Next generation media: A user friendly Android tablet application for business architecture modeling

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Science in
Applied Economic Sciences: Business Engineering

Joost Dumeez

with supervision from

Prof. Dr. Geert Poels, Co-promotor Maxime Bernaert
PERMISSION

The undersigned states that the content of this master thesis may be consulted and/or reproduced, if cited.

Joost Dumeez
Gedurende de afgelopen jaren is veel onderzoek gedaan om enterprise architectuur (EA) benaderingen voor grote en complexe ondernemingen te ontwikkelen. Bijgevolg is een scala aan instumenten ontwikkeld om deze grote ondernemingen te helpen bij EA management. Echter, de traditionele kleine en middelgrote ondernemingen (KMOs), die zeer belangrijk zijn voor de economie, zijn grotendeels genegeerd. Recent pas is er onderzoek gestart naar een nieuwe EA aanpak voor KMOs. Deze aanpak wordt CHOOSE genoemd. Ondersteuning door tools is bijna onmisbaar in complexe omgevingen, en tijdens het uitvoeren van case studies in KMOs werd de nood aan tools ook snel ervaren. Helaas vinden EA beoefenaars in grote bedrijven de bestaande tools reeds gebruiksonvriendelijkheid. Een andere aanpak was dus noodzakelijk om een tool te ontwikkelen die bruikbaar kan zijn voor managers in KMOs. In deze thesis worden alle stappen beschreven in de ontwikkeling van deze tool, van onderzoek over managers tot de uiteindelijke tool presentatie.
Acknowledgement

I would like to use this preface to thank all the persons, who have contributed to the realization of this master thesis, over the past two years.

First of all I would like to thank Prof. Dr. Geert Poels, my promoter, for his valuable feedback and his willingness to answer all my questions.

Furthermore, I want to thank Maxime Bernaert, for his assistance during the creation of this master thesis. His professional advice and critical remarks helped me to stay sharp and gave me the necessary impulses to go the extra mile. Additionally, with his help, I managed to get my first publication.

Finally, I thank my godfather and girlfriend for their unconditional support and encouragement. Special thanks go to my parents for the chance they gave me to complete my studies and their unshakable faith in me.
# List of Figures

| Fig. 1 | The amsterdam information model showcases the different areas of interest in information management research. | 2 |
| Fig. 2 | Four ingredients of an enterprise architecture | 4 |
| Fig. 3 | The Zachman framework | 5 |
| Fig. 4 | The architecture development method | 6 |
| Fig. 5 | Classification criteria | 7 |
| Fig. 6 | Criteria derived for CHOOSE | 9 |
| Fig. 7 | CHOOSE subject focus | 10 |
| Fig. 8 | Three layers of EA, from a focus on technology towards a focus on business | 11 |
| Fig. 9 | The CHOOSE metamodel | 11 |
| Fig. 10 | Need for tool support during case studies | 14 |
| Fig. 11 | Diagram showing a low usability rating | 16 |
| Fig. 12 | Typical EA tool | 16 |
| Fig. 13 | Traditional software development process | 17 |
| Fig. 14 | Software development process proposed by Cooper | 18 |
| Fig. 15 | Software development process, design step | 19 |
| Fig. 16 | Goal-directed design process | 19 |
| Fig. 17 | First step of goal-directed design process: Research | 20 |
| Fig. 18 | Step two of goal-directed design process: Modeling | 24 |
| Fig. 19 | Third step in goal-directed design process: Requirements | 27 |
| Fig. 20 | Fourth step in the goal-directed design process: Framework definition | 28 |
| Fig. 21 | The three main operating systems (from left to right: iOS, Windows, Android) | 30 |
| Fig. 22 | Two layer application hierarchy | 33 |
| Fig. 23 | Usage of skeuomorphisms by Apple | 34 |
| Fig. 24 | The continuum between the implementation and mental model | 35 |
| Fig. 25 | Information overload on today's websites | 36 |
| Fig. 26 | Wide hierarchic trees obtained during case studies | 38 |
| Fig. 27 | Combination of OSX and Windows | 39 |
| Fig. 28 | Arrow makes user actions more predictable | 40 |
| Fig. 29 | Navigation bar for easy access to other hierarchical layers | 40 |
| Fig. 30 | Easy to delete and modify | 40 |
| Fig. 31 | Screen overlays | 41 |
Fig. 32 Flexible search algorithms ................................................................. 42
Fig. 33 Nine page introductory guide ............................................................. 43
Fig. 34 Fifth step in the goal-directed design process: Design refinement ........... 44
Fig. 35 Heavy use of graphical representations in the application ..................... 44
Fig. 36 Use of contrast to draw attention (left Blinksale website, right CHOOSE application) .... 45
Fig. 37 Contrast ratios ..................................................................................... 46
Fig. 38 Present information upon request .......................................................... 47
Fig. 39 Third step in software development process: Application internals .......... 48
Fig. 40 Elements of software quality: Internal attributes dimension ................... 49
Fig. 41 First element of software quality: Functionality .................................... 49
Fig. 42 Parameterized query to avoid SQL injection .......................................... 50
Fig. 43 Second element of software quality: Reliability .................................... 51
Fig. 44 Android support library and snapshot from imported Android frameworks .... 51
Fig. 45 Example of annotations in Java ............................................................. 52
Fig. 46 Android activity lifecycle ...................................................................... 54
Fig. 47 Third element of software quality: Efficiency ......................................... 54
Fig. 48 Goals loaded when clicked .................................................................... 55
Fig. 49 Loading in all 9 pages of the guide .......................................................... 55
Fig. 50 Full text search example ........................................................................ 56
Fig. 51 Extra table for full-text search ............................................................... 56
Fig. 52 Fourth element of software quality: Maintainability ............................... 57
Fig. 53 Application structure ............................................................................ 57
Fig. 54 Cyclomatic complexity ......................................................................... 58
Fig. 55 Other code metrics ................................................................................ 59
Fig. 56 Readable code ....................................................................................... 59
Fig. 57 Final element of software quality: Portability ....................................... 60
Fig. 58 Essential metamodel ............................................................................ 61
Fig. 59 Database design .................................................................................... 64
Fig. 60 Fourth step in software development process: field test feedback .......... 65
Fig. 61 Page indicator ....................................................................................... 66
Fig. 62 Improvements after feedback ................................................................. 67
Fig. 63 ‘Add lower Goal’ button ......................................................................... 67
Fig. 64 Final step in the software development process: Tool presentation .......... 73
Fig. 65 Page one of CHOOSE guide ................................................................. 74
Fig. 66 CHOOSE guide step two and three ....................................................... 75
# Table of Contents

Chapter 1  Application Area  .................................................................................................................. 2

1.1  Information Management.................................................................................................................. 2

1.1.1  Enterprise Architecture .............................................................................................................. 3

1.1.1.1  Viewpoints ............................................................................................................................ 4

1.1.1.2  Process .................................................................................................................................... 5

1.1.1.3  Language ............................................................................................................................... 6

1.1.1.4  Repository ............................................................................................................................. 6

1.2  Importance of Small and Medium Sized Enterprises ................................................................. 7

1.3  Criteria for Enterprise architecture and Small and Medium Sized Enterprises .......................... 9

1.4  Enterprise Architecture Approach for Small and Medium-Sized Enterprises: CHOOSE ........ 10

1.4.1  CHOOSE Metamodel .................................................................................................................. 11

1.4.1.1  Goal Viewpoint ..................................................................................................................... 12

1.4.1.2  Actor Viewpoint .................................................................................................................... 12

1.4.1.3  Operation Viewpoint ............................................................................................................. 12

1.4.1.4  Object Viewpoint .................................................................................................................. 12

1.4.2  CHOOSE Method ....................................................................................................................... 13

Chapter 2  Problem Statement and Goal of the Research ............................................................... 14

2.1  Specific Research Problem ............................................................................................................. 14

Chapter 3  Related Research ............................................................................................................... 15

3.1  Existing Tool Support ..................................................................................................................... 15

Chapter 4  Solution Approach ........................................................................................................... 17

Chapter 5  Goal-Directed Design ....................................................................................................... 19

5.1  Research ......................................................................................................................................... 20

5.1.1  Literature Review ..................................................................................................................... 20

5.1.2  User Interviews ........................................................................................................................ 22

5.2  Modeling ....................................................................................................................................... 24

5.2.1  Identify Behavioral Variables ................................................................................................. 25

5.2.2  Synthesize Characteristics and Relevant Goals ...................................................................... 25

5.3  Requirements Definition ............................................................................................................... 27
Chapter 6

6.1 Code Quality ................................................................................................................. 48
  6.1.1 Functionality ............................................................................................................. 49
  6.1.2 Reliability .................................................................................................................. 51
    6.1.2.1 Maturity ............................................................................................................... 51
    6.1.2.2 Fault Tolerance ................................................................................................... 53
    6.1.2.3 Recoverability ...................................................................................................... 53
  6.1.3 Efficiency ................................................................................................................... 54
  6.1.4 Maintainability .......................................................................................................... 57
    6.1.4.1 Measure Maintainability ...................................................................................... 58

5.3.1 Data Requirements ..................................................................................................... 27
5.3.2 Functional Requirements ........................................................................................... 27
5.3.3 Other Requirements ................................................................................................... 27
5.4 Interaction Framework .................................................................................................. 28
  5.4.1 Define Form Factor .................................................................................................... 28
    5.4.1.1 Existing Tablet Platforms ....................................................................................... 29
    5.4.1.2 Tablet Operating System Security ......................................................................... 31
  5.4.2 Define Functional and Data Elements ....................................................................... 32
  5.4.3 Define Application Flow ............................................................................................ 33
  5.4.4 Interaction Design Principles .................................................................................... 34
    5.4.4.1 Simplicity, Less is More ....................................................................................... 36
    5.4.4.2 Familiarity ............................................................................................................ 38
    5.4.4.3 Safety .................................................................................................................. 40
    5.4.4.4 Availability ......................................................................................................... 41
    5.4.4.5 Flexibility ............................................................................................................ 42
    5.4.4.6 Beginners, Experts and Intermediates ................................................................... 43
5.5 Design Refinement ......................................................................................................... 44
  5.5.1 Visualization .............................................................................................................. 44
  5.5.2 Attract Attention ....................................................................................................... 45
  5.5.3 Content ..................................................................................................................... 46
  5.5.4 Visual Simplicity ....................................................................................................... 47

Chapter 6 Application Development: Internals ....................................................................... 48
6.1.4.2 Attributes of Maintainability ........................................................................ 59

6.1.5 Portability ........................................................................................................ 60

6.2 Database Design .................................................................................................. 61

6.2.1 From ER Metamodel to Relational Database Model ...................................... 62

Chapter 7 Field Test Feedback ................................................................................ 65

7.1 Feedback on Form and Behavior ........................................................................ 66

7.2 Feedback on CHOOSE Approach ...................................................................... 68

7.2.1 Goal Model ..................................................................................................... 69

7.2.2 Actor Model ................................................................................................... 70

7.2.3 Operation Model ............................................................................................. 70

7.2.4 Object Model .................................................................................................. 71

7.2.5 Links Between Models .................................................................................... 71

7.2.6 Conclusion ..................................................................................................... 72

Chapter 8 Tool Presentation .................................................................................... 73

8.1 Application Overview ........................................................................................ 74

8.2 Requirements Overview .................................................................................... 88

8.2.1 Data Requirements ........................................................................................ 88

8.2.2 Functional Requirements ................................................................................ 88

8.2.3 Other Requirements ....................................................................................... 89
Introduction

A common analogy is often made between enterprise architecture (EA) and architecture in construction. When an architect is asked to design a house, the focus lays on where the staircase should be or how many rooms or balconies there have to be. The specifics of the color or the brand of paint is not of interest yet. The future habitant agrees on some kind of blueprint, a master plan that will serve as the starting point for more detailed decisions. This high level blueprint will show the major functions of the house and how these have to be constructed. As future occupant of the house this high level and abstract representation is probably the most informative. On the other hand, the engineer or constructor is probably more interested in a detailed elaboration.

When companies enlarged and became more complex, the need for EA arose. The advantages of EA for large enterprises have been widely acknowledged in practice and literature. But while small and medium-sized enterprises (SMEs) are considered to be the backbone of our economy, little research has been done on EA approaches for those SMEs. Most research is only targeted towards architecture in large enterprises. A literature research concerning information systems in SMEs, didn’t find any paper on EA in SMEs, in a period from 1979 to 2008.

In response to this lack of research on EA for SMEs, Bernaert and Poels proposed a new EA approach for SMEs. A metamodel was created, focusing on the essential dimensions and characteristics of EA in the context of SMEs. The metamodel is called the CHOOSE metamodel, which is an acronym for "keep Control, by means of a Holistic Overview, based on Objectives and kept Simple, of your Enterprise". Just as tool support is widely considered indispensable in large companies, several case studies quickly identified the same need for it in SMEs. However, as existing tools are already considered low on usability by EA practitioners, a different approach is proposed in this master thesis: a software application for tablets. As SMEs usually rely on insight from the CEO and do not have any enterprise architects on staff, this software application had to be useable for management.

In the first chapter, the application area is discussed. An overview is made of several EA concepts, and an introduction is given about the EA approach CHOOSE. Next, in chapter two, the problem and goal of this research are stated. In chapter three, existing tool support is first considered. In the fourth chapter, a methodology to achieve user-friendly tools is introduced. In this chapter, a software development process proposed by Cooper is presented. That process consists of five steps: initiate, design, build, test and ship. Chapter five to eight will respectively handle these steps, except the initiate step. Methodologies for each of these chapters will be described at the start of each chapter.
Chapter 1  Application Area

1.1 Information Management

In the mid-nineties it was evident that some companies could reap great profits and benefits by using information technology (IT) effectively. Many others failed however, to capture any value from IT and where left with a financial hangover. Broadbent M. studied those market leaders that were able to capitalize on information technology. She argued in her book that information technology is as an investment like any other which could create or destroy business value (Broadbent, 1998). The technicality aside, it should contribute to achieve the desired business objectives. It should be evaluated like any other investment which has to achieve objectives and give a financial return. When IT technologies are seen as a portfolio of investment projects that have to achieve business objectives, well known concepts from traditional business management can be used; e.g. managing project portfolios, creating business value, deciding on capital investments and aligning resources with strategic goals. The large capital investments required for information technology should be enough reason to get management attention. Today, almost every business process is interwoven with information and communication technologies. This makes good management of IT resources even more important. But because of this interconnection with business processes, IT should not be managed as an independent functional area. As discussed above, IT should deliver value and be strategically aligned. Consequently, IT is an area that cannot just be managed in a narrow technology scope. However in practice this is easier said than done, the business–IT relationship is said to be that of a troubled marriage in need of guidance (Ward and Peppard, 1996). Caughlan et al. found that

Fig. 1 The amsterdam information model showcases the different areas of interest in information management research.
miscommunication and even non-communication are the main reasons of the misalignment between IT and business (Coughlan, Lycett and Macredie, 2005). As IT is often the largest capital expense in companies, the need for more research was evident. To decrease the gap between IT and business, an academic research area called “information management” was born. The most famous framework in this research area is the Amsterdam information Model (Fig. 1) (Maes, 1999).

1.1.1 Enterprise Architecture

Enterprise Architecture is an important part of the information management discipline. Its research area contains the middle row in the Amsterdam Information Model framework. It describes the crucial link between the strategy and the operations of a company (Red area in Fig. 1). Just like a blueprint from an architect offers an overview of a house, enterprise architecture (EA) should offer a holistic overview of the firm. Companies have become bigger and more complex over the years. Consequently, managers from different departments will optimize their department in a way they perceive to be optimal. One manager may set up a system that is very cost efficient but inflexible, while another might feel that cost efficiency should be sacrificed for more flexibility. Without a general overview and a vision with common goals, it is impossible to manage a company properly. According to Jonkers et al., this provision of a holistic overview is the most important benefit of EA (Jonkers et al., 2006). EA offers other advantages too. EA can enforce and support necessary changes in the organization, it can align IT with corporate strategy, and hence IT can be used in a more flexible and efficient way (Lindström et al., 2006, Daneva and van Eck, 2007, Radeke, 2011, Tamm, Seddon, Shanks and Reynolds, 2011) and finally it deploys a corporate strategy better suited to the business reality (Veasey, 2001, Hoogervorst, 2004). Jonkers et al. furthermore provide the following definition of EA.

“EA is a coherent whole of principles, methods and models that are used in the design and realization of an enterprise’s organizational structure, business processes, information systems, and infrastructure.”
According to Jonkers et al., all of these principles, methods and models can be classified into four different ‘ingredients’ (Jonkers, Proper and Turner, 2009). The four ingredients are: a process, a viewpoint, a modeling language and a repository to make the EA approach complete. These are represented as a bow tie in Fig. 2. EA approaches often don’t cover all of these components, or when they do, they often put focus on a certain aspect. With the help of this bow tie framework, EA approaches that are strong in one aspect can be combined with other EA approaches that are strong in other aspects. While presenting the different ingredients, the best known EA approach component for that area will be briefly introduced.

