CAUSAL MODELS FOR WELL-BEING
Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction

Steven Bosems
Causal Models for Well-Being

Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction

Steven Bosems
Graduation Committee

Chairman and secretary
prof. dr. P. M. G. Apers (University of Twente)

Supervisor
prof. dr. R. J. Wieringa (University of Twente)

Co-supervisor
dr. ir. M. J. van Sinderen (University of Twente)

members
prof. dr. ir. L. J. M. Nieuwenhuis (University of Twente)
prof. dr. M. M. R. Vollenbroek (University of Twente)
prof. dr. M. A. Neerincx (Delft University of Technology)
prof. dr. M. Goedicke (University of Duisburg-Essen)
prof. dr. O. Pastor (Polytechnic University of Valencia)

CTIT Ph.D. Thesis Series No. 17-453
Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE, Enschede, The Netherlands

SIKS Dissertation Series No. 2018-03
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This publication was supported by the Dutch national program COMMIT (project P7 SWELL).

ISSN: 1381-3617 (CTIT Ph.D. Thesis Series No. 17-453)
DOI: 10.3990/1.9789036544603

Copyright ©2018, Steven Bosems, The Netherlands

All rights reserved. Subject to exceptions provided for by law, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the copyright owner. No part of this publication may be adapted in whole or in part without the prior written permission of the author.
CAUSAL MODELS FOR WELL-BEING

KNOWLEDGE MODELING, MODEL-DRIVEN DEVELOPMENT OF CONTEXT-AWARE APPLICATIONS, AND BEHAVIOR PREDICTION

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. T. T. M. Palstra,
on the account of the decision of the graduation committee
to be publicly defended
on Thursday the 1st of February 2018 at 14:45

by

Steven Bosems
born on the 3rd of December 1985
in Apeldoorn, The Netherlands
This dissertation has been approved by:

Supervisor    prof. dr. R. J. Wieringa
Co-supervisor  dr. ir. M. J. van Sinderen
ABSTRACT

In recent years, we have witnessed an increase in the capabilities of smartphones. Not only are these portable communication devices becoming increasingly powerful, they are equipped with a growing number of sensors that allow them to measure the properties of the world around them. Applications running on these smartphones can benefit from these sensors if developers choose to make them context-aware. However, applications that are built to operate in a wide variety of situations have to be able to cope with and respond to any combination of measured context factors. Anticipation of these different contexts at design time is challenging. For certain domains the requirement for the application to behave exactly as anticipated at design time is key. One such domain is that of well-being. Problems predicting the run-time context may cause the application to behave in other ways than intended.

Current research efforts primarily focus on adding features to traditional software development methods to deal with the complexity of the domain of context-awareness. They aim to define a full set of requirements and design an application that is to satisfy these requirements. Furthermore, there are research directions aiming to solve the technological problems of context-awareness without dealing with the user-centric elements needed for well-being systems.

We see the field of well-being as consisting of variables and relations between them. To ease the documentation of the variables that the context relevant to the well-being application is made up of, we have designed a domain specific modeling language. In addition to variables, the models created in this language capture variable properties, such as their dimension and their normal range, and causal relations between them, modeling what happens to variable values if one variable in the context is increased or decreased. The modeling language is called the Dynamic Well-being Domain Model language and was developed to be user centric, allowing developers to model the objective well-being context of the user. Although it was designed with the well-being domain in mind, it can also be used in other domains where context variables play an important role.
The contributions of this dissertation are threefold. Firstly, we present a reference model for the well-being domain, relating physical and mental well-being. This model was validated by experts from both fields. We discuss how this model was created. Secondly, we describe a model-driven process for the creation of context-aware well-being systems. This process uses models of the well-being context as an input, and is partially automated using model transformations. Finally, we describe a structured analysis method that can be used to predict the behavior of context-aware systems based on their domain models. This prediction can be made at design time, allowing designers to evaluate the utility of the application based on the model, and preventing run-time problems. This method was validated by analyzing three context-aware well-being applications.

We find that the Dynamic Well-being Domain Model language can be understood by domain and technology experts alike. Its usage in a model-driven development process was deemed useful, and the ability to reason over future contexts was found to be both powerful and reliable for the cases used in the validation.
SAMENVATTING

In recente jaren zijn de mogelijkheden van smartphones gegroeid. Deze draagbare communicatie apparaten worden niet alleen steeds sneller, ze worden ook uitgerust met een groeiend aantal sensoren die ze in staat stelt om de eigenschappen van de wereld om zich heen waar te nemen. Applicaties die op deze smartphones draaien kunnen profijt hebben van deze sensoren als ontwikkelaars ze context bewust maken. Echter, aangezien applicaties worden ontwikkeld om in een grote verschijdenheid aan situaties te werken, moeten ze om kunnen gaan met elke combinatie van gemeten omgevingsfactoren. Om deze verschillende situaties tijdens de ontwikkeling van de applicatie te anticiperen is erg lastig. Voor sommige domeinen is van het grootste belang dat applicaties zich exact gedragen zoals tijdens het ontwerp bedacht is. Welzijn is een voorbeeld van een dergelijk domein. Wanneer het moeilijk is om deze voorspelling goed uit te voeren, kan dit er voor zorgen dat de applicatie ander gedrag gaat vertonen dan initieel gedacht.

Huidig onderzoek richt zich voornamelijk op het toevoegen van mogelijkheden aan bestaande software ontwerp methoden om zo om te kunnen gaan met de complexiteit van context bewuste systemen. Ze proberen over het algemeen een volledige set gebruikers-eisen te verzamelen, om zo een applicatie te ontwikkelen die aan al deze eisen voldoet. Een andere onderzoeksrichting is het vinden van oplossingen van de technische problemen van context bewuste applicaties, terwijl er niet wordt gekeken naar de menselijke kant. Deze focus is wel noodzakelijk voor welzijnssystemen.

Wij zien dat het welzijnsdomein uit variabelen en relaties tussen variabelen bestaat. Om het documenteren van variabelen waaruit een situatie is opgebouwd te vereenvoudigen, hebben we een domein specifieke taal ontworpen. De modellen die in deze taal gemaakt worden kunnen domein variabelen en hun eigenschappen representeren, zoals hun meeteenheid en bereik van normale meetwaarden, en causale relaties tussen deze variabelen, om zo te modelleren wat er gebeurt met de waarde van een variable wanneer de waarde van een andere variabele in de context groter of kleiner wordt. Alhoewel deze taal is ontworpen met het welzijnsdomein in gedachte kan
Samenvatting
deze ook gebruikt worden in andere domeinen waar context variabelen een belangrijke rol spelen.

De bijdragen van dit proefschrift zijn drieledig. Ten eerste presenteren we een referentie-model voor het welzijnsdomein waarin we fysische en mentale welzijn relateren. Dit model is gevalidateerd door experts uit beide domeinen. We bespreken welk process we doorgaan hebben om dit model samen te stellen. Ten tweede beschrijven we een model gedreven ontwikkelmethode voor het maken van context bewuste welzijnssystemen. Dit process gebruikt modellen van de welzijnscontext als invoer en is gedeeltelijk geautomatiseerd door middel van model transformaties. Ten slotte beschrijven we een analyse methode om het gedrag van context-aware systemen te voorspellen. Deze methode is gebaseerd op het bestuderen van de domein modellen van deze systemen. Deze voorspelling kan tijdens het ontwerp van de systemen reeds gedaan worden, wat fouten tijdens het uitvoeren van de applicatie kan voorkomen. We hebben deze methode gevalidateerd door hem toe te passen op drie context bewuste welzijnssystemen.

We zijn er achter gekomen dat de Dynamic Well-being Domain Model taal begrepen kan worden door zowel domein- als technologie experts. Het gebruik van deze taal in een model gedreven ontwikkelproces werd nuttig bevonden en de mogelijkheid om te redden over toekomstige situaties was zowel krachtig als betrouwbaar in drie afzonderlijke onderzochte gevallen.
ACKNOWLEDGMENTS

In front of you lies the summary of over six years of my PhD work. During this time multiple papers were written, published, and rejected, application code was written in Java, ATL, QVTo, LaTeX, and HTML, experts were interviewed, and many meetings were organized, all to the goal of getting the words of this dissertation on paper.

They say travel is not about the destination, but all about the journey. I think the same can be argued for a PhD dissertation, and this journey would not have been the same without the people met along the way. I would like to start by thanking my supervisors Roel and Marten. Roel, despite our meetings being infrequent during the first year of my contract, I think we caught up quite nicely during the later years. Our discussions were always fruitful, having me leave with renewed inspiration. Marten, our weekly meetings were always enjoyable, often mixing discussions that would shape my work to the form it is today with pleasantries.

Next I would like to thank everyone from the Information Systems/Information and Software Engineering/Services, Cybersecurity and Safety group (starting my contract at the IS group, I have seen two group merges, and four personal moves of office over the course of four years, making the UT a truly dynamic working environment indeed) for their kindness during my time there. I want to thank Suse and Bertine for their help during every phase of my PhD project. By now, I am pretty sure that all work in the group would come to a grinding halt if it were not for them. Elmer, Jan-Willem, Marco, Ali, Eleftheria, Prince, and the rest, I would like to thank you for the cheerful lunches and coffee breaks, they proved to be welcome distractions.

Working in a project always gives you additional colleagues. Being employed by the commit/swell project, I had the opportunity to work with a great group of people from TNO, Philips, Roessingh R&D, Radboud University and many more. It was a terrific experience to collaborate on an integrated tool that was to improve knowledge worker well-being, and I learned a lot from every one of you. Of those working in SWELL, I would like to pay special thanks to the other PhDs, Reinoud, Saskia, Maya, and Shoaib. Starting all
acknowledgments

at about the same moment with similar journeys, it was good to have some people around to share experiences with, both the joys and the woes. I hope we can collaborate in future projects and employments.

A good work-private balance is important, as stressed by commit/swell, so time not spent working was spent as enjoyable as possible: with friends. Plenty days were enjoyed scuba diving at ZPV Piranha, either teaching new (Advanced) Open Water Divers, or simply having fun with friends. I want to thank all the Dive Professionals and other divers of Piranha for what have been (in general) very relaxing dive days. Robbert, Bas, Siebren, Arjan, Wendy, Susan, Geert, and Ilse deserve special mention for being amazing friends both in and out of the water. I will always have fond memories of our diving holidays to Gozo and Bonaire, and for the countless hours spent playing every board game we could get our hands on.

What would acknowledgements be without a paragraph reserved for the parents. John en Mieke, you have always given me love and support, and helped me where possible through ever part of my life. It is hard to put in words how thankful I am for that.

The final place in a word of acknowledgments is often the most important one. This is also true for these. Renske, I want to thank you for all the love and support you have given me the years. You were there to share happy moments, and you were patient with me if I complained about things not working out the way I wanted to. You did not always like it when I went away for conferences, leaving you alone, and later with Bailey, at home. I promise you this: in the future, we will be traveling together.

Steven Bosems
Borne, January 2018
CONTENTS

ABSTRACT i
SAMENVATTING iii
ACKNOWLEDGMENTS v

I PROBLEM INVESTIGATION 1
1 INTRODUCTION 3
  1.1 Motivation 3
  1.2 Research context 5
  1.3 Research Methodology 5
  1.4 Research questions 6
  1.5 Contributions 8
  1.6 Scope 10
  1.7 Outline 11
2 CONCEPTUAL FRAMEWORK 13
  2.1 Context-awareness 13
  2.2 Well-being 16
  2.3 Conclusions 18
3 METHOD REQUIREMENTS 21
  3.1 Problem investigation 21
  3.2 The product 22
  3.3 The system 22
  3.4 The containing system 25
  3.5 The wider environment 26
  3.6 An example context-aware application 28
  3.7 Discussion 30
  3.8 Conclusions 30
4 STATE OF THE ART 33
  4.1 Requirements engineering 33
  4.2 Context-aware application design 39
  4.3 Well-being support 48
  4.4 Conclusions 54

II KNOWLEDGE MODELING 57
5 A DYNAMIC WELL-BEING DOMAIN MODEL LANGUAGE 59
  5.1 Language requirements 60
  5.2 Notation of Dynamic Well-being Domain Models 61
## Contents

5.3  Discussion 71  
5.4  Conclusion 74  
6  A well-being domain model 75  
6.1  A generic well-being domain model 75  
6.2  Construction process 80  
6.3  A step counter application DWDM 88  
6.4  Discussion 93  
6.5  Conclusion 94  
7  Validating the generic domain model and the suitability of the DWDM language 97  
7.1  Research design 97  
7.2  Research execution 101  
7.3  Results 103  
7.4  Discussion 115  
7.5  Conclusions 119  

### III  Model-driven development of context-aware well-being systems 121  
8  Model-driven development using DWDMs 123  
8.1  Model-driven development 123  
8.2  Modeling levels 125  
8.3  Development process 126  
8.4  Process automation 140  
8.5  Discussion 144  
8.6  Conclusion 147  

