected using other connections.

- **Data sources**: Almost all the data that is inputted into the SAP software using the SAP GUIs is put in by users which can lead to errors in this data because it is mostly manually copied from other sources. Mobile applications can use several data sources available on the mobile device (such as user location, camera) to get different information, and put in the system correctly.

- **Functionality**: Due to the physical limitations of a mobile device, such as screen size, memory and processing power, not all of the functions provided by the SAP software can be executed using a mobile application. Some functions are also not available for execution by an external source.
All these aspects distinguish mobile applications for SAP from the standard SAP GUIs. Some of them limit the applicability of mobile applications while other aspects favor the use of mobile applications. To develop such mobile application, a set of design guidelines is desired for all of the steps that have to be taken in order to create such mobile applications. In the next section a main question is formulated and along with it sub questions that define the different steps of building mobile applications that support SAP-involved business processes.

1.4 Research questions

Based upon the problem (section 1.3) this thesis tries to find an answer for, a main topic for this thesis is formulated in this section. Along with this main topic several sub questions or topics are also formulated.

1.4.1 Main Question

“How to build applications for mobile devices that will support business processes involving the ERP systems SAP”

The main question focuses on finding a framework or a set of design guidelines on which applications can be built that will support business processes involving SAP. These applications have to handle a great deal, if not all, of the human-SAP interaction in processes. This will most likely result in a decrease of total-cost-of-ownership (TCO) of a process, or at least will increase the efficiency of a process. Because these statements are not part of the scope of this thesis there is no evidence included that they are correct.

To be able to answer this main question, a number of sub questions have been formulated that cover different topics within the area of building mobile applications supporting SAP-involved processes. Basically each of the sub questions can be viewed a step in the development process of these applications.

1.4.2 Sub Questions

In this paragraph the sub questions will be shortly explained. The order of questions reflects the way in which their topics have been researched.

1. In which way can existing development methods or methodologies be reused or enhanced for mobile application development?

There are methodologies that are widely used for developing software, such as the waterfall methodology and agile methodology. Also SAP offers a methodology/method for implementations with its ERP system. Can these methodologies or methods based upon these methodologies be used when developing mobile applications for business processes? And if they prove insufficient, (how) can they be changed to be sufficient?
2. How to model business processes that will be supported by mobile applications?

The second step will be about modeling business processes that will be supported by mobile applications. Modeling a process can be done in various ways with various programs and techniques, of which each of them will have advantages and drawbacks. Modeling of a business process is needed to identify the steps involving SAP and other steps that can be supported or even executed with a mobile application.

3. What is a suitable way of exposing the SAP environment?

To be able to support SAP-involved processes mobile applications have to have access to all the needed data and functionality of SAP software that is involved in these processes. The SAP software offers several options to expose these functionalities and data, each with its pros and cons.

4. What architecture should be used for building mobile applications?

The last but not least step is what kind of architecture for the application should be used. The architecture of a mobile application is depending on several factors, such as the implementation platform (native or web-based) and the type of device (phone or tablet). Another important factor is the connection with the backend systems that is established by calling the services created in the previous step.

Limitations

Although the end result of this thesis is generalized for a wide area, the scope of this thesis is limited in several ways. Although there are many ERP software packages, this thesis limits its scope to the SAP package. Because of the fact that during the first couple months this thesis was written at the vendor of the ARIS modeling software, only this modeling software was used to create process models. The last limitation concerns the available SAP landscapes, which did not allow trying all the different scenarios to unlock SAP Data and functionality. Further limitations can be found in section 6.5.

1.5 Research method

The way in which the questions formulated in the previous section are answered resembles the way Action Design Research (ADR) (Sein, Henfridsson, Purao, Rossi, & Lindgren, 2011) works.

“ADR reflects the premise that IT artifacts are ensembles shaped by the organizational context during development and use”.

The ADR consists of 4 phases, each with its principles:
The first phase defines the problem. The second phase shapes the IT artifact by building it, intervene it in the organization and evaluate it. The third phase reflects upon the first phases and defines the lesson learned. These phases are iterated until an end result is created. When the end result is created, the last phase of ADR is the formalization of the lesson learned.

This research method was applied by starting a project to develop an iOS application (the main IT Artifact) for the iPad that supports SAP-involved business processes. The goal was not to create a working and marketable app, but to learn from the process of creating such an application. The problems are defined in section 1.3 and 1.4 of this thesis. Accordingly, for each sub question or problem, a solution was built (although it was not always an IT artifact) and a reflection was given. These results are resembled in chapters 2 to 5. The final Formalization of Learning is found in the sixth chapter of this thesis, where the design guidelines from the previous chapters are generalized.

1.6 Deliverables

The deliverable of this thesis is a set of design guidelines for building mobile applications that support SAP-involved business processes, and the total set of these design guidelines can be found at the end of this thesis. These design guidelines are based upon the finding of the literature research and the experience of the project, both described in section 1.5, and the design guidelines for each step/sub question can be found at the end of the chapter that deals with that particular sub question. Using this set of design guidelines while developing mobile applications for SAP business processes should lead to mobile applications that indeed support these processes.

1.7 Structure of the thesis

The structure of this report broadly follows the way of working when building the demo application. The first part of this thesis is looking at existing development methodologies for software and describes whether these can be used. Thereafter the steps that were taken to build the demo application are explained. Each chapter starts with an overview of the available options for that step of the process, and accordingly a comparison between these options is made which is based upon a literature study. The second part of each chapter contains the findings of the project, and the chapters are ended with the design guidelines based upon the literature and project findings. The structure of the document is the following:

Part I: Preliminaries

The first chapter introduces the main goal, and provides an overview on the steps that were taken to reach that main goal. It furthermore provides the sub questions that are answered in order to reach the main goal. It also contains an overview of this thesis.

The second chapter explains existing, widely used software development methodologies and the implementation method that is used to implement SAP software. It answers the question whether these methodologies can be used in creating mobile applications that support business processes.
Part II: Creating the Foundation

Chapter 3 describes the step of modeling processes that will be executed with the support of mobile applications. Several modeling languages are available to model these processes, and the main question is which one of those is recommended for modeling. To model processes several tools are available, and design guidelines are formulated which tool to choose.

In chapter 4 the architecture of the SAP landscape that enables mobile connections is the subject of discussion. Several options are available when installing the components that enable data exposure to the outside world, with all of them having their own strengths and weaknesses. Developing services with these components, that make the data available, can be done in various ways, and is discussed as well.

Part III: Implementing the application

Chapter 5 results in design guidelines for building an architecture for the mobile applications. It describes several aspects of mobile application building, such as choosing a platform and of device on which the applications should run. Some other important aspects such as security issues are discussed briefly, but have not been the main topic of this thesis because they are common aspects for all kind of mobile application development.

Part IV: The result : Guidelines

Chapter 6 contains an overview of the questions and the set of design guidelines that are used to answer them. Based on these guidelines it tries to provide a set of generalized guidelines. It furthermore lists the limitations that were in place during the executing of the project for the demo mobile application. It ends with a short overview of future work possibilities.
Using existing methodologies

Methodologies for the development of software already exist for a long period of time, and have been optimized by the experience gained during all the projects in which they were used. Every existing methodology has its own advantages and disadvantages, and choosing one methodology over the other has to be done for good reasons. Mobile applications are a special kind of software with more restrictions and demands compared to applications for computers. Most of the standard methodologies have shortcomings when it comes to building mobile applications, and almost none of the methodologies deal with building mobile applications that have to support a business process. To find out whether existing methodologies can be used or leveraged this chapter focuses on four existing methodologies and their ability to support mobile application building. The first three that will be explained will be three of the most commonly used methodologies in software development, the waterfall methodology, the spiral methodology and the agile methodology. The fourth methodology (or method) is the ASAP methodology, developed by SAP. This methodology has been the preferred implementation methodology for implementation of SAP related software.

2.1 Common methodologies

Considering that the field of software development does already exist for a long time, many methodologies for implementing software have been presented. Some of these methodologies are being used for a broad scope of software, while other methodologies focus on a specific field or domain. This section will describe three of the most common and widely used methodologies of software development: the Waterfall model, the Spiral model and Agile programming.

2.1.1 Waterfall

The Waterfall model was introduced in the seventies of the previous century. Already being used in the industrial world, it found its way to the software development world as well. Although the term Waterfall was never officially introduced (Weisert 2003), the method seems to lend its name from the way of working. The waterfall methodology exists of the following stages:

1. Requirements Analysis: This step contains the gathering of all the requirements and demands for the software. A problem arises, and a solution has to be found.
2. Design: A solution was found, and the design of the solution is drawn. The solution is given a physical body.
3. Implementation: The design of the solution is transformed into a working prototype. Code is generated, and hardware is put in place.
4. Testing: The working prototype of the solution is tested extensively to make sure there are no flaws in it. Testing can be executed in many ways, from human testing through scripted testing to automated testing.
5. **Integration/Installation**: Once the prototype has passed the test, it can be integrated within the existing environment, or be installed in a new environment.

6. **Maintenance**: After being put to work, the solution has to be maintained in order to ensure it works the way it was supposed to.

Due to the fact that all these stages are executed after the previous one was finished, and therefore create a downward flow, this way of working became known as the Waterfall model. The picture below depicts the visual interpretation:

![Waterfall Model Diagram](image)

**Figure 5 The Waterfall Model**

Several variants of the waterfall exist such as the incremental waterfall model where the several stages are being run through several times. Another variant of the waterfall model is the ‘sashimi’ waterfall model introduced by Peter Degrace⁵, where the stages do have some overlap with each other. Before ending a phase, the next phase of the waterfall methodology is already started. This ensures that errors found in this next phase can be corrected in the previous stage without a large increase of costs. The downside is that milestones are more difficult to specify due to overlap in phases.

### 2.1.2 The Spiral Model

Because the Waterfall model is focused on documentation of the development process, B.W. Boehm introduced the Spiral Model (Boehm, 1988). Instead of focusing on documentation or coding, this model’s primary focus is on risks that can occur during developing software.

The Spiral model works in cycles, and each cycle consists of the following steps:

---

1. **Identification**: Identify the objectives of this cycle, along with the alternatives that can be used to achieve this objective and the constraints.

2. **Evaluation**: Evaluate the alternatives with regard to the objectives and the constraints. If risks occur during this phase, a strategy to resolve these risks should be the next step.

3. **Development**: Depending on the nature of the remaining risks either an evolutionary approach is followed or an iterative waterfall approach. The former approach is followed when performance and/or user interface risks are dominant, while the latter approach is followed when risks regarding program development or internal interface control are dominant.

4. **Review**: At the end of each cycle a review is executed with the primary stakeholders. The executed cycle is reviewed, and plans and resources for the next cycle are determined.

![Figure 6 The Spiral Model (Boehm, 1988 page 64)](image)

The main advantage of the spiral model is its focus on risk control. Instead of determining and documenting everything before the development is started, during each cycle it determines the greatest risk present and tries to find a solution for that risk.

The focus on risk control is also a foundation for the disadvantages of this model. It is difficult to create a good planning using this model, because determining all the risks of a project is not always possible. Furthermore determining the impact of the risks is difficult and requires knowledge and experience.
2.1.3 Agile Programming

Agile Programming is not so much a method as it is a collection of different methods that share the same methodology (Cohen, Lindvall en Costa 2003). Most of these methods were developed because developers were not satisfied with the traditional methods of developing software, which were executed according the Waterfall model (2.1.1.). These traditional methods were seen as inflexible methods that could not deal with changes afterwards in for example user requirements. The agile methods on the other hand are either or both incremental and iterative, which means they can deal with changes made in next phases more easily. Because they iterate over each phase several times, errors from a previous iteration can be corrected in the next one. Changes of requirements can be implemented during each iteration. They all embrace the principles of the Agile Manifesto (Beck, Beedle, et al. 2001), which was written in 2001. One of those principles is that “Business people and developers must work together daily throughout the project.” This is an important principle when developing mobile applications supporting business processes involving SAP because business people execute these processes. Another important principle from the manifesto sounds: “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly”. Both these principles are used in the following two methodologies which both are agile.

**Extreme programming**

One of the more known agile methods is eXtreme Programming, XP (Beck, 1999). This method is focused on teamwork. Where the waterfall methodology starts with an exhaustive amount of documentation, the XP method allows developers to start almost immediately with the implementation part. Each time small parts of the project are implemented, while the developers follow the major practices of the XP method. One of these practices is pair programming, which signifies the way of working of XP. This is done by pairing two developers and put them behind one computer to implement their part. According to the method developing this way will reduce the number of bugs in the code. Implementing small parts at a time, the project can be easily adopted to changing user requirements or other change factors. The XP method emphasizes communication with the customer and teamwork, to make sure all the needs of the customer are fulfilled.