1.1.1.1 **Viewpoints**

As we described above, different stakeholders have different objectives. All of these interests have to be aligned to have an efficiently run company. Modeling the enterprise in one big model, taking into account all of the perspectives of the company, would be too complex. The company is represented by different viewpoints that each covers the perspective of a certain stakeholder. Each stakeholder should be positioned in the bigger picture, though, to achieve a holistic overview. The most famous framework is the Zachman framework (Zachman, 1987). Zachman tried to capture most viewpoints with a two dimensional classification matrix (see Fig. 3). On the X-axis, rudimentary questions are asked to cover the subject focus: who, what, where, how, when and why. On the Y-axis, the participants of the EA design process are represented: planner, owner, designer, builder, programmer and user. Note the similarities with architecture in construction. In this 6X6 matrix, 36 viewpoints are covered. The 2-dimensional structure makes the framework easy to read. For example, if the ‘Owner’ is interested in ‘How’ something works, he will probably be most interested in a business process model.

![Diagram of bow tie framework](image-url)
For an inexperienced cook, knowing the ingredients of a dish is not enough to serve a decent meal. He will probably need some instructions, some kind of process to make the dish. In the same way, EA approaches need a process (method) to create and manage the architecture. The best known process is part of The Open Group Architecture Framework (TOGAF) (Josey, 2009). Contrary to Zachman, who only offers a framework for viewpoints, TOGAF includes all 4 ingredients of an EA approach. The best known and most used ingredient in TOGAF is the process ingredient, though. More specifically this process is called the Architecture Development Method (ADM). The ADM consists of a continuous cycle of 8 steps (see Fig. 4). By continuously going through the cycle, an EA can be created and maintained. In the first cycle, the different models, and other architectural artifacts are created. In subsequent cycles, models can be refined or adjusted. Note that this is a continuous cycle.
1.1.1.3 Language

To create the different models to describe the EA, a certain language is needed. As natural language would be difficult to interpret, graphical representations are used. These ‘languages’ are mostly standardized. Consequently, tools can be made for certain languages to make development or analysis of the models easier. Additionally, a standardized language allows for unambiguous communication of the models. Throughout the years, many languages have been proposed to model the different viewpoints. For example, for a business process model, the business process modeling notation (BPMN) has become a standardized language (White, 2004). More recently, efforts have been made to make one language that can cover every viewpoint. In that way, it is much easier to link the different viewpoints and end up with one integrated model of the company. An example is the language called Archimate (Iacob et al., 2012).

1.1.1.4 Repository

Finally, the last ingredient of an EA approach is a repository. This is basically a central storage of everything that is related to EA. Compare it with a cook book where ingredients, instructions and pictures of the resulting dishes are stored.
1.2 Importance of Small and Medium Sized Enterprises

Small and medium sized enterprises are defined in the EU as enterprises with fewer than 250 persons employed, provided that they don’t have an annual turnover of more than 50 million euro or a balance sheet that exceeds 43 million euro (European Commission, 2003). In 2003 a third class named micro enterprises was introduced. SMEs are now broken down into three main categories: micro, small and medium-sized. Their classification criteria can be found in Fig. 5.

![Classification criteria](image)

Micro, small and medium-sized enterprises are often referred to as the backbone of the European economy. There are close to 20.8 million SMEs in Europe, which account for 99.8% of all companies. Around 70% of European jobs are provided by SMEs. In the private sector they account for 58.4% of total gross-value added production. (European Commission, 2010)

Different governments recognized the central role of SMEs in economic recovery after recession hit in 2008. In the EU, the small business act was adopted while in the US the small business job act was introduced. The small business act aims to increase competitiveness through the following measures (European Commission, 2008):

- Make the administrative burden lighter.
- Facilitate access to finance.
- Help SMEs expand their business in foreign countries.

Although these actions have made life easier for SMEs, only 50% of all start-ups survive for more than 5 years. And a mere 33% lasts for more than 10 year. In 2009 it was estimated that 1.7 million jobs were lost because of bankruptcy, mostly due to insolvency (European Commission, 2007). This number shows it is of the utmost importance for the stability of the economy to prevent these bankruptcies from happening.
Lybaert found in his research that owners/managers in an SME who have more strategic awareness, employ more information in their decision making process. And by using more information, companies seem to get better results (Lybaert, 1998). Hannon and Atherton came to the same conclusion. They also found that there is a strong positive correlation between strategic awareness and success (Hannon and Atherton, 1998). As described in 1.1.1 (supra, p.3), EA can offer a holistic overview of a company. Done well, this holistic overview is a great source of information, which leads to a higher success rate in SMEs.

According to Smith and O’Regan et al., SMEs who included a strategy in their business plan rather than just drawing up a financial plan, seem to perform better (Smith, 1998, O’Regan and Ghobadian, 2004). However, if the strategy is not deployed well throughout the organization, even a great strategy will not deliver any benefits. A link is often missing between strategy and operations. EA can offer this crucial link between strategy and operations.

Another major problem in SMEs is growth. When they start growing, they face many constraints. Insolvency is the first struggle for growing companies and often cause them to go bankrupt (European Commission, 2007). Also in this case, research found that EA could aid SMEs in periods of growth (Jacobs, Kotzé, van der Merwe and Gerber, 2011).

In literature, there seems to be ample research about the advantages of EA, even in the context of SMEs. Furthermore a lot of research about EA approaches, as described in 1.1.1 (supra, p.3), has been done. Landenberg and Wegmann investigated which aspects of EA are being researched. One of their conclusions was that interest in the field of EA is currently growing, given the increase in publications (Langenberg and Wegmann, 2004). Furthermore, Ernst et al. found in their survey that, also in practice, EA is of growing importance (Ernst, Lankes, Schweda and Wittenburg, 2006). However most of this research is oriented at architecture in large enterprises. Unfortunately, due to some inherent differences between SMEs and large enterprises, research results from these large enterprises cannot be generalized and transferred to SMEs (Bernaert, Geert, Snoeck and De Backer, 2013). Advantages of EA in practice were apparent in large enterprises (supra, p.3). As described above, EA could have great advantages in SMEs as well. But unlike large enterprises, SMEs hardly ever heard of EA, let alone used EA (Wißotzki and Sonnenberger, 2012). So despite the apparent advantages, it was quite surprising that it is not used in practice and even more surprising, that little research has been done in the academic world. In a literature research done by Devos, concerning information systems (IS) in SMEs, a summary of the EA literature was given (Devos, 2011). Not one paper was found from 1979 to 2008 about EA in SMEs. It was only recently that efforts have started to develop an EA approach for SMEs (Aarabi et al., 2011, Jacobs, Kotzé, Merwe and Gerber, 2011, Bidan, Rowe and Truex, 2012, Bernaert et al., 2013).
1.3 Criteria for Enterprise architecture and Small and Medium Sized Enterprises

In the remainder of chapter one, the EA approach by Bernaert et al., called CHOOSE, will be introduced. Before the development of this new approach, research about the characteristics of EA and SMEs was done. In this extensive literature study, these characteristics were summarized in several criteria. (Bernaert et al., 2013). These criteria were used as a starting point for the development of the CHOOSE metamodel. This model will be discussed in the next section. Five criteria about EA and 6 criteria about SMEs were derived and integrated (Fig. 6). The five criteria of EA are listed below and the six SME characteristics are integrated in the 4th criteria about EA.

**Evaluation model EA**

![Evaluation model EA diagram](image)

Fig. 6 Criteria derived for CHOOSE

1. **Control**: As enterprises are becoming more complex, EA has to aid in controlling that complexity.
2. **Holistic Overview**: As noted above, Jonkers described this criterion as the most important one. EA has to capture the essentials of the enterprise.
3. **Objectives**: As EA is the middle row in the Amsterdam Information Model (Fig. 1), it has to fulfill the link between strategy and operations.
4. **Understandable for its target audience (SMEs)**. As discussed by Bernaert et al., SMEs have some inherently different characteristics compared to large enterprises. The EA approach has to take these characteristics into account.
   4.1. **Allow working in a time efficient manner on strategic issues.**
   4.2. **A person with limited IT skills should be able to apply the approach**
   4.3. **It should be possible to apply the approach with little assistance of external experts.**
   4.4. **The approach should enable making descriptions of how things are done in the company**
   4.5. **The CEO must be involved in the approach**
4.6. The expected revenues of the approach must exceed the expected costs and risks.

5. Enterprise wide. This should enable the optimization of the company as a whole instead of doing local optimization within individual domains.

1.4 Enterprise Architecture Approach for Small and Medium-Sized Enterprises: CHOOSE

After the above criteria were derived, development of the EA approach could start. A complete EA approach contains the four ingredients that were discussed in section 1.1.1 (supra, p.3). However as this is just an introduction to CHOOSE, only the viewpoint and method ingredients will be discussed. Bernaert et al. started with deducing the viewpoints and creating a metamodel. The metamodel is part of the viewpoint ingredient. It offers a more detailed elaboration on the different viewpoints and links them.

In the Zachman framework, that was discussed in 1.1.1.1 (supra, p.4), two essential dimensions were used. The subject focus on the X-axis and the participants of the EA design on the Y-axis (Fig. 3).

![Fig. 7 CHOOSE subject focus](image)

Bernaert et al take over these two dimensions and propose a simplified version to fit the SME context. The subject focus consists of four dimensions, why (goal dimension), who (actor dimension), how (operation dimension) and what (object dimension). Leaving out the when and where dimensions (see Fig. 7).
The participant dimension, the Y-axis in the model of Zachman, in essence consist of just three layers: a business layer, an information layer and a technology layer. Traditionally, EA was an answer to the growing complexity of IT in companies. It helped companies to manage their complex IT systems and IT infrastructure in a holistic way. The technology layer was perceived to be the most important one (Fig. 8 left hand side). Recently, more and more frameworks start to stress the importance of the business layer where the motivational dimension (why) is suggested to be the most important component (Fig. 8 right hand side) (Braun and Winter, 2005, Van’t Wout, 2010). In the CHOOSE approach, the business layer is stressed and is built around the 4 dimensions (why, how, what, who) as shown in Fig. 7. An inverse pyramid is suggested in Fig. 8 at the right hand side. It clearly shows the importance of the why dimension within the business layer and the business layer as a whole.

1.4.1 CHOOSE Metamodel

Below, in Fig. 9, the complete metamodel is shown. In the next sections we will go into more detail about the most important components.
1.4.1.1  **Goal Viewpoint**

In an EA approach where the business layer is central, the ‘why’ is the most important dimension. An essential element here is to construct a goal tree. The AND/OR refinement allows to construct different possibilities to achieve an upper goal. For example, the goal ‘saving money’ can be achieved by ‘lowering wages’ AND by ‘buying cheaper raw materials’ OR the second possibility to achieve the goal ‘saving money’ is to ‘increase efficiency’ AND to ‘increase employee productivity’. The AND/OR refinements allow to construct these different possibilities. The conflict refinement allows for linking goals that may be contradictory. The goal ‘saving money’ may be in conflict with the goal ‘increase employee satisfaction’.

A goal can be linked with the actor model (who dimension) through a wish or assignment relationship. An owner may wish to fulfill the goal ‘make more profit’ but the CEO may be assigned to that goal. Furthermore a link with the operations model (how dimension) can be made. A certain operation has to be performed to realize a low level goal. If that goal isn’t operationalized by any operation, then it will be difficult to achieve it. Finally a goal can be linked with certain objects (what dimension). For example the goal ‘reduce time by reorganizing the layout of the plant’ will be concerned with the object ‘factory layout blueprint’.

1.4.1.2  **Actor Viewpoint**

The most obvious part of the actor model is people (human actors) and it is straightforward to construct an actor tree including every person from the organization. Actors don’t have to be human, though. Actors can have different specializations and can be defined as a Role, a Software Actor or a Device. For example, if a person is replaced by a robot, the actor is no longer a human actor but is now a device. That device then actively performs a certain task as an actor. An actor can be linked to the goal model, as described above. Additionally, an actor can be linked with an operation through a RACI relationship. The actor can be Responsible, Accountable, Consulted or Informed over a certain operation. Finally an actor can also monitor or control a certain object.

1.4.1.3  **Operation Viewpoint**

The operation model consists of the different processes or projects in a company. These processes or projects may further be build up from sub processes or sub projects. The link with the goals and actor model are described above. An operation needs certain input, a machine for example needs certain raw materials to work. When there are inputs, mostly there is some output too. These are the two relations (input/output) that an operation can have with an object.

1.4.1.4  **Object Viewpoint**

An object is an item that does not actively performs anything. It is something passive by nature. It may be a delivery note, raw materials, a website etc. Objects may have links with other objects. For
example, a tire is part of a car (specialization relation). But more important are the links with the other dimensions of the models. These have been discussed above.

1.4.2 CHOOSE Method

The second ingredient that will be introduced is the CHOOSE method. During case studies, a six step process was derived. The process is an essential ingredient in the bow tie model (supra, p.5). The six steps are:

1) In the goal dimension, the CEO is asked to define the companies’ highest level goals. This step can be aided by thinking about the four dimensions in the balanced scorecard (Kaplan, Norton, Dorf and Raitanen, 1996).

2) After defining the highest level goals, a goals tree can be constructed by asking ‘Why/How?’ questions.

3) Human actors and roles can be added in the actor dimension. These can be found by interviews and other secondary sources like organograms.

4) Operations are added to the operations model. This step is aided by thinking about the different dimensions found in porter’s Value Chain (Porter and Millar, 1985).

5) Object and relations between models are added.

6) The final step is a validation round.

Furthermore, for each dimension certain stop criteria are suggested.

1) Goal dimension: Is every lowest level goal operationalized by a process? If not, it is either an alternative (goal OR refinement) or there is a misfit between operations and strategy.

2) Actor dimension: Is every actor at least linked to one operation? Furthermore, is everyone from the company included in the model?

3) Operation dimension: Is each operation assigned to at least one actor and contributing to the fulfillment of a goal?

4) Object dimension: Does every operation has its appropriate input and output specified in the object model?
Chapter 2  Problem Statement and Goal of the Research

2.1 Specific Research Problem

While developing the EA approach CHOOSE, case studies were performed in SMEs. Even after the first tests, it became evident that tool support to implement the CHOOSE model was indispensable. Working on a blackboard with post-its was not practical (see Fig. 10). First of all, input was very slow, having to write every element on a separate post-it. If a goal had to be corrected, a new post-it had to be made. Once the model was finished, it was impractical to store and save three blackboards and present it on a meeting the week after. More importantly, finding an element could take a while and making any sort of analysis was out of the question. Additionally, tool support is part of the repository ingredient and consequently is an essential part of an EA approach (supra, p.6). Academic sources also widely support the necessity of software tool support.

“For EAs to be useful and provide business value, their development, maintenance, and implementation should be managed effectively and supported by tools.” (Schekkerman, 2009)

“EA management should be supported by tools, which support distributed access to consistent data, offer the possibility to structure the information managed, and also aid users in filling out their role in the EA management process.” (Ernst et al., 2006)

The objective of this master thesis is to design and build a tool, suitable to support SMEs in developing EA, based on the EA approach CHOOSE. This part was already described in the paper ‘Development of Software Tool Support for Enterprise Architecture in Small and Medium-Sized Enterprises’, which was accepted by BUSITAL to be published in their conference proceedings of 2013 (Dumeez, Bernaert and Poels, 2013). Furthermore a second objective is to try out the tool in a real case, which allows to evaluate the tool and the EA approach in an SME. First conclusions will be presented in this master thesis.

Fig. 10 Need for tool support during case studies
Chapter 3 Related Research

3.1 Existing Tool Support

Kurpjuweit and Winter identify three different objectives for which tools can be used. They can be used to document and communicate, they can be used to analyze and explain or they can be used to design the EA (Kurpjuweit and Winter, 2007). Typically, after selecting an EA approach and a preferred modeling language, there are software tools available to model your enterprise. There are several options on the market, for example IBM’s system architect (IBM) or Metis from Troux technologies (Troux). However most of these tools are focused on the design aspect of EA and are solely used to model the enterprise.

Based on a survey performed by the University of Munich, Ernst et al. summarize how different tools perform according to different criteria and different scenarios (Ernst et al., 2006). They came up with the following kiviat diagram, showing the minimum, maximum, and median score obtained by the different tools (see Fig. 11). We can see that even for EA practitioners, the average usability of the different tools scores rather low. One reason was that all tools came with predefined metamodels. Some came with up to 400 entities with corresponding associations. Even for daily practitioners, this is perceived to be very complex, making the tools hard to work with in practice. Especially the visualization needed improvement. Two major issues occur when visualizing an EA with existing tools. First, the automated generation of adequate visualization is usually not possible. Second, the semantics of those visualizations are often not properly defined. This leaves a lot of room for interpretation. These issues are both explained by the simple fact that most tools are drawing tools, where visualizations are manually created. This often causes practitioners to use certain symbols or links out of context. Although the model is understandable for themselves, the tool can no longer interpret the results correctly. The result is a drawing rather than a model.