### 9  Development of applications using DWDMs 149  
9.1  Research problem 149  
9.2  Research design 150  
9.3  Research execution 157  
9.4  Results 158  
9.5  Discussion 165  
9.6  Conclusions 167  

### IV  Behavior prediction 171  
10  Context behavior analysis using DWDMs 173  
10.1  Analysis process 173  
10.2  Discussion 178  
10.3  Conclusion 179  
11  Application behavior prediction using DWDMs 181  
11.1  Research problem 181
11.2 Research design 182
11.3 Research execution 184
11.4 Results 186
11.5 Discussion 197
11.6 Conclusions 198

V contributions and conclusion 201

12 conclusions and future work 203
12.1 Answers to research questions 203
12.2 Future work 212
12.3 Conclusions 216

VI appendix 219
A dwdm editor 221
A.1 Technologies used 221
A.2 Model Transformation 223
B reference well-being domain model 231
B.1 Initial model 231
B.2 Improved model 236
C subject material reference model validation experiment 239
D subject material application development experiment 247
D.1 Introduction 247
D.2 Dynamic Well-being Domain Model and editor 247
D.3 Case description 248
D.4 Activity Coach domain model 250
E tno/swell fishualization 253
E.1 Overview 253
E.2 Goal 253
E.3 Means 253
E.4 Sensors 254
E.5 Domain model 254
F philips/swell mbeats 257
F.1 Overview 257
F.2 Goal 257
F.3 Means 257
F.4 Sensors 258
F.5 Domain model 258
Contents

G ROESSINGH RESEARCH AND DEVELOPMENT ACTIVITY
COACH 261
G.1 Overview 261
G.2 Goal 261
G.3 Means 261
G.4 Sensors 262
G.5 Domain model 262

References 279
Previous publications by the author 282
SIKS DISSERTATION SERIES 283
LIST OF FIGURES

Figure 1  A simple CLD model [105]  62
Figure 2  The DWDM metamodel  63
Figure 3  DWDM variable  64
Figure 4  Causal relations  68
Figure 5  A simple causal path  69
Figure 6  Two causal loops  70
Figure 7  A norm associated to a variable through a norm relation  71
Figure 8  Reference commit/swell domain model: Physical well-being  77
Figure 9  Reference commit/swell domain model: Mental well-being  79
Figure 10  Questionnaire: physical well-being  104
Figure 11  Questionnaire: mental well-being  105
Figure 12  Results of the well-being domain model questionnaire  114
Figure 13  Improved reference commit/swell domain model: Physical well-being  117
Figure 14  Improved reference commit/swell domain model: Mental well-being  118
Figure 15  Overview of the partially automated development process.  127
Figure 16  Class Diagram of the reference architecture  128
Figure 17  Component interaction when obtaining data  129
Figure 18  Component interaction when performing interventions  130
Figure 19  The PIM level Class diagram for the step counter application  131
Figure 20  Dynamic application structure deduced from the processing type property  132
Figure 21  Dynamic application structure derived from norms  135
Figure 22  The Activity diagram for control flow derived from the Physical Activity variable norm  136
List of Figures

Figure 23    The Activity diagram for the `getValue()` function of the `PhysicalWellbeingConcept` class 137
Figure 24    The PSM level Class diagram for the step counter application 139
Figure 25    The structure of the DWDM tool 140
Figure 26    The model-driven process 142
Figure 27    Metamodeling stack instance for the DWDM to UML model transformations 143
Figure 28    The Eclipse GMF Dashboard [110] 169
Figure 29    Variables influenced by the TNO/SWELL Fishualization when introduced into the context 188
Figure 30    Variables influenced by the Philips/SWELL mBeats application when introduced into the context 191
Figure 31    Variables influenced by the RRD Activity Coach when introduced into the context 195
Figure 32    Reference well-being model - Physical 232
Figure 33    Reference well-being model - Mental 234
Figure 34    Improved reference well-being model - Physical 236
Figure 35    Improved reference well-being model - Mental 237
Figure 36    The Activity Coach interface 249
Figure 37    The DWDM for the reduced Activity Coach case 251
Figure 38    DWDM for the TNO/SWELL Fisualization 255
Figure 39    DWDM for the Philips/SWELL mBeats application 259
Figure 40    DWDM for the RRD Activity Coach 262
LIST OF TABLES

Table 1  Research questions  7
Table 2  Dissertation outline  9
Table 3  Classification of context-aware systems  15
Table 4  Method requirements  29
Table 5  Requirements satisfied by requirements engineering methods  40
Table 6  Requirements satisfied by development methods  49
Table 7  Physical well-being support applications  51
Table 8  Mental well-being support applications  52
Table 9  DWDM construction: From scratch  81
Table 10  DWDM construction: Derived model  84
Table 11  Questions asked to establish reference well-being model validity  100
Table 12  Model validation experiment participants  102
Table 13  Levels of the OMG Meta-Object Facility [76]  125
Table 14  Exit questionnaire model-driven process experiment  155
Table 15  Model metrics for evaluating changes made to models by participants  156
Table 16  Model-driven development experiment: exit questionnaire results, level of experience on a 5 point scale  159
Table 17  DWDM analysis questions  185
Part I

PROBLEM INVESTIGATION
1 INTRODUCTION

“A whole is that which has beginning, middle, and end.”
– Aristotle (philosopher)

1.1 MOTIVATION

During the last few years, the miniaturization of computer chips has increased the computing power of portable and embedded devices used by the general population, up to the point that high performance smartphone devices have become the rule rather than the exception. These mobile computing platforms have more capabilities than the portable telephones known from the 1980’s and 1990’s: besides being able to make phone calls and send text messages, smartphones offer the user ever increasing functionality, such as high quality media consumption, gaming, and navigation. In addition, these devices are being equipped with a growing number of sensors. Where a few years ago only high-end models were equipped with GPS receivers that allow for accurate localization and tri-axial accelerometers that can detect acceleration, these features are currently available in devices in every segment of the market. High-end models now contain even more means of collecting data. These include, among others, gyroscopes to measure changes in rotation of the device, light sensors to measure ambient light, barometers that can be used to measure atmospheric pressure, and magnetometers that capture magnetic field strength and direction.

Developers of smartphone applications have eagerly adopted these sensors and the data that is collected through them. Using this added information, increasingly smart applications have been created. The aim of developing such applications is to make them able to reason about the data collected through the sensors and adapt their behavior in such a way that it better fits the user’s current context. We call such applications context-aware. We will provide a more complete definition for this later on.
The development of applications that gather information about their surroundings and adapt their behavior based on this information is, however, challenging. Practice shows that the future context of the system under development is unlimited in diversity, so it is not possible to anticipate all combinations of context factors at design time, and to always specify the behavior that ought to be exhibited by the system in such situations to be as useful as possible to the user. Additionally, as this type of application is novel, user requirements collection is difficult, and sometimes impossible. Research shows that the requirements a user has regarding a system are dependent on the current context s/he is in [107]. With a partially unknown future system context, gathering a complete set of user requirements is impossible. Furthermore, a discrepancy may exist between what a user wants from the system and what the user needs, i.e. what is objectively good for the user. To circumvent this, assumptions regarding user requirements and needs have to be made at design time. These assumptions might, however, not hold at run-time when the system has been introduced in unanticipated contexts. As a result, the system behavior might not fit the user’s needs. User adaptation of the application will be troublesome if such problems exist.

The domain of context-aware systems is broad, and providing a solution that is applicable for all systems using sensor technology is infeasible. In this dissertation we will focus on those systems that are to help knowledge workers, i.e. people who “make their living by accessing, creating, and using information in ways that add value to an enterprise and its stakeholders” [117], in improving their physical and mental well-being. The work performed by these employees is characterized by a certain degree of time and place independence, which causes the boundary between personal and professional life to fade. This can result in both psychological and physical problems. By actively coaching the worker to keep a balanced work/private life, it is possible to mitigate these problems. Coaching can be done by health care professionals, or by providing the worker with the right (software) tools. These tools, however, should be able to reason about their environment: they should be context-aware.

Although smart context-aware coaching applications seem to be a cost effective step in the right direction to improve the work/private balance for knowledge workers, their development is not straight
forward. In this dissertation, we look further into the challenges of context-aware application design, focusing on applications that are to encourage and support well-being specifically.

1.2 RESEARCH CONTEXT

The research described in this dissertation was carried out as part of the Dutch national program commit, project swell. The project started from the assumption that it is difficult for knowledge workers to balance work and family life caused by the freedom they enjoy to often work time and place independently. Additionally, the assumption was made that the jobs of knowledge workers are often sedentary, resulting in reduced overall physical activity. The goal of the project is to develop methods, tools, and algorithms to allow users to attain and maintain a balanced, active lifestyle. COMMIT/SWELL recognized two types of well-being to be supported: physical and mental well-being.

The research performed while working on this dissertation provided COMMIT/SWELL the overall, high level requirements [19], and the overall system architecture [15].

1.3 RESEARCH METHODOLOGY

In this research, we use the design science approach of Wieringa [132]. At the basis of this approach is the design cycle. We move through the following three phases:

PROBLEM INVESTIGATION First it should be clear who the stakeholders are, and what their goals are. We should describe what the problem context is, and what potential for improvements exist in this context so the goals of the stakeholders can be achieved.

TREATMENT DESIGN Second, we can specify one or more treatments. Wieringa defines a treatment as the introduction of an artifact, which can be anything from a (software) tool to a method or a process, in the problem context that will help stakeholders achieve their goal. We should specify requirements regarding this artifact and how they contribute to the
goal. Available solutions should be analyzed and new ones are to be designed.

**TREATMENT VALIDATION** Thirdly, we should validate the treatment/solution in order to provide evidence that it brings the stakeholders closer to their goals. By observing the treatment in a laboratory setting, analyzing the effects and performing analysis regarding the sensitivity of the approach, we can make predictions regarding the effects of real-world implementations. When we define such a research design, we specify an **object of study**, i.e. who or what are we observing to draw conclusions about our treatment, a **treatment design**, i.e. how will the treatment be used (by the objects of study), and a **measurement design**, i.e. in what way are results obtained from the experiment.

We describe the problem investigation we have performed in chapters 2 through 4. We discuss the treatment design and validation processes in parts II – IV of this dissertation.

### 1.4 RESEARCH QUESTIONS

In this section we formulate the research questions that are answered in this dissertation, the answers forming our contributions to the field of context-aware well-being application design and development.

Wieringa [132] divides research questions into two categories. **Knowledge questions** (KQ) are asked to gather information about the world, **design problems** (DP) call for change toward a state in the world where it is possible for stakeholders to reach a certain goal. We use the same structure.

Our research starts by obtaining information about the domain of interest, i.e. we establish the knowledge context for the improvement design. This results in our first main research question: how are context-aware well-being systems currently designed and developed (RQ1)? To answer this question, several sub-questions will have to be answered first. Analyzing the question, we find that we need definitions for context-awareness and well-being (RQ1.1). Furthermore, we need to know how requirements engineering is currently being performed for this type of systems (RQ1.2), and what
1.4 RESEARCH QUESTIONS

Table 1: Research questions answered in this dissertation. KQ = Knowledge Question, DP = Design Problem

sets these systems apart with regard to the design process (RQ1.3): are there certain properties of context-aware well-being systems that require alterations in this process?

When the structure of the problem domain has been established, we propose treatments that are expected to improve the development of context-aware well-being systems (RQ2). This is our second main research question. However, first we need to specify a set of requirements a system development method should adhere to (RQ2.1). Next, we need to obtain an overview of the current development methods available, and verify whether these satisfy the requirements formulated (RQ2.2). Due to the promises made by model-driven development (MDD) with regard to increased development speed and reduced error rates, we focus on this type of development method. After this, we have to design a method for developing context-aware well-being systems that we expect to im-
prove upon the currently available methods with regard to the requirements posed (RQ2.3). Finally, we should develop a method to verify whether the scoping of the designed application, i.e. what the application measures and which part of well-being it tries to influence, actually fits the user needs, and as such can help reduce the chance of incorrect application scoping (RQ2.4).

Once the treatments have been designed, we need to confirm our contributions and validate the solutions (RQ3). As our model-driven method uses a domain model as an input, we will first have to validate whether the domain specific modeling language in which it is created is suitable for the intended use, and whether the model is correct (RQ3.1). Next, we have to look at the effect our method has when introduced into the intended context, i.e. the process of developing a context-aware well-being system (RQ3.2). To test this, test subjects will use our method to construct an example context-aware well-being system. Using the experiences obtained from this case study, we can verify to which extent the desired effects of our method satisfy the method’s requirements. Finally, we should verify whether our analysis method actually can reduce scoping mistakes (RQ3.3).

We list an overview of our research questions and sub-questions in Table 1. Table 2 gives an overview of the chapters in which these questions are answered.