**Scrum**

Another well-known and widely applied agile methodology is the SCRUM methodology (Beedle en Schwaber 2009). This methodology divides the developers in teams that execute sprints, short periods of weeks in which a part of the project is been implemented. Such a sprint can last from 1 week to a month, and has a definitive end goal (where the team is ‘sprinting’ towards). During this sprint no changes can be made in the list of design guidelines until the sprint has been finished and the next sprint will be set up. During a sprint at the beginning of each day a team meets and configures the result of the last day, create a list of goals for that day and tries to solve any problems that have come up. At the end of the sprint the result is demonstrated.
2.2 SAP Methodologies

The purpose of this chapter is to find out whether existing software methodologies can be used or related to when building mobile applications for SAP-involved business processes. The first section listed a few standard, widely used methodologies. This section describes the method and a variant of that method that is created by SAP.

2.2.1 ASAP

Based upon the experience of thousands of SAP implementations, SAP provides customers with an implementation method of its own called AcceleratedSAP, ASAP. This method consists of 6 phases, which resemble the steps from the waterfall methodology. These phases are: Project Preparation, Blueprint, Realization, Final Preparation, Go Live Support, and Run (See figure 4). Each phase is divided into several areas or topics that have to be dealt with during that phase. The ASAP method furthermore provides templates for deliverables and milestones that have to be created or reached during a phase. Besides an implementation method the ASAP method at the same time is a project management method providing the different roles for each phase, and the work areas these roles have to focus on.

![Figure 7 The 6 ASAP phases](http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/com.sap.km.cm.docs/lw/asap%20methodology/asap%20methodology%20for%20implementation/99BFE0F648A74B589CBA0C056E892833/ASAP%20Roadmap%20-%20RoadMap%20Template_files/image008.jpg)

The first phase is the Project Preparation. During this phase, all the necessary preparations have to be taken, and an initial planning should be created.

The next phase is the Business Blueprint. The results of the previous phase are processed, and a detailed design is created. This design shows how the result of the project should be used within the company.

Phase three, Realization, uses the business blueprint to implement the solution. At the end of this phase a solution is delivered, along with testing plans and training materials.

Continuing with phase four, Final Preparation, the ASAP methodology directs the project into preparing the solution to be used within the company. At the end of the phase the solution is well tested and users are trained and ready to use the solution.

The Go Live Support is the fifth phase in which the solution is activated within the company. The solution is monitored and issues that occur are gathered and solved. At the end of the phase the solution is working properly within the company and the lessons learned are documented.
The last phase is the Run phase. This is not so much a phase as it is a status of the project. During this phase the solution is monitored, and improvements are made to optimize the solution. Until the solution is no longer sufficient or needed, the solution will have this status.

The ASAP method has the structure of a waterfall methodology (Ghandi, 2012). Each of the phases can be linked to one of the phases in the waterfall model. Besides the standard ASAP method SAP offers also add-ons which enhances the ASAP method with specific objects and steps for certain projects. One that could be used for implementing mobile applications is the mobile add-on. Another add-on is the Agile add-on, which will be discussed in the next paragraph.

2.2.2 ASAP Agile Add-on

Besides the standard ASAP method SAP offers customers also the opportunity to use ‘business add-ons’ for ASAP. These add-ons offer specific functionality when implementing SAP projects for specific business areas or other fields. In early 2011 SAP released the ASAP Agile Add-on, which extends the ASAP method with an agile implementation approach.

![Overview of Agile Business Add-on to ASAP](http://scn.sap.com/servlet/JiveServlet/showImage/38-57712-72049/AgileBusinessAddOn.JPG)

**Figure 8 ASAP Agile add-on**

Using this ASAP agile Add-on the Preparation and Blueprint phase have been expanded with several agile techniques such as determining a project scope and using a prioritized project backlog.

---

The biggest difference with the traditional ASAP method is the execution of the realization phase. The traditional ASAP method delivered the whole of the software and the end of the realization phase. Using the Agile add-on, functionalities are released during the realization phase. This enables the developers to prototype the software during the realization. This requires a close cooperation between the developers and the business people that are going to use the software. The Final Preparation, Go Live Support and Run phase are executed according to the standard ASAP method. Using this Add-on is not undisputed. Some people find this add-on unreliable because agile methods only support small projects.

---

2.3 Methodology or approach of demo project

The previous section described the two mainstream development methodologies in software development and highlights the development/implementation method that is offered by SAP. This section describes the project and the approach that was taken for creating the demo application that was built along with this thesis. It furthermore describes whether there is any resemblance between the approach that was followed and the methodologies and method described in the previous sections.

2.3.1 The Project

The goal of this thesis is to create a list of design guidelines for developing mobile applications that support SAP-involved business processes. To come up with design guidelines a project was started with the goal of creating a mobile demo application that supported at least one business process involving SAP, but preferably more. A standard business process involving SAP was selected which had to be supported by this application. The main goal of this project was to experience the difficulties of modeling the process from the standard as-is to the to-be process in which it is supported by the mobile application, and experiencing the choices to make when exposing the SAP systems and consuming the services offered by the SAP systems. The demo application was created using several steps (phases). These phases are represented in the different chapters of this thesis. Figure 5 depicts the order of these phases, but the third and fourth phase were mostly executed at the same time. The remaining text of the paragraph describes these phases.

Modeling the Process (chapter 3)

The first phase of the process was selecting a standard business process in SAP that will be supported by the application. This standard process was analyzed to find steps that could be improved by using mobile technology. The process that was selected was the ‘Sales quotation process’, in which a quotation for an existing customer is created. Future options are to implement the total SAP Best Practice (BP) scenario of which this
process is part of\(^9\). A SAP BP is like a template for implementing business processes in SAP and is based upon the experience of thousands of SAP implementations. Most companies start using SAP applications by implementing these BPs and create their own BPs from there on out.

This phase started with deciding which process should be used in the project. The next step was choosing the right modeling technique and tool. Final step of this phase was the modeling of the as-is and the to-be process with the selected tool and technique. The to-be model represented the process being executed while using mobile technology.

**Configuring the back-end (chapter 4)**

After modeling the to-be process, the second phase is to make the needed data and functionality from the SAP systems available to the application. The first step in this phase is to configure the SAP systems to reach this goal. Configuring the SAP systems means to install the needed modules and configure all the right software settings. There are several configurations that can be used to open the backend to the application, each with its perks and setbacks. During the project executed for this thesis 2 different configurations were used. Because setting these configurations has a big impact on a SAP landscape, more configurations could not be tested.

After the data and functionality has been made available, the next step is to create a data model of all the needed objects and their attributes. This data model is implemented in the SAP systems, which makes the data available for the application. The data model was for the bigger part derived from the concept of the application, which was realized by creating mockups of the different screens of the application. In the project this data model was expanded a couple of times because more objects or attributes were needed. Depending on the configuration of the backend systems realizing the data model was easy or difficult.

**Creating the application (chapter 5)**

Having modeled the process and prepared the backend for the data exposure, the final phase of the project was to create the mobile application that could retrieve the data and add new data or enrich the existing data. After having selected a platform and device for which a demo application should be created, an iterative way of working was followed in which little pieces of functionality were added to the application. During this iteration sometimes missing data or functionality was discovered that was needed to execute the process that was modeled, and the previous phase was executed again to ensure all the necessary data was available. A detailed description of this phase can be found in chapter 5.

---

2.3.2 Similarity and conclusion

To see whether one or more of the standard methodologies (including ASAP) fulfill the demands for developing mobile applications for SAP-involved business processes each of the described methodologies is compared with the approached followed in building the demo application.

Waterfall versus Own Approach

The first methodology described in this chapter was the waterfall methodology. This methodology uses a phase-after-phase approach, with each phase only executed after its predecessor was finished. Compared to the approach followed during the development the waterfall model does not seem a likely candidate, because several phases were iterated over due to findings in other phases. There however might be mobile application development projects in which such a methodology can be useful, for example when security has the highest priority, and a flawless design is needed to achieve total security. In that case changes in the blueprint cannot be applied afterwards as it might endanger the security of the application.

Spiral versus Own Approach

The primary focus of the Spiral methodology is set on Risk control. Although several risks were presented in the development of the demo application, risk control was never a significant part of the development process. This was mainly caused due to the fact that the goal of the development of the demo application was to get experience in the total process, and doing good risk control requires experience that was not hold by the author of this thesis (this requirement is also noted as one of the disadvantages of this methodology).

Agile versus Own Approach

The third methodology was the agile approach. Methods that embrace the principles of this methodology (such as SCRUM or XP) are more likely candidates to be used in mobile application development projects. As seen in section 2.3.1 most of the phases executed during the development of the demo application were iterated over several times, and changes to the design were made throughout the project. The approach that was used in the project of the demo application has similarities with the SCRUM method with gatherings between developers and clients on a regular basis in which the finished parts were discussed and new milestones were set (just like using sprints).

ASAP versus Own Approach

Comparing the followed approach for the project with the ASAP methodology (with the agile add-on) there seems to be similarity between some of the phases. The phases of the project can be linked to some of the phases of the ASAP methodology:

1. The first phase of the approach, which was modeling the process, can be compared to the blueprint phase of this approach, with the difference that the ASAP approach requires a blueprint as deliverable for this phase, and in the
followed approach only the process that should be supported by the solution (demo application) was delivered.

2. The second and third phases of the followed approach can be linked to the realization phase of the ASAP agile approach. Like the ASAP agile approach these phases were iterated over several times. In chapters 3 and 4 it shows that this was done in a way that looks a little bit like the agile SCRUM approach, in which developers and users meet on a regular basis.

The ASAP agile methodology does contain more steps that were not executed during this project. Although preparations were taken to start this project, most of the preparation was done at the moment that knowledge or skills were needed. This was the result of the fact that for every phase of the project knowledge was needed, and it was decided to gain the necessary knowledge or skills when needed. The benefit of iterating over the second and third phase is that gained knowledge is used over and over again, and improving.

Conclusion

To state a good conclusion, the following table compares the 4 described methodologies with each other and the approach used to develop the demo application.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Waterfall</th>
<th>Spiral</th>
<th>Agile</th>
<th>ASAP</th>
<th>Own Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexibility</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Documentation</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Sizeable</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Ease-of-Use</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Structured/Evolutionary</td>
<td>Structured</td>
<td>Hybrid</td>
<td>Evolutionary</td>
<td>Structured</td>
<td>Evolutionary</td>
</tr>
<tr>
<td>Stakeholder participation</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Prototyping</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 1 Comparing methodologies
The following list explains the different aspects the methodologies are compared on:

- **Flexibility:** The ability to change the requirements during a project. As described earlier, the basic waterfall methodology does not allow changes in requirement. Due to iteration or evolutionary design the other methodologies allow changes in requirements.

- **Documentation:** Amount of created documents during a project. The Waterfall methodology requires documentation before implementing. The Spiral method requires a review of each cycle. The Agile methodology was founded to limit the amount of documentation. The ASAP method provides a lot of templates that can be used in a project.

- **Sizeable:** The waterfall model can be used in small and big projects. The spiral model is not fit for smaller projects (Munasar & Govardhan, 2010). Agile methods are considered not fit for big projects (Boehm, 2002), although there are attempts made to create hybrid agile-plan based approaches to use Agile development (Barlow, Giboney, Keith, Wilson, & Schuetzler, 2011). ASAP can be used on small and big projects.

- **Ease-of-Use:** The waterfall model is easy to use. The Spiral model requires experience with risk management. Agile methods can be easily used as well, depending on the method. To use the ASAP methodology some experience is required, but sufficient documentation can be found.

- **Structured/Evolutionary:** The Waterfall methodology uses a structured approach because of its step-based working. The Spiral methodology uses a hybrid approach, because it runs in structured cycles, but the cycles itself can contain an evolutionary approach. The agile methodology is based upon an evolutionary approach. The ASAP method is a step-based, and therefore structured methodology.

- **Stakeholder participation:** Using the waterfall methodology, stakeholders are only involved in the beginning (requirements gathering) and the end (testing and releasing). The Spiral methodology requires a stakeholder review at the end of each cycle. One of the principles of the agile methodology is a high stakeholder involvement. ASAP provides roles for each step, and these roles include stakeholders.