Some tool developers claim to deliver an EA solution for SMEs, but offer nothing more than a lightweight ERP system (Business Management Division Sage Software, 2008). Not one tool for EA was found that could be used in an SME context or that was designed from the ground up to be used in that context.
The graphical user interface of a typical EA tool is usually composed of three different components: the content explorer, a canvas to do the modeling itself and a concept explorer with different modeling language constructs (see Fig. 12). Often this works by dragging and dropping the different constructs from the concept explorer to the canvas and thus slowly building up the different models.
Chapter 4  Solution Approach

While it has been made clear that tool support is needed, one of the key words in the abbreviation CHOOSE is ‘Simple’. Given the resource poverty of SMEs (Beck and Demirguc-Kunt, 2006), they are unable to staff an enterprise architect, or pay expensive consulting fees (Hankinson, Bartlett and Ducheneaut, 1997). As described above, modeling tools are typically aimed towards enterprise architects. So while developing a tool for this new approach, it was clear that contrary to the above tools, a different approach would be required for small companies. Even practitioners in big companies rated the existing tools rather poor in usability. Furthermore these tools expect the users to be fully aware of the models and modeling languages.

In the past, software was developed completely by one programmer. He would be doing the coding and testing, and eventually spread the software to some tech enthusiast or other programmers. When software businesses started appearing, a professional manager was appointed. He was responsible for translating market opportunities into product requirements. As the market further grew and became more mature, testing became a big part of the development process. Later on, when graphical user interface (GUI) design became more important, designers were brought in at the end to make buttons and other elements look nicer. However, how can a good product be created when it does not take into account the users’ goals, needs and motivations?

To find a proper approach towards tool development, a methodology aiming to achieve user-friendly interfaces was adopted from Cooper et al (Cooper, Reimann and Cronin, 2012). He was one of the first to criticize traditional software development processes and was the first to pioneer a software
development process that was based on the users’ needs. The improved software development process that he proposed now emphasizes the design aspect (Fig. 14).

The first step, called ‘initiate’, is the step where the need for a certain program is identified. The need for tool support has already been discussed as being necessary to support an EA approach. The necessity of this new EA approach has also been shortly dealt with in introduction. The design of the application will be discussed in the next chapter. The design step is followed by a more technical chapter explaining some of the decisions that were made on a code level. Next, a test of the application and EA approach are done through a case study. Finally, we end with a presentation of the tool.

Fig. 14 Software development process proposed by Cooper
Chapter 5  Goal-Directed Design

The design step in Fig. 15 should not just be about the appearance of a product. When properly deployed, it should identify user requirements and take into account his behavior. Design should be about product definition, based on the goals of users, needs of the business and the constraints of technology. Design is defined in a broader sense.

Like an analogy was made with architecture when arguing that EA is needed, a similar reasoning can be made here. In the case of an architect, he will have to understand how the people live and work before he can start designing a house. Then, he should sketch the spaces to support and facilitate those behaviors. For the architect, designing a house is more than making it look pretty.

This broader approach towards design is called ‘interaction design’. Interaction design is more than making sure something looks nice. Interaction design is about understanding the users and knowing cognitive principles.

An approach to implement this design philosophy is proposed by Cooper (Cooper et al., 2012). In short, this approach is called ‘goal-directed design’. The main steps followed in a goal-directed design approach are shown in Fig. 16. We will discuss the first five steps: research, modeling, requirements definition, design framework and design refinements.
5.1 Research

The research phase mainly consists of understanding the future user and knowing who that future user will be. Contrary to other parts of science, social scientists soon realized the value of qualitative research. Human behavior is often too complex to capture in hard data. In general, the advantage of qualitative research is that social phenomena can be researched in their natural setting rather than an experimental one (Pope and Mays, 1995). In a design setting, Cooper et al (Cooper et al., 2012) found that qualitative methods were cheaper, faster, and provided better answers to relevant questions like: ‘How does a product fit into people’s lives?’, ‘What goals motivate people to use the product and what tasks helps them achieve this?’, ‘What are the problems with the current way of working?’. Many different types of qualitative analysis have been proposed by practitioners in these social sciences. Design and usability practitioners have used many of the same techniques. The types of qualitative research that are being used are literature review, user interviews and a subject matter expert. The findings from the subject matter expert will be presented in Chapter 7 (infra, p.65). The other research methods will be discussed next.

5.1.1 Literature Review

In the research done to develop the CHOOSE approach, it was already clear that the CEO would have to be closely involved. There are two reasons illustrating the need for the CEO’s involvement. Firstly, in smaller companies, employees generally don’t know the structure, let alone know why they are doing something. It is only the entrepreneur who knows the whole working of the company. This distinguishes SMEs from larger companies, because a CEO can have an overview of its SME. Secondly, in SMEs, the job description for employees is often vaguely defined and an overview of tasks or responsibilities is missing. (Kamsties, Hörmann and Schlich, 1998)
In a study performed by Hankinson (Hankinson et al., 1997) which involved 11 countries, owner-managers from almost 800 SMEs were questioned. Owner-managers responded that they worked between 50 and 60 hours a week. A lot of time was spent on the phone, almost 40% of total time. Most time was spent in meetings and informal interactions (50%). The remainder was spent travelling. Only 5 hours or less than 10% of their time was used on strategic issues, personal reflection, and analyzing results. Notably, managers generally indicated that their time management was poor. This also meant that little time was spent with family, although most claimed to have stable happy family lives. When managers were asked what their motivation was to have their own company, social status and remuneration were said to be important. But even more important was that they were independent. This was rated as being vital. Power, contributing to the economy or creating jobs were not seen as an aim, but rather as a consequence of striving towards the first motivational factors. Very little managers were prepared to trade their current job for a management job at a big company when they would receive the same wage and have fewer responsibilities. The job security and fewer responsibilities could not weigh up against the independence. Contrary to popular believe, money is not the main objective. Another important finding was, that the respondents found their companies to be big enough, but not profitable enough. The average age in the survey was 53 years and the respondents were mostly male. Education wise, there was a wide variety of different diplomas and backgrounds. The typical owner-manager thus doesn’t necessarily has a university degree. Skill wise, marketing, finance and computing were indicated to be a weakness. Their strengths came from experience, enthusiasm and their hard work. It’s remarkable that there was little interest in further education to improve their weaknesses. They reasoned that their strong points could compensate or that they simply did not have any time for it. Similar findings were found in a follow up research, three years later (Hankinson, 2000).

Research done by Vankatesh and Brown identified that managers in SMEs generally have limited IT knowledge. In larger companies, the same is true, but they have internal experts who play a critical role in the adoption of IT. Unfortunately, SMEs seldom have internal IT experts and the decisions regarding IT often are up to management. In the introduction from Chapter 4, we already mentioned that hiring experts is difficult for SMEs because of financial limitations (Beck and Demirguc-Kunt, 2006). Furthermore the survey above found that almost no SMEs had ever hired any consultants or experts. Therefore, the limited knowledge of IT can form a barrier for effective IT implementation in SMEs (Palvia and Chervany, 1995, Venkatesh and Brown, 2001). Unfortunately, little efforts are made within SMEs to set up formal training programs for employees and managers.

When developing the CHOOSE approach, a more thorough literature study has been conducted by Bernaert M. His findings are summarized in six criteria as introduced before (supra, p.9). These will serve as additional input for our persona definition in the next section (infra, p.24).
5.1.2 User Interviews

Usually after a software project is initiated, stakeholders already have some kind of vision about what they expect from it. Managers, customers and employers may have completely different visions on what the application has to look like. It is important to gather feedback from all stakeholders and harmonize the different perspectives. However, in our case, managers generally have never heard of EA. Asking them what they expect from an EA application will be to no avail. This is why instead, a contextual inquiry has been done, pioneered by Beyer and Holtzblatt. A contextual inquiry is a customer focused process of finding out who the customer really is and how they work on a day to day basis. This process is based on a master-apprentice model. The interviewee is the master while the interviewer is the apprentice. By joining the master for a day, asking him questions on how to learn the craft, a lot of valuable information can be obtained (Beyer and Holtzblatt, 1997). Unfortunately, this technique is only possible when the project has been initiated within the own company. Therefore, we use certain principles from the technique. To find behavioral patterns, Holtzblatt and Beyer propose to ask the following questions: goal-oriented questions, system-oriented questions, workflow-oriented questions and attitude-oriented questions.

An interview has been conducted with Jan Provoost, the CEO of Stanwick. Stanwick is a consulting firm with approximately 30 employees, active in three countries. The different questions, suggested by Beyer and Holtzblatt, were asked. These are the findings:

Goal-oriented findings:
Contrary to findings in literature, the CEO of Stanwick had a really clear vision of what the priorities were in the company. When asked to tell me what a ‘good year’ meant for him, he was able to give very concrete goals with accompanying metrics. Stanwick even has systems in place to gather sufficient information to decide upon these strategic goals. By asking him to explain these techniques more in detail, I found out they had a complete method to perform strategy in their SME. Therefore, a second interview was conducted with the same manager in a later stage, where more information about their method was obtained and valuable feedback was given about the CHOOSE approach. A detailed elaboration on this interview can be found in Chapter 7. The manager is then consulted as a subject matter expert.

System-oriented findings:
When asking about how he felt about technology in general, he was quite skeptical. Even during their consulting jobs, they often prefer more traditional robust techniques, rather than relying on what he calls ‘technology that never works’. He only accepts technologies that offer real benefits, like a smartphone for example. This device offers great benefits: have your agenda with you at all times and receive e-mails instantly everywhere. As a consultant, during projects in other companies, the only
time he could check his e-mail was often when he finally got home. For the same reason, recently a tablet was purchased. Answering e-mails from a couch is more relaxing then doing it in your home office. Although his smartphone was great to read the e-mails during the day, he did not find it really practical to write e-mails on his smartphone. Furthermore, as he travels a lot, bringing along his tablet to check his e-mails and the news in his hotel room was a lot easier than having to take his heavy laptop with him. That being said, he only uses his smartphone and tablet for checking e-mails or as an agenda.

He has never visited the application store and never installed an application. When asked about the technology policies in the company, he told that everyone received a smartphone and a laptop. But only the laptops were integrated on their company network. Furthermore, laptops were completely locked so no one could install software of their own. He explained they used to have one IT guy on their payroll but after repeated problems with security they outsourced it to a third party. Now that everything works, they feel reluctant to make any major changes to their network, or allow new devices to be connected to it. These findings confirm literature findings that managers are generally not technology savvy.

**Workflow-oriented findings:**
When asked about what his typical day looked like, he answered that no such thing exists. Because he combines his job as CEO with regular consulting projects, his weeks can be very hectic. When I looked at his agenda for the week, it was jam packed. After our interview was finished around lunch time, he still had two meetings in two different places to attend. Furthermore he admitted that he also often worked during the weekend. And when he didn’t work, he often thought about his job. Contrary to findings from literature, they did have formal meetings about strategic issues.

**Attitude-oriented findings:**
When asked where he saw himself in 5 years, he came back to the topic about technology. He said everything technology wise is evolving very rapidly. He recently started reading about ‘cloud computing’. But trying to follow the IT trends is very difficult without an IT background. Despite that, he knows that using more technology will probably be inevitable. The speed of transformation in every domain is the most daunting.
5.2 Modeling

Designing the application for just one person would not be very effective when this application is to be used by many. Even worse is designing an application without a user in mind. In the modeling step, a persona is defined. A persona is the portrait of what the ideal user would be. It is a technique to gather all the information that was found in the research step and add it together in one model. If you try to design something that pleases everyone, you will likely end up with a features packed application. As discussed later, features packed applications please no one in the end. By trying to satisfy everyone, no one is truly satisfied. This is why it is better to make the application for a specific type of person with specific needs. By completely defining a ‘persona’, programmers, designers and managers always keep focus on who they are making the program for. Usually, programmers talk about ‘the end user’. By making the end user an actual person, it restricts them of stretching the ‘end user’ to fit the situation. ‘This is easy’ might be true for the programmers, but may not be the case for the defined persona, and obliges programmers to rethink their approach to fit the persona’s needs. Contrary to task-analysis-based approaches, by defining a persona and defining his/her goals, the end-state is always taken into account. Tasks on the other hand are needed to achieve a goal. Activity centered design approaches successfully address the “what” of user behavior but neglect the why. Another advantage of taking into account the users’ goals rather than activities or tasks, is that goals are more stable over time. The activities and tasks to achieve a goal are often very dependent on the technology at hand, making them fluctuate heavily over time.

Because a complete persona requires much more research, a provisional persona is proposed. A provisional persona is very similar to a real persona, but the designer needs to make more assumptions based on existing research and best guesses. This is not ideal, but provisional personas offer better results than using no model at all.

To construct a persona, we follow several steps proposed by Goodwin et al. (Calde, Goodwin and Reimann, 2002). Since we are constructing a provisional persona, only two steps will be described: identify behavioral variables and synthesize characteristics and relevant goals.
5.2.1 Identify Behavioral Variables

In the first step, a list of behavioral variables is made. Normally, variables would be observed during the one day master-apprentice exercises and augmented through other research. Here we will use research from literature and the interview to deduct a list of variables. Goodwin et al. propose to focus on 5 variables that describe behavioral patterns.

**Activities:** We have seen both in research and in our interview that managers from SMEs were busy with day to day activities. Furthermore these activities are very diverse and differ widely every day. Additionally, managers generally have jam packed schedules with little time in between.

**Attitudes:** Attitude towards technology is rather skeptical. In general, managers aren’t the most technology savvy people. Only if technology can show real added value, managers will take the plunge.

**Aptitudes:** Contrary to managers in larger companies, owner-managers in SMEs can have any type of education and aren’t necessarily ‘smart’. Willingness to learn is rather low, as most consider their time to be too valuable.

**Motivations:** Often, managers engage in the IT domain because it is necessary, not because they like it. More specific engagement in the EA domain is non-existent since few managers in SMEs have ever heard of the term.

**Skills:** Managers tend to have more knowledge about general management techniques. The terminology in the EA domain should not be too hard to grasp. Unfortunately skills in the IT area are generally poor.

5.2.2 Synthesize Characteristics and Relevant Goals

In ‘emotional design’, Norman described three levels of cognitive processing and its importance for design. These three levels are called: visceral, reflective and behavioral cognitive processing (Norman, 2007). When designing for a visceral response, design tries to excite involvement from what the initial senses perceive. Or easier said, we want to make user feel good by only looking at our design. But, making something beautiful does not imply that it is useful. Be that as it may, research has shown that something beautiful can be perceived as being more useful. People will tend to rate the perceived ease of use to be high, regardless of the actual usability. Users will invest more time in learning a program, when they are presented with a beautiful design (Dillon, 2002).

When designing for reflective processing, the aim is to induce a long term relationship with the product. Unfortunately this is impossible to describe in design guidelines.

When designing on a behavioral level, the behavior of the application should complement a user’s own behavior. A mental model is an important concept in this area and will be further discussed in section 5.4.4 (infra, p.34). This level can be considered to be the most important one. Designing for a behavioral level influences the actual usability. In this regard, visceral and reflective processing have a
supporting role, because they can increase the actual usability by increasing the user’s perceived usability.

An example to clarify the difference between the three levels is the Apple iPod. Although there were mp3 players that had higher actual usability, the combination of the beautiful design and high actual usability made it a winner combination. This is reinforced by the strong connection people have with their music collection, which has a lot of reflective potential. A combination of these three levels made the Apple iPod the iconic product it has become.

In ‘Perfecting your personas’, Goodwin K. describes the importance of adding user’s goals to a persona. He proposes three types of user goals, which correspond to the three processing levels, described above. Visceral, behavioral and reflective cognitive processing levels are respectively translated to experience goals, end goals and life goals (Goodwin, 2001).

**Experience goals:** Experience goals describe how someone wants to feel when using a product. These goals are often quite universal for many people. One of the most important goals is that users don’t want to feel stupid. People want to feel in control of what they are doing. This is especially important for managers who don’t have much affinity with technology. They would quickly resent the product if it makes them feel stupid.

**End goals:** The end goals describe what the user’s motivations are for performing tasks with the product. In our case, it is difficult to answer these questions. As described above, it was not possible to ask managers what they expect from an EA application since they never heard of the concept. However criteria from the EA domain were derived, that describe certain of these end goals. First, the application has to offer a holistic overview of the company. It needs to assist in thinking strategically. Next, the application has to aid in linking the strategy with operations. And finally, managers need to get information of how things are done in the company. However, future research will have to show what managers will and will not value. In section 7.2 (infra, p.68), other possible end goals (described as output there), are proposed.

**Life goals:** Life goals are beyond the context of the application and describe the ‘why’ someone wants to achieve the end goals. Research has shown that social status, high remuneration and most importantly independence were found to be important life goals of owner-managers.