1.5 Contributions

Our research targets the intersection of software development, context-aware application design, and well-being support. The following contributions are planned:

1. A modeling language that is better suited to capture the well-being domain for the use in context-aware software development than current languages. Although we can use current general purpose modeling languages to model the domain of well-being, expressiveness is lacking, as we will show in chapters 4. In chapter 5 we introduce a Domain Specific Language (DSL) for this that is described in a metamodeling language.

2. A validated reference model of mental and physical well-being. It is generally acknowledged that physical and mental
### 1.5 Contributions

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Research questions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Part I: Problem investigation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 1</td>
<td>Introduction</td>
<td>–</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>Conceptual framework</td>
<td>RQ1.1</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>Method requirements</td>
<td>RQ2.1</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>State of the art</td>
<td>RQ1.2, RQ1.3, RQ2.2</td>
</tr>
<tr>
<td><strong>Part II: Knowledge modeling</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 5</td>
<td>A dynamic well-being domain model language</td>
<td>RQ2.3</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>A well-being domain model</td>
<td>RQ2.3</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>Validating the generic domain model and the suitability of the DWDM language</td>
<td>RQ3.1</td>
</tr>
<tr>
<td><strong>Part III: Model-driven development of context-aware well-being systems</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 8</td>
<td>Model-driven development using DWDMs</td>
<td>RQ2.3</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>Development of applications using DWDMs</td>
<td>RQ3.2</td>
</tr>
<tr>
<td><strong>Part IV: Behavior prediction</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 10</td>
<td>Context behavior analysis using DWDMs</td>
<td>RQ2.4</td>
</tr>
<tr>
<td>Chapter 11</td>
<td>Application behavior prediction using DWDMs</td>
<td>RQ3.3</td>
</tr>
<tr>
<td><strong>Part V: Contributions and conclusion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 12</td>
<td>Conclusions and future work</td>
<td>RQ1, RQ2, RQ3</td>
</tr>
</tbody>
</table>

Table 2: Dissertation outline
well-being are closely related. We provide a model, which has been validated by experts from both fields, that captures and explains this relation. This model is scoped by priorities set in commit/swell. The process of creating such a model is detailed. We discuss the model and the process of constructing it in chapter 6.

3. **A model-driven development method that exploits knowledge captured in domain models.** Due to the high degree of expressiveness and detail of the modeling language created, the domain models mentioned above may serve as an input for a model-driven development process that is partially automated through the use of model transformation tools. We introduce this process and the tools used in it in chapter 8. Furthermore, we discuss other project considerations when dealing with the development of context-aware well-being applications in this chapter.

4. **A structured analysis method to predict the the effect of a well-being support system on the context at design time.** Next to allowing software development from context models created using our domain modeling language, these models can also be used to predict the behavior of the well-being context when systems to support elements of it have been introduced. The well-being domain models are based on medical knowledge, allowing us to reason about cause and effect relations between elements of well-being. For example, we can use them to identify shortcomings in a context-aware well-being application’s design at design time, so we can create alternative solutions.

### 1.6 Scope

**What this research is about** In this dissertation, we look at context-aware well-being support systems. For such systems, we are concerned with models, the process of creating models, and the usage of them during the design and development of the software. When discussing the application development and analysis process based on domain models, we consider the design and development of the application architecture, physical distribution of the system and the reasoning logic embedded within it.
Due to the nature of the domain models, design and development processes are executed in a user-centric fashion: the person that will be using the application is of higher importance than the solution technology required to implement it. Technology, both hard- and software, will evolve, but the well-being context will remain largely the same.

**WHAT THIS RESEARCH IS NOT ABOUT**  Although this work is about user-centric applications, we do not aim to collect user requirements for context-aware applications and services, or provide means of doing this. Since user requirements for context-aware applications are hard or impossible to elicit, we circumvent the problem by modeling phenomena that we do have knowledge about and that impact the desired software properties, and use that model to develop software that satisfies unstated user requirements better than it would otherwise have. The resulting application will therefore perform in a way that is objectively good for the user, rather than being developed in a technology-centric way.

The software design process and considerations discussed in this work are not claimed to be complete. We focus the research on the ease of design and development of the application, rather than the creation of a user interface and user interaction, or the implementation of security and privacy concerns. Work on these topics is performed in the fields of Human-Computer interaction, and computer security respectively.

1.7 **OUTLINE**

This dissertation is structured as follows:

**CHAPTER 1** is this introduction.

**CHAPTER 2** provides the conceptual framework required to perform our research.

**CHAPTER 3** details requirements for a solution direction to the problem we identified in chapter 1.

**CHAPTER 4** lists state of the art work in the field of context-aware system development and requirements engineering for this class of systems.
CHAPTER 5 introduces the Dynamic Well-being Domain Model language. This modeling language is used throughout the research to capture the domain of context-aware well-being systems, models which are later on used as basis for a software development and a prediction process.

CHAPTER 6 describes how the DWDM language was used for the construction of an initial reference domain model.

CHAPTER 7 provides more details about the reference domain model reiterations, improvements, and validation.

CHAPTER 8 explains how we can develop context-aware well-being applications in a model-driven way, using DWDMs as an input, focusing on lexical, i.e. language-to-language, transformations.

CHAPTER 9 tests the validity of the development method proposed.

CHAPTER 10 provides explanations on how to predict future application behavior using domain models.

CHAPTER 11 tests the validity of the prediction method.

CHAPTER 12 provides conclusions and directions for future research.
In the introductory chapter, we introduced context-aware well-being systems, and noted that they are challenging to develop. We have, however, not yet given a definition of context-awareness or well-being. In this chapter we therefore answer the following research question:

**RQ 1.1 [KQ]** How are context-aware systems and well-being defined?

We start by giving an overview of the concept of context-awareness and provide the definition that we use in the rest of this dissertation (section 2.1). Next, we explore the concept of well-being (section 2.2).

### 2.1 Context-awareness

Before we start defining context-awareness, we first need to know what *context* is. The classical definition used here, is that by Dey [30]: “any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” The observant reader will notice that this definition regards context information, not context itself. Context is defined as the situation of an entity. Wieringa [129] defines context as encompassing all entities that are directly needed or affected by the system under development. The author defines three types of entities: (i) physical, (ii) conceptual, and (iii) lexical. Physical entities are those that “have a mass and a location in real space,” such as natural objects. Conceptual entities on the other hand are not physical, but are entities that “people use to
structure their world.” Examples given include rights, permissions, claims, roles, and contracts. Lexical entities are defined as “physical symbol occurrences,” such as pieces of text. Dey does not make this distinction, focusing only on what Wieringa calls physical entities.

Schilit and Theimer [92] were the first to use the term “context-aware.” They use it to distinguish systems that can use location information and track mobile objects from systems that cannot obtain and use this information. Continuing, the authors define “context-aware computing” as “the ability of a mobile user’s applications to discover and react to changes in the environment they are situated in”; the authors focus on location as the primary attribute of context. For our purposes, we shall not limit context-aware systems to application on mobile devices or location context; stationary systems are also able to utilize context information if the hardware or software required is present.

Besides “general” context-aware systems, it is possible to identify special classes of systems that can react to their environment. Firstly, we find “pervasive systems,” which are first defined by Weiser [128] as systems that blend in with the user’s every day life. Second, there is “ambient intelligence,” which, according to de Ruyter and Aarts [27], “refers to electronic environments that are sensitive and responsive to the presence of people.” Finally, there are “cyber-physical systems,” which are defined by Lee [60] as “integrations of computation with physical processes.” All of these types of systems have the capabilities to collect information about their context, however, the focus of their role within the environment differs. For example, the term ambient intelligence is often used in connection with smart surroundings, such as smart rooms or buildings. These systems are statically embedded in the context and offer services to those who pass through this context. Cyber-physical systems play an important role in bridging the gap between the real and the digital world when dealing with certain processes. Examples of such can be found in robotics and manufacturing.

When we look at the categories of context-aware systems, we see that the distinction between them can be made based on a number of system properties, the combination of these properties defining the category of system. Firstly, some of the systems are designed to move between locations, offering their services in changing physical contexts, whereas others remain in the same location. Secondly, there are systems that are designed to affect their physical environ-
ment, while others are intended to offer information or influence a digital environment. Thirdly, the system can be designed in such a way that it blends in with the environment and the user’s every day life, being unobtrusive. We summarize the properties of the systems identified in Table 3.

An element missing from the definition given by Schilit and Theimer [92], is the concept of usefulness. According to the definition, a context-aware system should be able to discover and react to a changing environment, but no mention is made whether this reaction should result in a situation the system’s user or other stakeholder benefits from. Without the system providing enhanced qualities by being context-aware, we could argue that context-awareness has little use. Some might reason that a system can be context-aware without altering its behavior to changes in the physical or virtual surroundings, and that just the ability to store such information would already make an application aware of its context. We argue, however, that only storing context information without using it to alter the system behavior is equivalent to a (user interface) button without an action attached to it: if there is no consequence to changes in the measured value, not even the visualization of the measurements through a number or graph, the ability for the application to record this measured value is useless: its reaction to the change in the environment does not become apparent to its user. In the definition below, we explicitly state that this reaction requirement should be fulfilled for a system to be regarded context-aware.

We have formulated our own definition for context-awareness. We use this definition throughout the rest of this dissertation.
Systems are context-aware if they can obtain information about their physical and virtual surroundings through hardware and software sensors, and adapt their behavior to better support the user based on this information.

In this definition, hardware sensors are physical devices that can be used to obtain information about physical phenomena and translate this information into a digital format. Software sensors are computer applications which can be utilized to gather information about virtual entities. A thermometer is an example of a hardware sensor that allows the application to observe properties of physical entities, a key logger and a text parser being software sensors that make it possible for the context-aware application to observe virtual and lexical entities.

2.2 WELL-BEING

Merriam-Webster [64] defines well-being as “the state of being happy, healthy, or prosperous.” As per this definition, the concept of well-being is highly subjective: two people in the same situation may feel differently about their well-being. When regarding the field of well-being, the World Health Organization [135] identifies three different types of well-being: (i) physical, (ii) mental, and (iii) social. These types concern the objective or subjective ability of a person to perform physical activities, the way a person perceives his/her own mental state, and the way people feel about interactions with others. Subjective well-being is how persons themselves feel about their well-being, whereas objective well-being regards their state of being happy according to fixed, objective criteria. Varelius [123] notes that these two perspectives might not align: something may be objectively good for a person, but the individual might not like it.

PHYSICAL WELL-BEING Physical well-being is to a great extent related to health care, i.e. “efforts made to maintain or restore health especially by trained and licensed professional” [64], and medicine. Ailments and disabilities may affect a person’s physical well-being. Such ailments may be of a temporary nature, being curable through medication or other forms of treatment, or permanent. The World
Health Organization [133] provides a framework for the classification of ailments.

Current research on improving physical well-being is primarily aimed at stimulating and increasing physical activity. A sedentary lifestyle may cause obesity, which is seen as a risk factor for physical well-being [134].

Physical well-being is a broad concept that covers all aspects regarding a person’s real-world, physical body. It is often possible to observe or measure physical well-being objectively, although subjective elements can also play a role, such as temporary physical conditions.

Mental well-being  Mental well-being does not only encompass mental and behavioral feelings and disorders, but also the way a person reacts to stressors [133], and factors as self perception and levels of energy. This highly subjective nature makes the field of mental well-being difficult to quantify.

Attempts have been made to make mental well-being measurable [109, 126], however, these are, understandably, limited in their applicability. We argue that it is not possible to measure every aspect of mental well-being, as some aspects of it are unquantifiable. All aspects of the life of a person are of influence on the mental well-being and vice versa: if a person is, for example, in good physical health, this will benefit the person’s perception of their own mental health. If the person is suffering from physical ailments, aspects of his/her mental perception may be affected too.

Mental well-being as a phenomenon is highly subjective. The ability to cope with negative situations not only differs per person, but for each person it may also differ per day: situations will be perceived differently depending on previous events. Because of this, we cannot define universal conditions that define mental well-being for all people at all times. Questionnaires or other tools available, such as the International Classification of Functioning, Disability and Health [135], are therefore only suited for the interpersonal judgment of mental well-being and not for intra-personal comparison.

Social well-being  Social well-being concerns elements pertaining to interpersonal interactions and relationships. Social interaction may benefit a person’s well-being by preventing loneliness and social isolation. For example, Cattan et al. [23] note that pre-
vention of social isolation in elderly is effective if specific groups are targeted. Looking at the working population, Johnson et al. [49] correlate the “iso-strain”, i.e. the “combination of social isolation and job strain.” [49], with cardiovascular disease mortality rates, finding that jobs (both white- and blue-collar) with a high iso-strain have significantly increased chances of suffering from these diseases. Furthermore, social interaction can increase or decrease subjective mental well-being. For example, if social interaction is comprised of a comparison of income, this can result in a change of subjective well-being [31, 37], resulting in the person thinking higher or lower of him/herself or about the other person. Contact with others is therefore important for a person’s overall well-being.