- **Prototyping:** Amount of prototypes and beta products offered to customers and end-users. The waterfall methodology only offers the complete product at the end of the project, as is the ASAP methodology (unless using the Agile Add-on). The spiral methodology involves customers at the end of each cycle to evaluate the project so far (thus including prototype testing). One of the principles of the Agile methodology states: “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale”.

The following scale shows the different methods in relation with each other:
Based upon these comparisons the approach used tends to be an agile approach and therefore less similar with the Waterfall methodology. Although the followed approach differs from the standard ASAP method, applying the ASAP Agile Add-on will enlarge the similarity between the followed approach and the ASAP method. The ASAP method was not used in the project because of experience and time limitations. Using the ASAP method to develop software (mobile applications are software) for SAP systems is recommended because it supports the participants to document the project in such a way that other professionals can easily understand the progress of the project.
2.4 Design guidelines for the project approach

The previous sections of this chapter describe two of the most common software implementation methodologies and the methodology that is offered by SAP. Based upon the findings of these sections this section will list design guidelines that are useful when implementing a mobile application for SAP processes.

Design guideline 2.1: When not restricted to waterfall like methodologies, agile approaches should be considered over traditional approaches, mainly because agile approaches can deal with changes a lot faster, which will occur a lot especially when developing for the first time.

Based upon the findings in section 2.3 it seems that agile methods are more capable of dealing with changes in design and demands than traditional waterfall methodologies. Some cases will require waterfall like approaches (for example applications that will deal with very sensitive information which have to be extremely secure), but when possible it is recommended to develop in an agile manner, especially when there is little to no knowledge of the environment (SAP’s ABAP, Android, iOS etc.) in which the development will take place. Side note by this design guideline is the fact that mobile solutions are not yet mature, and a lot of research is still done on how to implement mobile applications in an effective way.

Design guideline 2.2: Try using the ASAP methodology (with the agile add-on or perhaps the mobile add-on) when developing mobile applications for processes (partly) supported by SAP.

According to experience gained in the project approach used for the demo application (see section 2.3), an application can be built following the steps of the (agile) ASAP methodology. This correlates with design guideline 2.1 due to the agile part of the ASAP methodology. Another benefit will be that document templates are provided which comply to standard SAP document styles. Given that this methodology is based upon previous implementation of SAP modules and functionalities it will help other developers to understand the followed approach.

The ASAP methodology might also be used in other, non-SAP involved mobile application development, but because this was not in the scope of this thesis, the second design guideline is narrowed down for only SAP-involved implementation projects.
Part 2: Creating the foundation

Figure 11 BPMN, Gateway and OData
3 Modeling processes

To develop mobile applications that support SAP-involving business processes, the first step or phase in the development process is requirements gathering. One aspect of this requirements gathering is determining how these processes are currently executed (as-is process), and sketching the execution of these processes with the help of a mobile application (to-be process). To capture these processes process modeling is applied. To model a process a modeling language is used to formalize how the process model is modeled, and a tool is used to create a model of the process with the language chosen.

The first section of this chapter describes several modeling language or techniques. The second section describes a tool that can be used. The third section of the chapter describes modeling the processes in the demo application project, and the experience obtained during that phase of the project. The fourth section lists design guidelines based upon the literature and experience obtained during the project.

3.1 Modeling languages

The first step in modeling processes is to determine which language will be used to model the process. There are several languages available, varying from the standard Unified Modeling Language (UML), to more advanced and specific languages such as Event-driven Process Chain (EPC), which is more of a technique and part of the modeling language used in the ARIS tool (see 3.2), and Business Process Model and Notation (BPMN). Each of these languages has its advantages and disadvantages. This chapter will discuss the UML technique, the EPC technique and the BPMN technique. The end of the section will describe the selected technique.

3.1.1 UML

The UML10 is one of the most common modeling language in the world of computer science, and especially in the area of object oriented software. Being officially published for the first time in 1997 by the Object Management Group (OMG), the UML has grown to version 2.x. The UML is a big package containing all kinds of diagrams ranging from Use Case diagrams to Communication diagrams and Class diagrams. For modelling business processes activity diagram are commonly used.

Activity diagram

An UML Activity diagram (AD) stepwise depicts all the actions and activities in a process. ADs always start with a start node (the initial node) and end with an end node (the activity final node). Connections between activities are represented by edges. Activity diagrams can also contain forks and joins, which represent a set of parallel activities. Forks mark the start of those activities, and joins mark the waiting point for all these activities to be finished. There are also decisions and merges. Decisions

10 http://www.uml.org
represent multiple options after an activity, with conditions for each option. Merges have multiple incoming edges, and the process can continue after at least one of these edges is triggered. Furthermore activity diagrams can contain partitions, which are similar to swimlanes and represent who is responsible for which activities. A simple Activity Diagram (without partitions or forks/joins) is depicted in figure 12.

![Figure 12 A Simple UML 2.0 Activity Diagram](image)

### 3.1.2 EPC

Event-Driven Process Chains are the main models used for modeling processes in the modelling tool ARIS (see 3.2.1). These EPCs are used to represent the dependencies between activities in a business process (Mendling 2008), and their main purpose is to describe the business logic in these processes. When designing information systems it represents the integration of the different perspectives involved in designing information systems, such as functional and data perspective. The following figure shows the most common symbols used in EPC models:

![Figure 14 Standard EPC symbols](image)

The purple hexagon represents an Event, which is used to describe the pre- and post-conditions for a function. The green rectangle represents a function, which represents an activity in the process. Formally every function is followed by an Event, and vice versa. Only exceptions are the start

![Figure 13 A simple EPC model](image)
and end event. The overlapping rectangle on the rectangle represents a Process Interface. This symbol can be found at the start or end of a process, and represent a process that is executed before or after the current process.

The 3 circles represent the following:

- The round with the X represents the XOR connector, which represents that the process has to make one choice from multiple options
- The round with the arrow pointing up represents the AND connector, which represents that the process is going to execute multiple branches
- The round with the arrow pointing down represents the OR connector, which represents that the process is going to execute one or multiple branches connected to it

3.1.3 BPMN

The Business Process Modeling Notation (BPMN) technique is another standard managed by the OMG. The BPMN technique can be used to describe business processes in a graphical notation. (Chinosi and Trombetta 2012). It is a notation that can be understood by business users that have to understand the process and also by technical users that need to implement the functionalities need to support this process. It has some similarity with the UML Activity diagrams discussed in 3.1.1. The current version of the BPMN technique is 2.0. In BPMN 2.0 the following categories of models can be modeled (OMG, 2011):

- Private Processes
- Public processes
- Choreographies

When to use which category to model a certain process is not specified and left in the hands of tool vendors that offer BPMN 2.0 functionality with their tools.

Like the UML activity diagrams and the EPC diagrams the BPMN diagrams represent a workflow with activities, events and edges to link these activities and events. But the BPMN allows for a more detailed description of these events and activities by offering different symbols for these events and activities.

3.1.4 Choosing a technique

The purpose of this thesis is to provide design guidelines when building mobile applications that support processes that are (partly) supported by SAP. In order to achieve this the current processes have to be changed to cope with the mobile applications. This results in a different way of working for the business people executing
these processes, and they need to be involved in changing the current processes. A modeling technique is needed that is understood by both these business people executing the process as well as the technical people that have to implemented the mobile applications and make sure the applications offer the needed functionality. Simply put, there are several stakeholders that are involved in modeling the process, and according to (Chuang, 2010) BPMN covers the most stakeholders, while EPCs is a good runner-up. The UML AD are very technical models and therefore less suitable for this purpose, and therefore not taken further into account.

Besides covering possible stakeholders view, selecting a modeling technique can also be done on several other criteria. Several discussions (Stein, 2010) (Velitchkov, 2011) on choosing between EPC and BPMN resulted in the following comparison table:

<table>
<thead>
<tr>
<th>Criterion</th>
<th>EPC</th>
<th>BPMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Control flow expressiveness</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>2. Linking other dimensions</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>3. Readability</td>
<td>+</td>
<td>+/-</td>
</tr>
<tr>
<td>4. Resource assignment</td>
<td>+</td>
<td>+/-</td>
</tr>
<tr>
<td>5. Enterprise Architecture</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>6. Semi-structured processes</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>7. Exceptions and transactions</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>8. Loops</td>
<td>+/-</td>
<td>+</td>
</tr>
<tr>
<td>9. Managing data</td>
<td>+</td>
<td>+/-</td>
</tr>
<tr>
<td>10. Process Notation for ERP</td>
<td>+</td>
<td>+/-</td>
</tr>
<tr>
<td>11. Tool support</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 2 Comparison between EPC and BPMN based upon several discussions

Table 1 provides a comparison on several criteria between EPC and BPMN. For the goal of this thesis not all these criteria are equally relevant. The criteria that are important for the cause of this thesis are the following:

- **Readability**: As mentioned earlier in this section, the process model has to be understandable to different kind of stakeholders. According to discussion, EPC is slightly better to read, but there is not much difference as (Recker & Dreiling, 2008) shows that people understanding EPCs do not have difficulty understanding BPMN as well.
- **Managing data**: One possibility to support processes with mobile applications is by adding data to the process derived from resources available on the mobile
device (such as location). Modeling data needed in the process accurately will help identify possible resources. EPC seems slightly better for this criterion.

To conclude this section it seems like choosing a modeling technique is depending on the purpose of the model. To create a model that can be understood and shared by business people as well as the technical people (developers) BPMN seems to be the technique to go with. When the purpose is to create a model that is more expressive and executable, EPC models are preferable.
3.2 Modeling tools

The next step in modeling the process is selecting the tool that will be used to model the process. The choice for the tool is depending on the choice of technique, which is made in the first step. There are several tools available for modeling, ranging from free tools\textsuperscript{11} such as ARIS Express, Bonita, BizAgi and Tibco Business Studio to commercial modeling suites such as ARIS and Bizagi. According to (Chinosi and Trombetta 2012) BizAgi is the most preferred tool to model BPMN models, while Bonita was the most preferred free (open-source) tool do so. There was no extensive look into the several tools due to the fact that this thesis was partly written during an internship with the vendor of the ARIS tool.

3.2.1 ARIS

The ARIS suite is developed by IDS Scheer and is based upon the Y-model developed by IDS Scheer founder Dr. August-Wilhelm Scheer. ARIS stands for: ARchitecture of integrated Information Systems. The different parts of the suite can be divided into 4 major categories: Process Strategy, Process Design, Process Implementation and Process Controlling. The part of the suite used in this thesis is housed in the Process Design area. The tools that will be used in this thesis to model the process will be the ARIS Designer, which allows users to model processes in all kind of different sorts of diagrams, and ARIS Architect, which allows users to not only model processes but also maintain process modeling projects and repositories, and set the options for a project. The latest also includes the Designer.

The ARIS tool has its own language for modeling business processes, with EPCs serving as the main models. Other models are Value-Added Chain Diagrams and Function Allocation Diagrams.

ARIS also supports other modeling languages, such as BPMN or UML. It also provides semantic checks that can be used to check if the created model applies to the rules of the chosen protocol.

When merely using ARIS to model some processes, the free variant ARIS express can also be used. It allows users to model in many different ways, but it lacks some features that are available in the ARIS Designer/Architect, such as storing all the created objects and models and the ability to reuse these objects and models.

\textsuperscript{11}\url{http://bpmfundamentals.wordpress.com/2009/07/10/free-bpm-modeling-tools/}
3.3 Process modeling during demo application

In section 3.1 several techniques were discussed that could be used when modeling a business process. In section 3.2 the ARIS tool was described which will be used to model the process for the demo application. This section depicts the model that was made for a standard business process involving SAP, and furthermore it lists the steps in the process that can be supported by a mobile application.

3.3.1 The process

For the demo application, a standard and widely used SAP process was selected that should be supported using a mobile application. SAP offers Best Practices that include standard SAP processes that can be executed. One of the SAP Best Practices for Customer Relationship Management is the Best Practice for Order and Quotation management, labeled C64\(^12\).

Figure 17 SAP CRM Best Practice C64

This best practice can start with an opportunity, which is according to the SAP wiki “…a verified possibility to sell a product or service to a specific customer as well as the process of pursuing that sale”. Based on this opportunity or for example a visit to the customer, a sales employee creates a quotation, which has to be accepted by the customer. The customer creates a purchase order in his or her ERP system and sends this purchase order to the sales employee. The sales employee creates a sales order in SAP based upon the purchase order of the customer.