When constructing a complete persona, an important step is to draft a third person narrative. By making a story around your persona, that persona is even more ‘personified’. People involved in making the application will now have a very specific idea of who they are designing for. Identifying behavioral variables and synthesizing characteristics and relevant goals are only two of multiple steps to construct a persona. But we feel that these two steps already give a good idea of who the application has to be designed for.
5.3 Requirements Definition

After the research and persona definition, we can start extracting the personas' needs or requirements. These requirements can be thought of as consisting of objects, actions and contexts. For example for a calling application, this might be something like: ‘Call (action) a person (objects) directly from an appointment (context)’. As it can be quite difficult to extract information in this format, it is easier to separate them into data, functional and contextual requirements. (Shneiderman and Ben, 1998)

5.3.1 Data Requirements

These requirements are the objects and information that our users want from our tool. We discussed in the persona’s end goals that these data requirements are not yet complete and further research is needed. What we did find, was that the application will have to:
- Contain all relevant information about actors/goals/operations/objects;
- Contain the links between the different models;
- Give strategically important output;
- A holistic overview of the company;
- A description of how something is done.

5.3.2 Functional Requirements

The functional requirements are all of the actions that have to be possible with the application. It has to be possible to:
- Add, remove and modify goals/actors/operations/objects;
- Navigate easily within and between the different models;
- Generate automatic visualizations;
- Output data in popular formats like PowerPoint/Excel/Word.

5.3.3 Other Requirements

Business Requirements:
- It has to be cheap to implement. (Beck and Demirguc-Kunt, 2006)

Experience Requirements:
Since a beautiful design can greatly enhance the perceived ease of use, a beautiful design is important. Furthermore, the application has to be easy to understand and easy and fun to use. We want managers to feel in control and don’t feel stupid.

**Technical Requirements:**
- It has to work on a platform that can complement a manager’s usage pattern.
- As the CHOOSE metamodel is still being refined, the software architecture should be flexible to allow changes to be adapted quickly.
- Finally, it has to be a safe solution that doesn’t need integration in existing infrastructure.

### 5.4 Interaction Framework

![Goal-directed design process](image)

Fig. 20 Fourth step in the goal-directed design process: Framework definition

Considering the issue of usability with existing tools for EA practitioners, making the tool easy to use for non-technology savvy managers would be challenging. Going back to the roots of usability, Vitruvius was the first student of ergonomics and usability (Colborne, 2010). His questions were based around three key words:
- **Firmitas:** How durable is the design and does it have strength?
- **Utilitas:** How useful is the design and does it fit the user’s needs?
- **Venustas:** How beautiful is the design?

Note the similarities with the three levels of cognitive processing, proposed by Norman in section 5.2.2 (supra, p.25). The proposed interaction framework, will try to achieve utilitas or how to make a useful design that fits the user’s needs. It defines the high level structure of the application. This includes the form factor, the functional elements that have to be presented on the screen and the structure of the application. Throughout this section we will use screen shots of the actual application to show how certain design guidelines have been used. Note that these screenshots will be used to illustrate the design choices, not showcase the functionalities.

#### 5.4.1 Define Form Factor

Ever since the first laptops were available for consumers to buy, people have tried to bring their personal devices to work. Nevertheless, IT departments traditionally have been very reluctant to allow these devices to be used on the business network. However recently, since the introduction of the first iPhone and iPad, smartphones and tablets have become big business. Marketsandmarkets, a market...
research company, estimated that the smartphone market on its own will be worth 150 billion dollars by 2014 (Marketsandmarkets). In these times of high market growth, an array of new operating systems has emerged, each trying to become the dominant player. Firefox OS, BADA, Tizen, Android, iOS, Windows phone, Blackberry 10, Ubuntu phone are just some of the new operating systems that have emerged. Back in 2000, Microsoft’s Windows operating system ran on nearly 97% of all computing devices. Goldman Sachs calculated that today, including all smartphones and tablets, their market share has plummeted to 20% (Goldman and Sachs, 2012). It is safe to say that times have changed. More and more employees want to work on their platform of choice, with their devices of choice. Avenade, an IT consultancy firm, did a research on ‘bring your own device’ (BYOD) in 2011. They found that from the people that were allowed to bring their own device, 25% did this because the company did not provide the device they wanted, another 25% because the devices offered, didn’t have the required functionality and another 20% wanted to have a different brand. Intel, the world largest semiconductor chip maker, started testing the concept since 2010 and allowed employees to bring their own device when they subscribed to the ‘personal device program’. After just one year, already 17000 (out of 80000) of their employees enrolled in the program. Intel reported that the program offered great business value. Employees became more productive and were more satisfied, additionally costs for providing devices decreased. IT costs rose slightly but IT flexibility increased simultaneously (Intel, 2012).

Many employers embraced BYOD, seeing that it provides a win-win situation for the employer and the employee. In the same research done by Avenade, 73% of the executives of large enterprises, reported that BYOD was a top priority. Furthermore, if we include executives from smaller firms, already 88% reported that employees are allowed to use personal devices (Avenade, 2012).

During the requirement research, it became clear that CEO’s from smaller companies were very busy with day to day activities. A portable device could be a great solution, since it fits in their day to day activities and could thus be more successful than a traditional desktop application. In a research done by the small business authority, 50% of small business owners reported that tablets are already used in their company. Furthermore, 55% of the business owners reported that they see themselves using a tablet in the future. When asked what their next technology purchase would be, they chose a new tablet over a new laptop (Forbes, 2013).

5.4.1.1 Existing Tablet Platforms

Since the launch of the iPhone on the smartphone market, Apple’s market share has been declining in favor of Android. More than 70% of all smartphones that are shipped today run Android. The same trend is visible on the tablet market. Where Apple started with close to 100% market share, by the end
of 2012 that market share declined to 51%. The International Data Corporation (IDC), the premier global provider of market intelligence, estimates that already in 2013, Android’s market share will surpass iOS’ on the tablet market (IDC, 2013, IDC, 2013). Contrary to the smartphone market, little other operating systems are available. In the same research by the IDC, 99.4% of all tablets currently run either Android, iOS or Windows (See Fig. 21). We will thus only consider these three operating systems. In the next section, we will give some insights on what are important business considerations.

**Apple’s iOS**

Apple likes to keep control into its own hands. They use just a few hardware configurations and are able to optimize their software for just a couple devices. This makes the devices reliable and quite secure. Additionally they keep a close eye on what apps appear in their application store. In this way they try to prevent malware from getting onto their users’ devices. The downside for businesses is that they can’t just install homebrew applications. If they want to distribute software outside the app store, this will cost 300 dollars a year. Not insurmountable, but given the resource poverty of SMEs, this might be another cost they may not be willing to make. Concerning management of iOS devices, Apple has worked hard to provide some tools for easy management. However, these require MAC or OSX servers in the company. Again, for smaller companies that largely rely on Microsoft software, this may be another extra cost. Security wise, because of the closed nature of Apple’s iOS, it is inherently pretty secure. Little malware apps get into the store. Additionally iOS has some other security features built in. It has 256 bit hardware encryption and allows users to remotely wipe and lock a phone. This decreases the chance of losing company data. Apple’s main disadvantage is that companies that want more advanced security features are completely dependent on Apple. (PcPro, 2013).

**Google’s Android**

Contrary to iOS, Android is completely open source, meaning that the source code is freely available for everyone to download and use. This offers a very flexible platform. Hardware manufacturers can compete on features, price, quality and design. For companies, the flexibility allows them to choose the devices that fit their usage scenario and their manufacturer. Android has tried to find the middle

![Fig. 21 The three main operating systems (from left to right: iOS, Windows, Android)](image)
ground between the intuitiveness of iOS and the power and flexibility of traditional desktop operating systems, like Windows. Because the operating system (OS) is open source, a lot of management solutions are offered by third parties, like Samsung and 3LM. Security wise, Android is perceived to be the least secure one. Because Google doesn’t check applications as closely as Apple, and because Google allows users to install applications from outside their store, people run the risk of installing malware. For businesses, these risks can be almost nullified, though. With tools from third parties, it is possible to completely control what gets installed. Nowadays, there are solutions on the market that can put two accounts on the tablet or smartphone, one for business and one for home use. When going onto the business account, installing apps can be restricted and advanced security features for accessing the company network can be added. This offers the best of both worlds: people have no restrictions when using their smartphone at home and employers don’t have to fear security risks when checking in on the company network. The flexibility of the platform also allows third parties to offer complete business solutions. Tablets, ready for business use that include all of these features, are offered by manufacturers like Lenovo and Cisco. Companies don’t have to be reliant on Google.

Microsoft’s Windows

Windows 8 has positioned itself to be a complete replacement of the traditional desktop. They claim to offer best of both worlds. As it is less intuitive than the previous two platforms, it might be more difficult for employees to understand. This might be a disadvantage for companies, since training could be required here. Furthermore, Windows software has traditionally been quite costly. Management wise and security wise, Windows 8 is miles ahead of the competition though. Windows is most often the platform of choice for businesses. Windows 8 products are very easy to integrate and manage. Furthermore, compatibility with traditional software is guaranteed. A PowerPoint presentation can be made on a computer and shown during meetings on a tablet without any issues.

Ultimately there is no best platform, since it is up to the employees to bring their device of choice. At the time of writing this master thesis though, the Windows 8 market share for tablets is still almost negligible. Furthermore, if the same trend from the smartphone market continues on the tablet market, Android may also soon conquer this market. Additionally, the development tools for Android are free and homebrew applications can be freely tested. Android thus was the platform of choice.

5.4.1.2 Tablet Operating System Security

The Avenade survey above found that more than half of the companies that allowed consumer technology at work, reported a breach in security. Around 80% also admitted that their IT infrastructure still needs improvements to counter possible security issues. Most companies that
experimented with BYOD also reported that security is their number one risk. Even in larger companies, little money was spent on training, to point out these security risks to employees. Given that SMEs have even less money to spend on IT infrastructure or training, and given that the mobile platforms are still in its infancy concerning security, the decision was made to not allow the application to connect to company networks or databases. The tablet application can completely run independently without the need for internet connection or company network access. Once the operating systems and tools have matured, the application could easily be integrated in existing company networks.

A major disadvantage of designing for tablets, is that so far, this device has been primarily used for media consumption. Typing on the onscreen keyboard is a lot slower than typing on a physical keyboard and is thus not great for productivity. However, recently the focus is shifting away from just media consumption to a more complete computer like experience, where the user can also be productive. Convertible solutions, where a keyboard can be clicked onto the tablet, are available today. (Microsoft surface, ASUS transformer, Bluetooth keyboards...) Furthermore, a lot of effort is being put in increasing the accuracy of voice input.

5.4.2 Define Functional and Data Elements
The functional and data elements have been defined in the requirements definition above (supra, p.27). Instead of listing every functionality and data element that has to be available on every screen, or showing sketches that were made initially, an overview of all the functionalities is shown and explained in Chapter 8, where the tool is presented (infra, p.73).
5.4.3 Define Application Flow

Demarco and Lister define ‘flow’ as a “Condition of deep, nearly meditative involvement.” (DeMarco and Lister, 1999). Once a person gets into his flow, he becomes really productive. It is thus important to design our application to support and enhance this flow. Consequently, it is important to avoid flow-disturbing behavior. In this regard, an important concept is excise. Excise is the interaction that requires extra work for users, while not helping in the achievement of their goals. The word ‘excise’ is derived from the Dutch word ‘accijns’, which originates from the Latin word ‘accensare’. This simply means ‘to tax’; adding a cost without adding any value.

Navigating through different screens is very disorienting. Users have to completely shift their attention and are forced into a new context. If this only happens sporadically, this is not much of a problem but when users have to constantly switch between different screens, they will be quickly disoriented and frustrated. This flow-disturbing behavior is avoided by taking into account how users will be using the application. In the CHOOSE method (supra, p.13) it was clear that it was easiest to build up each model first, before linking the different models together. The application starts from a start screen, where the four dimensions are at the center. A click brings the user to the chosen model and the back button brings the user back to the main screen. The output section and guide work in the same way. Consequently only two hierarchical layers are present in the application structure (Fig. 22). Once you are within a certain model, the user never has to switch to another screen. Even for linking the different models, no screen switching is required. This will become clearer in the application presentation (section Chapter 8). All of this helps in keeping the user in his ‘flow’.

Fig. 22 Two layer application hierarchy
5.4.4 Interaction Design Principles

Norman, an academic researcher in interface design, argues that design is often constrained by culture. People want to be able to foresee what an object will do. When an interaction with a system is deeply engrained through cultural learning, it is hard to change the way people interact with an object (Norman, 2004). The easiest way to make an interface design consistent with how a user expects it to behave is by mimicking the physical product. Apple for example is known most for mimicking physical products in software (See Fig. 23). This technique is called a skeuomorphism.

For example in the books application (top left on Fig. 23), people’s mental model of how a book works in the real world is perfectly aligned with the behavior of the application. However, skeuomorphisms are not always the optimal solution. If we take a look at the calendar application, even though we are in week 3, we still see our appointments of the past 2 weeks. With technology, we could make the calendar more efficient by starting with the current week and showing the appointments from the future rather than the past. Furthermore, as is the case with our application, there is not always a physical object that can be mimicked. The insight to decrease the discrepancy between how an object behaves and how a user expects it to behave is offered by Norman (Norman, 2002), Cooper (Cooper et al., 2012) and IBM (Design, 1992). They have discerned three models of a system:

-Mental Model: How a user perceives that a product works;
-Implementation Model: The actual way the system works internally (programmer’s perspective);
-Represented Model: The way the application is represented to the user by the designer.

Davindson et al. offer a crucial link between usability and mental models. (Davidson, Dove and Weltz, 1999). The closer a user’s mental model is matched, the higher usability will be rated. Alternatively,
the program will be perceived easier to use and easier to understand the closer the represented model resembles the mental model (See Fig. 24).

![Fig. 24 The continuum between the implementation and mental model](image)

Reason argues in his book ‘Human Error’ that many disasters could have been avoided if controls would be designed, taking into account the mental model. Even though airplane pilots are trained extensively in the controls of a plane, if these controls were designed more intuitively, plane crashes could have been avoided (Reason, 1990). While pilots are well trained, computer users are no longer the stereotypical nerds. The computer has become available for almost everyone and interface designers have to accommodate this more diverse audience. These users are less technical, more demanding and most of all impatient. They no longer want to browse through manuals, they want things to be intuitive and useable, out of the box. (Gribbons) The burden should lie on the designer to research the mental model of the user, not on the user who has to adapt to the implementation model.

As mentioned, skeuomorphisms aren’t always available when designing a user interface. In that case, software designers have to aid users in building a productive mental model of a system. Several design methods are used to support and influence these mental models of the user. We will elaborate on the following topics: Simplicity, familiarity, availability, flexibility and safety. These have been proposed by IBM (IBM, 1992). By following these methods, the represented model will better match the mental model and consequently usability will be higher.
5.4.4.1 Simplicity, Less is More

More or Less Information?
When visiting a website, it can sometimes feel like walking through Times Square. When walking through Times Square, you see a myriad of different advertisement billboards, each with catchy slogans like: “The next big thing”, “Probably the best beer in the world” or “The real thing”. Many of these billboards are very flashy with vibrant colors and lights flashing, trying to get the passer-by its attention. Every brand competes with each other to make the billboard that pops out the most, trying to capture the so important passer-by glance. More and more websites and applications start to look like them, trying to throw as much information at the visitor as possible (See left side of Fig. 25), hoping that at least something that is shown will be of interest to them.

However, people are impatient and they can only process a limited amount of information (Hertwig and Todd, 2003). The more content you show, the smaller your chances of people noticing your most important content. Less really is more in this case.

Even small details like words, colors and buttons can add to the already heavy load a person has to process. Removing irrelevant options, slogans and content decreases this load on users. Designing a user interface in a clever way can further diminish this load and enhance the user experience (Colborne, 2010).

More or Less Features?
Unfortunately this answer is not straightforward. Thompson, Hamilton, and Rust (Thompson, Hamilton and Rust, 2005) did an experiment to find out the answer to this question. More specifically, they tried to find out if users preferred features over usability, or vice versa.

They concluded that features sell a product better, but when users are able to test the product before their purchase, they would buy the product with better usability over the product with more features.

In software, the cost of adding extra features is close to nothing. Considering people seem to buy
products based on features, this leads to features piled upon features. The problem is that with each extra feature added, the product becomes more difficult to use.

**More or Less Choice?**

Iyengar and Lepper set up multiple experiments to test whether people like more or less choice. All experiments were set up similarly. The results show significant higher sales revenues when offering only a handful of options. Also, people were more satisfied with what they bought when being offered fewer options (Iyengar and Lepper, 1999).

Because it offers a person a sense of control, people prefer to have a choice over no choice at all. However if that choice exceeds a handful of options, the choice can get overwhelming. The same statement can be made for technology, where most people get anxious if they are presented with a whole array of options and buttons.

Often when designers are unsure whether to take approach one or two, they just offer users both and let the user choose. But a simple user interface should not let users make these kinds of decisions. If a developer is unsure, he should do some research whether users prefer approach one or two. And if users like both, then a developer cannot really do anything wrong by choosing one of both.
5.4.4.2 Familiarity

When something looks and feels familiar, user can get started quickly and make progress instantly. Furthermore, when a certain approach is used consistently throughout the application, the user will feel more confident. Once the approach is learned, it can be applied in a variety of situations. Users will perceive the user interface to be easier and more productive (IBM, 1992).