Ormel et al. [78] discuss the Social Production Function theory, which states that “people produce their own well-being by trying to optimize achievement of universal goals, within a set of resources and constraints they face.” According to this theory, social well-being is formed by a combination of factors: status, behavioral confirmation, and affection. Increasing one’s social status, having actions affirmed by others, or having good emotional connections with others will increase the perceived social well-being. These goals are deemed instrumental: individuals may have different preferences regarding each of the factors.

In this dissertation, we will only focus on two of the three types identified: physical well-being and mental well-being. The choice not to focus on social well-being was made due to the scope of the commit/swell project, which excludes social well-being. Furthermore, we only focus on those elements of mental well-being that have measurable, physical manifestations. An example of such an element is the level of experienced stress; we could measure indicators of this using a sphygmomanometer to measure blood pressure and a heart rate sensor to measure a person’s heart rate.

2.3 Conclusions

The research question we aimed to answer in this chapter was to find a definition for context-aware systems. Combining definitions found in literature, we defined such systems as being able to “obtain information about their physical and virtual surroundings through hardware and software sensors, and adapt their behavior to better
support the user based on this information.” This can be especially important when dealing with the domain of well-being. We divided well-being, i.e. whether a person feels happy, healthy, prosperous, into three sub-categories: physical, mental, and social well-being. We do not regard this last category in the rest of this dissertation. We can often measure physical well-being objectively, but have to resort to questionnaires to measure mental well-being.

Applications that are context-aware and are to support well-being have to take the specific challenges posed by both domains into consideration. Applications operating in these domains do not always behave in a way best suited for the context they are operating in at that moment. As a result, the users experience the application as failing them. We propose methods for the design of context-aware well-being systems that do take these requirements into account in the following chapters. We start by listing requirements for such a development process, and identify current state of the art methods for the design and development of context-aware applications, after which we identify the gap between the requirements and current solutions. We then propose a method that bridges this gap.
Having discussed the conceptual framework in the previous chapter, we continue by specifying requirements for a solution direction that will help with the design and development of context-aware well-being applications. We answer the following research question in this chapter:

RQ2.1 [DP] Define and motivate the requirements for developing context-aware well-being systems.

We start by discussing different levels of requirements (section 3.1), after which we continue by identifying the stakeholders who benefit or are otherwise affected by the development of context-aware well-being applications (sections 3.2 – 3.5). Finally, we discuss an example context-aware application (section 3.6). The requirements identified should hold for the development process of this application. We discuss how this is achieved in later chapters.

3.1 PROBLEM INVESTIGATION

The first step in the engineering cycle, is the problem investigation. We start by identifying the stakeholders and their goals. Alexander [4] defines four different levels of stakeholders or stakeholder roles: (i) the product, (ii) the system, (iii) the containing system, and (iv) the wider environment. We use the levels and stakeholder types identified by Alexander in this chapter. Stakeholders may or may not have desires that the system under development should fulfill. If a stakeholder has such a desire and commits to allocate resources to achieve this, Wieringa [132] defines these as stakeholders goals. The author notes that because of the finite nature of resources, not all desires can be promoted to goals. In the following sections we discuss the stakeholders and their requirements for each level.
The requirements discussed in this chapter were obtained through informal interviews and discussion with stakeholders throughout the committ/swell project.

3.2 The Product

This is the system under development (SUD) itself. This level does not contain any humans, and is purely technical in nature. Our product of interest is a method for designing and developing a context-aware system. This includes the process steps to be taken, the considerations per step, and the tools to support this process.

The product itself does not have specific requirements.

3.3 The System

This is a socio-technical level that includes the product and those who directly interact with it. This interaction may be the routine use of the product (“normal operator”), or keeping the product operational (“maintenance operator”).

3.3.1 Normal Operator

The normal operators, or end-users, of the development method are those who are tasked with the creation of context-aware well-being application, i.e. software designers and developers. Furthermore, domain experts are also part of this group as they deliver the knowledge on which the application is based.

Software Designer

Software designers translate end-user requirements and information gathered from domain engineers into a design for the system under development. The design should be as complete as possible, as missing details in the design will cause problems in later stages of the development process, or when testing or deploying the system.

In order for software designers to perform their jobs to their best extent, the requirements and domain knowledge provided should be understandable and should not require the designer to interpret the received details to a great extent. Interpretation of input in-
formation may cause interpretation errors, which in turn result in errors in the software design.

R1 Domain knowledge and system requirements should be easily understandable by the software designer.

A development method is to guide the software designer from the input information, i.e. domain knowledge and requirements, to the design of software structure and behavior. Structured steps can help a designer in this process. Furthermore, tool support may help designers perform their job better, however, the tool offered should be similar to the ones currently used, such as modeling tools for the Unified Modeling Language, for the designer to be effective at performing the required tasks without a lengthy period of getting used to the tools.

R2 The steps in the process should be clearly documented.
R3 The designer should be supported by tools.
R4 The tools offered should be similar to existing tools, or allow tooling of personal preference.

Software developer

A software developer is tasked with the implementation of the design, i.e. writing application code that satisfies the system design and the requirements elicited.

The move from an application design to application code will require the interpretation of the design artifacts. If the design is highly detailed for both the application’s structure and the application’s behavior, less interpretation is required. This in turn will result in less potential for errors to occur during the interpretation process. Software developers also need information regarding the rationale behind design choices, i.e. the relation between design and user requirements. Having information about this rational can prevent misinterpretations and as a result implementation errors.

R5 The relation between requirements and domain information, and software design should be documented.
Software developers need tools to support them in their work, such as editors, IDEs and frameworks. A new method should either provide for proper tooling, or should allow developers to use the tools they are accustomed to.

**R4** The tools offered should be similar to existing tools, or allow tooling of personal preference.

*Domain expert*

An example of domain experts in the field of well-being systems are health care professionals. They are asked their professional opinions regarding parts of the domain the application under development is to operate in.

Well-being experts rarely have a background in software engineering. A software engineering method in which they are to participate should therefore be easily understandable where their input is concerned, and should not focus on technical details they are not knowledgeable in. When their knowledge is documented, it should still be understandable for the expert, so s/he can verify that the information is correct.

**R6** The means of documenting domain information should be readable by domain experts.

Furthermore, the time required of experts in the design and development process should be minimized, reducing the burden of the project on the experts.

**R7** The time required of the domain expert should be as small as possible.

*Project manager*

Project managers should keep track of a project, making sure it remains on time, and within budget. Their primary requirement is therefore a development method that can be tracked properly, i.e.
consists of clearly defined steps, and which allows for an easy calculation of required resources. A new process method should be usable in existing project management styles, such as PRINCE2 [9] and Agile [39].

R8 The process should allow for the use of existing management methods.

3.3.2 Maintenance operator

Maintenance for a physical system entails tasks concerned with repairing the system so it operates properly. The same can be done for a process or method: the process is monitored and changes are made in order for it to operate efficiently. This is, for example, done as part of the “continual improvement” strategy defined in ITIL [6]. As this is part of project management, the requirements of the project manager also apply here.

3.4 THE CONTAINING SYSTEM

Like the system level, this level also is of a socio-technical nature. Here, we find those who are directly affected by the product, such as people benefiting from the product’s operation (“functional beneficiary”), and those who are responsible for having the product developed (“purchaser”).

3.4.1 Functional beneficiary

Those benefiting directly from the functionality offered by an application development method, are software designers, developers, and software development project leaders. An improved development process will help these stakeholders perform better at their jobs. We discussed the requirements of these stakeholders earlier. We discussed the requirements for these stakeholders earlier.
3.4.2 Purchaser

The company performing the context-aware well-being application development will likely be the party purchasing the development method. An improved way of constructing software will be financially beneficial for this party, as a shorter development project times, and less problems during this process will result in reduced costs. These are benefits resulting from the requirements of the normal operators, so we do not need to add additional requirements for this stakeholder.

Software development company

The exact requirements for a development method will differ per company, based on size, culture, and focus. What all companies have in common is their desire to maximize profit. For a software development company, this means that the development cycle of an application should be kept at a minimum, while ensuring high software quality, i.e. no bugs in the software, and adhering to user requirements.

**R9** The chance for software errors should be minimized.

3.5 The wider environment

The last level of abstraction is mainly social in nature. Here, we find those that are harmed by the product (“negative stakeholder”), roles benefiting from the system in terms of money (“financial beneficiary”), and those directly involved with the product development (“developer”), supporting development (“consultant”) and provision of components for the product (“suppliers”).

3.5.1 Negative stakeholder

The negative stakeholders group consist of those people or companies who can no longer sell their products, i.e. training in software development methods or tools, when the development method discussed in this dissertation is adopted. These stakeholders do not generate additional requirements for our method.
3.5.2 Financial beneficiary

In case of our software development process, the purchaser (discussed in 3.4.2) and the financial beneficiary are likely to be the same party, i.e. the software company developing context-aware well-being applications. We therefore do not gain additional requirements from this stakeholder role. We stated that the requirements for the purchaser were the same as for the normal operator, so the requirements for the financial beneficiary are those elicited for the software designer (R1, R2, R3, R4), for the software developer (R4, R5), and the domain expert (R6, R7).

There are additional parties benefiting from an improved development process, such as parties receiving benefit from improvements in the product being developed, i.e. the software, however, we consider these outside the scope of our research.

3.5.3 Developer

The author of this dissertation is the developer of the development process for context-aware well-being applications. The only requirement he has is for this method to be successful. However, as this is trivial, we will not be adding it to our list of requirements.

3.5.4 Consultant

Consultants may train others in using the development method discussed in this dissertation. The consultants’ customers may consist of those who will apply it when developing software, i.e. designers, developers, and project leaders.

For consultants to be able to train designers and developers in the use of a method, it has to be proven and well documented. Without proof of added benefits to the trainee, there will be no incentive to learn it, and without proper documentation training how each process step should be taken will not be possible.

| R2 | The steps in the process should be clearly documented. |
3.5.5 Suppliers

As the treatment suggested here consists of a software development method, the suppliers considered are those who provide context-aware application designers, developers, and project leaders with the tools necessary to perform their jobs. These tools may consist of modeling tools, tools for writing application code, project management tools, and others.

Suppliers of products that are used in conjunction with a software development method, are primarily vendors of software design and development tools. Their primary interest will be to continue or increase their sale of tools, despite the method of software development.

R4 The tools offered should be similar to existing tools, or allow tooling of personal preference.

3.5.6 Other stakeholders

Henkemans et al. [41] discussed the stakeholders for the commit/swell project. These stakeholders, however, are primarily concerned with context-aware well-being applications themselves, rather than the process of designing and developing such applications. As our treatment concerns the development process, we find that the stakeholders found by Henkemans et al., such as health care insurance providers, and health care professionals, are not the same as our identified stakeholders: they are concerned with the software resulting from the development process, rather than the process itself.

3.6 An example context-aware application

This section introduces an example context-aware application. We use this case throughout this dissertation to illustrate the methods and techniques discussed. The method to create this application will have to adhere to the requirements discussed in this chapter.
**Process Requirements**

- **R1** Domain knowledge and system requirements should be easily understandable by the software designer.
- **R2** The steps in the process should be clearly documented.
- **R5** The relation between requirements and domain information, and software design should be documented.
- **R6** The means of documenting domain information should be readable by domain experts.
- **R7** The time required of the domain expert should be as small as possible.
- **R8** The process should allow for the use of existing management methods.
- **R9** The chance for software errors should be minimized.

**Tool Requirements**

- **R3** The designer should be supported by tools.
- **R4** The tools offered should be similar to existing tools, or allow tooling of personal preference.

Table 4: Requirements for a development method of context-aware wellbeing applications

We regard a step counting application as an example. This is a highly simplified version of the Activity Coach discussed in Appendix G. The application has the following requirements:

- The application must count the steps taken by the user.
- The application must show the counted steps on a graphical user interface.
- The application must indicate how far the user is toward the daily recommended number of steps.
- The application’s interface must show the user’s progress toward his/her daily goal, encouraging or discouraging more activity based on the amount of activity required.

Unlike the Activity Coach, this application does not learn from or anticipates the user’s behavior. It should indicate if a user should be more or less physically active, but personalization as offered by the Activity Coach is not required.
METHOD REQUIREMENTS

3.7 DISCUSSION

The requirements discussed in this chapter are those for a development method for software applications. Combined with the specifics of the fields of context-awareness and well-being discussed in the previous chapter, they give us insight in the challenges posed when designing and developing context-aware well-being applications. The requirements discussed should not be mistaken for requirements on the applications themselves. Well-being applications have additional stakeholders, e.g. the end-users and health care providers, which will result in additional requirements. The idea is that by satisfying the method requirements, it is possible to better satisfy the user requirements with regard to the application itself.