3.3.2 Model mobile support opportunities

In some southern European countries sales representatives execute the process by visiting customers on location and handwrite a quotation that they have to put into the SAP system at the moment they are back at the office. Offering a mobile application that will allow them to create the quotation at the location of the customer will support the process and increase the throughput time of the process. To model the process for a scenario with mobile support the first step is to identify the subprocesses that could be executed with the help of a mobile application:

1. Create quotation: Instead of creating a handwritten quotation first, the mobile application offers the functionality to create a quotation and send it to the SAP ERP system. The system checks whether the quotation is correct and if further

\(^{12}\) http://help.sap.com/saap/sap_bp/BBLibrary/documentation/C64_CRM702_BPD_EN_XX.doc
approval is needed. This result is communicated back to the sales employee using the application. The application offers support creating the quotation by offering a catalog with products and their price indication.

2. Create Sales Order: Based upon a quotation or just a new sales order, the application offers the functionality to create a sales order and send it to the SAP ERP system. The application supports this step by offering a catalog with products and indicates the availability and possible delivery date of each product.

Because the benefits of executing this project using mobile devices has to be explained to business people it was chosen to model the to-be process using the BPMN technology and the ARIS tool. This picture depicts the result of modeling the Create Quotation process:

![Figure 18 Create Quotation (Appendix A)](image)

To obtain this to-be model of the Create Quotation process, the as-is process was modeled, and in this process the steps were listed that required human-machine interactions. After the list was created, each step was checked to see if mobile support could be implemented for this step, and in figure 18 three steps are modeled as executed by a mobile device. Two of these steps send a quotation to the backend system, and one step is processing the result of the validation check on this quotation. An identical model can be created for creating sales orders.
3.4 Design guidelines based upon the project

In the previous sections of this chapter some of the most common modeling techniques were discussed as well as ARIS, a tool to create this models with. This section provides design guidelines for modeling a process that is supported by the use of mobile applications, and these design guidelines are based upon findings in the literature and the project to build a demo application.

*Design guideline 3.1: Use BPMN to model the processes.*

According to the literature and the discussions between the experts there is not that much of a difference between using EPC or BPMN. UML is a more technical technique used mainly in the software development world, and not suitable for explaining process models to business people. BPMN does cover more stakeholder views than EPC because EPC is a modeling technique focusing on business people while BPMN was developed to bridge the gap between the technical people and the business people. The project for the demo application in which both the techniques were tried proved that BPMN got a better response with both areas and was equally easy to understand for people that did not have knowledge of BPMN yet.

*Design guideline 3.2: Start from the as-is process to model to a to-be process with mobile support.*

Instead of just modeling a new process the first step to take when modeling is to model the current process. That way mobile support opportunities will become visible. Using these points to create a to-be model of the process ensures that it will become clear what the actual support of the mobile application is and what the changes are in the execution of the process.

*Design guideline 3.3: Identify data input in the process as accurate as possible to enable external source support in the mobile application*

One of the advantages of an application on a mobile device is the possibility to contact external sources to get (additional) information. In the creation of a quotation or a sales order the application extracts its data from the SAP systems in the backend, but when creating a new prospect in the system the application can extract additional information from free sources such as Google Places or paid services such as Dun and Bradstreet. For creating contact persons mobile applications can extract information from sites like LinkedIn.
4 Opening up the back-end (SAP)

The goal of this thesis is to find a set of design guidelines or framework that supports building mobile applications for business processes involving SAP. The previous chapter discusses the modeling of the process and identifies the steps or points in this process that require interaction with backend SAP systems. This chapter discusses the different steps that have to be taken in enabling this interaction. It starts with an overview of the different SAP applications that can be part of the backend landscape. The next section depicts the ways in which the application can be connected to these SAP applications (together also called the SAP Business Suite). The third section describes the configuration of the SAP Landscape for the demo application project. The last section lists design guidelines for configuring the backend for mobile applications.

4.1 SAP

This section provides a short description of the two most used applications of the SAP Business Suite. A more detailed description is given for the SAP NetWeaver Gateway application, the application that allows external sources to connect to the SAP applications.

4.1.1 Applications

SAP offers several applications for supporting business processes in a company. Besides offering these applications separately, SAP offers the Business Suite, which contains several applications. This term is used in the remains of this thesis to denote a collection of SAP applications. Each application is installed on a separate (virtual server). A server can have multiple instances of the application, which are called clients. The data from each client is separated from the other clients, but (changes to) the functionality of the application is spread out through all the clients.

The following list describes the two most common applications:

- Enterprise Resource Planning (ERP)
  The ERP application is the heart of the SAP Business Suite. This application is based upon the R/3 core. The ERP application has the R/3 application as its core, and several modules can be installed upon this core, such as FiCo(Finance and Controlling) and PP(Production Planning). Using these modules most if not all processes in a company can be supported. The other applications in this list focus on a set of processes. Although these applications can be installed and run independently from the ERP application, common practice is to install the ERP application along with these applications.
• Customer Relationship Management (CRM)
The CRM application focuses on one of the important aspects for a company, its customers. With this application contact with customer can be maintained, and exhaustive information about a customer can be stored. Processes supported are creating leads for new customers to maintaining contact persons and marketing information of a business partner.

• Other Applications
Besides the CRM and ERP application SAP also offers applications for Supply Chain Management (SCM), Supplier Relationship Management (SRM) and Product Lifecycle Management (PLM), which all offer extra functionality for certain areas of business processes.

To connect these applications, SAP developed an integration platform called NetWeaver. Besides integrating all the different SAP applications this platform also offers development functionality and integration with other applications (such as Oracle) and technologies (such as .NET and Java EE). One of the modules for this integration platform is the NetWeaver Gateway. This is the module that enables data and functionality exposure to the outside world. This module is described in the next paragraph.

4.1.2 SAP NetWeaver Gateway
To understand the SAP NetWeaver Gateway it is necessary to have some background knowledge of the implementation of SAP applications. Since the release of the R/2 system SAP has developed its own programming language, the Advanced Business Application Programming (ABAP), which looks a bit like COBOL and/or Open SQL. Most of the applications of the SAP Business Suite are programmed in this language, and some in Java. These modules are placed on top of a database layer. ABAP add-ons are pieces of ABAP code that add functionality to SAP applications.

The SAP NetWeaver Gateway module (SAP AG 2011) is a collection of ABAP add-ons that allows users to access the data and functionality of the SAP systems from the outside world. Basically it provides an API that can be called from any platform or device to provide data. The only requirement is that the device or platform is capable of dealing with HTTP(s) calls (REST protocol) and the OData protocol (see paragraph 4.2.2).

Figure 20 The SAP NetWeaver Gateway
4.2 Connecting with SAP

In the previous section it was made clear that the module to interact with the SAP Business Suite was the SAP NetWeaver Gateway. This section depicts options for the installation of this module and the options to implement the services for data exposure. The first paragraph gives an overview of the options to install the Gateway on the SAP Business Suite. In the second paragraph the options for developing the services are examined. The third paragraph explains the OData protocol that is used to expose the data and functionality to the outside world through the services. The fourth paragraph describes the Sybase Unwired Platform, another (similar) way to expose the SAP data to the outside world.

4.2.1 Installation of the Gateway

The NetWeaver Gateway module is a set of ABAP add-ons (components) used to expose the SAP data and functionality to the outside world. The following tables contain the required components of the Gateway and the optional components:

**REQUIRED ABAP Add-ons**

<table>
<thead>
<tr>
<th>Component</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>IW_FND</td>
<td>Framework of Gateway hub</td>
</tr>
<tr>
<td>GW_CORE</td>
<td>OData Libraries</td>
</tr>
<tr>
<td>IW_BEP</td>
<td>Business Enablement Provisioning component. (Local option)</td>
</tr>
</tbody>
</table>

Table 3 Required Gateway components

**OPTIONAL ABAP Add-ons**

<table>
<thead>
<tr>
<th>Component</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>IW_CNT</td>
<td>Metadata description, data mapping and more</td>
</tr>
<tr>
<td>IW_CBS</td>
<td>Basis functionality, similar to IW_CNT</td>
</tr>
<tr>
<td>IW_SCS</td>
<td>Screen Scraping (only for backend systems)</td>
</tr>
<tr>
<td>IW_BEP</td>
<td>Business Enablement Provisioning component. (Remote option)</td>
</tr>
</tbody>
</table>

Table 4 Optional Components of SAP Gateway

To install the Gateway module there are the following 2 options:

1. Local: The Gateway module is installed on each server/application. All the required components need to be installed on each server. If Screen Scraping will be used the IW_SCS module has to be installed on each server/application as well.
2. **Remote**: The Gateway module is installed on a dedicated server, serving as a hub for all the different applications in the backend. All the required components are installed on a single server, and connections are made between the Gateway server and the other servers/applications. The IW_SCS module is again optional, and the IW_BEP module can be installed on the backend systems if development is needed on the backend servers (i.e. services that are executed on the backend).

The first option results in a lower total-cost-of-ownership because there is no need for an extra server. Because the Gateway is installed on top of the application the runtime will be faster than the second option in which remote function calls (RFCs) have to be made to get the data.

The second option has the advantage that installation of the Gateway in a (existing) SAP Business Suite can be executed without touching the backend. Updating the Gateway therefore also has less impact when installed remotely, because it just uses RFCs to connect with the backend (Appleby, 2012). A third advantage is that a remote Gateway can be installed in a ‘demilitarized’ zone that is located between the internal network where the SAP servers are located and the outside world (the Internet). In this case users can have one single point of entry (and login) to access all the backend systems.

### 4.2.2 Developing the services

The final step to expose the data and functionality is to create services based on the OData protocol (4.2.3.) using the different options available. These options, listed in order of difficulty, are:

1. **Standard SAP content**: With the Gateway module SAP offers some standard services that can be easily activated. These services are already implemented and offer some standard SAP data such as business partner information or sales document information. This content can only be activated in the Gateway module.

2. **RFM/BOR Generators**: The Gateway also offers generators that create services based upon Remote Function Methods (RFMs) or Business Object Repositories (BORs). RFMs are ABAP based functions that can be used to execute functionality like creating new sales order or retrieve business partner information. BORs contain all the functionality (RFMs) for a specific object like a business partner. The generators can only be used from within the Gateway module.

3. **ABAP coding**: The services can also be implemented manually. This requires ABAP skills of the developers. Implementing the several classes in SAP needed for the OData service allow the developer to create an OData service that meets the requirements. The coding can be done within the Gateway application with calls to the backend systems or when the IW_BEP is installed on the backend systems the coding can be done on these systems. In the last case the Gateway server merely functions as a hub between the services and the outside world.
In the fourth Service Pack (SP4) for Gateway 2.0 a new functionality is offered called the Gateway Service Builder. The Service Builder offers the functionality to create services using the second or third option described above and generates all the needed classes and settings to enable the services. The enabled services are called using the REST protocol and the language of the documents retrieved or sent is the OData protocol. This protocol is described in the next paragraph.

### 4.2.3 OData

The OData protocol is according to [www.odata.org](http://www.odata.org):

"... a Web protocol for querying and updating data that provides a way to unlock your data and free it from silos that exist in applications today."

In other words the protocol is created to extract data from the applications onto the web and also allow users to update and maintain this data.

#### Core of OData

The OData protocol is based upon the AtomPub, which is used to publish and update web resources. This protocol is based upon the XML language that is used to encode documents on the Internet to be readable to both human and machines. The XML language is used on top of the HyperText Transfer Protocol (HTTP), which is the foundation for the World Wide Web (WWW), the Internet. Due to this fact the OData protocol also complies with the Representational State Transfer (REST) architecture, which is the architecture for the WWW. Because it is compliant with the rules of this architecture the OData protocol is therefore ‘RESTful’. One of the implications is that the OData service enables 5 operations:

<table>
<thead>
<tr>
<th>OData operation</th>
<th>REST operation</th>
<th>Kind of Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE</td>
<td>POST</td>
<td>Create a new entity in the designated collection</td>
</tr>
<tr>
<td>GET</td>
<td>GET</td>
<td>Get an entire collection of entities</td>
</tr>
<tr>
<td>READ</td>
<td>GET</td>
<td>Get a specific entity in the designated collection</td>
</tr>
<tr>
<td>UPDATE</td>
<td>PUT</td>
<td>Update a specific entity in the designated collection</td>
</tr>
<tr>
<td>DELETE</td>
<td>DELETE</td>
<td>Delete a specific entity in the designated collection</td>
</tr>
</tbody>
</table>

**Table 5 The CRUD operations of the OData protocol**

In the figure on the left the top level is the OData for SAP. SAP developed its own annotations for OData to use with the Gateway, but with the release of Service Pack 3 of the Gateway SAP ceased with using these annotations. It still supports its own annotations but SAP recommends using the standard OData language.
Format

The OData services are located in specific packages, where the ‘sdata’ package contains the services that use the OData for SAP notations and the ‘odata’ package contains the services that are build with the standard OData notation. An OData service basically allows users to execute the operations listed in table 4 upon a collection of entities that are all the same. The only thing a user needs to know to start using an OData service is the URL of server offering this service.