After a first test in practice of the CHOOSE approach using post-its, it was clear that when modeling a goal tree or actor tree, wide hierarchic trees were obtained (See Fig. 26).

A wide hierarchic tree consists of just a few hierarchic layers, but contains a lot of elements in each of those layers. While looking for a fitting way of navigating through such trees on a tablet screen, an analogy with a typical file manager, used on desktop applications (Windows Explorer, Apple Finder), was discovered. After this analogy was found, designs from different file managers were analyzed.

In the Windows operating system, a file path has been present from the very first version of Microsoft’s operating system. In version 3.1, a tree view was introduced but quickly removed in the next version. The wide trees, typical for file managers, were hard to show graphically and difficult to grasp by users. The file path however, is present in every version and appears to give users the best sense of where they are. In later versions the file path became clickable to allow for easier navigation through the file tree. In Windows XP the ‘up button’ was introduced to move to the above hierarchic layer. In Windows Vista and Windows 7 this feature disappeared to make a more minimalistic design. However this ‘up button’ was the most requested feature to come back to the file manager by Windows users, so is now reintroduced in Windows 8 (Sinofsky, 2011).

OSX, the operating system from Apple, takes on a slightly different approach. When browsing through a file tree, all hierarchic layers become visible. A file path on the contrary, is not visible here. Important for both approaches is that users are offered easy navigation within their tree, either by clicking on a link (Windows) or by scrolling back (OSX).

Since a full graphical representation of a wide tree was already rejected on a computer screen, it is certainly not advisable for a tablet screen. Our application combines the Windows and OSX approach.
Instead of the up button, two hierarchic layers in an OSX style are shown. This makes easy navigation possible between two layers, while maintaining easy access to other layers with the clickable file path. The color-coded picture gives a graphical explanation of the combined design elements (see Fig. 27).
5.4.4.3 Safety

It is important to make users feel confident about their actions. This can be done by making actions predictable. Take for example, the use of a little arrow next to the name and description in our application. This arrow is shown when there are lower level elements available. Like this, the way the interface reacts after a click will become more predictable (see Fig. 28).

Safety can also be enforced by allowing users to try out something, without fearing for irreversible consequences (IBM, 1992). It has to be obvious how to easily go back to a previous state. By offering a navigation bar at the top of the screen, users can easily navigate back to the previously selected elements. It also prevents users from getting lost in their hierarchical tree (see Fig. 29).

Finally, modifying and deleting should be easy. In other tablet applications, modifying or deleting an item can be achieved by pressing and holding items, similar to right clicking an application on the desktop. In our application this is done the same way (see Fig. 30).
5.4.4.4 Availability

Users should never have to recall names, previous settings, usage patterns or interactions. The program should give a list of options, cues, reminders or other techniques, so that users never have to rely on their own memory (IBM, 1992). In our program, we have opted to show an overlay over the normal user interface (UI). When users install the application, first they’re presented with a guide about the CHOOSE approach (infra, p.43). When they finish the guide, overlays are shown to help users on their way (See Fig. 31). In the overlay on the start screen, users get a reminder of how to start. The CHOOSE method is implicitly used here. It points out what every button does and what everything means. These overlays can afterwards always be recalled by clicking on the question mark in the top right corner.

Another way of assisting users is by using smart defaults. Smart defaults can be obtained by tracking the user’s behavior. Or when a user is new to a program, by showing the most likely defaults based on other users their preferences. Although users may have to change the default settings, it is always better than being offered a blank canvas. Defaults can save people a lot of time, effort and can furthermore reduce cognitive overload. Please note for example that during the CHOOSE guide, some elements can already be added. When going to the actual models, several goals/actors/operations/objects can already be seen. Now the user does not start from a blank canvas.

Fig. 31 Screen overlays
5.4.4.5 Flexibility

When searching for a picture, taken on your last vacation in France, you will probably search for it in the pictures folder of your computer, and then navigate to the folder of the appropriate holiday. Getting all nostalgic, you want to start planning your next holiday and remember you made a planning schema for some previous holiday. Unfortunately you have no clue for which holiday it was. But you do remember that the file was called ‘Holiday planning’. Luckily there is a search button which will probably be quicker than browsing through your complete pictures folder. Basically, it is important to offer enough flexibility that allows for different ways of interacting with a program. Dependent on the situation, different ways of interacting will be preferred. One has to be careful to not offer all possible options though. This would be in contradiction with what we discussed under simplicity (supra, p.36). It is up to the designer to figure out what the most common way of interaction will be. For this application, two ways of navigating have been implemented. The main way of navigating is by clicking your way through the hierarchical tree. However, sometimes users may not remember the hierarchical path towards the element. Then, they can search that element by clicking on the search button. It will refresh the results after every letter that has been typed and will search through the whole database based on name or description. In many applications, even though you still remember part of the name, if you don’t know the first letters, you will not be able to find that item. On the contrary, our search algorithm is very flexible in that regard. For example, when searching for someone in the actor model, maybe you don’t remember his name but remember that ‘mr. X’ was a salesman. After pressing just two letters, you are already presented with a list of salesmen of your company. Similarly, it is possible to only type part of a name.
5.4.4.6 Beginners, Experts and Intermediates

Another important factor to keep in mind is that managers are not expert technology users. Users can be divided in three types: experts, beginners and intermediates. (Colborne, 2010, Cooper et al., 2012). Intermediates are people who don’t use technology for the fun of using technology. They use it to get something done. They use just a few key features and don’t care about the advanced ones. Constantine was the first to recognize that most of the effort should be targeted towards these mainstream users. Most beginners become intermediates fairly quickly but few eventually become expert users (Constantine and Lockwood, 1999). Some specialized software specifically targets expert users, like development software for applications. For this type of applications, we expect the user to invest significant time and effort to master the application. Since we aren’t dealing with expert users in our case, the application should be designed for a mainstream user.

However, everyone usually starts as a beginner. To get a beginner on board, it is often good to offer extra help in the beginning. A small guide, launched at first use and possible to access when needed, is a useful tool for beginners (See Fig. 33 or go to tool presentation at Chapter 8 to see the detailed guide) (Cooper et al., 2012). Furthermore, an overlay is shown which points out what everything does and means. Having said that, an important concept that should be taken into consideration again, is the concept of excise (supra, p.33). If we would show this guide and overlay every time the program starts, user would become quickly annoyed. Especially in the case of expert users, they prefer no excise at all. However, if every form of excise would be removed, it would be very difficult for new users to get started. In our case, after the first time the guide and overlays are shown, these are never shown again and are available by clicking the question mark buttons.

Fig. 33 Nine page introductory guide
5.5 Design Refinement

While the interaction framework was set up to achieve a usable design, the goal in the design refinement is to make a beautiful design or ‘Venustas’, as it was called by Vitruvius. As discussed, a beautiful design can greatly enhance the usability by increasing the perceived usability. As websites have to appeal at first glance to new users, it is really important to quickly catch the user’s attention. Therefore, a lot of research has been done on web design. Although not every design element or guideline for websites can be transferred to mobile devices, a lot of interesting insight can be obtained. Many books have been written and guidelines have been set forward by renowned designers. In the remainder of this chapter, some of these guidelines will be introduced. Then, we will show how these are used in the application. Again we don’t go into the functionality of the application yet.

5.5.1 Visualization

Visuals have certain major advantages over textual representation. Research found that it is easier for humans to cognitively ingest information, shown in a visual representation (Larkin and Simon, 1987, Winn, 1991). Visual information makes the viewer synthesize, whereas text makes the user analytically decompose (Unnava and Burnkrant, 1991). Visuals are processed in the affective system of the brain and evaluate stimuli as a whole. This greatly increases the processing speed (Cacioppo, Gardner and Berntson, 1999). Another major advantage is that it is easier to remember and recall graphics (Nesbit and Adesope, 2006). These principles have been used extensively in our application and almost anything has a graphical representation. Some examples can be seen on Fig. 35.
5.5.2 Attract Attention

The first way to catch attention is by creating contrast. Contrast occurs when two things are different. A basketball player may not seem big when he is playing with his team. But if you would meet him after his game, his large posture suddenly becomes obvious. Contrast is a relative measure that compares an element with his surrounding elements. Attracting attention has been an important area of interest for websites. The website of Blinksale is a good example. When you look at the left hand side of Fig. 36, the first thing that you probably notice is the screenshot. This is the point on the screen with the highest contrast ratio. Next, your eyes will probably be drawn to the yellow button. The health site CareLogger reported that their sign-ups went up by 34% after changing the color of their sign-up button (CareLogger, 2010). Playing with contrast can be a very effective way of directing the user to where you want them to look.

Similarly, in our application, the greatest point of contrast is the right hand side of the application where visualizations are shown. This is where we want to user to focus his attention to. By using shadings around the white area, this effect is accentuated further.

Another way of attracting attention is to play with colors. Tiger color, an award winning program for designing harmonious color schemes, proposes to use just three colors (Tiger Color). When using more colors, the user interface can quickly start to look busy and unprofessional. If you use too few colors, your application will look boring and uninteresting. These three colors should not be used to the same extent. The primary color should create the overall style of the application. The secondary color is the color that has to provide contrast and attract visual interest. The third color or accent color has to give your application a touch of elegance. So a good starting point appears to be three colors, which makes visual interest and variation possible. In our application, the primary, secondary and accent color are respectively gray, white and light blue (See right hand side Fig. 36).
Closely related to these concepts of contrast and color is the concept of readability. Text and icons will be better readable, the higher their contrast is. The international standardization organization proposes a contrast ratio of at least 3:1 (ISO-9241-3 (ISO)). The contrast ratio has been tested for all our text and icons, by using the ratio formula provided by the W3C (W3C). As you can see on the left hand side of Fig. 37, the lowest contrast ratio of text/background is 9.49. This means that the text is perfectly readable. Also note the very low contrast ratio of 2.02 for the blue line. This is by design, the color is meant to accentuate, not to draw attention. The same reasoning applies to the icons: by using a relatively low contrast ratio of 3.46 at the top right of the application, we don’t want to draw attention to it. On the right hand side, the visualizations are done. As you can see, these contrast ratios aren’t ideal yet, though this will be discussed more in depth in the next section.

5.5.3 Content

The visualization on the right hand side of the application (see Fig. 37) is closely related to the language ingredient we discussed in section 1.1.1.3. We explained that a language is a necessary component in an EA approach. Unfortunately, this component has not yet been developed. No research or effort has been done in refining the visualizations. Basic colors have been used to easily discern from which model an element is. More research is needed on what has to be visualized for the user and how these visualizations have to be drawn.
5.5.4 Visual Simplicity

Get rid of half the words on each page, then get rid of half of what’s left.’

In ‘Don’t make me think’, Steve Krug describes that ‘what can be cut, should be cut’. Preventing cognitive overload is the main focus point. Getting rid of all the unnecessary text has, besides preventing information overload, two other important advantages. First of all, it makes sure to draw your attention on what is really important. Second, it makes people feel more secure whether they got it or not (Krug, 2009).

Sometimes, there can be unnecessary introductions, lengthy instructions like ‘click this button if you want to order this item from our website’ or explanations, that are often annoying, once the user knows how to use a site. As discussed in the visualization section, if icons or graphics can be used, they should be used, and text can be cut. One last guideline to not clutter a page with text, is by showing the explanation upon request. For example a link with ‘show more’ or an information icon can be used.

In our application, an information button is added that shows the user information upon request (See Fig. 38). The overlay, shown in 5.4.4.4 (supra, p.41), is then visible.

![Fig. 38 Present information upon request](image)

However, some caution is needed to not overdo the cutting spree. When Apple introduced the iPhone in 2007, it set itself apart from competition in many ways, but one was that it just had one button in the front. It made it very easy for people to navigate back and forth to the home screen and gave the users a sense of control. However in their Apple Store in Japan, Apple reinvented the elevator by cutting all of its buttons and text, so the elevator just stopped at every floor. But instead of simplifying the experience, they complicated it. Users were left frustrated, not being able to control where they were going.

Simplicity has been one of the key words throughout the development of the CHOOSE approach (the S stands for Simple). When researching about design, Karvonen showed that simplicity and a beautiful design greatly influence the user’s experience and interpretation of the design. Again simplicity is an important key word (Karvonen, 2000).
Chapter 6  Application Development: Internals

6.1 Code Quality

When you would ask an Apple user to compare the quality of their iPhone to other phones, they will most likely talk about how great the user experience is, how sturdy yet light it feels, how stable it is and never crashes or how magical the device feels. Often when we talk about quality, we talk about the quality in use or how the actual end user perceives it. More broadly, this view reflects whether or not users can achieve their goals. Many of these external quality attributes will heavily influence the quality in use. In the development of our tool, up till now, we focused mainly on these external quality attributes. We talked about how to make it simple and easy to use and how to create a great user experience. However, there is an important second dimension. When using your phone, you would never expect it to break into thousand pieces when squeezing it in your hand. In the factory, they make sure that the alloys have the correct characteristics and that all the parts are within tolerance margins. The internal quality attributes are all the attributes that can only be seen by examining the device rather than using it. In software, internal quality attributes are everything that can be examined without running the code.

The ISO/IEC 9126 standard is an international standard, used to evaluate software quality. This standard offers a holistic view on quality. It describes external and internal attributes. The quality model, which is the first part of the standard (ISO 9126-1), classifies software quality according to six dimensions: functionality, reliability, usability, efficiency, maintainability and portability. In the book ‘Code quality: the open source perspective’ by D. Spinellis, this quality model of the ISO standard is adapted to focus on internal quality attributes (Spinellis, 2006). He discusses five dimensions, each with specific internal attributes (Fig. 40).

---

Fig. 39 Third step in software development process: Application internals

Fig. 40
When thinking about the functionality of software, the first thing that hits our mind is the question of what we can accomplish with it. What are all of the possible features or outputs that we expect to get from the application? This external dimension has already been discussed above, we will only zoom into the internal attributes of functionality, consisting of security and accuracy.

Accuracy is how close a value is to the quantity it actually represents. Saying that ‘this is page 49’ is perfectly accurate as the value is equal to the quantity it represents. However, the mathematical symbol π is often represented by 3.14, but can hardly be called very accurate. As our application isn’t using any mathematical calculation or stores no numerical data, we don’t go into further detail about this topic.

Since we are developing for a business environment, the attribute security is much more important. The problem with software security is that it depends on the weakest link. No matter how secure your program is, if the operating system is not, efforts will be in vain. Not all information is critical though. If you are making an application for storing cooking recipes, the user will probably not lose his sleep over potential security problems. On the contrary, if you’re designing an application for receiving personal e-mails, users will find security an important issue. The necessity and importance of security should always be judged according to how critical the data is. When offering SMEs an application where they can input their strategy, this can be classified as critical data. In this regard, a
secure application is essential. As described before, tablet platforms aren’t as mature as their desktop counterparts. For the time being, to nullify potential data breaches, the decision was made to disable internet connection. The application only communicates with the internal memory of the device. Still, some preventive measures were already taken to make the program secure. One often recurring problem in database heavy programs is SQL injection. For example, a standard sql query may look like:

\[SQL statement= \text{"SELECT * FROM GoalTABLE WHERE C\_ID = ' ' + ActorId + ' ';"}\]

After the user is asked to input his ActorId (for example a password) in a text field, this query shows all the available goals that the user is allowed to see. If that user would pass the string ‘ OR ’1‘=‘1, the query would now become

\[SQL statement= \text{"SELECT * FROM GoalTABLE WHERE C\_ID = ' ' OR '1‘=‘1’"}\]

While the statement C\_ID = ‘ ‘ will return no results, 1 will always be equal to 1. The user would now get all of the goals returned to him, even if he did not have the right clearance level. To make sure these sql injections are not possible, Android offers parameterized queries. In Fig. 42, the ‘WHERE’ clause is now broken down into two separate parameters. For now, the application and accompanying database can only be used by the owner of the application. In the future, when the application is made available to multiple stakeholders using the same database over the internet, the security measures are already in place.

```java
public String GetGoalName(String ActorId) {
    SQLiteDatabase db = dbHelper.getReadableDatabase();
    Cursor c = db.query(DbHelper.GoalTABLE,
            new String[] { DbHelper.C\_NAME }, DbHelper.C\_ID
            , new String[] {ActorId}, null, null, null);
    if (c.getCount() > 0) {
        c.moveToFirst();
        GoalName = c.getString(NameColumnIndex);
        return GoalName;
    }
}
```

Fig. 42 Parameterized query to avoid SQL injection
6.1.2 Reliability

The reliability of software concerns how performance is maintained under certain conditions. Three internal attributes of reliability will be further zoomed into: maturity, fault tolerance and recoverability.

6.1.2.1 Maturity

The maturity is the amount or the absence of software faults that lead to failures. Maturity is broken down into seven attributes: Input, output, logic, computation, concurrency, interface and data handling problems.