The different stakeholders identified in this chapter are not necessarily different persons. They may also be roles a person has during several stages of the development process: a project manager can be a software designer, and it is possible that a software developer is also a domain expert.

3.8 CONCLUSIONS

In this chapter, we identified stakeholders who are affected by a development method for context-aware well-being applications, and the requirements these stakeholders have for such a method. Any process that is identified as a possible means of developing this type of applications more efficiently is to satisfy these requirements. We discussed software designers, who would benefit from easily understandable domain knowledge and system requirements in order to prevent errors made. Additionally, designers should be supported by tools, which operate in a similar fashion as currently used tools, or the development method should allow for designers to use their own tools of choice.

Software developers benefit from clear relations between requirements and designed software elements to be able to reason when a system could be deemed complete with regard to the requirements. Developers also require tool support, tools which are provided or sold by suppliers. If software is created efficiently and with a reduced number of errors, the development company will benefit because of higher sales and reduced cost of correcting errors.
When the ideas and opinions of domain experts are required, the domain information provided should be documented in a format readable by both the experts, and the software designers and developers. The time required of the domain experts should also be minimized.

During the development process, a project manager wants to keep track of progress. With multiple management methods available today, the software development method should allow for the use of these existing means of management.

The final stakeholder identified was the consultant, i.e., a person that trains the end-users of the method (the designers and developers) in the usage of the development method. For this class of stakeholders, documentation of the development method is key.

For each of these stakeholders, we have defined and motivated requirements, answering research question RQ2.1. These requirements are summarized in Table 4 on page 29.
“For me context is the key - from that comes the understanding of everything.”

– Kenneth Noland (artist)

We answer the following research question in this chapter:

RQ1.2 [KQ] Which requirements engineering approaches have been proposed for context-aware systems?

RQ1.3 [KQ] What challenges are specific for the design of context-aware systems and how are they addressed by current methods?

RQ2.2 [KP] Which MDD methods for developing context-aware systems currently exist?

This chapter discusses previous work in the fields of context-aware application design and well-being support obtained through systematic literature research while writing previously published works (as listed on page 282). With regard to context-aware application design, requirements engineering for this type of system is discussed (section 4.1) and design strategies are looked at (section 4.2) in order to answer RQ1.2, RQ1.3, and RQ2.2. Finally, we discuss software application support for the identified well-being types (section 4.3).

4.1 REQUIREMENTS ENGINEERING

When gathering and eliciting requirements for non-context-aware systems, the methods we can chose from are plentiful [5, 79, 81, 82, 120, 129, 130]. For example, we can gather requirements at design time through stakeholder interviews and questionnaires. These requirements can then be prioritized and implemented in the application. By providing a prototype of the system to the user, the we can test the elicited requirements (do the requirements correspond
to features needed by the user?), and changing or adding to them if needed. This process works well if stakeholders have the time and ability to specify their requirements up front, and to thoroughly test the prototype.

In situations with unpredictable contexts of use, the traditional linear process is less suitable. Context-aware systems are examples of this. Users will use these systems in a diverse range of situations and locations, making requirements engineering for this type of system challenging: when dealing with a different set of context elements, and element properties, the user might want the system to operate in a different fashion, according to Seyff et al. [96]. Such different contexts might not be foreseeable by the designers, and with users having little experience with this type of system, they too might be unable to specify their wishes, i.e. the ways the system should operate in different contexts. Novel requirements engineering methods are required to help requirements engineers deal with this challenge.

As Sitou and Spanfelner [100] state, expectations regarding context-aware systems are high, and such systems often fail to live up to their expectations due to a discrepancy between what users expect and how the system behaves. The authors describe an overall system architecture and a requirements elicitation method. The elicitation method focuses on requirements per part of the system architecture. The requirements are gathered by playing through specific scenarios, eliciting requirements about the basic system functionality first, after which usability concerns are addressed by performing task analysis. The process is iterative, and re-executed to gain additional requirements. However, when we analyze the method proposed, we see that in order to elicit requirements, the domain has to be known very well, and it should be possible for the requirements engineer to predict future situations of these systems. Without this knowledge, we cannot specify scenarios. Because of this approach, it is not possible to gather requirements for those situations which are unanticipated, leaving the system behavior undetermined. The authors note that a system part that aims at “identifying needs that could be automatically recognized by the system at runtime and at converting them into adaptation requirements.” How this should happen, however, is not discussed. Comparing this method to our development method requirements as listed on page 29, we see that it does not satisfy any of them.
Seyff et al. [96] propose to use in-situ requirements engineering to solve the problem of users being unable to specify their requirements up front by providing them a prototype application. A requirements engineer then observes the user while s/he uses the prototype. If discrepancies between functionality offered by the system and the demands of the user arise, the context factors are recorded and new requirements are specified or existing requirements altered. Although this method does not contradict any of the requirements we elicited in the previous chapter, it does not help with the initial requirements analysis process either: a prototype still has to be created, after which requirements can be checked using the method proposed by the authors. It could therefore be classified as requirements evolution or refinement, rather than engineering. Furthermore, as a requirements engineer should follow an end-user, the method is costly: firstly, end-users have to be found who are willing to participate in a long-term test of new software, and a requirements engineer has to follow this person, which requires a significant amount of person hours. To reduce this investment of time, Seyff et al. [97] propose to supply the intended end-user with an application to capture their own requirements in situ. This would eliminate the need for a requirements engineer to follow and observe the user. The authors indicate that the tool should be able to capture contextual information, end-user needs, and a rationale for this need and a task they wanted to perform. The tool presented in the paper allowed the users to capture a picture and GPS information, which were timestamped as the user entered his/her need and rationale in plain text (this satisfies our requirement R6). To test the tool, we provided it to end-users, and requirements for a commuting application and for a shopping application were collected. The authors conclude that the tool allows end-users to document their individual needs and that analysts can deduce well-defined requirements from these needs. Although this method does solve the problem of the need for a requirements engineer to constantly follow the user, the method is tested only for an application where the goal of the user was clear and the user knew what could be expected from the application under development. This is not the case with well-being applications, where personalization often has to happen at run-time, rather than at design time. Additionally, the subjective, personal requirements of users regarding a well-being application might not align with what is objectively good for the
user. We can argue that if subjective requirements are collected up front, the domain experts could verify these and correct them to be objectively beneficial to the user, which satisfies our requirement R7.

Salifu et al. [87] note that context-awareness entails a constant monitoring of the environment, and the switching of behavior in accordance to these changes. To facilitate constant monitoring, the authors propose a solution based on the Problem Frames approach proposed by Jackson [47]. Monitoring and switching problem frames are created, taking variability of problems into account. The problem frames are then used to derive state charts that capture the desired application behavior. This solution assumes that monitoring can identify new requirements. Salifu et al. [88] build further upon this work. The authors suggest the specification of requirements should be performed using boolean logic, which allows the application to verify at run-time, using boolean solvers, whether the requirements are still satisfied by the application in the current context. The authors claim that their solution improves on previous work with regard to time required by monitoring the context and switching behavior. The methods proposed in [87, 88] both allow for highly detailed description of system requirements and for automatic interpretation of these at application run-time. However, due to the notation used by the authors (boolean logic) the requirements are difficult to understand. This violates two of the requirements for a context-aware well-being application development process we formulated, i.e. that the requirements should be readable and understandable by software designers (requirement R1) and domain experts (R6) alike. Furthermore, the focus of the method is highly technical, instead of taking the human factor into account as well.

In [46], Inverardi and Mori describe the difference between changes in the behavior of a system that designers can anticipate at design time and those that have to be decided upon at run-time, calling these foreseen and unforeseen evolutions. To anticipate these evolutions, the authors propose a framework which uses a model of the context and the context requirements in a features based system. Requirements for the system can be added, removed or altered by the user at system run-time. Requirements should be specified through boolean statements, which allows the system to reason over the requirements, and adapt its behavior accordingly. The authors motivate the method using an e-health application. The primary problem we see with the solution direction identified by the authors,
is that it is highly technical, both in the focus and in the style of specifying requirements: the authors are primarily concerned with hardware details, and do not seem to consider the human aspect of the e-health system under development. Furthermore, the technical structure of the notation makes it impossible for domain experts, i.e. health care professionals, to verify whether the requirements they have provided are documented correctly, which is a violation of our method requirement R6 and R7.

Huang et al. [45] chose not to only look at the elicitation of requirements while developing the first version of the software. Instead they look at the requirements of evolving software throughout multiple versions, and how these requirements evolve as well. The framework proposed by the authors is therefore claimed to be suitable both for requirements elicitation and for requirements recovery from existing systems through application code analysis. Although requirements elicitation is mentioned, the authors do not clarify how this is to be done using their framework. Additionally, the requirements discussed throughout the paper do not seem to anticipate unknown future situations. The way of coming to these requirements demands full domain knowledge and perfect anticipation. Because, the method requires detailed knowledge of both the domain and technical elements of the application design, we deem it unsuitable for domain experts to work with. None of our method requirements is satisfied by the framework proposed.

The method proposed by Desmet et al. [29] is aimed at gathering requirements for context-aware systems through a systematic approach. The authors claim it is designed to be simple so all stakeholders can use it, but expressive enough for the requirements to evolve toward the solution space, i.e. application design and programming. Although the method proposed by the authors allows for the modeling of behavior variations, these heavily rely on domain knowledge and on the perfect anticipation of future contexts, which, as we concluded earlier, is not possible in the systems we are considering. The paper focuses primarily on the modeling language to capture the requirements, rather than on the actual elicitation process. Finally, the method is geared at the soft- and hardware requirements, rather than looking at the needs of the end-user. We argue that the method satisfies our requirements that it should be easily understandable and the output readable by all involved (R1 and R6), the technology focus of the method, combined with the
lack of tool support and description how to deal with unanticipated future settings makes this method unsuited for our needs.

Rather than looking for specific requirements, Ming et al. [69] look for the intentions the future end-users have with the application. Starting from user goals, the authors propose to collect both user response to the application’s behavior and behavior patterns to improve the formulated goals during the reiteration steps. By analyzing past behavior, the authors claim that future the can predict and anticipate behavior, and so future user needs. When analyzing the method, however, we see that rather than purely focusing on the goals, the authors look at requirements derived from goals. They argue that these requirements “may not cover all the possible situations.” Although this method is more user oriented than other requirements engineering methods discussed in this section, the process of reiterating through multiple versions of the software before a version is created that satisfies all, or at least as many as possible, user requirements seems labor intensive to us. Furthermore, the role of domain experts is not discussed by the authors.

Sawyer et al. [90] propose to use “requirement awareness,” i.e. the ability of the system to verify which requirements it is currently adhering to, as a run-time solution to the design-time inability to make proper predictions on the future system context. The authors compare this quality to middle-ware system being able to use reflective means to introspect their architectural configuration. The authors introduce a DSL to capture requirements. They claim the models in this language can be analyzed at run-time and allow behavior adaptation. If requirements are conflicting, the user should be consulted. Requirements are formulated in semi-structured natural language, so they are easily readable by designers and domain experts alike (R1). As other methods discussed, the method proposed by the authors is highly technology focused, rather than looking at requirements from the actual end-user. The authors do not provide a concrete example how their method should be used, which causes the explanation to remain abstract.

Focusing on requirements engineering for pervasive health care systems, Jørgensen and Bossen [50] do not define a system that adapts at run-time. Rather, a three-tier approach is proposed to improve the requirements collection. Firstly, requirements are postulated in free-form prose, describing use cases and system behavior (R1). Secondly, executable models are created from these stories.
Finally, animated applications are provided to the user to interact with. These applications depict the future usage setting and allow the user to see system behavior visualized. The relation between artifacts is, as such, well documented (R5). The method proposed by the authors, however, assumes a predictable environment in which the system will operate: although the required system behavior is not yet known (this should be elicited), where the system will be is known, the context elements that will be there are also known. This approach may work when regarding systems that are statically embedded in a specific context, such as ubiquitous systems, but may not be suited for mobile context-aware applications, such as the well-being applications we consider in this dissertation.

Finkelstein and Savigni [38] also recognize the inherent problems posed by the context-aware capabilities of systems: “A changing context means essentially that one cannot, while analyzing requirements, rely on reassuring assumptions about the world.” Instead of looking at system requirements (the way the system should operate in certain situations), the authors focus on system goals (the reason why the system should be delivering certain services). The proposed solution is targeted on the “why” of the system, not the “what” or “how,” i.e. goal-driven, rather than requirements- or specification-driven. By reflecting on these goals at run-time, behavior can dynamically be adapted. The authors, however, do note that due to the early stage of the research, the exact way of implementing this framework should be determined. This framework assumes a set of fixed user goals, but is flexible with regard to specific requirements. Although the work seems promising, the authors make no mention of a method of gathering or formulating the goals for the system, whereas we are looking for a fully documented way of moving from user goals to a system design, rather than an abstract framework description.