Requesting this URL (which is the same as executing a GET operation on this URL) will return a service document (Appendix C, section A), which contains all the URLs to the services contained in this package.

Putting the tag $metadata behind the URL will return a document (Appendix A section B) that gives an overview of all the metadata of the services in that package. This metadata describes all the different entities and the kind of attributes these services have. When using the OData for SAP notations it also provides information whether an entity or attribute is creatable and/or updatable.

Putting the service name behind the URL will return a document containing a list of entities from that service. Using specific operators allows users to filter on attributes, number of entities or allows the user to skip the first $n$ number of results. Within the SAP systems users can define what kind of attributes will be represented in retrieving a list of entities, and which attributes will be shown in calling a specific entity. An example of a document containing a specific entity can be found in Appendix C, section C.

4.2.4 Sybase Unwired Platform (SUP)

An alternative way to use the OData capabilities of the SAP NetWeaver Gateway is the services of the Sybase Unwired Platform. This platform operates as middleware between the SAP environment and mobile devices.

The Sybase Company was acquired by SAP in July 2010, mostly for its Unwired Platform. This platform enables developers to connect mobile devices to backend databases, web services and most importantly to SAP landscapes using the Gateway. It supports several kinds of application types (Corbridge 2011):

1. Off-line applications. This type of application is a native coded based application that uses Mobile Business Objects (MBOs) to get data. The MBOs contain the translation between SAP objects and the outside world, and also contain the different operations that can be executed on these objects. According to (Corbridge 2011) these type of applications lend themselves especially for applications that are run on tablets. The downside is that the application is written in native code, which implies that for each platform a separate version has to be written.

2. Online applications. These applications are also written in native code, but they merely consist out of the interface and contain no data. This data is retrieved
through the Gateway connectivity. No data is stored on the device. These applications are to be used in cases with ‘always-connected’ devices and applications that require real-time data (such as dashboard applications). A minor drawback is the separate coding for each platform: a bigger drawback is that they only work when a device is connected to the SUP. In this case the SUP merely functions as a bridge, and offers some benefits as mobile device management.

3. Online+ applications. These are applications that are written in a platform independent language, with as good example HTML5. The only thing that has to be coded for each platform is the container that will show the HTML5 data. This enables push notifications, so new events can be instantly send to a device. These notifications can be stored on the device as well, so that the user can process them even when the system is ‘offline’.

Besides offering the functionality to connect mobile applications with SAP Business suites, the SUP offers other functionality such as Afaria, which can be used for Mobile Device Management (MDM). It also offers for some platforms, like Android, the possibility to create an application store in which the application can be offered to the users. Besides Afaria the SUP enables the possibility to work with messages (notifications). The disadvantage of the SUP is the extra license costs.
4.3 Configuring the backend for the demo application

This section explains the configuration of the SAP landscape that was used to build the demo application. It gives insight in the choices made, and the reason why these choices were made. During the execution of the project there was a change of company at which the project was executed. This will be reflected in each paragraph by describing both situations, but the biggest impact of this change was in the service development. The first paragraph explains the configuration of the gateway. The second paragraph depicts how the services were implemented. Because implementing the OData services was done twice for the project, the third paragraph discusses the difference between the two implementations.

4.3.1 SAP Landscape and Gateway configuration

The first step described in 4.2 is the installation of the Gateway (and its component). There is a choice between local and remote installation. In both situations during the project the Gateway was installed on a remote server, with connections to both a server with a SAP ERP application and a server with a SAP CRM application. On these systems the IW_BEP module was installed to enable service development on the backend systems and using the Gateway merely as a hub to make these services available to the outside world.

Gateway version

During the first situation the Gateway that was installed was version 2.0 with Service Pack (SP) level 2. SPs are updates of the Gateway with new and/or improved functionality. In the second situation the Gateway was up to version 2.0 with SP level 5, which included the Gateway service builder in contrast to SP2. The difference in SP level can make a difference in the development of the services, and during the project it was experienced that keeping up with the current SP level increases the speed of developing new services.

Availability from the Internet

In the first situation the Gateway was installed on a remote server due to the ease of updating it without touching the backend systems (too much), but there was no connection to the outside world. Connecting to it had to be done by using a Virtual Private Network (VPN) connection. Once the VPN connection was established the Gateway could be reach, but so could the ERP and CRM system. In the second situation a web dispatcher (WD) was used to gain access to the OData services of the Gateway. The difference between using a VPN or a WD is that the first option is safer because connecting to the gateway can only be accomplished when mobile devices are connected to the local network. The second option is easier because mobile devices just can connect to the URL to get access to the Gateway.
Available Data

In the first situation the Gateway server was connected to clients on both applications that already contained data from other projects. This required filtering in the OData services to retrieve only the project relevant data. In the second situation on both applications a new client was created without any data, except the data that was created during the installation of the SAP Best Practices. This data was initially used for testing the services, but this data had to be filtered out when new data was created that fit the requirements for testing new services. Therefore for good testing of the created services it is recommendable to use new and empty clients in which all new data can be customized to serve as testing material for the new services.

4.3.2 Developing OData services

In paragraph 4.2.2 three different options were provided for developing ABAP classes that enable OData services on the Gateway server. During the first time of developing the services all the options were tested to see how they worked and the result they delivered. The second time the services were developed the development phase started with creating a Data Model that describes the needed objects and their attributes. This model was based on the objects used during the first time of the development and is as follows:

<table>
<thead>
<tr>
<th>Object:</th>
<th>Create</th>
<th>Read</th>
<th>Update</th>
<th>Delete</th>
<th>Filterable</th>
<th>Sortable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALESDOCUMENTS (HEADER)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-OrderID</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-OrderType</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-Description</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-DocumentDate</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>-CustomerID</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>-SalesOrganization</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-DistributionChannel</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-Division</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-NetValue</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-Currency</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-RequestedDeliveryDate</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>-CustomerPurchaseOrderNumber</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-DeliveryStatus</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-BillingStatus</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-InvoiceStatus</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

| SALESDOCUMENTS (ITEM) |        |      |        |        |            |          |
| -OrderID | -      | +    | -      | -      | -          | -        |
| -ItemNumber | +      | +    | -      | -      | -          | -        |
| -Material | +      | +    | -      | -      | -          | -        |
| -Description | -      | +    | -      | -      | -          | -        |
| -Plant | +      | +    | -      | -      | -          | -        |
The filterable option indicates whether when requesting a collection of objects a filter argument for that attribute is allowed. The sortable option indicates whether the OData services should return a sorted collection of objects sorted on that attribute. The Data model was built without keeping the location of all the different attributes in mind and just based upon what was needed for developing a working application. Using mockups of what the application should look like helped determining which attributes of each object were needed and whether these attributes should be creatable or not.

The demo application had to support creation of sales document and quotations, and therefore several of the attributes are besides readable also creatable. Updating and deleting a sales order or quotation was not part of the process and therefore none of the attributes can be updated or deleted. Although updating was considered deleting was out of scope because deleting an object in a SAP environment can have serious repercussions and some objects cannot be deleted at all in SAP (objects such as Business Partners, they can only be archived).

**Used development tools**

As indicated the development of the services was done twice, and during the first time all options were explored.

The first option was to use the standard content services of the SAP NetWeaver Gateway, which includes services for Quotations and Contacts. It turned out that these services offer a lot of data, but they were missing vital pieces of data and offered a lot of data that was needed. These standard services are using a lot of Function Modules (FMs) from the backend systems. FMs can be used to get information (read) but also to store or update information (write). And as SAP systems are like big databases, these FMs can be seen as queries on those databases.

The second option was to use the RFM/BOR generators. RFMs are FMs that can be called from outside the application, while BORs are a collection of these RFMs for a
specific object within SAP such as a Sales Order or a Business Partner. These RFMs return a lot of data (mostly attributes for an object) but the generators enable selection of the data that should be offered by the service that will be generated. Enabling filtering of the data is depending on the implementation of the RFM. But in most cases the RFMs offered some input fields that could be used for filtering. Whether the output result was sortable also depended on the implementation of the RFM. But like the standard content none of the RFMs offered the output and input options that were needed for the project. Combining several of the RFMs offered more functionality than the standard content, but an OData service can only use one RFM for each of the CRUD operations. Using for example multiple Read operations to get all the needed data for one object therefore means creation of multiple OData services, and multiple calls from the application to all the services to get all the data for that object.

When trying to implement a service that offers all the needed information and the generators and standard content services cannot fulfill the requirements, the third option is to develop and implemented these OData with ABAP coding. In the first situation the service for the Create and Read operation of Sales Documents (such as Sales Orders and Quotations) were implemented by coding the service in ABAP. A tutorial\(^\text{13}\) was available but did not offer all the functionality needed (such as filtering) and more code was added to create a service applying to the needs in the data model (table 6). The basic idea behind the service implementation is that depending on certain parameters the service decides whether the incoming request is for a Sales Order or a Quotation. Instead of having to create 2 services, one for each type of Sales Document, the application can just call one service that will process the given input and gets the wanted output. Several RFMs were used in the service, but this is not visible to outside applications that call the service.

The second time the services were implemented it was done by another company that got involved in the demo application project, and they developed the services by implementing them using ABAP coding. For the Read operations no RFMs were used, but instead all the tables in the SAP backend systems were read that contained the data needed for the service. Although this offered a more freedom to choose the needed data it also required manual implementation of the filtering functionality. Instead of giving input parameters to the RFMs that were called, in this case during the reading of the tables in the backend systems the filter had to be checked whether the data complied with the filter.

The following table compares the three options of developing an OData services based upon the experiences from the demo application project:

---
\(^{13}\) [http://scn.sap.com/docs/DOC-22305](http://scn.sap.com/docs/DOC-22305)
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Standard Content</th>
<th>RFM/BOR Generators</th>
<th>ABAP coding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty</td>
<td>Easy</td>
<td>Easy</td>
<td>Hard</td>
</tr>
<tr>
<td>Data Availability</td>
<td>None</td>
<td>Only input/output of RFMs (Custom RFMs can be implemented)</td>
<td>Full ABAP</td>
</tr>
<tr>
<td>Filter options</td>
<td>Only standard</td>
<td>Depending on used RFM</td>
<td>Filter on anything</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Automatically</td>
<td>Mostly automatic</td>
<td>Manually</td>
</tr>
<tr>
<td>Reusability</td>
<td>Easy (standard available)</td>
<td>Medium (Services can be transported)</td>
<td>Difficult (depending where developed)</td>
</tr>
<tr>
<td>Availability at Gateway</td>
<td>Differs per service per SP of gateway</td>
<td>Always, but available RFM differ</td>
<td>Always</td>
</tr>
</tbody>
</table>

Table 7 Comparison of OData service development options

Most of the criteria are derived from the description of the both situations, but a few need explanation:

- **Maintenance**: With each update of the Gateway or backend systems, the service might need an update. The standard content is maintained by SAP, and most of the generator-developed services will be as well, because the standard RFMs are maintained by SAP as well. All the manually ABAP implemented services have to be maintained and might require additional development when installing updates.

- **Reusability**: Once services are developed for an application, these services might have to be installed on a different (client) system as well. SAP offers transport packages that contain all the ABAP modules that have to be transported. Generator-developed services can be easily transported because they will be located all on the Gateway server and can be bundled in once transport package. The same goes for ABAP-coded developed services if they have been developed on the gateway application, but otherwise several different transport packages are needed to transport all the implemented functionality.