Input: Programs aren’t completely built from scratch. Often existing libraries or frameworks are used. This input is generally thoroughly tested. In our tool, the Android support libraries are used together with a whole array of existing Android and Java frameworks. Most of our application handles these frameworks, and our coding is consequently fairly high level. Input can be considered mature.

Fig. 43 Second element of software quality: Reliability

Fig. 44 Android support library and snapshot from imported Android frameworks

```
import android.annotation.SuppressLint;
import android.app.ActionBar;
import android.app.ActionBar.LayoutParams;
import android.app.Activity;
import android.app.AlertDialog;
import android.appProgressDialog;
import android.app.SearchManager;
import android.content.ComponentName;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.util.Log;
import android.view.ContextMenu;
```
Output: Problems concerning output are more subtle than input problems. Output problems include incomplete or missing output, formatting issues, spelling or grammar errors. For example, when using the frameworks discussed above, existing methods from parent classes can be overridden. But because those classes are mostly hidden in libraries, it can be tricky to know exactly what type of output is expected. In Java, annotations can be used to solve this problem. By adding an annotation, the compiler will check whether or not the correct output has been passed. Instead of having a crashing application, the compiler will throw a compilation error when output is incorrect. For example in the following figure (Fig. 45), the class ObjectsSurfaceView, that holds the code to generate drawings for the objects model, uses the existing Android class Surfaceview as a parent class. By overriding the onDraw() method, the default drawing method is overridden by what is coded under the onDraw() method in ObjectSurfaceView. By placing the @Override annotation, the compiler checks whether everything that’s drawn is accepted. Like this, problems are prevented even before running and testing the application.

```
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class ObjectsSurfaceView extends SurfaceView implements SurfaceHolder.Callback {

@Override
public void onDraw(Canvas canvas) {

if (getRunning) {

    canvas.drawRGB(255, 0, 0);

    int GoalWidth = (int) Math.floor(canvas.getWidth())*(0.40);
    int GoalHeight = (int) Math.floor(canvas.getHeight())/(2);
    int ObjectWidth = (int) Math.floor(canvas.getWidth())*(0.60);
    int ObjectHeight = (int) Math.floor(canvas.getHeight())*(0.60);
    int AccrWidth = (int) Math.floor(canvas.getWidth())*(0.80);
    int AccrHeight = (int) Math.floor(canvas.getHeight())*(0.40);
    int OperationWidth = (int) Math.floor(canvas.getHeight())*(0.40);
    int OperationHeight = canvas.getHeight()/2;
    int maxWidth = 140;
    int maxHeight = 60;

    canvas.drawLine(10, ObjectHeight, ObjectWidth, ObjectHeight, LightlinePaint);
    canvas.drawLine(ObjectWidth, 10, ObjectWidth, canvas.getHeight()-10, LightlinePaint);
    canvas.drawLine(ObjectWidth, GoalHeight, canvas.getWidth()-10, GoalHeight, LightlinePaint);

    Fig. 45 Example of annotations in Java
```

Logic Problems: The logic is concerned with a program’s control flow. This is a vague definition because it includes a lot of little problems that are often hard to detect. The only way to avoid these is to follow coding conventions. One example of where control flow often goes wrong is the ‘off-by-one’ error. This is an error encountered by incorrectly defining iterative loops like the ‘for’ loop in Java. ZeroPointerExeptions is what often comes up as an error in this case. The convention to follow here is to always use an asymmetric range of elements. This means that the ‘for loop’ expression includes the first element of the range but excludes the last element or vice versa. for (int I = 0; I < files.length; i++)
Computation Problems: Often, problems arise when you try to make a complex algorithm yourself. It may work for the cases you use to test the algorithm, but may give errors when used by other people. Again, the golden rule is to use conventions and existing frameworks or algorithms. As discussed above, a lot of existing frameworks have been used to decrease these computational errors.

Concurrency and Timing Problems: These problems appear when using multiple threads. When using multiple threads, code can be executed asynchronously. Even for seasoned practitioners, problems arise that can be extremely hard to solve. Unfortunately, in our application, multiple threads had to be created to allow the user to interact with the interface while a picture is being drawn. Existing code, provided by tutorials, was used and left unchanged to not face these problems. In other areas of the application, multithreading could be useful but is not implemented because of the timing problems it could introduce.

Interface Problems: This was a problem, encountered when applications had to communicate with hardware themselves. Fortunately this is now handled by the operating system.

Data handling problems: These will not be discussed because we only use one data source. Data handling problems occur in massive software projects with large amounts of data sources.

6.1.2.2 Fault Tolerance

A system is completely tolerant when it can compensate mistakes at the hardware or software level. For example, when a data read error occurs at one database, a duplicate database should be available where the same information can be requested. Thinking about handling every possible exception and providing duplicate solutions, is something that has not yet been done in this application. Consequently, this should be improved in future versions.

6.1.2.3 Recoverability

Recoverability is the extent to which a system is recoverable in case of a failure. Although fault tolerance should help in avoiding failure, if it does happen, the system has to handle this failure. How does it recover and how does the system handle the lost data? Luckily Android has a predefined activity lifecycle. When overriding one of the methods shown in figure Fig. 46, the default behavior can be changed. For example when the activity crashes, it will call the onDestroy() method. By overriding this method, custom behavior like saving the data and reinitiating the previous activity can be done. In our application, the program always returns to the start screen after a crash.
6.1.3 Efficiency

Efficiency has two dimensions, a time and a space dimension. These are often two opposing elements. For example, an internet browser could load a complete website onto your computer’s memory. Although browsing will be very fast, a lot of memory will be used. The two have to be balanced. Furthermore, many optimizations exist to optimize the time or space dimension. Several measures exist to compute time performance. The most important attributes are: latency, throughput, processor time requirements, real-time response and time variability. By using these metrics, the algorithms’ efficiency can be measured and tweaked to find better performing ones. Similar measures exist for improving the space requirements. That being said, in the very influential book ‘the principles of program design’, AM. Jackson warns that code optimization has to be done with caution. He has two rules concerning program optimization. Rule one: ‘don’t do it’. Rule two: ‘if you are an expert, don’t
do it yet’ (Jackson, 1975). Other experts gave similar warnings. D. Knuth even calls premature optimization ‘the root of all evil in programming’ (Knuth, 2007). There are several reasons why code optimization is so dangerous. In general, the more efficient an algorithm is, the more complex it becomes. Consequently, reliability and readability suffer. Another reason is that when making optimizations for a certain platform, portability to other platforms is reduced. Finally, to further optimize code, special cases of routine input is exploited. This destroys the simplicity and clarity of the code. As a non-expert programmer, no code optimization is done. Some clear tradeoffs were made between the time and space dimension though. These will be shortly discussed next.

One obvious tradeoff where time was sacrificed in favor of space, was the use of cashing. One possibility was to load in all of the goals/actors/operations/objects at the start of opening a model. However, in case of larger goal trees with for example 100 goals, this would mean storing 100 class objects in memory. Therefore, in the start, only the main goals are loaded. When pressing a goal with lower level goals, these lower goals are fetched from the database and loaded onto memory (Fig. 48).

Other tradeoffs were also made in the opposite direction. For example, the CHOOSE guide consists of multiple pages where the user can swipe through. Standard in Android, only three pages are loaded in memory, however this behavior is overridden to load in all nine pages (Fig. 49). This greatly improves performance but sacrifices space.

```java
MyPagerAdapter adapter = new MyPagerAdapter();
ViewPager myPager = (ViewPager) findViewById(R.id.myfivepanelpager);
myPager.setAdapter(adapter);
myPager.setOffscreenPageLimit(9);
myPager.setCurrentItem(0);
```

Fig. 48 Goals loaded when clicked

Fig. 49 Loading in all 9 pages of the guide
Finally, another tradeoff worth mentioning is the use of full text searches. When a user for example wants to find all the salesmen in the company, the application will give feedback after every letter that is typed. After just two letters, we can see all of the salesmen. Because the database has to be completely searched on name and description every time the user types a letter, traditional relational databases would be too slow. In this regard, a database supporting full text search is used.

Unfortunately, this type of database only supports limited query terminology. So for more complex queries, we still had to use a different type of database. Two databases mirror each other leading to more memory and storage usage (Fig. 50). In this example, we sacrificed the space dimension in favor of the time dimension.

```java
// Table for full-text search
String sql0 = "CREATE VIRTUAL TABLE " + GooiTABLE_FTS + " USING fts3 (" +
    C_ID + ", " + C_NAME + ", " + C_Emschrijving + ", " + C_TYPE + ");";
// Table to perform normal queries
String sql1 = String.format("CREATE TABLE %s (%s String, %s String, %s String, %s String)",
    GooiTABLE, C_ID + " [PRIMARY KEY | UNIQUE] ", C_NAME, C_Emschrijving, C_TYPE);
```

Fig. 50 Full text search example

Fig. 51 Extra table for full-text search
6.1.4 Maintainability

The program’s maintainability is concerned with how hard it is to modify the existing application. How easy is it to fix bugs or how easy is it to integrate new features? As the CHOOSE approach is still a very young approach, that is still being tested and refined, the program should be able to quickly adapt to these changes. In Fig. 53, the application structure and dependencies are shown. In the

---

**Fig. 52 Fourth element of software quality: Maintainability**

The program’s maintainability is concerned with how hard it is to modify the existing application. How easy is it to fix bugs or how easy is it to integrate new features? As the CHOOSE approach is still a very young approach, that is still being tested and refined, the program should be able to quickly adapt to these changes. In Fig. 53, the application structure and dependencies are shown. In the

---

**Fig. 53 Application structure**

---
com.CHOOSE.project class, the general classes are present, including the main screen of the application. Similar to the dimensions of CHOOSE, the rest of the application is structured around the four dimensions. Each of these packages holds a main class, plus some additional helper classes that can be seen on the left hand side of Fig. 53. The listview package and sql package contain all the code, used by all four dimensions. From a developer perspective, a lot of the interface handling and drawing is similar across the four models. It may have been more efficient to integrate many parts. By using a structure where the four models are completely separate, it will be easier to alter a certain dimension later. For example, if the operations model would be transformed to include activity sequences, this part of the code and visualizations can be changed independently from the other models. Note that code to define UI elements is written in XML and is not visible on the picture. So structure wise, the application is built up to be flexible and easy to maintain.

### Measure Maintainability

In this section, some metrics will be shown, which are often associated with maintainability. These are measured with the tool Codepro, which is offered as a free plugin for eclipse by Google. Cyclomatic complexity is a measure that indicates how complex a program is. In programming it is often used to measure the complexity of methods. It measures the number of independent paths that are possible in that method. For example a method that has multiple nested if and switch statements will generally be perceived to be complex. The more paths there are available, the more difficult the code will be to read and the harder it will be to fix bugs or change code. Generally, it is agreed upon, that simple methods have a cyclomatic complexity lower than 10. Everything up to 15 is acceptable, though (Watson, McCabe and Wallace, 1996). In Fig. 54 the average cyclomatic complexity is shown. An average of 3.26 is very good, however, there are some methods that have very high complexity. One method even reaches a value of 44. After doing the same analysis on a class level, we saw problems with certain classes again. Especially the classes that handle the main interface interactions

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Minimum and Maximum</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Cyclomatic Complexity</td>
<td>3.26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>actors</td>
<td>3.76</td>
<td>minimum</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>com.CHOOSE.project</td>
<td>0.0</td>
<td>maximum</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>com.CHOOSE.project</td>
<td>1.17</td>
<td>minimum</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>com.viewpageindicator</td>
<td>0.0</td>
<td>maximum</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>goals</td>
<td>4.03</td>
<td>minimum</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>information</td>
<td>3.15</td>
<td>maximum</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>listviews</td>
<td>2.53</td>
<td>minimum</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>objects</td>
<td>3.77</td>
<td>maximum</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>operations</td>
<td>3.70</td>
<td>minimum</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>output</td>
<td>1.87</td>
<td>maximum</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>search</td>
<td>2.45</td>
<td>minimum</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>sql</td>
<td>2.23</td>
<td>maximum</td>
<td></td>
<td>44</td>
</tr>
</tbody>
</table>

Fig. 54 Cyclomatic complexity
are large. Consequently, readability and maintainability are reduced. There is definitely room for further improvement regarding code complexity.

The comment ratio is the number of comments, divided by the amount of lines of code. With a ratio of 8%, a lot of effort has been put in documenting the code. Other code metrics are shown in Fig. 55 but not further discussed.

<table>
<thead>
<tr>
<th>Lines of Code</th>
<th>17,778</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Characters</td>
<td>957,452</td>
</tr>
<tr>
<td>Number of Comments</td>
<td>1,508</td>
</tr>
<tr>
<td>Number of Constructors</td>
<td>35</td>
</tr>
<tr>
<td>Number of Fields</td>
<td>1,737</td>
</tr>
<tr>
<td>Number of Lines</td>
<td>24,977</td>
</tr>
<tr>
<td>Number of Methods</td>
<td>753</td>
</tr>
<tr>
<td>Number of Packages</td>
<td>18</td>
</tr>
</tbody>
</table>

Fig. 55 Other code metrics

6.1.4.2 Attributes of Maintainability


The analyzability dimension is concerned with how readable an application is. Besides complexity measures like described above, other possibilities are available to achieve readable code:

- Consistency: This encompasses how consistent statements are formatted, indented, named and commented.
- Naming conventions: Names have to be easy to understand and immediately tell what their role is. In Fig. 56, the method `getGoalName(String id)` makes it immediately clear that we want to get the goal’s name that matches a certain identifier.

```java
public Cursor getMainGoal() { }
public Cursor getUnassignedGoal() { }
public String getId() { }
public String getGoalName(String goalId) { }
public void setGoal(String goalName, String goalDescription, String goalType) { }
public void modifyGoal(String id, String goalName, String goalDescription, String goalType) { }
```

Fig. 56 Readable code

- Comments: Comments help to better understand what certain methods do or what certain statements try to accomplish. Especially since some complex methods are used in the program, by explaining what every path does, the analyzability greatly improves. As discussed above, the comment / lines of code ratio is very good.

Changeability is about how quickly we can implement a certain modification. How quickly can we find where to modify? How complex is our code? Does adding code break other parts of the software? The
answer to these questions is closely linked with what we have discussed above: application structure, analyzability and complexity. The stability of the code after modifications is highly dependent on the application structure. As we discussed before, the application structure has been constructed to be flexible. Furthermore, the dependencies in Fig. 53 are constructed fairly robust. The more everything is interconnected, the more chance of instability after a change. This is not the case in our program, though. Testing can be viewed at different levels. In our case, we will define testing on the unit level. A unit is defined as a single class. Unfortunately, testing is highly dependent on the cyclomatic complexity. As we have seen from our measures, some methods are fairly complex. If complexity is decreased, automatically testing will improve.

6.1.5 Portability

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Reliability</th>
<th>Efficiency</th>
<th>Usability</th>
<th>Maintainability</th>
<th>Portability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 57 Final element of software quality: Portability

The final dimension of the quality model is the portability dimension. Several sub dimensions exist when talking about portability: Operating systems, processor architectures, graphical user interface environment, regions, hardware devices and platforms.

The application has been written in Java. This has some huge advantages because Java code is platform independent. Java code is compiled to byte code for a virtual machine. This virtual machine is available for different platforms and processor architectures. Then, this virtual machine handles the translation to code, that is interpretable for the specific platform or processor architecture. Because there is an extra layer of translation, code can be written independent of the platform. The choice for the Android platform has been discussed in section 5.4.1 (supra, p.28). Since Android applications are written in Java and XML, at first sight, it may seem that it will be quite easy to port the application to other platforms. However, as discussed in section 6.1.2.1 about maturity (supra, p.51), a lot of Android frameworks were used. By using these frameworks, input problems were minimized. Consequently the portability has been sacrificed. The same goes for the graphical user interface. Although the visual look is mostly designed specifically for this application, many Android interface elements have been used that may not be available on other platforms. Android is very flexible in supporting all kinds of devices though. It could run on smartphones, tablets, desktops and televisions. Only tablets are supported at this point, but it is very easy to port Android apps to be used on other devices running Android.
6.2 Database Design

In this section, we translate the ER meta-model to a relational database model. The starting point is the ER diagram from Fig. 58. Notice that the latest version of the metamodel has changed dramatically (supra p.11). Unfortunately, although efforts were made to make the application structure flexible, the database is by definition inflexible. Drastic changes in the metamodel lead to drastic database redesign and consequently also code has to be redone. Once the metamodel is final, this translation exercise could be redone and code could be adjusted. For now, we will make the exercise on this ER metamodel. To translate the ER metamodel to the relational database model, a mapping tool is used (Vandenbulcke and Lemahieu, 2001). By following 9 steps, an ER metamodel can be translated to a relational database schema. Only the steps that are relevant are described below.

![Essential metamodel](image-url)
6.2.1 From ER Metamodel to Relational Database Model

**Step 1:** For each strong entity type from the ER diagram, a relation is created with all singular attribute types from this entity type.