The requirements satisfied by the methods discussed here are summarized in Table 5.

4.2 CONTEXT-AWARE APPLICATION DESIGN

In current work, we can roughly identify three categories of directions for improvement for the design of context-aware systems: (i)
State of the Art

Table 5: Requirements satisfied by requirements engineering methods

<table>
<thead>
<tr>
<th>Method</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
<th>R8</th>
<th>R9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sitou and Spanfelner [100]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seyff et al. [96, 97]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Salifu et al. [87, 88]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inverardi and Mori [46]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Huang et al. [45]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Desmet et al. [29]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ming et al. [69]</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sawyer et al. [90]</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jorgensen and Bossen [50]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finkelstein and Savigni [38]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

improved design processes, (ii) better ways of capturing the design, and (iii) architectural solutions for context-aware systems.

4.2.1 Design process

Silva et al. [99] note that the development process of context-aware systems can be time consuming. To improve this, the authors introduce a process which enables developers to rapidly create prototypes of the application. The framework uses state transition systems to capture dynamic behavior, which is in turn visualized in a three-dimensional simulation. It is this simulation with which the users
can interact. Most steps of the approach discussed are tool supported, satisfying our requirements R3 and R4. Furthermore, software engineers will be accustomed to working with state transition systems, making it understandable to work with (R1). The documentation of the process steps could be improved (R2), and no mention is made of the requirements gathering and processing. Furthermore, no information regarding the role of the domain expert is given.

Sheng et al. [98] present a platform that should simplify the development of context-aware web services. The platform is based around the notion of model-driven engineering: contexts are defined in a language called ContextUML, which is a specialization of the UML created for the specification of context-aware systems, these context models being automatically translated to executable implementations of the target web service platform. The end-user of the web service can interact with these executables. The method proposed satisfies a number of our requirements. It is understandable by software designers, and fully tool supported (R1, R3, R4). The authors do not, however, clarify what the role of domain experts in the design process is. Furthermore, the models that are created to represent the context and the context-aware web service are highly technical, making them unsuitable as a means of communicating with experts.

Kang et al. [53] introduce a tool for the design of context-aware systems. In the paper, the authors focus on health care systems, designing a ubiquitous system for a smart home. The design is performed according to a framework which consists of five components: (i) a context provider that obtains context information through sensors, (ii) a context integrator that combines context information from multiple sources, (iii) a context interpreter that derives additional information from the obtained sensor data, (iv) a profile manager that handles user and service profiles, and (v) a context manager that drives the system as a whole. The tool provided by the authors is based on the Eclipse platform, and allows designers to create context-aware systems through a structured process. The solution direction proposed by Kang et al. satisfies our requirements with regard to understandability by the designer (R1), the need for well defined process steps (R2), and provides tool support (R3, R4). The authors do not discuss how requirements should obtained and translated. Neither is the role for the domain experts clearly stated.
In [104], Sottet et al. focus on the development of user interfaces for context-aware systems, rather than on the full system design. The authors propose a model-driven approach for this development process. The authors specify five principles which should improve the quality of the user interfaces that will be generated through the process. The user interface is first specified through models, which are transformed automatically using ATL model transformations. The context in which the application will be used is modeled in an ontology, rules for dealing with context changes are specified using Event-Condition-Action (ECA) rules. The context switches that are anticipated by the authors are related to a change in application platform, i.e. a move from a desktop system with a mouse and keyboard, to a small screen smartphone et cetera. The approach taken by the authors seems to prevent errors while coding the user interface (our requirement R9), and developers with experience in model-driven engineering can clearly understand the models and transformations created (R1), using tools known to them in the process (R3 and R4). The role of the domain expert, however, is not clearly stated, nor is an example given for the models that are to serve as an input for the transformations.

4.2.2 Capturing design

To capture the design of context-aware systems, Abeywichrama et al. [1] propose “State Of The Affairs” (SOTA). SOTA is a formal modeling language that allows designers to capture labeled transition systems that represent the workings of the application under development. At the basis of the modeling language is the state of affairs at a certain time. This state of affairs can move through the “SOTA space” through transitions. To capture the intended behavior of the system, developers can specify goals, along with a constraint on how to reach it. As the system is specified in a formal way, model checking analysis may be used to verify the system is error free. Although the research direction taken by the authors seems promising, it is intuitively clear that the notation used by the authors is not easily understandable, especially if large systems with multiple goals are specified in it. This is a violation of our requirements R1 and R6. Furthermore, it is not clear how the authors intent to support...
the use of the language, other than allowing it to be checked using Labeled Transition System Analyzers.

Muñoz et al. [70, 71] propose a domain specific modeling language to capture the structure of pervasive systems. The language is called Pervasive Modeling Language (PervML) and is situated at the Platform Independent Model (PIM) level of the MDA: the systems are described in an abstract way, not dealing with future underlying implementation technologies. The PervML language allows for the capturing of information regarding the interoperability of systems, triggers, pre- and post-conditions, and service aggregation. Through the use of model transformations, it is possible to generate OSGi models, which in turn can be used to generate Java code for services. Serral et al. [95] uses PervML and the PervML Generative Tool (PervGT) created by Cetina et al. [24] to specify a model-driven method for context-aware application development. Improvements over regular development include decreased development time, increased reusability, and improved software quality. The PervML and the model-driven process based on it seem to satisfy a lot of our requirements. The models are easily understood by designers (R1), the process of attaining a software design is documented (R5), the process would allow for existing management methods (R8), because of the model-driven nature the risk of errors is reduced (R9) and the steps are clearly documented (R2). Cetina et al. [24] adds tool support that is similar to the support designers currently have, satisfying R3 and R4. The requirements not dealt with, however, are those of the domain expert: the PervML is geared at the technical design, and not at the elicitation or capturing of domain knowledge, which is where our requirements R6 and R7 are aimed at.

Paganelli et al. [80] introduces two types of models to capture context information and to drive context reasoning. The models are called “Context Acquisition and Delivery Data Model” (CADDM) and “Context Reasoning Data Model” (CRDM). The first type of model allows designers to capture context information and is object-oriented, the second allows for reasoning over context element relations and is ontology-based. The authors describe the usage of these models in the process of designing a context-aware tourism application. Although the duality in the approach seems interesting, additional detail regarding the actual development of the tourism application example should have been provided: the authors mention that the CADDM and CRDM models are used, however, do not provide
examples of the model implementations for this specific example. What is clear, is that the approach suggested satisfies a number of our development method requirements. Firstly, system designers will be able to understand it (R1), the move from requirements to the models is clear (R5), and given the nature of the models, it is possible to provide tool support (R3, R4). The move from the models to the rest of the design and programming is not detailed, and the role of the domain experts is not elaborated on.

Henricksen et al. [43] propose a method of capturing the technical system context of context-aware systems. The method allows for the description of the concept of a ‘Person’, who is authorized to use, or is located near a certain ‘Device’. The Person also has a (communication) ‘Channel’, which in turn requires a Device. ‘Location Coordinates’ are used to locate the Person and the Device. The authors allow for the use of different association types in the models, including ‘Static associations’, ‘Derived associations’, and ‘Sensed associations’. Dependencies can be added between associations in order to guarantee a stable system. Furthermore, associations can be annotated with quality parameters, indicating accuracy and certainty of these connections between entities. Henricksen and Indulska [42] use this modeling approach to guide the analysis and modeling of context requirements, creation of an infrastructure, application design, and testing. The modeling method introduced by the authors is primarily aimed at modeling physical entities and their relations. Activities and other attributes can be added or annotated, but these are of lesser significance in the models. Because of this modeling focus, only the context situation at a specific moment in time can be represented in the model, while dynamic context behavior, i.e. the transition from one situation to another, cannot be modeled. Although the modeling language uses a specific notation, it would probably be understandable by software designers if some training was provided (R1). Due to the technical nature, however, the models are unlikely to be easily readable by domain experts. The role of experts is not discussed.

Bolchini et al. [14] provide a survey of 16 context modeling languages. The languages are compared on features such as space, i.e. location information, time, i.e. temporal aspects, type of formalism, such as described earlier, and the ability to reason over the context. The conclusion of the authors is that languages that aim to be general purpose fail to be effective, as the requirements for a
context-aware application design

general purpose language are different from those for a domain specific language. Languages that target specific problem areas should therefore be preferred. Bettini et al. [11] also give an overview of context modeling languages, however, the authors have different requirements regarding the language, such as heterogeneity, i.e. the ability to express different types of context information, modeling of imperfections, and usability. Bolchini et al. [14] disregard usability as a criterion, as this is a subjective measure. The authors propose a hybrid modeling approach that captures three levels of detail: (i) low-level sensor data, (ii) data representation and reasoning, and (iii) ontological representation and reasoning. Although the models individually are readable for software designers and domain experts (R1 and R5), spreading the information over several models, the relation between modeling elements not formally defined, can be confusing.

Dockhorn Costa et al. [32] characterizes context as either being intrinsic or relational. The intrinsic contexts includes types of context which belong to a single context entity, and do not depend on relations that entity has with other entities, such as geographical location or heart rate. The relational context type does depend on relations between entities, such as a social network, in which a person is related to other people. Using these context types, it is possible to build situational models. These models are instances of a context model. The models build by the authors are presented as UML class diagrams, and are therefore understandable by software designers (R1). The situation models are further simplifications and should be readable by experts (R6). Dockhorn Costa et al. [33] propose to use context models and situation models to drive a rule-based reasoning engine. The models created in the modeling language described in this chapter does not make an explicit distinction in intrinsic or relational contexts, rather, the measurability of context items is regarded as an important aspect. Tool support is not offered, which violates our requirement R3.

4.2.3 Architectural solutions

To aid and ease the development of context-aware systems, Salber et al. [86] proposed the Context Toolkit. This toolkit consists of reusable building blocks called widgets. These widgets aim to hide
specifics of the context, while managing the details of it, not unlike their counterparts found in user interface design. The authors do not propose a method to be used for combining these widgets, this is left to the developer. Reusability is useful for developers, requiring less work to build applications. However, the authors make no mention of how the widgets should be used together. They authors fail to clarify how their method eases the development of systems that are “distributed […] and use unconventional sensors.” The method proposed in the paper only satisfies our requirement R₁, being that it is readable for designers and developers.

Lane et al. [59] describe the architecture of a context-aware wellbeing system. The BeWell system consists of a smartphone and an infrastructure part, the smartphone elements providing means for sensing and providing the user with feedback, the infrastructure hosting a web portal and performing calculations on raw data. The process of requirements engineering, and design and implementation of the system are not discussed in detail, the primary focus of the paper being a technological one. van Sinderen et al. [121] give a similar description for context-aware systems in general, providing an architecture that as designed to be more scalable and modular. The operation of both systems does differ. While Lane et al. focus on informing the user of the application through a relatively simple feedback loop, van Sinderen et al. use context reasoning and ECA rules to directly interact with the environment. As is the case with Salber et al. [86], both Lane et al. [59] and van Sinderen et al. [121] do not clarify how their methods are used to improve software quality, reduce errors, and reduce development time. Domain knowledge and domain experts are mentioned in neither of the papers.

Capra et al. [22] explore a middle-ware solution for the design of context-aware systems. The authors propose to use reflection to be used in the middle-ware layer. Through this reflective property, the application running on the middle-ware can access the context information. The middle-ware makes sure that the internal representation of the context is up to date and policies based on the context are not conflicting. Conflicts can be a problem as two separate applications might want to exhibit different behavior based on context information.

To deal with the need of context-aware to adapt their behavior at run-time, Nunes et al. [75] propose an architecture that consists of four main components. These components, the Aggrega-
tor, Mediator, Maintainer, and Actuator, are tasked with capturing and identification of context elements and changes, identification of adaptation needs when a context change is detected, management of context definitions, and making adjustments to the process in the work environment when needed respectively. These components are controlled by a context manager, which also keeps repositories of contexts, and context rules. Although the architecture defined in the paper seems fitting for context-aware systems, the authors do not provide a convincing application scenario: instead of “normal” context-aware applications that operate in a real-world context, the authors chose to elaborate the design based on the iTunes music store. As this is a regular web service that does not operate in unpredictable environments, we find it hard to regard this as a context-aware system. Even though the solution direction seems promising, the authors do not elaborate on the development process of the systems based on their architecture, or show how this architecture is an improvement over systems using the traditional sensing/processing/reasoning structure.

Zhang et al. [136] propose an architecture for a system that should help elderly with mild dementia to live at home. The architecture consists of four components, namely a smartphone based assistant application which provides means of interacting with the user, a home gateway that communicates with the other system components and performs the context reasoning, sensors to obtain information, and a server that logs events from the gateway. Caregivers can set rules and reminders. The type of system concerned by the author is, despite the use of a smartphone for user interaction, statically embedded in the context. Despite the architecture being clear, the paper does not answer any of our requirements specifically. Especially knowledge from domain experts is not regarded, except for opinions regarding the working of a pilot version of the system.