- **Availability at Gateway**: With each new SP for the Gateway more standard content is made available. This means that systems running lower SP cannot offer yet standard content on which an application is running. The generators are available since the roll out of Gateway 2.0 and ABAP coding has been available since the first version of the Gateway.
Comparison between the developments

Because the development of the services has been done twice, a comparison can be made based upon experiences gained during the developments. During the first development all possible options were used, while during the second development another party that only used the ABAP coding option developed the services. Some comparison has already been done in table 7 between the different options. Using the standard content was rejected because of the absence of options for data selection (input/output). The two remaining options were using the generators or coding the services in ABAP. Based upon the experiences gained with both options, using the Generators seems to be the better option when developing services for an application that will follow SAP Best Practice guidelines. For most steps in the processes of these Best Practices RFMs exists that enable the execution of these steps and can be easily transformed into OData services using the generators. When services are needed that offer functionality that is not standardly available in the SAP applications there is a choice between developing new RFMs that can be transformed into services using the generators or to manually code the service in ABAP. The first option means that usually development is done in 2 applications, the Gateway application where the service is generated and the application that will offer the RFM. The second option can be done with programming the service in one of the two applications.
4.4 Design guidelines for configuring the backend

In section 4.2 an overview was given of the configuration options of the SAP NetWeaver module and the development of the OData services, while section 4.3 provided an overview of the choices and implementations made during the development project of the mobile application. This section will list design guidelines based upon the findings in those section.

*Design guideline 4.1: Install the Gateway on a remote server to enable quick modifications and easy updates and securing connections to the backend.*

In section 4.2 both the remote and local installation option were discussed, and installing the Gateway on a remote server eases securing the connection from the outside world into the SAP Business Suite. This is confirmed by the Gateway configuration of the demo application project were in one case a connection could be made using a web dispatcher, and in the other case using a VPN connection. Only these connections had to be secured because they were the only incoming connections from the outside world. Furthermore the literature made clear that for maintaining the Gateway it is better to remotely install it so that updates does not affect the other SAP applications, unlike when the Gateway were to be installed locally. The benefit of having a quicker response when installed locally could be an argument to install locally, but during the project the ‘slower’ response of a remote installed Gateway was not noticeable.

*Design guideline 4.2: Implement OData services with a top-down approach in difficulty, i.e. start with standard content with as last option ABAP programming.*

In this chapter several different implementation options for the OData services were discussed and used during the project. The amount of freedom to choose the data for each of these options also defines the amount of maintenance time, i.e. the standard content does not offer to choose the data used in the service but is maintained by SAP and therefore needs no maintenance of the developer while ABAP coding a service allows to choose almost any data wanted but at the same time the service needs to be maintained by the developer without any help from SAP. When implementing the services it seemed better to first check whether the standard content delivers the needed data. If not then check whether the data can be retrieved with the help of RFMs and if not then the service might be implemented by coding it in ABAP.

*Design guideline 4.3: Use a data model to describe objects and their attributes and depict the operations that have to be implemented for each attribute.*

In order to decide which implementation method to choose (see design guideline 4.2 as well) it will be helpful to create a data model based upon the to-be model created during the previous phase (described in chapter 3). This data model should contain a list of the objects that have to be provided by OData services, and the attributes that these objects should have. For each of these attributes it should be stated which of the CRUD operations should be implemented. Based upon this data model a development method
(standard content, ABAP coding...) can be picked for each object, keeping design guideline 4.2 in mind. During the first implementation of the services in the demo application project such a data model was not created leading to a cluttered gathering of implemented services. Replicating the same objects during the second implementation was therefore difficult and done in a different way (only ABAP coding) using a data model based upon the first implementation.
Part 3:
Implementing the application

Figure 23 iOS Development Tools
5 The mobile application

The goal of this thesis is to deliver a set of design guidelines that can be used when building a mobile application that will support a process (partly) executed in SAP systems. The previous chapters of this thesis describe the steps necessary to model this process and prepare the SAP systems (backend) to expose data and functionality. This chapter describes the next and final step in this project approach, the implementation of the application itself. The first section discusses several topics about implementing mobile applications. The second section describes the choices made regarding these topics during the implementation of demo application that is connected to SAP. The third section lists design guidelines based for building mobile applications based upon the first 2 sections.

5.1 Mobile applications

In the last few years, development of mobile applications is becoming more and more a hot topic in the world of software development. According to a report Gartner\textsuperscript{14}, by the year 2015 for each PC software project there will be 4 mobile application development projects. This same report also predicts that in the next years more and more employees of companies will be using tablets instead of laptops.

These trends make an interesting case to develop mobile applications in an enterprise environment. Several studies already proved that mobile applications can improve processes executed in a company (Mohelska & Tomaskova, 2010) (van der Heijden & Valiente, 2002). SAP also offers opportunities to connect mobile applications to their software. The following paragraphs will describe various aspects of mobile application development that should be considered when building a mobile application for processes in SAP.

5.1.1 Platforms

One of the choices that have to be made when implementing a mobile application is the selection of the desired platforms the application should run on. The main competitors in the market of mobile platforms are Apple’s iOS and Google’s Android. According to www.netmarketshare.com, iOS has a share of 60% in the mobile operating system market, while Android is present in 27%. Although iOS still has the biggest share, according to Gartner\textsuperscript{15} Android is the best selling Operating System (OS), and will relieve iOS from the first place sometime soon. Another trend that becomes visible is the rise of the operating system Windows Phone. With the release of the OS Windows 8 Phone this platform is expected to become another great player in the field of mobile operating systems.

Besides the so-called ‘native’ platforms (platforms that are based upon the code in which the operating system is written), there is also the possibility to create an application

\textsuperscript{14} http://www.gartner.com/it/page.jsp?id=2131115

\textsuperscript{15} http://www.gartner.com/it/page.jsp?id=2017015
that is web-based that means that it is created as a web page. HyperText Markup Language (HTML) is the programming language that is used to specify and make up pages upon the World Wide Web, and the latest released draft version of HTML, HTML5, offers such a lot of functionality that it can be used to create applications for mobile applications. Some projects have already proven that HTML5 can be used as a supplicant for native coding for building a mobile application (Gawley, Barr, & Barr, 2012).

Figure 24 Most important mobile platforms

Combining the native platform with web-based platform creates a third kind of platform, the hybrid platform. Some believe that this will be the most likely outcome in the battle between native implementation and web-based implementation (Charland & Leroux, 2011). SAP already collaborates with companies that offer tools to create hybrid applications.¹⁶

A fourth option in selecting the platform for development are the tools that create application that can be compiled cross-platform. These tools offer the functionality to implement the application in one language, and compile that language into different languages and platforms. SAP has the Sybase Unwired Platform (see 4.2.4 and 5.2) that offers this functionality. This tool is supposed to be the all-in-one tool (Sybase, 2012) that offers Mobile Device Management (MDM), connection with the SAP systems and a workspace in which the mobile application can be developed and cross-compiled.

5.1.2 Devices

Another choice that has to be made is the kind of device that applications will be running on. Although this is dependent on the kind of platform that has been or will be selected, all the platforms listed in 5.1.1 are running on both mobile smartphones and tablets.

The sale of smartphones and tablets has been growing the last couple of years, and according to Gartner\(^{17}\), the number of sales will pass the one billion mark in 2013. Although most of the sold smart devices (the collection of tablets and smartphones) are bought for personal use, there is also an increase in the sales of smart devices for business purposes. Gartner expects the sales to rise from 13 million units in 2012 to 53 million units in 2016. Around 67% of employees will have a smartphone at that time.

Due to the different properties of smartphones and tablets (i.e. the screen of tablet is sometimes twice the size of a smartphone screen), developing one application that will run on both types of smart devices cannot be arranged without loss of quality due to problems with scaling and other problems. In this case a choice has to be made whether the application will be implemented for both types, which has a serious impact on time and effort because the application has to be implemented twice (although there will be some overlap), or to just choose one type of device to develop for. When choosing to develop for one native platform, this narrows the number of devices down to only a few choices, which could result in a less marketable application.

![Smartphone (Galaxy S3) and tablet (the new iPad)](image)

**Figure 25 Smartphone (Galaxy S3) and tablet (the new iPad)**

### 5.1.3 Other development aspects

Besides the choice for platform and device there are some other important aspects of mobile development that require attention. Because implementing mobile applications is just a part of the scope of this thesis and several other studies such as (Wasserman, 2010) and (Nah, Siau, & Sheng, 2005) already describe these aspects, they will not be discussed in this thesis. Only the aspect of security will be briefly mentioned, because it was observed that possible customers were concerned about this topic, and it was discussed during the implementation of the demo application.

**Security**

The type of application this thesis discusses about uses data from company systems. This data is very sensitive and has to be protected; because when the wrong people get hold of this data it can have far-reaching consequences, especially when it is very private information about persons. There are several security options such as Virtual-Private-Network (VPN), sending encrypted data and use password protection.

\(^{17}\) [http://www.gartner.com/it/page.jsp?id=2227215]
5.1.4 Connecting with the backend

When implementing the mobile application, an important aspect is the way in which the application will be connected to the backend. In chapter 4 the configuration options for the backend are described, which led to the activation of the OData services. These services are used for connecting with the backend systems (SAP systems). Besides connecting directly to the SAP Gateway, there is also the possibility to use the Sybase Unwired Platform (SUP, see 4.2.4), offered by SAP. The SUP also uses the Gateway. The following picture depicts 4 ways of connecting the application to the backend systems:

![Diagram of four ways of connecting to the backend]

**Figure 26 Four ways of connecting to the backend**

The colored lines in figure 15 represent the following ways:

- The **orange** line shows the option in which a native application connects through the gateway with the backend systems. The **green** line represents the option in which instead of native coding a cross-platform compile tool is used to create an application, but stills it is directly connected to the gateway.

- The **red** line shows the dataflow between the application and the backend when using the Sybase Unwired Platform (SUP). The SUP offers the functionality of Hybrid Web Containers (HWCs). The interface of these HWCs is built in the native code, but the content is distributed in HTML format. This means that only several interfaces have to be build for different platforms and devices, but content has to be generated only once.

- The **blue** line shows the other option of connecting to the backend. The SUP offers the functionality to create native applications that use Mobile Business Objects (MBOs), which contain all the logic and functionality of objects. These MBOs are exchanged between the application and the SUP, and allow offline usage.
Besides using the SAP NetWeaver Gateway for connecting with the SAP systems there are some other possibilities such as the SAP NetWeaver Portal en SAP Business Objects. The latter one is mainly used to expose financial data, while the NetWeaver Gateway was developed just to alleviate mobile development through the gateway, and therefore make the portal useless.
5.2 Implementing the demo application

The previous section discussed topics that are important when building a mobile application that supports processes involving SAP. This section describes the decisions that were taken in the project accompanying this thesis where a demo application was created. Each of the topics in the previous section is described and discussed.

5.2.1 Choosing a platform

In paragraph 5.1.1 several possibilities were described regarding the platform the application could be implemented on. Discussions about what platform offers the best implementation possibilities for enterprise applications are found everywhere\(^{18}\), and there is no right answer. One way to decide which platform to choose is by checking out the current existing applications distributed by SAP and note on which platform they are running. Taking a look at the chart by Moy (Moy, 2012), it stands out that the most of the current application at that time (January 2012) are implemented for iOS. For the Android platform only 3 applications were available and Windows Phone (Windows Mobile at that time) is only good for 4 applications. There are also some applications for the BlackBerry platform, but the common expectation is that this platform is on its return from the business market. Hybrid applications are not listed, but are also not totally undisputed (Bestvina, 2012). They also require the SUP to be developed. The following table compares the 3 different platforms (native, HTML5, hybrid) on several criteria:

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Native</th>
<th>HTML5</th>
<th>HWC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>-</td>
<td>+</td>
<td>-/+</td>
</tr>
<tr>
<td>BYOD (Bring-Your-Own-Device)</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Offline</td>
<td>+</td>
<td>-</td>
<td>-/+</td>
</tr>
<tr>
<td>Deployment</td>
<td>-/+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Use hardware</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 8 Comparing development platforms

The first criterion is development. Developing native applications requires a lot of work, especially when development is needed on several different platforms. HTML5 is supported by most of the standard browsers. The HWCs require implementation of the containers in native code, but the content is implemented once, and distributed across the different platforms.

Second criterion is about Bring-Your-Own-Device (BYOD) policy. When companies use this policy, they allow employees to use their own smart devices to do their job. This

\(^{18}\) http://www.networkworld.com/community/tech-debate-android-ios
means that an application has to be written for almost every kind of platform. Around 25% of the Dutch companies use this policy\textsuperscript{19}.