- Refinement(Id)
- Goals(Id, Name, Description, GoalType)
- Actors(Id, Name, Description, ActorType, Refinement)
- Operations(Id, Name, Description, Type, ProjectOrProcess)
- Objects(Id, Name, Description, Type)

**Step 2:** For each binary 1:* relationship type, the primary key of the relation that originated from the ‘owner’- entity type has to be added as a foreign key in the other relation.

- Refinement(Id, goalId) goalId is a foreign key which refers to Id in Goals, NULL not allowed

**Step 3:** For each binary *:* relationship type, a new relation has to be added

- ANDRefinement(RefinementId, goalId)
  - RefinementId is a foreign key that refers to Id in Refinement, NULL not allowed
  - goalId is a foreign key that refers to Id in Goals, NULL not allowed

- GoalActorLink(goalId, actorId, Refinement)
  - goalId is a foreign key which refers to Id in Goals, NULL is not allowed
  - actorId is a foreign key which refers to Id in Actors; NULL is not allowed

- GoalsOperationLink(goalId, operationId, Refinement)
  - goalId is a foreign key which refers to Id in Goals, NULL is not allowed
  - operationId is a foreign key which refers to Id in Operations; NULL is not allowed

- GoalsObjectLink(goalId, objectId, Refinement)
  - goalId is a foreign key which refers to Id in Goals, NULL is not allowed
  - objectId is a foreign key which refers to Id in Operations; NULL is not allowed

- ActorOperationLink(actorId, operationId, Refinement)
  - actorId is a foreign key which refers to Id in Actors, NULL is not allowed
  - operationId is a foreign key which refers to Id in Operations; NULL is not allowed

- ActorObjectLink(actorId, objectId, Refinement)
  - actorId is a foreign key which refers to Id in Actors, NULL is not allowed
  - objectId is a foreign key which refers to Id in Objects; NULL is not allowed

**Step 4:** For each recursive *:* relationship type, a new relation has to be created
ObjectAssociation(objectId1, objectId2, Refinement)

Object1Id is a foreign key which refers to Id in Objects, NULL is not allowed
Object2Id is a foreign key which refers to Id in Objects; NULL is not allowed

ActorSupervision(supervisorId, superviseeId, Refinement)
supervisorId is a foreign key which refers to Id in Actors, NULL is not allowed
superviseeId is a foreign key which refers to Id in Actors, NULL is not allowed

OperationSuperior(superiorOperationId, suboperationId, Refinement)
superiorOperationId is a foreign key which refers to Id in Operations, NULL is not allowed
suboperationId is a foreign key which refers to Id in Operations, NULL is not allowed

GoalConflict(conflictgoal1Id, conflictgoal2Id, Refinement)
conflictgoal1Id is a foreign key which refers to Id in Goals, NULL is not allowed
conflictgoal2Id is a foreign key which refers to Id in Goals, NULL is not allowed

1.1.1. Resulting Database Design

The end result consists of 16 tables. The design of the database can be seen on Fig. 59.

Goals(Id, Name, Description, GoalType)
Actors(Id, Name, Description, ActorType, Refinement)
Operations(Id, Name, Description, Type, ProjectOrProcess)
Objects(Id, Name, Description, Type)
Refinement(Id, goalId) goalId is a foreign key which refers to Id in Goals, NULL not allowed

ANDRefinement(RefinementId, goalId)
RefinementId is a foreign key that refers to Id in Refinement, NULL not allowed
goalId is a foreign key that refers to Id in Goals, NULL not allowed

GoalActorLink(goalId, actorId, Refinement)
goalId is a foreign key which refers to Id in Goals, NULL is not allowed
actorId is a foreign key which refers to Id in Actors; NULL is not allowed

GoalsOperationLink(goalId, operationId, Refinement)
goalId is a foreign key which refers to Id in Goals, NULL is not allowed
operationId is a foreign key which refers to Id in Operations; NULL is not allowed

GoalsObjectLink(goalId, objectId, Refinement)
goalId is a foreign key which refers to Id in Goals, NULL is not allowed
objectId is a foreign key which refers to Id in Operations; NULL is not allowed

ActorOperationLink(actorId, operationId, Refinement)
actorId is a foreign key which refers to Id in Actors, NULL is not allowed
operationId is a foreign key which refers to Id in Operations; NULL is not allowed

ActorObjectLink(actorId, objectId, Refinement)
actorId is a foreign key which refers to Id in Actors, NULL is not allowed
objectId is a foreign key which refers to Id in Objects; NULL is not allowed
OperationObjectLink(operationId, objectId, Refinement)
operationId is a foreign key which refers to Id in Operations, NULL is not allowed
objectId is a foreign key which refers to Id in Objects; NULL is not allowed
ObjectAssociation(object1Id, object2Id, Refinement)
Object1Id is a foreign key which refers to Id in Objects, NULL is not allowed
Object2Id is a foreign key which refers to Id in Objects; NULL is not allowed
ActorSupervision(supervisorId, superviseeId, Refinement)
supervisorId is a foreign key which refers to Id in Actors, NULL is not allowed
superviseeId is a foreign key which refers to Id in Actors, NULL is not allowed
OperationSuperior(suprioroperationId, suboperationId, Refinement)
suprioroperationId is a foreign key which refers to Id in Operations, NULL is not allowed
suboperationId is a foreign key which refers to Id in Operations, NULL is not allowed
GoalConflict(conflictgoal1Id, conflictgoal2Id, Refinement)
conflictgoal1Id is a foreign key which refers to Id in Goals, NULL is not allowed
conflictgoal2Id is a foreign key which refers to Id in Goals, NULL is not allowed

Fig. 59 Database design
After a more general interview in section 5.1.2 (supra, p.22), the same CEO has agreed to be the guinea pig to test the prototype of the tool. To recapitulate, Jan Provoost is the CEO of a consultancy company called Stanwick. With approximately 30 employees across three different countries, they fall under the category of small companies (supra, p.7). They perform implementation consulting which means that they aid companies from start to finish in their projects to maximize involvement and acceptance. One of their activities as implementation consultants is strategy deployment. This is usually requested by bigger companies where they help in achieving the goals management has set. Over the years, they have adapted their strategy deployment methods they use in large companies to fit their own needs as an SME. One of the main reasons for the development of CHOOSE was to aid SMEs in thinking more strategically (supra, p.9). Although this company adopted a method to fit exactly their needs, making a comparison with CHOOSE would be very interesting. Furthermore the method they implemented has been very successful, so getting feedback from someone who works almost daily on strategic issues in their own and other companies would be very valuable. The case study was structured in the following way. First, enterprise architecture was introduced together with an explanation about CHOOSE. He didn’t use the term enterprise architecture but as an expert in the field, he was very accustomed to the terminology that was used. Consequently he easily understood the CHOOSE approach. Next, the CEO was asked to perform certain tasks on the application. He was asked to think aloud about what he thought and what he wanted to do. In literature, this is called a usability test. To test and refine an existing design, usability tests are the most effective way. In a usability test, a user is observed in a structured manner in a laboratory-like setting. While they perform certain prescribed tasks, the user is asked to think aloud. Here everything is of importance, like what do they expect that a certain button will do, what are they trying to do. Consequently, the evaluator can see what the user’s problems are, what is not as evident as expected, etc. (Rubin, 1994). The findings and refinements that were apparent will be discussed in the following subsection. Finally we had a long discussion about what he thought about the value of the application and the CHOOSE approach. These finding will be discussed after the application refinements. To better frame how he thought about the CHOOSE approach, the method
they developed to implement strategy in their SME is first introduced. The CHOOSE approach is then compared, offering valuable feedback as input for further refinement of the CHOOSE metamodel. His enthusiasm about the subject matter reflected in the fact that we spend almost six hours discussing various topics, ending our day long past the usual working hours.

### 7.1 Feedback on Form and Behavior

During the usability testing, the CEO was asked to perform certain tasks. Since a small presentation about EA and CHOOSE was already given, the guide about CHOOSE that is normally shown at the start was skipped (supra, p.43). This guide holds the same information as the presentation did, but in a more condensed format. One remark here is that the CHOOSE method was not yet explained. Because of the overlay with additional information (supra, p.41), it was immediately clear where to start and what to do. The overlay pointed out that step one was to build up the goal tree. The first thing he did was then go into that goal model. Here, he was also greeted by an overlay which showed what everything was and what each button did. He had brought a PowerPoint presentation with his companies’ goals so I could immediately ask him to add his six main goals. He immediately pressed the right button and started with the wizard to add a new goal. He filled in the goal’s name and description but then it wasn’t instantly clear that he had different steps to go through. On the bottom of the different steps, a graphical page indicator is added to make this more obvious (See Fig. 61).

![Page indicator](image)

Additionally, during the exercise, as we often had a discussion during the steps of the wizard, sometimes he forgot what goal he was adding. Starting from step two, the name is now shown in the top left corner (See Fig. 62).

After leaving the second step blank because there were no upper goals or conflicting goals, the second problem became apparent. When the application asked to provide links with the other models, he felt obliged to do so, even when I told him that these steps were optional and that links could be made afterwards. He still tried to add links with each goal he added. During the case studies to develop the CHOOSE method it became clear that it was easier to first construct a goal tree, actor tree and add some processes or project and then start linking the different models together (supra, p.13). It may be a good idea here, to leave out these steps, where links with the other models could be made. And only
give the option to link with the different models later. Before making drastic changes, more research should be done with multiple CEO’s. For now, a text line is added to the steps, stating that these steps are optional and that it is easier to first complete the individual models before linking them together (See Fig. 62).

The next task I gave was to work out one of the main goals in more detail. When the goal tree was still empty it was evident to push on the ‘add goal’ button. However, once these main goals were added and I asked to add some sub goals the next problem arose. He started pressing the existing main goal and then hold-press it to see if he could add the sub goals that way. But by pressing the goal, he just got a screen to either modify or delete that goal. Pressing on the ‘add goal’ button and then reselecting the main goal, adding it as an upper goal felt like an additional step. An additional option was added to the popup screen. While holding a goal, next to modify and delete, it is now possible to click ‘add lower Goal’ on the popup screen (see Fig. 63).

When I said to no longer fill in the three steps with links to other models we quickly completed one main goal up to three layers. Navigating through the tree was instantly understood and seemed to go very easy. In the same way, I asked him to construct his organogram and he quickly added two layers of actors. Then some of their projects and processes were added to the operations model. Then I asked him to revisit the goals model and start linking goals with actors and goals with processes. Here the next problem became apparent. During the process of adding a new goal, he already created some links as he felt obliged to fill out every step. Now, when he wanted to make a new link with an actor, the first thing he tried was pressing in the area where the existing links were. As soon as I showed him that he had to click modify, he was quickly able to perform the different links. During another case
study in the FEB at the University of Ghent, I got similar feedback. But here they expected when pressing on one of the links to be brought to that specific element in the appropriate model. More feedback from testing is required to decide what feedback is needed when pressing on the area where links are drawn.

Other questions and problems arose during this exercise but were more related to the CHOOSE approach itself. These will be further discussed in the next section where feedback about the CHOOSE approach is given. Generally the application was perceived to be user friendly and many actions like browsing through goals were instantly understood. Though some rough edges still exist and more research is needed to find the appropriate solutions.

### 7.2 Feedback on CHOOSE Approach

Every year in August, seven people come together in a series of meetings. More precisely, the CEO comes together with the six managers in the company. First, they revisit their vision, mission and values. These are refined if necessary but normally remain constant over time. To achieve their longer term vision and mission, they set forward, as they call them, ‘focus points’. These are their six most important goals (first level) which have similar dimensions to the ones seen in the balanced scorecard. These normally also remain steady over time. Next, they refine these six focus points in what they call, ‘critical goals’. These critical goals (second level of goal setting), in contrast with the focus points, change heavily year after year. For example, one of their focus points or main goals is to professionalize their marketing and sales. This year, as critical goals they defined two new critical goals. First they want to go to the market with a new Unique selling proposition, next they want to increase the fraction of large customers in their portfolio. To achieve these critical goals they define a series of subgoals (third level) belonging to the critical goals. They end up with three layers of goals and objectives. Even at the third level, these goals remain relatively high level. All of these ‘critical goals’ are set during brainstorm sessions. They use ishikawa diagrams as a tool to consider different possibilities. Note that this is a very organic process. After each meeting, everyone is assigned to work out some new possibilities which includes getting some bottom up feedback from the employees. Usually after three meetings, they agree on all of their critical goals that have to be achieved for the next year. Important here is that they already define quantifiable targets and metrics for each critical goal. Once management has performed this exercise, they give a presentation to all of the employees where everyone is asked to sign in on one of these critical goals. This offers their employees a sense of purpose and greatly enhances their cooperation. They work with different operational teams where a team leader is appointed. Some critical goals are deployed to these operational teams. Since the critical goals are still quite high level, these teams have a lot of liberty to come up with a game plan. In November, each of
the teams is asked to present all of their goals and their game plan. These team goals are called operational goals and some of these are linked to the critical goals. In the game plan, they propose a number of actions and projects to achieve their goals. Finally each employee then gets an individual sheet about what his/her objectives are for the next year. Some of these will be linked to the critical goals that are general for everyone, some will be linked to the operational team goals. Finally when everything is approved they draw up one sheet of paper called ‘the tactical plan’ were all of the projects are listed categorized according to the six main focus points and where the contribution of each individual becomes apparent and transparent.

A large overlap can be seen with the dimensions of choose. We can easily see that their method concerns the goals dimension, actor dimension and operation dimension (more specifically projects and actions). In the next sections, feedback that was given about each dimension will be discussed.

### 7.2.1 Goal Model

As described above, they set up 3 layers of goals (1. Main goals or focus points, 2 critical goals, 3 subgoals of the critical goals) which are still not narrowly defined. They leave the implementation over to their employees and give them enough freedom to come up with their own game plan. How the operational goals are realized and split whiting each team, is not of direct interest to the CEO. The ‘how’ after this third level is very organic and can change quickly. The CEO and management team rather concentrate on the implementation of the critical goals (unless the operational goals are not met). Periodically, the team that is assigned to a critical goal, reports what they are doing and whether they are on track to reach their targets. It would also be possible to add the operational team goals to the model but the added value would be very limited. What is interesting to him is to know who is assigned to which critical goal, and what the metric and target is of each critical goal. He furthermore felt that constructing different scenarios through the AND-OR relations was a bit redundant. This is why they held brainstorm sessions in august. They draw up an ishikawa diagram on the blackboard and suggest and discuss different possibilities. This is also a very organic process where the tool would not add much value. After the meeting, they summarize their findings on a one pager and assign everyone to work out several options before the next meeting. By the time they reach the final session they already considered all the pros and cons and end up with a balanced set of critical goals. So the alternatives were discussed and a decision of the final critical goals was taken. Adding these alternatives felt redundant. Furthermore he felt that instead of constructing alternatives it would make more sense to split goals into two categories: goals they were sure of and all of the other possibilities that still had to be looked into. The same reasoning was made about conflicting goals. As they ended up with a balanced set of goals, he felt no need for adding conflicting goals. What he really would have
liked to see in the model was the possibility to assign targets with different metrics and a deadline to the goals. Now the description attribute was used to enter this data. In conclusion, he feels being able to construct a goal tree is interesting. Now they use PowerPoint to store their goals. In the application, these goals could be seen in a more holistic way. However instead of having relations to construct different alternatives and conflicting goals he feels that being able to store metrics and targets for different time horizons would be much more interesting.

7.2.2 Actor Model

The company has just three hierarchical layers. The first layer is the CEO with below him six other managers in the second layer, together they decide about the critical goals discussed before. In the third layer finally, whether it are consultants or secretaries, they all fall under the third layer. As the company is still relatively small with 30 employees and the structure is very flat, he still knows everyone by name. When asked to construct the organogram of the company, he quit after constructing the first and second layer. In the goals model where the different teams were free to set their own targets (in line with main goals and critical goals), the way targets were deployed did not interest him. These team goals were followed up by the team leaders. Similarly, the six members of the management team are also the only ones he is really following up on. The members of the management team in turn are responsible for the employees and they decide about what targets they need to achieve. For the CEO, knowing some personal information (competences, project experience and interests) about his people was more important than what goals the team leaders set for them or what specific projects they were working on. So while the company was still small he felt like the organogram as such wasn’t too interesting and that he still knew a lot of what was going on at the individual and personal level. Once they grew it could become interesting to look up some personal information about his people. Again rather than having different actor refinements, it would make the actor tree more interesting to see more information about the (human) actors themselves. As will be discussed later, the link between the other models did interest him. In conclusion, the actor model is very interesting to later assign them to the other models, but the organogram would only become interesting if more information about actors could be added.