Jaroucheh et al. [48] focus on the distributed nature of context-aware systems. The authors identify, among others, coordination, interoperability, mobility, and scalability as challenges faced when working with this type of system. As a solution direction, the authors propose a three layer architecture, consisting of (i) a network layer, (ii) a mobile middle-ware layer, and (iii) the application layer. The first layer should handle all network related issues, the middleware layer taking care of, among others, resource management, context management, event notification and security, while the appli-
cation is tasked with personalization, interoperability, and mobility. Comparing the solutions offered by Jaroucheh et al. with our design method requirements, we see that it satisfies requirement R1 (understandability by designers), R5 (relation between requirements and design), and R9 (minimize software errors). However, no clear process is specified, no tool support is offered or suggested, and the role of the domain expert is unclear.

A summary of the requirements satisfied by development methods discussed by these sources is provided in Table 6.

4.2.4 Discussion

Two things become apparent from Tables 5 and 6. Firstly, nearly none of the requirements engineering methods discussed in this chapter contribute to the posed method requirements. Secondly, only one paper is concerned with the readability of requirements by domain experts, and none of the papers deals with the minimization of time required from the domain experts (R6). Understandability by software designers is something addressed by all papers, tool support is the second subject dealt with most often. Overall, we can conclude that methods currently available to design and develop context-aware applications are primarily focused on solving problems that are technological in nature. For well-being, however, what is offered is of a much higher importance than how this is done. Well-being systems require a user-centric approach when it comes to the development, taking both objective and subjective requirements into consideration.

4.3 Well-being Support

In this section we explore the support applications currently available for physical and mental well-being. We did not look for social well-being support applications, because, as we stated in section 2.2, we consider this type of well-being out of our scope.

4.3.1 Support applications

In the field of well-being support, applications often offer support for a single type of well-being: applications that support physical
4.3 WELL-BEING SUPPORT

R1. Domain knowledge and system requirements should be easily understandable by the software designer.
R2. The steps in the process should be clearly documented.
R3. The tools offered should be similar to existing tools, or allow tooling of personal preference.
R4. The relation between requirements and domain information, and software design should be documented.
R5. The means of documenting domain information should be readable by domain experts.
R6. The time required of the domain expert should be as small as possible.
R7. The process should allow for the use of existing management methods.
R8. The chance for software errors should be minimized.
R9. The designer should be supported by tools.
R10. The tools offered should be similar to existing tools, or allow tooling of personal preference.

Design process

<table>
<thead>
<tr>
<th></th>
<th>Silva et al. [99]</th>
<th>Sheng et al. [98]</th>
<th>Kang et al. [53]</th>
<th>Sottet et al. [104]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
| Capturing design
| Abeywichrama et al. [1] | -                 | -                 | -                 | -                   |
| Muñoz et al. [70, 71] | X                 | X                 | -                | X                   |
| Cetina et al. [24]   | X                 | X                 | X                | X                   |
| Paganelli et al. [80] | X                 | -                 | X                | X                   |
| Henricksen et al. [43] | -               | -                 | -                | -                   |
| Henricksen and Indulska [42] | - | -               | -                | -                   |
| Bolchini et al. [14] | -                 | -                 | -                | -                   |
| Bettini et al. [11]  | -                 | -                 | -                | -                   |
| Dockhorn Costa et al. [32, 33] | X | -               | -                | X                   |

Architectural solutions

<table>
<thead>
<tr>
<th></th>
<th>Salber et al. [86]</th>
<th>Lane et al. [59]</th>
<th>van Sinderen et al. [121]</th>
<th>Capra et al. [22]</th>
<th>Nunes et al. [75]</th>
<th>Zhang et al. [136]</th>
<th>Jarroucheh et al. [48]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 6: Requirements satisfied by development methods
well-being do not try to influence mental well-being and vice versa. COMMIT/swell [83] is a project that aims to bridge this gap.

Physical well-being support

For physical well-being support, we see two types of applications: those that are to be used by the person without any help from health care professionals, which often bought and downloaded by people curious about their own well-being, and tele-treatment systems which are supplied by a health care professional.

Achterkamp et al. [2] list several existing physical well-being applications. Most of these applications are aimed at improving physical activity [26, 40, 63]. Accelerometers and tri-axial gyroscopes are often used to measure physical activity. Achterkamp et al. [2] lists well-being support applications that users can purchase on Apple iTunes or in the Google Play Store. The applications discussed by the authors are shown in Table 7. The authors make three distinctions: (i) applications that operate locally on the smartphone, (ii) applications that operate locally and store information in an external database, and (iii) applications that operate locally, store information in an external database, and provide the user with additional feedback which is provided directly by a health care professional.

Considering tele-treatment systems, the focus seems to be primarily on physical well-being support. Examples of such applications are listed in Table 7 as “Local feedback including feedback through a database and health care professionals.” Because treatment, rather than well-being support, is out of the scope of this dissertation, we will not go into detail here.

Mental well-being support

Koldijk [56] describes a tool that should support knowledge workers to improve well-being at work. The application runs on a personal computer system and uses sensors in order to recognize the user’s context. The information sources proposed are the keyboard and mouse, but also application information. Using these, the application can deduce which task the user is working on. By incorporating sensors such as microphones, GPS sensors and cameras, attempts can be made to deduce the user’s social context, i.e. the people the user is with. A combination of these may give us an indication
### Table 7: Physical well-being support applications

<table>
<thead>
<tr>
<th>Application name</th>
<th>Application type</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>UbiFit</td>
<td>Local feedback systems</td>
<td>Gamification (growing plants and garden)</td>
</tr>
<tr>
<td>StepUp</td>
<td>Local feedback systems</td>
<td>Step counter</td>
</tr>
<tr>
<td>Office Exercise</td>
<td>Local feedback systems</td>
<td>Exercise suggestions</td>
</tr>
<tr>
<td>Mijn activiteit</td>
<td>Local feedback systems</td>
<td>Activity tracking</td>
</tr>
<tr>
<td>Routinely</td>
<td>Local feedback systems</td>
<td>Remind user of tasks to establish habits</td>
</tr>
<tr>
<td>Be Fit, Stay Fit Home</td>
<td>Local feedback systems</td>
<td>Activity tracking</td>
</tr>
<tr>
<td>Fitness Challenge</td>
<td>Local feedback systems</td>
<td>Activity tracking</td>
</tr>
<tr>
<td>Exercise Booster</td>
<td>Local feedback systems</td>
<td>Body measurements and weight tracking</td>
</tr>
<tr>
<td>HealthLog</td>
<td>Local feedback systems</td>
<td></td>
</tr>
<tr>
<td>Nexercise</td>
<td>Local feedback including feedback through a database</td>
<td>Activity tracking, gamification (competitive)</td>
</tr>
<tr>
<td>Runkeeper</td>
<td>Local feedback including feedback through a database</td>
<td>Exercise tracking</td>
</tr>
<tr>
<td>P4Well/Wellness Diary</td>
<td>Local feedback including feedback through a database and health care professionals</td>
<td>Insight, tracking</td>
</tr>
<tr>
<td>TuneWalk</td>
<td>Local feedback including feedback through a database and health care professionals</td>
<td>Exercise tracking</td>
</tr>
<tr>
<td>Philips DirectLife</td>
<td>Local feedback including feedback through a database and health care professionals</td>
<td>Activity tracking</td>
</tr>
<tr>
<td>Roessingh Research and Development activity monitor</td>
<td>Local feedback including feedback through a database and health care professionals</td>
<td>Activity tracking</td>
</tr>
</tbody>
</table>

about the perceived workload. The author does not discuss specific interventions.

Bosems et al. [18] list applications that are commercially available, and aim at helping the user gain insight into his/her working behavior [84, 102], structuring data [36], and planning [34, 66]. Such aids should reduce stress in the user. Other tools are there to prevent distraction, allowing the user to focus on his/her work [116]. These are shown in Table 8.

Kailas et al. [51] discuss the move toward more personalized health care to be implemented in society, as this could reduce the overall cost of health care. They identify an increase of health-related applications for smartphone devices, and provide insights regarding challenges for wellness applications and systems such as security, user interface restrictions, memory restrictions, interference issues and data collection challenges.
As we have discussed in chapter 2, the well-being domain is broad, encompassing the entire spectrum of feelings a person may have regarding any situation, may it be physical to his/her own body, social interaction, or on an emotional level: all these factors are related. Because of this interrelatedness, providing a single means of improving every aspect of the well-being spectrum is challenging.

The notion of “subjective well-being” adds additional challenges to providing well-being support. A person decides upon how s/he feels given all aspects of the current well-being situation combined. For example, a person suffering from a permanent physical disability may perceive his/her well-being as good, whereas a person with a temporary ailment may judge his/her own situation as not being good at all. Objectively, the well-being situation of the person with the disability can be judged to be a lesser one due to the permanent nature of the condition. However, we can also reason that the person dealing with a permanent situation may have grown accustomed to it, as it is his/her every day life. The person with the temporary problem, on the other hand, is not used to his/her situation, it impacting their subjective experience to a greater extent.

Another example is that of exposing two individuals to the same external stressors. Due to the different personalities of both persons, these stressors are unlikely to be experienced identically: some people have higher resistance to stressors, or are by nature better able to cope with them. Different people perceive stress differently, even though the external, real-world stressors were identical.
A final example regards the subjective nature of social interaction. All people have basic needs to this regard, but interactions are subjectively experienced differently based on time, earlier events for the individual, and their inter-personal relations. If two people are interacting in a similar fashion, with one of them has a different socio-economic status, then this will affect the way either person perceives the interaction. This also holds true when two individuals have a different opinion of each other: if one person likes the other, s/he will experience the inter-action differently than when there is a feeling of dislike.

Due to the subjective nature of well-being, posing interventions in the life-style of a person is also challenging. Interventions are, or should be, objectively good for a person. However, an objectively good life-style change may have a subjectively negative impact. For example, presenting a person with the objectively sound intervention that they should improve their diet and be physically more active, may have an adverse subjective reaction: life-style changing interventions, as such, should be posed in such a way that the person receiving them can internalize them, so the change can come “from within” in the long run.

The commit/swell project aims to support both physical and mental well-being through both targeted, and integrated solutions. Firstly, applications specifically aimed at improving either physical or mental well-being are created. Providing better physical well-being support entails an improvement of existing well-being applications such as the Activity Coach [77] by including a better evaluation of the user’s base-line fitness and goals, and by adding persuasive techniques to improve user adherence to the exercise regime proposed by the application. The Activity Coach is discussed in more detail in Appendix G. Mental well-being support applications are systems that can detect a user’s working habits and projects s/he is currently working on. The system can suggest documents and e-mails relevant to the task the user is working on, helping in his/her information need. Secondly, by integrating solutions for physical and mental well-being, commit/swell partners will develop a system that can help the user into being more active, while not disrupting his/her daily activities.
In this chapter, we researched the current requirements engineering, design and development approaches available when dealing with context-aware systems in order to answer research questions RQ1.2, RQ1.3 RQ2.2: which RE and MDD methods have been proposed to develop context-aware systems, and how do these methods attempt to solve the specific challenges regarding the design of context-aware systems?

The primary challenge identified by all authors is the inability to fully predict the future context in which the system will be used. Authors in the field of requirements engineering attempt to solve this by improving the way requirements are gathered up front. Other authors recognize that up front requirements engineering might not be feasible for context-aware applications, and instead suggest the system under development to be goal-oriented, specifying only system goals and allowing the system to adapt its operation to satisfy these goals. Authors that look at the design process of context-aware applications on the other hand focus on technological improvements that increase the flexibility of the application, again allowing run-time alteration of behavior.

We looked at 11 methods for requirements engineering. Of these, only Seyff et al. [96, 97], Desmet et al. [29] and Jorgensen and Bossen [50] satisfy two of the requirements we posed in the previous chapter, Sawyer et al. [90] satisfying one requirement. Authors often assume the ability to specify requirements up front, or chose not to elicit requirements at all and only provide an application goal. To improve upon the requirements we posed in the previous chapter, we propose a domain modeling language in chapter 5 that should help developers and designers capture the problem domain. We postulate that this can aid in the elicitation of requirements. Rather than focusing on technological possibilities, we look at the information found in the well-being domain itself. Furthermore, we propose a method for verifying application requirements based on these domain models in chapter 10.

Looking at 20 proposed methods for design and development of context-aware applications, we found that none of the methods proposed took the role of the domain expert into consideration as the documentation of the domain information was not readable by them, and that the time required by the experts was not minimi-
zed. The methods are predominantly focused on the technological solutions to what is a socio-technical problem domain: the social, user-centric elements are often overlooked. Cetina et al. [24] satisfied seven of our nine requirements, leaving the two regarding the role of the domain expert unanswered. However, the authors have a highly technology centric approach to solving the challenges of developing context-aware systems, not dealing with user specific challenges. Again, we think that the modeling language we propose in chapter 5 can help with the development of applications.