The third criterion is the offline capability of the application. Using the SUP native applications can be used offline, but only when connected through MBOs (Corbridge, 2011). All other scenarios with native applications support offline capability because the data can be stored on the device. A remark is that the synchronization process requires a robust implementation. HTML5 applications are not meant for offline usage, although the HWC could be implemented to work offline, but it requires a lot of extra implementation.

Deployment is considered as another criteria. Native applications can be deployed using the application stores (App store, Play store etc.), but this limits the freedom of publishing them. HTML5 applications can be published freely by just running them on a server. HWC applications consists of a native container, which also has to be published through an app store.

The last criterion considered is the usage of the available hardware. Native applications can easily access the hardware, while HTML5 applications have more trouble accessing the available hardware. It has to be noted that the expectation is that HTML5 in the future will be able to access the hardware better. Because HWCs use a native coded container, they can also use all the available hardware.

**Platform choice for demo application**

For the project executed along with this project all listed possibilities were discussed. From the start of the project it was said that the application should run under iOS, but the cross-platform tools also looked promising. But when further research of them was done, it became clear that although they could deliver the applications, the support for the OData Software Development Kit (SDK) was not there at the moment, which meant that this piece of software could not be cross-compiled using the cross-compilers. The other option with the cross-compilers offers the possibility that the OData SDKs could be integrated after compiling the codes for the different platforms, but this would require knowledge of each the platforms that would be compiled, and it was chosen to select only the iOS platform to develop for. Given the fact that the BYOD policy also was loosing terrain, developing for one particular platform, which allows for best use of the hardware, seemed justified. Developing the application for multiple platforms in a parallel way was considered but rejected because it would result in loss of time, and the main purpose was to create a demo application as quickly as possible. Using HTML5 was also considered, but let go after it became apparent that it limited the possibilities of the applications. Another argument is that even big companies agree that HTML5 is not the best option yet (Olanoff & Zuckerberg, 2012), and native development will most likely result in better applications.

\textsuperscript{19} http://www.computable.nl/artikel/nieuws/infrastructuur/4572950/2379248/nederland-raakt-byorvoorsprong-kwijt.html
5.2.2 Selecting a device

The chart from Moy (Moy, 2012) shows that the vast majority of the applications at that time were developed for the smartphones, and considering tablets only the iPad had more than one application. But in paragraph 5.1.3 it already becomes clear that market of the tablets is growing faster each year, and more and more businesses are equipping themselves with tablets to execute their work.

Choosing which devices to develop for depends on several factors such as whether there is a BYOD policy in place and the functionality requirements of the application. One of the main questions is whether a 3g connection should be available (most tablets do not offer this yet) or a Wi-Fi connection should do.

Device choice demo application

For the project executed for this thesis the iPad was chosen as the device to develop for. The fact that the iOS platform was chosen left no choice but iPhone and/or iPad, the iPad offers more possibilities from a user experience perspective. From a hardware point of view the iPhone and iPad offer most of the same functionality (although the iPhone always has 3g connection, while not all iPad have these), but a main reason for native coding is the increased user experience, and using a tablet will increase this experience even more. The bigger screen of a tablet will offer more functionality at once, while a smartphones screen limits the amount of functionalities that can be offered.

5.2.3 Connecting with the backend

In paragraph 5.1.3 several options were discussed in how to connect with the backend. Two of these four options used the Gateway directly; the other two options were connecting by using the Sybase Unwired Platform. When developing the demo application the 2 latter options were dropped because of the following reasons:

- Availability: At both companies were the project was executed the SUP was not available. The option to purchase the SUP was explored, but rejected (see next argument).
- Costs: The costs of buying and using the SUP were too high in comparison with the extra functionality it had to offer. The several available futures turned out to be used for only certain types of platforms or devices
- Implementation limitations: When choosing the SUP as way of connecting, it also limited the possibilities for implementing the application. Using SUP compels to use the cross-compiler of the SUP or use the SUP SDKs for connecting. The benefit is that the SUP provides a set of tools for options like Mobile Device Management and Firewalling.

The other two options were the native development that connected through the Gateway and the use of cross-platform compilers. As already stated in paragraph 5.2.1 using the cross-compiler platform was considered but not chosen due to the fact that there was no OData SDK that could be cross-compiled. Furthermore SAP offers generator tools that
compile classes and even simple example applications for most of the native platforms such as iOS, Android and Blackberry. Using these tools to create classes for the objects from the OData services and connecting to those OData services eased the implementation of the application a lot.

The tool uses the standard OData SDKs, available at www.odata.org/libraries, and creates a complete package of classes with all the needed methods. This is done based upon the metadata model provided by an OData service. The generated classes can be imported into a project and used to connect to the service and retrieve or send data.
5.3 Conclusion and design guidelines

This chapter described three major choices that have to be made when developing a mobile application that supports processes in SAP. The chapter discussed possible options for each of the choices, and showed the choices made in the project that was executed for this thesis. Based upon the findings in the literature and the experience of having build a mobile application for processes in SAP, the following design guidelines can be made:

**Design guideline 5.1**: The choice of implementation platform and device relates to the functionality the application has to offer and the need to access hardware.

The chapter showed that several platforms are available for developing a mobile application, ranging from native platforms through web-based platform to cross-platform compilers. A table was provided in which these options were compared, and it result was that a bigger need for hardware access results in a bigger need for native development. Native coding also offers a lot more functionality, but will in most cases require also more time and effort to accomplish a result, especially when developing for multiple platforms. The advantage when choosing native development is the availability of generator tools, which make connecting with the backend faster and easier.

**Design guideline 5.2**: Implementing an application in native code will speed up the implementation process.

If the implementation of an application is done using native code a set of (free) tools available to quickly implement a simple application. SAP offers generator tool for every large language (such as Objective-C and Android), and App stores are also available for each of these language. Furthermore almost every programming language offers a free to use IDE (such as XCode). When using multi-platform compilers, connection with the backend systems via the OData channels still have to be implemented in the native language because generator tools for multi-platform compilers are not (yet) available. The disadvantage is that the application has to be re-implemented if multiple platforms are available, but because the Bring-Your-Own-Device policy is not widely used, this disadvantage does not outweigh the advantages of native implementation.

**Design guideline 5.3**: Use the standard (free) tools to be able to use agile methods as implementation approach, especially when developing mobile applications for SAP processes for the first time.

If a project is executed using the standard tools, a simple and working application can be created quickly. Based on this basic application new features can be applied, and the application can be enhanced. Especially when building such applications is done for the first time, it can help getting more experience with the implementation tools, while not having to solve all the difficult implementation options at once. Applying for example the SCRUM methodology will guide the project in this way of working.
Part 4:
The result: Guidelines

Figure 27 Guidelines (or Directions)
6 Conclusions

Mobile applications are more and more embedded in the daily lives of people. They are executed on mobile devices, which offer a lot of additional resources compared to personal computers and laptop. SAP is widely used across companies. This big software suite is used to support processes in all kind of business areas. A new opportunity opened up to combine these two topics and to come up with mobile applications that support business processes involving SAP. Because this area of mobile applications that supported SAP-involved processes is relatively new, design guidelines are needed on how to execute a project that will deliver a mobile application that does indeed support SAP-involved processes. To gather these design guidelines, the first sub question investigated was whether current software development methodologies would suffice (section 6.1). The second sub question (section 6.2) involves modeling the processes in such a way that mobile application support can be easily determined. After modeling the process and determining the possibilities for mobile application support, the next sub question (section 6.3) investigated is how to make connection to the SAP suite available. The final step and sub question (6.4) investigated dealt with deciding on which architecture to use for the mobile application itself. An evaluation and reflection of the design guidelines is provided as well (6.5). The last section (6.6) of this chapter lists limitations of this thesis and future work possibilities.

6.1 Using existing methodologies

The first sub question that was answered contained the possible usage of commonly known, existing methodologies for software development and the methodology of SAP itself for executing projects with SAP software. Although these methodologies might be helpful when building a mobile application that supports business processes, none of them fulfill all the needs when building such a mobile application. A mix of these methodologies, such as the ASAP Agile Add-on (the partly agile version of the standard SAP methodology) already seems to be making a good candidate. The approach/methodology used when building the demo application also proved that agile approaches seem to be more favorable candidates. The following 2 design guidelines are based upon these findings during the implementation of the demo application and the available literature (section 2.4):

Design guideline 2.1: When not restricted to waterfall-like methodologies, agile approaches should be considered over traditional approaches, mainly because agile approaches can deal with changes a lot faster, which will occur a lot especially when developing for the first time.

Design guideline 2.2: Try using the ASAP methodology (with the agile add-on or perhaps the mobile add-on) when developing mobile applications for processes (partly) supported by SAP.
6.2 Modeling Processes & identifying mobile opportunities

Besides giving design guidelines on whether to use existing methodologies or not, this thesis also provided design guidelines for the different topics involving development of mobile application for supporting SAP-involved business processes. The first topic discussed and recommended upon was process modeling. To be able to develop mobile application that indeed support business processes, the first step to take is to determine in which steps or part of the process the mobile application can give support. To achieve this a model has to be created of the current situation, and a model of the future situation (involving a mobile application). Several modeling languages were discussed and tried out during the project for the demo application. Modeling tools were not discussed due to the fact that the company supporting this thesis was supplier of a modeling tool, ARIS. The following design guidelines (section 3.4) were made based upon discussions, literature and experience from the demo application project:

Design guideline 3.1: Use BPMN to model the processes.

Design guideline 3.2: Start from the as-is process to model to a to-be process with mobile support.

Design guideline 3.3: Identify data input in the process as accurate as possible to enable external source support in the mobile application

6.3 Opening up the back-end (SAP)

Having modeled a to-be process including support from a mobile application, next question answered or topic recommended on was the exposure of data from the backend SAP systems. To be able to support business processes mobile applications have to retrieve and send data to the SAP suite. The SAP suite includes a module that allows these operations and it is called the SAP NetWeaver Gateway. This module can be installed and configured in several ways, and the design guidelines for this topic are design guidelines on how to do this. The following design guidelines were made:

Design guideline 4.1: Install the Gateway on a remote server to enable quick modifications and easy updates and securing connections to the backend.

Design guideline 4.2: Implement OData services with a top-down approach in difficulty, i.e. start with standard content with as last option ABAP programming.

Design guideline 4.3: Use a data model to describe objects and their attributes and depict the operations that have to be implemented for each attribute.
6.4 The Frontend: Mobile application

Final step in developing a mobile application supporting business processes involving SAP is the actual implementation of the mobile application itself. The world of mobile devices and platforms is a relatively young one, and many variations are available. The answer to the sub question of which architecture to choose when implementing such a mobile application is a set of design guidelines for choosing a platform and type of device, choosing implementation tools and connecting to the services created in the backend SAP systems. The design guidelines are listed below:

Design guideline 5.1: Connecting with the backend can be done with simple tools, especially when not developing for multiple platforms.

Design guideline 5.2: Implementing an application in native code will speed up the implementation process.

Design guideline 5.3: Use the standard (free) tools to be able to use agile methods as implementation approach, especially when developing mobile applications for SAP processes for the first time.
6.5 Generalization

As this chapter conforms to the fourth stage in the ADR method (see section 1.5), the final step in this thesis is to try to generalize the design guidelines formulated in the previous sections. This is done by first generalizing the problem (main question) of this thesis, and accordingly generalize the guidelines.

6.5.1 Generalizing the main question

The main question of this thesis is “How to build applications for mobile devices that will support business processes involving the ERP system SAP”. This question is already fairly generalized, including only one limitation factor, the specific ERP system. Removing this limitation provides the general problem: “How to build applications for mobile devices that will support business processes involving ERP software”.

6.5.2 Generalizing the design guidelines

Generalizing the design guidelines is the next step of generalizing the findings of this thesis. This section contains the generalization, if possible, of the design guidelines for each phase of the development process:

Use of existing methodologies

1. Select an agile method to develop the mobile applications if possible, with preference to methods provided by the ERP vendor.

Using agile methods to implement mobile application enables making changes to requirements and enables high customer participation. Unless mobile applications have to apply to certain requirements such as robust security, agile methods are preferred. If such method are provided by the vendor of the ERP software the applications are developed for, they have preference because most of these methods provide documentation templates, which will likely improve the understandability.

Modeling processes & identifying mobile opportunities

1. Use a modeling language understandable for both IT and business.

To ensure that both the IT (the mobile application developers) and the business (the end-user and customers) get grip of the modeled process, a modeling language has to be selected that can be understood by both parties. BPMN 2.0 is such a language, both other language may suffice as well.