7.2.3 Operation Model

The company has only nine predefined processes. Everything they do to achieve their goals is either defined as an action, when it is something that can be done in less than two month, or otherwise it is defined as a project, when they predict it will take longer or be more complex. As the CEO is only directly responsible for his management team, knowing which projects they are working on or what actions they are taking was much more interesting than adding the companies’ processes to the application. Every process had some operational goal with accompanying key performance indicators.
(kpi’s). Every month, he receives a printout of their kpi’s and how they evolved. He feels that this is more than enough information. Putting these into the application would not add any value to him. Though, if the kpi’s were not evolving as planned, these operational goals would become critical goals in the next strategic meeting in August. Then, some projects and actions would be started to achieve this new critical goal. Furthermore it was often not clear what the difference was between the how question to construct the goal tree and the accompanying process. Especially adding a process to achieve a high level goal was unclear. One remark was made about the usage of RACI to link projects or processes with actors. In their experience as consultants, RACI of RASCI often led to misinterpretation about what exactly someone’s role was. More specifically, in practice people are often unsure about the specific semantics of RA(S)CI. People often don’t understand what the difference is between who is ‘accountable’ and who is ‘responsible’. Instead, they classify people as either ‘LW’ (leading the work) or just ‘W’ (work). As the application allowed to define a new type of link this was not a problem during the modeling exercise. Because all of the information is available to everyone in the company, everyone can know everything. Consequently ‘I’ and ‘C’ are not used. In conclusion here, being able to link the projects and actions with actors and critical goal was interesting (see below). Adding processes to the application were not seen as value adding.

7.2.4 Object Model

While their own strategic method has a lot of overlap with the existing CHOOSE model dimensions, they don’t in any way use any information about objects. How operational goals are achieved with what inputs is up to his people. He also found the concept quite hard to grasp. Only for the first two goals that were added he filled in some object but for all the other goals he quickly gave up. He was also not sure what could be the added value to him to know these objects. He felt like this part of the model was redundant.

7.2.5 Links Between Models

In the beginning of the interview, the managers printed out all available information they stored about their strategy. He showed me the following documents: a PowerPoint with their critical goals with metrics and targets for them, a PowerPoint of each of the six management members describing what their objectives were for the following year and finally he had a one pager with their tactical plan. This tactical plan is a summary of the projects that were agreed upon by the employees and the management, which were categorized according to their six focus points. If the application and model would be slightly altered where each dimension had more attributes, these three documents could easily be drawn up once the links were added. So by just inputting his critical goals, the first and second layer of his organogram, their projects and linking these, he would be able to get all of the information.
he needs. So especially after the links between the models were added, the manager did see a lot of added value to having some integrated approach.

7.2.6 Conclusion

In conclusion, while using the application, the added value became apparent when we brainstormed about different ways of output that could be extracted from the application: drawing up goal sheets for individual employees with their targets and projects for the following year, making a printout of critical goals with targets and metrics and constructing a tactical plan where all projects are appointed to the six focus points of their company. Especially in thinking about this output, the power of the model became clear. With further fine-tuning of the application and the model, it could be something that could really add value according to the CEO of Stanwick.
In the last chapter of this master thesis, the application is presented. In the first section, a complete overview of the application will be shown. During the case study, as described in the previous chapter, a manager was asked to perform certain tasks. Similarly, the application will be presented by going through several tasks, visualized with screen shots. Some design rules, described in Chapter 5 should become more clear during the presentation (supra, p.19). The chapter will be concluded by revisiting the requirement definition (supra, p.27). A requirement check is done to see whether all of the requirements have been satisfied.
8.1 Application Overview

When professional applications are started for the first time, it is frequently very daunting and requires time and effort to get started (especially for managers who are not technology savvy and don’t want their precious time to be wasted). Efforts have been made to make the acquaintance with the program as pleasant as possible (supra, p.40, p43). When starting up the application, the user is presented with a guide, consisting of 9 pages. This guide introduces EA and the different dimensions of CHOOSE. On the first page of the guide, the purpose of the program is shortly explained (see Fig. 65).

![Fig. 65 Page one of CHOOSE guide](image-url)
Next, the different dimensions of CHOOSE are explained. Two pages explain each dimension. On Fig. 66, the two pages explaining the goal dimension are shown. In the first screenshot on Fig. 66, the CHOOSE metamodel is utilized to explain what the goal model entails (infra, p.12). In the screenshot below, the CHOOSE method is used. In this regard, the user is asked to think about his main goals, according to the four dimensions of the balanced scorecard. The purpose of allowing the user to already fill in some information, is to avoid presenting the user a blank canvas when the program starts (supra, p.41). Similarly, the guide proceeds with the other three dimensions of CHOOSE.

Fig. 66 CHOOSE guide step two and three
In the final step of the guide, a roadmap of how to proceed is shown (top screenshot Fig. 67). This roadmap is based on the CHOOSE method (supra, p.13). After the guide is finished, the user arrives at the start screen. Because this is the first start-up, an overlay is drawn on top of the start screen to show what every button does. Furthermore, it summarizes the roadmap, explained on the last page of the guide (bottom screenshot Fig. 67) (supra, p.41). To get back to the guide, the user can click on a question mark, shown on the bottom right of the start screen. The user is now ready to get started.
The start screen is playful, while remaining simple. It has to entice the user to get started with the program. As discussed, a beautiful design can improve the perceived ease of use (supra, p.25). In Fig. 67, the user was invited to start with the goal model. When we get into this model, another overlay is shown pointing out what everything does. Again, this is only shown at the first start-up, but can be recalled when pressing the question mark button. Notice that we don’t present the user with a blank canvas. The four goals, added during the guide are already shown (supra, p.41) (See Fig. 68).
An important aspect of the tool is to construct a goal tree. The showcase this functionality, an extract from the case study is presented. Going back to the case study, one of Stanwick’s goals was to increase turnover in both their Belgian and French branch. When you look at the goal path at the top of Fig. 69, you can see this goal was needed to achieve the main goal ‘Maintain organic growth’. One way of increasing their turnover in both France and Belgium, is to increase their name recognition. By pressing and holding ‘Increase turnover BE’, we get a menu where we can choose ‘Add lower Goal’. This will start a wizard that allows us to add a new goal. In Fig. 69, the first step is shown, where the name and description can be added. Other attributes like targets or deadlines could be added in the future.
In the next step, the upper goal ‘Increase turnover BE’ is already entered automatically. But Stanwick also wants to increase its name recognition to increase the turnover in France. After navigating to this upper goal, by clicking on the right arrow, this goal can be added as an upper goal. A menu asks whether this has to be added as a new alternative or not. In this case, we need to achieve both ‘name recognition’ and ‘business development’ to achieve ‘Increase turnover FR’. Notice how abstraction is made from the CHOOSE metamodel terminology (AND/OR refinement) and the user is presented with natural language. Similarly, conflicting goals can be added by navigating at the left and clicking the appropriate right arrow.

Fig. 70 Add a lower goal: Step 2
In the third step of the wizard, actors can be assigned to the goal we’re adding. These steps are optional though, as it was advised to build up each model separately first. For this reason, in Fig. 70, a ‘double arrow’ was added in the second step of the wizard, to skip these links with other models and immediately go to the end of the wizard. For this example, part of the other models were already built up. Similar to how an upper goal was added in the previous step, an actor can be linked with a goal. In this case, we navigate to Karine Roodhooft who was assigned to this goal by Stanwick and click the right arrow. A menu is shown to select what type of link the actor has with the goal. If users are unsure, they can select ‘Just add’ rather than being obliged to make a choice. In Fig. 71, on the bottom screenshot, the flexibility of the search algorithm is shown. Although we are in the goals models, the search is contextual and will now search through the actor database. Finally, notice that the user never has to switch screens to make links with other models. The user remains in his ‘flow’.

Fig. 71 Add a lower goal: Step 3
Another feature closely related to this concept of flow, is that users can already create an actor, operation or object from within the goal model. For example, to improve name recognition, Stanwick started a project to redesign their website. Unfortunately, this was not yet added to the operations model. We could leave the fourth step blank, finish the wizards, add the operation in the operation model and then link the operation to our goal. But this would heavily disturb a user’s flow. We have allowed the user to add elements from other models by typing in a name (see top screenshot Fig. 72). Later, we will discuss how this elements can be further completed. In the fifth step, a link can be made with the object model. Again, this is done similar to the previous steps. In this case, our goal ‘Name recognition’ is concerned with the object ‘Website’.

Fig. 72 Add a lower goal: Step 4 and 5
In the final step, a summary is given of what is entered during the previous steps. Everything can be saved by pressing the bottom right ‘save’ button. Thereon, the user is presented with the final result (bottom screenshot Fig. 73). On the right hand side, a visualization is drawn, where links with the different models and the upper goals are shown. This visualization can easily be augmented. For example, we could check whether there are any links between the other models. We could then see that Karine Roodhooft is responsible for redesigning the website and that the website serves as input and output for the redesign project. More research is needed to improve this part though.

Fig. 73 Add a lower goal: Final step and end result
Another important aspect of every program is that it has to be easy to modify and delete existing elements. We will showcase the modify functionality by modifying the operation ‘Redesign website’ that was added during the wizard. We want this operation to be specified as a project, add a description and make links with other models. To go to the operations model, we press the back button (Android specific button, not shown on picture) to get back to the start screen and go to the operation model. As you can see on Fig. 74, the operation ‘Redesign website’ we created before, is stored under ‘Unassigned Operations’. At the top right of the screen, the notification center also notifies the user there are uncompleted operations. More on this notification center is explained later. To modify the operation, pressing and holding the operation will show a menu where ‘Edit Operations’ can be selected.

Fig. 74 Modifying an element
After clicking ‘Edit Operations’ in Fig. 74, the user can select what exactly he wants to modify (top left screenshot of Fig. 75). In our case, we first want to add a description and define this operation as a project (top right screenshot of Fig. 75). The end result is shown on the bottom screenshot of Fig. 75. Note that the notification has disappeared at the top right.
Finally, when the different models are built up, the notification center warns the user if there are any stop criteria that are not satisfied (supra, p.13). If every stop criteria is satisfied, the user is done modeling. Another important part, that has not been covered, is the ‘general output’ part. By going back to the start screen, the general output can be accessed on the bottom left (bottom screenshot Fig. 76).

Fig. 76 Stop criteria
Unfortunately, in the output section, only the visualization of a RACI chart has been implemented. In the case study some other outputs have been proposed: a tactical plan, critical goal list with targets and an individual sheet for every employee with his tasks, targets and goals for the following year. More research is needed to find out what output can really add value to managers. The section that can really add value, regrettable isn’t adding much value yet. But by pulling the handle on the left hand side of the screen, more output can be added in the future. On the bottom screenshot of Fig. 77, note the little share button. This will be the last feature that will be shown in this tool presentation.
By clicking on the share button, different sources that can handle the generated output are shown. A CSV file is created that can be opened in spreadsheet software, like Microsoft Excel. In the example on the bottom screenshot of Fig. 78, an email with the RACI chart of Stanwick was sent to their CEO, Jan Provoost.
8.2 Requirements Overview

To conclude the tool presentation chapter, we will check whether or not the requirements that were set in section 5.3 are satisfied (supra, p.27). To recapitulate, our requirements were categorized according to: data requirements, functional requirements and other requirements.

8.2.1 Data Requirements

Data requirements were defined as “all of the objects and information the user wants from our tool”. We came up with the following requirements: The application needs to:
- Contain all relevant information about actors/goals/operations/objects;
- Contain the links between the different models;
- Give strategically important output;
- A holistic overview of the company;
- A description of how something is done.

The first requirement has been achieved to a great extent. This requirement was deliberately defined very broad though. For example, during the case study in Stanwick, the CEO missed the possibility to set targets, metrics and accompanying end dates for goals. While more case studies are performed, the metamodel, representing the relevant information, can be further refined. So far, the application supports the goal model and operation model completely, as they were defined in the latest metamodel (supra p.11). The actor and object model are supported too but don’t yet include all of the refinements proposed in the latest metamodel.

The second requirement has been achieved completely, as all of the links that were proposed in the metamodel are supported.

The third requirement is only been attained to a lesser degree. Within the different models, the visualization of the links between models is of strategic added value. The real added value, has to come from the output section. This has only been developed to a lesser extent.

The fourth requirement is accomplished by a combination of: easy navigation through hierarchical trees and the links with other models. These links are shown on the right hand side of the application. The final requirement also needs further refinement. The description of how something is done is achieved by the visualization on the right hand side.

8.2.2 Functional Requirements

The functional requirements are all of the actions that have to be possible with the application. It has to be possible to:
- Add, remove and modify goals/actors/operations/objects;
- Navigate easily within and between the different models;
- Generate automatic visualizations;
- Output data in popular formats like PowerPoint/Excel/Word.

While presenting the tool in this chapter, all of these functionalities have been showcased. Further improvements to the functionality could be obtained through feedback in case studies. Especially the automatic visualizations still need improvement. Once the language ingredient of the CHOOSE approach is finalized, these visualizations can be updated.

8.2.3 Other Requirements

Business Requirements:
The most important business requirement was that it had to be cheap to implement. First, Android tablets come in all price categories and can be very cheap. Second, the installation of the application does not induce any costs. And finally, a lot of effort has been put in making the application usable without help from experts, since these are too expensive for SMEs to hire. The ultimate aim is that the application can be downloaded in an app store and that managers can get started right away, without any help. More case studies are still required to bring the application to this point though.

Experience Requirements:
During the application design, several guidelines have been taken into account to make a beautiful design. By playing with different colors, using a lot of graphics and having a playful start screen, we aim to increase the perceived ease of use. Furthermore, abstraction was made from the CHOOSE model by representing everything in natural and easy to understand language. This helps the managers feel in control and not feel stupid. Finally, a complete interaction framework with interaction design principles was built up to make an easy to use application (supra, p.34). During the case study, a first usability test was done and initial improvements were made to increase the ease of use. Although the application was already considered easy to use by the first CEO, more usability testing is needed to make further improvements.

Technical Requirements:
After a study about a manager’s behavioral variables, we concluded in section 5.4.1 that a portable device, like a tablet, could perfectly fit into his lifestyle (supra p. 28). Another important requirement was that the application had to be safe. We achieved this by prohibiting the application from entering the internet (supra, p. 31 and p.49). Finally, the application architecture had to be flexible to quickly adapt the CHOOSE metamodel refinements. In the section about maintainability, we described several actions that were taken to create a flexible architecture. In this regard, we noted that there is still a lot of room for improvement.
Conclusion and Future Research

After years of economic crisis, companies struggle to keep their heads above water. “Another record month for bankruptcies”. It is daily fare to see these kinds of headlines. Especially SMEs are vulnerable, as large companies have a lot more capabilities to resist the economic downturn. By using EA, large companies have greatly improved their flexibility and overall performance.

The aim of the EA approach CHOOSE, is to bring the same advantages EA delivers to large enterprises, to SMEs.

In this master thesis, a tool is presented that supports the CHOOSE approach and delivers these advantages to SMEs. By using goal-directed design in the tool development process, a radically different approach is proposed. A vital aspect of the approach is that it takes into account the user’s goals and motivations. By starting the tool development from scratch and doing proper research about the end user, some important insights were obtained. The first important finding revealed that the user would be the CEO. In contrast with larger companies where EA experts are hired, SMEs lack the means to hire these experts. While doing research on managers, it was clear that they had very little time to work on strategic issues and were not technology savvy at all. During the design of the application, we took into consideration all of these insights and translated them to several requirements. For example, one requirement demanded that the tool would have to fit into a manager’s busy schedule. To comply with this requirement, a tablet application has been proposed.

In section 8.2 of the previous chapter, we check to what extent all of the requirements have been achieved. And although they have been achieved to a certain extent, some clear limitations became visible. First, as I have programmed this tool without a background in computer science, many improvements can be made on a technical level. Next, a good design requires input from users and a lot of testing in practice. After each iterative round of testing, the design can be refined. Unfortunately, in this master thesis, input from users and iterative rounds of usability testing have been limited. Much more background research and case studies are required to further improve the application. Still, the first case study has already been very promising. The tool is already found to be user friendly and the manager was enthusiastic about using a tablet application to work on strategic issues.

The contribution to the field of EA is, that by using the methodology proposed in this master thesis, many more stakeholders could be involved in EA. For the first time, EA has been made accessible for managers. But the goal-directed design approach could easily be expanded. Further research could find ways of better involving other stakeholders from small to large companies with easy to use applications. For example, employees could use a smartphone application to get an overview of their tasks and responsibilities, connected to the companies’ goals. This can offer employees a sense of
purpose. Shareholders on the other hand, may be interested in what the companies’ long term goals are and if targets are being achieved. This could help them to focus more on the long term rather than on short term financial figures. Further research can be done to include these stakeholders in EA.
- Daneva, M. and van Eck, P. (2007). "What enterprise architecture and enterprise systems usage can and can not tell about each other."


Gribbons, D. W. M. "Knowledge-infused design, the “ultimate solution” to product usability." Help 99: 153-156.


Jackson, M. A. (1975). Principles of program design, IMA.


Maes, R. (1999). A generic framework for information management, Universiteit van Amsterdam, Department of Accountancy & Information Management.


- W3C. "W3C contrast ratio formula." from <http://www.w3.org/TR/AERT#color-contrast>.