2. Create a to-be model from an as-is model by identifying mobile application opportunities.

Modeling the current (as-is) process helps to identify the possible steps in a business process that can be supported by a mobile application. Modeling the future (to-be) process might help to let the business understand the changes and benefits mobile applications can offer to them.
Opening the backend

1. **Use a data model to describe objects that can be communicated and the services that can be executed.**

In case of this thesis in order to create a working data connection between the application and the ERP software development work was needed on both ends of this connection. To make sure that both sides understand what is expected of them, a data model is required that describes the objects (along with their attribute types etc.) or services that will be communicated. As the data model is basically the only thing connecting the application developer and the ERP-side developer, sticking to that data model is required.

2. **Use a top-down approach regarding ease of implementation**

Easy implementations are (based upon the experience in this thesis) limited in what they can offer, but tribute to the agile development of the mobile applications. Most likely the easy implementations are services already offered by the ERP vendor. Using these services is recommended because the ERP vendor will maintain them.

The mobile application

1. **Implement the application in the native coding language of the platform it runs on.**

Implementing the application using native coding offers in most cases much more freedom to use the available data sources of the mobile device. Furthermore it offers options to ensure the fastness of the application because memory and processing power are also directly addressable.

2. **Use available tools offered by the mobile platform vendor or the ERP vendor to implement the application.**

If tools such as code generators are provided they should be used to quickly create prototypes. Providing customers early on with prototypes will help to identify possible requirements changes and allows the customers to participate in the project, and therefore steer the direction in which the application development is going.
6.6 Contribution

The main contribution of this master thesis is the set of design guidelines presented in the previous sections (6.4 and 6.5). The first set is a specific set for implementing mobile applications for SAP-involved business processes using the SAP NetWeaver Gateway. These are founded on a literature research and the experience gained during the creation of demo mobile application. The latter set is a generalized version of the first set that could be used for building these kind of mobile applications without limiting the scope to specific ERP software.

A side contribution was made towards Scheer Management by the creation of working demo application that can be used to show customers the possibilities of using mobile applications and the SAP NetWeaver Gateway maintain and read data from the SAP software.

6.7 Evaluation

In the previous sections design guidelines were presented for developing mobile applications for supporting business processes involving ERP software. These guidelines were formulated after reading literature and getting experience by developing a demo application.

Building upon the result of this demo application, the project is continued at Scheer Management BV to build a mobile application for sales representatives. Following the guidelines formulated in this thesis new functionality is added to the application and more processes are supported by the application (such as entering new customers and contact persons into the system, and adding marketing data and pictures of customers).

Regarding the design guidelines for the use of existing methodologies they are not followed (yet), the same agile approach is still used as described in this thesis (design guideline 2.1), thus not using one of the existing methodologies (design guideline 2.2). As a result documentation is missing, but customer participation is used to improve the application.

The guidelines for modeling the process and identifying mobile opportunities are used to add new functionality to the application. Although not every added process is modeled in its to-be situation (design guideline 3.2), the current situation models are used to identify support opportunities (design guideline 3.3). The to-be models will be created as well to explain to customers the benefits of using mobile applications for their business process.

The development on the SAP software is done by another company, which has a SAP landscape complying with design guideline 4.1. This provides a good opportunity to compare a situation where the guidelines formulated in thesis were followed and a situation where they are not. Instead of using the tools and generators provided by SAP to create the OData services, these services are now developed by coding. This leads sometimes to little errors and bugs due to human errors, which could have been prevented by using the provided tools (design guideline 4.2). Furthermore a good data
model was missing at the beginning of the cooperation, but has been created and is used to communicate what objects and services are needed (design guideline 4.3).

Developing the application itself is still done by applying the design guidelines as well. The application, an iOS application for the iPad, is implemented using XCode (Apple’s own IDE) and the generator provided by SAP to create proxy classes to connect to the services on the SAP Gateway, thus following guideline 5.1 and 5.3. The implementation is done in iOS’s native language, Objective-C (design guideline 5.2). This ensures that all needed resources of the device, the iPad, can be addressed and used.

Overall, using the design guidelines presented in this thesis, already one application has been created and published in Apples App Store\(^\text{20}\). Several possible customers are testing this application and their first feedbacks are positive.

6.8 Reflection & Limitations

This thesis (and the internship attached to it) was started to present Scheer Management a set of design guidelines for creating mobile applications that support SAP-involved business processes. During the beginning of the internship the focus was on learning business process modeling and the use of ARIS. The latter limited the scope of this thesis regarding using a modeling tool. Regarding business process modeling a choice was made between EPC (the native ARIS modeling language) and the upcoming language BPMN.

After having modeled 2 processes the focus of the internship shifted to learning SAP and all its components. After following a basic SAP training course the focus was aimed at the SAP Gateway. Furthermore the Sybase Unwired Platform was investigated as possible candidate. Because this platform was not installed and involved extra costs when installed it was dropped as an option, limiting the scope to unlocking SAP data and functionality through the possibilities the SAP NetWeaver Gateway offers. Investigating the different options and how to use these options took some time as well.

Having unlocked some data and functionality, the focus was shifted to implementing an iOS application. Android was considered as well, but dropped because of the many variants of devices that are available for that platform, all of them with their own (unique) specifications. The first step in implementing was learning Objective-C, the native iOS language, and the use of the tools Apple provides for building an application. Having learned this, next step in implementing was coding the application and creating a connection between the application and the SAP NetWeaver Gateway. This was the final step in implementing the application, and a first working prototype was created.

Halfway during the entire internship the group that initiated the internship switched companies, which led that the project was stopped at the old company (Software AG). Because they founded a new company, Scheer Management BV (as a subsidiary of Scheer Management GmbH), SAP systems were not available from the start of the internship at this new founded company. During this time, a great part of this thesis

was written. The benefit of this switch was that after having SAP systems available the OData services implemented at the old company had to be re-developed again, creating the chance to use a different approach. This leads to conclusions that can be found in the previous section.

Overall, a lot of time has been invested in learning all the different programming language and tools such as Objective-C, XCode and the SAP language ABAP. On one side this limited the scope regarding implementation options of the mobile application, but it also was an opportunity to experience every part of developing a mobile application that supports SAP-involved business processes.

6.9 Future work

Future work opportunities will rise in the area of connecting the mobile application with the SAP backend systems. This area is upcoming and still moving, and with every new release of the SAP NetWeaver Gateway module new opportunities are presented to connect to the backend system.

Process modeling is also a moving area, especially the BPMN language. BPMN 2.0 was officially released in January 2011, but it is still a work in progress. It still has gaps that have to be filled, and it will be worth researching newer versions to see if they will better fit the need for modeling processes involving mobile applications.
References


Recker, J. C., & Dreiling, A. (2008). Does it matter which process modelling language we teach or use? An experimental study on understanding process modelling languages


<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIS</td>
<td>ARchitecture of integrated Information Systems</td>
</tr>
<tr>
<td>ASAP</td>
<td>Accelerated SAP</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business Process Model and Notation</td>
</tr>
<tr>
<td>BYOD</td>
<td>Bring-Your-Own-Device</td>
</tr>
<tr>
<td>CRM</td>
<td>Customer Relationship Management</td>
</tr>
<tr>
<td>EPC</td>
<td>Event-driven Process Chain</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>NW</td>
<td>(SAP) NetWeaver</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>SAP</td>
<td>Systems Applications and Products in Data Processing</td>
</tr>
<tr>
<td>SUP</td>
<td>Sybase Unwired Platform</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>UML AD</td>
<td>UML Activity Diagrams</td>
</tr>
<tr>
<td>WD</td>
<td>Web Dispatcher</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
<tr>
<td>XP</td>
<td>eXtreme Programming</td>
</tr>
<tr>
<td>WD</td>
<td>Web Dispatcher</td>
</tr>
</tbody>
</table>
Appendix A: Quotation Process
Appendix B: Simple EPC
Appendix C: OData Format

A. OData Service Document

```xml
  <app:workspace sap:semantics="things">
    <atom:title>Things</atom:title>
  </app:workspace>
  <app:workspace sap:semantics="data">
    <atom:title>Data</atom:title>
    <app:collection href="/create_bupaCollection" sap:creatable="true" sap:updatable="false" sap:deletable="false">
      <atom:title>createBupaCollection</atom:title>
      <atom:title>create_bupa</atom:title>
    </app:collection>
    <app:collection href="/bupaCollection" sap:creatable="true" sap:updatable="false" sap:deletable="false">
      <atom:title>bupaCollection</atom:title>
      <atom:title>bupa</atom:title>
    </app:collection>
    <app:collection href="/bupa_classCollection" sap:creatable="true" sap:updatable="false" sap:deletable="false">
      <atom:title>bupaClassCollection</atom:title>
      <atom:title>bupa_class</atom:title>
    </app:collection>
    <app:collection href="/et_partnerrolesCollection" sap:creatable="false" sap:updatable="false" sap:deletable="false">
      <atom:title>et_partnerroles</atom:title>
    </app:collection>
    <app:collection href="/bupa_roleCollection" sap:creatable="true" sap:updatable="false" sap:deletable="false">
      <atom:title>bupaRoleCollection</atom:title>
      <atom:title>bupa_role</atom:title>
    </app:collection>
    <app:collection href="/bupa_sales_orgCollection" sap:creatable="true" sap:updatable="false" sap:deletable="false">
      <atom:title>bupaSalesOrgCollection</atom:title>
      <atom:title>bupa_sales_org</atom:title>
    </app:collection>
    <app:collection href="/z_contactCollection" sap:creatable="true" sap:updatable="false" sap:deletable="false">
      <atom:title>zContactCollection</atom:title>
      <atom:title>z_contact</atom:title>
    </app:collection>
    <app:collection href="/z_materialCollection" sap:creatable="false" sap:updatable="false" sap:deletable="false">
      <atom:title>zMaterialCollection</atom:title>
      <atom:title>z_material</atom:title>
    </app:collection>
  </app:workspace>
</app:service>
```
B. OData Metadata Document

```xml
<complexType Name="create_bupa" minOccurs="0" maxOccurs="1" sap:content-version="1">
  <complexContent/>
</complexType>
```

- `<edm:Edmx Version="1.0"/>
- `<edm:DataServices m:DataServiceVersion="2.0">
- `<Schema Namespace="ZIPAD">
- `<Entity Name="create_bupa" m:HasStream="false" sap:content-version="1">
  - `<Key>
    - `<PropertyRef Name="value"/>
    - `<PropertyRef Name="scheme_agency_id"/>
    - `<PropertyRef Name="scheme_id"/>
  </Key>
  `<Property Name="lv_category" Type="Edm.String" MaxLength="10" sap:label="BP category" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="lv_searchterm" Type="Edm.String" MaxLength="20" sap:label="Search Term 1" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="first_name" Type="Edm.String" MaxLength="40" sap:label="First Name" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="house_no" Type="Edm.String" MaxLength="10" sap:label="House Number" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="postal" Type="Edm.String" MaxLength="10" sap:label="Postal Code" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="value" Type="Edm.String" Nullable="false" MaxLength="72" sap:creatable="false" sap:updatable="false"/>
  `<Property Name="scheme_agency_id" Type="Edm.String" Nullable="false" MaxLength="16" sap:creatable="false" sap:updatable="false"/>
  `<Property Name="org_name" Type="Edm.String" MaxLength="40" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="lv_group" Type="Edm.String" MaxLength="4" sap:label="Grouping" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="region" Type="Edm.String" MaxLength="3" sap:label="Region" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="correspond_language" Type="Edm.String" MaxLength="1" sap:label="Correspondence lang." sap:creatable="true" sap:updatable="false"/>
  `<Property Name="city" Type="Edm.String" MaxLength="40" sap:label="City" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="last_name" Type="Edm.String" MaxLength="40" sap:label="Last Name" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="country" Type="Edm.String" MaxLength="3" sap:label="Country Key" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="scheme_id" Type="Edm.String" Nullable="false" MaxLength="30" sap:creatable="false" sap:updatable="false"/>
  `<Property Name="title" Type="Edm.String" MaxLength="4" sap:label="Title" sap:creatable="true" sap:updatable="false"/>
  `<Property Name="street" Type="Edm.String" MaxLength="40" sap:label="Street" sap:creatable="true" sap:updatable="false"/>
</Entity>`
C. OData Entity Document