In order to leverage the models created in the suggested modeling language, we propose to use a model-driven method for the actual development of context-aware well-being applications. We describe this method in chapter 8. By using a model-driven method, the information gathered in the domain models by interviewing experts can be used directly for the application development, rather than it having to be interpreted by the developers. Such interpretations may be cause for error, resulting in undesired application behavior in the end. An additional benefit of reusing the domain models, is that the time required of the domain experts is limited to the construction of the model, instead of needing them for the actual application development too. By supporting the designers and developers with semi-automatic model-transformations and design tools, development speed of application is increased and the chance for errors is reduced. This would satisfy all requirements identified in the previous chapter. Further more, by employing model transformations, we have direct traceability in the development process: we can directly indicate which features of the application are a result of elements of the input model.

In this chapter, we also looked at current work performed in the field of well-being support, focusing on the sub-domains of physical and mental well-being. We argued that interventions that are “good” for the population in general, might not be for the individual.
Part II

KNOWLEDGE MODELING
A DYNAMIC WELL-BEING DOMAIN MODEL LANGUAGE

“Well-being cannot exist just in your own head. Well-being is a combination of feeling good as well as actually having meaning, good relationships and accomplishment.”

– Martin Seligman (psychologist)

In the previous part, we have looked at the context of this research and a conceptual framework in which it has been executed. Based on this context and this framework, we have elaborated on the requirements for a software development process for context-aware well-being systems. We looked at the current state of the art, and compared these solution directions with our requirements. We found that, because of the unpredictability of the context of a user, traditional methods for software requirements gathering and elicitation are infeasible. Other methods proposed focus on solving technological challenges for context-aware systems, rather than aiming to improve the usefulness of and the user’s experience with the software system. Such methods try to strictly define a highly unpredictable, or even unknown, context.

Instead of specifying the solution domain, i.e. how the software should operate, we focus on capturing the problem domain, i.e. the domain of well-being. We use causal structures for this. This approach has two advantages. First, the causal structures in the domain do not change with changes in the domain itself. Second, the causal approach allows for causal reasoning over the modeled domain. We can use this in context-aware applications to offer improved functionality to the user: the system could reason what functionality would be desired, instead of asking the user for feedback in this regard. Rather than “In situation C, perform action A” because this will satisfy user goal G, the system can, using a causal model as input, reason “in situation C, perform action A” because this will result in effect E, which is better for the user’s health and well-being.
We start by specifying requirements for a domain modeling language for the domain of context-aware well-being applications (section 5.1). Next, we describe the notation of the Dynamic Well-being Domain Model (DWDM) language (section 5.2), after which we elaborate on the usage of DWDMs for the well-being domain and in other domains (section 5.3).

We move toward answering the following research questions:

**RQ2.3 [DP]** Design a method for developing context-aware well-being systems that improves current methods with respect to the requirements specified in chapter 3.

We do not describe a development method in this chapter, this will be done in part III. However, the modeling language discussed here plays a key part in the development method introduced in later chapters.

### 5.1 LANGUAGE REQUIREMENTS

The requirements stated in this section are based on the overall method requirements stated in chapter 3 and the solution direction elaborated on at the start of this chapter.

The disadvantage of current modeling languages is that they do not allow for the capturing of both structural and behavioral properties of a system context in the same model. Instead, at least two different models are required to represent both structure and behavior. More models are needed for the capturing of additional detail or improved readability.

The first requirement for the DWDM language is the ability to capture the static structure of the well-being domain, i.e. the different elements of the domain, and whether they are related. For example, a person’s physical activity and his/her heart rate are related. Sterman [105] provides us with a solution direction for this requirement, namely Causal Loop Diagrams (CLDs).

Secondly, it should be possible to capture the dynamic properties of the well-being domain. It is important not only to know *if* domain elements are related, but also *how*. For example, if a person undertakes physical activity, his/her heart rate will increase. Again, CLDs [105] would fulfill this requirement.

Thirdly, the DWDM language should be easy to use, i.e. when provided with information regarding the structure of the language,
the user, most likely a technology expert or software engineer, should be able to capture well-being information without much effort, documenting which elements exist in the domain, and how they are related. Software designers and developers should be able to use the models to create well-being software applications and systems, domain experts should be able to use them to reason about their own area of expertise and verify whether the information depicted in them is correct. CLDs might not be the best solution with regard to this requirement; as CLDs are not commonly used in the field of software engineering, those designing and developing software might not be accustomed to their syntax.

Fourthly, the models created in the DWDM language should be able to capture technical information that developers can use when developing well-being applications. This will allow users of the language to utilize it for capturing both the domain structure and the technical aspects, such as required sensor technology. For example, if a person’s heart rate should be measured, it should be possible to indicate that a sensor should be used for this, and that the sensor should record the measurement in terms of beats per minute. CLDs do not foresee in this requirement.

Finally, models in the DWDM language should be easily understandable for both technology experts and well-being domain experts. DWDMs should be usable as a tool for communication between these two groups, so mutual understanding of the information captured in the model is of key importance. CLDs might not be the best choice due to the same reason given for requirement three.

In this chapter, we describe the DWDM language, in chapters 6 and 7 we test whether these requirements hold for our design.

5.2 NOTATION OF DYNAMIC WELL-BEING DOMAIN MODELS

This section introduces the grammar of the DWDM language.

5.2.1 Language Development

The development of the DWDM language started with the graphical syntax of Causal Loop Diagrams (CLDs) [105]. CLDs consist of named variables, connected by causal relations that have a certain polarity. We show a simple example of a CLD in Figure 1. This sy-
stem shows the relation between chickens and eggs: if the number of eggs increases, the number of chickens will increase as well. At the same time, an increase in the number of chickens will result in an increase in the number of eggs.

By creating CLDs for the cases discussed in Appendix E, F and G, we found that additional constructs are needed for the CLD language: a requirement for the models to be usable in software development is the ability to capture the measurability of variables. Without this information, software engineers would be required to have extensive knowledge regarding available sensor technology. As a workaround, the variable names were prepended with the annotations “[o],” “[d],” and “[c]” to indicate a variable was observable, derivable, controllable respectively, but a more elegant solution was deemed needed.

The next language development iteration resulted in the addition of units of measurement and measurement scale properties for variables. We found that without these we could not express the domain knowledge unambiguously. Furthermore, norms were added to capture accepted boundary values for certain variables. Without this information, software engineers would need medical knowledge before being able to write an application.

During the development, a choice was made to give the possible values of the measurability property the names “observable,” “derivable,” and “controllable”. We can argue that the first two of these possible values should be renamed to better indicate measurability, such as “measurable” and “unmeasurable.” In other fields, however, similar terminology as ours is used: in physics and statistics, an observable (verb) is something that can be measured. Throughout this dissertation, we will use the measurability property in a similar way, discussing observables/derivable, instead of calling them variables that are measurable/unmeasurable.
5.2 Notation of Dynamic Well-Being Domain Models

The DWDM grammar is specified in the metametamodelling language Ecore [113]. Ecore is part of the Eclipse Modeling Framework (EMF) [35]. The DWDM metamodel is shown in Figure 2.

The DWDM language is a graphical modeling language, allowing the visual depiction of elements of the well-being domain. According to our requirements, its constructs are simple enough to be understandable by experts in the field of well-being allowing them to reason about their area of expertise in a structured way, yet powerful enough to capture the technical details required for the development of well-being systems base on them.
Figure 3: The representation of a variable in the DWDM language. Left the structure of the variable, right an example

DWDMs are constructed using two types of elements (variables, section 5.2.3, and norms, section 5.2.5), which are connected using relations (positive or negative causal relations, section 5.2.4).

5.2.3 Variables

We can associate many different elements with well-being. Some of these elements are physical properties of a person or context, others are mental properties. All of them, however, are described by information that captures the current state of the person’s world. In DWDMs, this information is represented in variables. Figure 3 depicts such a variable. The figure shows how a variable is visualized in the graphical DWDM language. The variable’s properties are stored as attribute/value pairs, similar to the way UML diagrams are depicted and stored. The order of the attributes is fixed, the fields are not explicitly labeled when the models are visualized, as this would increase the size of the visualization of DWDMs. The labels of the fields should, however, be shown in an editor to aid the designer.

A variable has four properties, not all of which necessarily having a value. One property a variable does not have in the DWDM, is a value. The reason for this is that the modeling language should be used to document domain models, rather than run-time (real-time) behavior of the well-being context.

Name

In order to be able to distinguish between variables, each variable has a name.
Measurability

A variable can reside at any level of abstraction in the well-being domain. Because of this diversity, it is not possible to measure all variables. The measurability property allows us to express this. It can have any of three values:

**Observable** A variable is marked as being observable if, given the current state and utilization of techniques, it is possible to directly obtain the variable’s value through the use of sensors. We distinguish between two types of sensors. First, hardware sensors can capture a real-world phenomenon and convert its attributes into a digital signal which the system can use as an input. Second, software sensors are applications that capture information regarding the virtual environment the user is operating in and translate its properties to a format which may be used by a context-aware application. An example of a hardware sensor is a thermometer, software logging the active system applications is an example of a software sensor. The observable variable values such sensors would be collecting are “Temperature” and “Number of applications” respectively.

Whether we can mark a variable as observable depends on the current state of the art regarding sensor technology: new types of sensors may turn previously unmeasurable variables measurable.

**Derivable** We cannot measure all variables directly. Rather, some variable values have to be computed by looking at the values of related variables. We call such variables derivable. Some might argue that a derivable variable can be observable, if a deterministic relationship between an observable and a derivable variable exists, i.e. the value of the derivable variable can be directly calculated from the observable variable’s value. However, as per the definitions of measurability presented here, the derivable variable is still denoted as such, because a specific calculation based on the value of other variables has to be performed in order to derive a derivable variable’s value. Variables of this type are often abstract concepts or properties that are not physical in nature. It is therefore not always possible to assign a numerical value to the variable. An example of a derivable concept is the amount of stress perceived by a
A dynamic well-being domain model language

person. To say something about the level of stress a person is experiencing we have to take into account the stressors acting on the individual, such as deadlines and distractions, and the factors that help the person to cope with stress, such as relaxation and social support. Note that we use the term “experiencing” rather than undergoing, as stress is a subjective concept, being highly dependent on the coping mechanisms of the individual.

Like observable variables, derivable variables are technology-dependent. New technology can render a derivable variable observable.

**Controllable** Some variables can directly be influenced by the software system, often through the use of actuators. Such variables are called *controllable*. In a system that maintains the indoor climate of a building, a controllable variable may model a valve’s position which, by changing its value, will regulate the temperature. This temperature in turn can have a positive or negative effect on a person’s feeling of well-being, depending on their personal preference.

A variable that is controllable cannot be observable or derivable at the same time and vice versa. By altering the value of a controllable variable, we create a situation where other variable values can change as well. Looking at the climate control system mentioned above, it is not possible for the software system to directly control the temperature, which is observable through thermometers, but by altering a heater’s valve position, we can change the temperature.

Variables that can be influenced directly by the software system are, compared to observable and derivable variables, more rare, as they tend to include only those context variables that are part of actuators.

**Dimension**

If a variable can have a quantifiable value, a *dimension* is needed to be able to say something about said variable value. For example, the value “3” of a variable with the name “Length” is of no meaning without a dimension, i.e. did the author of mean 3 millimeter, 3 foot, 3 parsecs, or 3 beard-seconds [74]?
5.2 Notation of Dynamic Well-Being Domain Models

We can categorize dimensions as either being independent, i.e. variables values that can be used on their own, or dependent, i.e. to use the variable’s value, additional data is required such as the value of other variables. Kelvin, used for expressing temperature, is an independent unit, while body fat percentage is a dependent unit that relies on, among others, the total body weight, and the water and muscle percentage. If any of these change, the body fat percentage changes as well. Furthermore, units can be time dependent or time independent. An example of a time dependent variable is the heart rate expressed in beats per minute. The classification of the variable and the way it should be treated by the software system is not enforced by the modeling language, and is left to the software developer.

*Measurement scale*

To reason about variable values, we need to be able to compare values and perform arithmetic operations on them. Not all scales, however, allow for the same types of operations. To make this distinction, we add the notion of a *measurement scale* as an attribute of a variable. We use the four types of measurement scales identified by Stevens [106].

**Nominal** The nominal scale only allows for the equality operation: two values of a variable on a nominal scale are equal, or they are not. Typically, we can regard the values of a nominal variable as different classes of the measured phenomenon. Examples of items on such a scale are apples and oranges: we can see that these are not equal, but we cannot sort them objectively.

**Ordinal** Rather than only allowing for distinction to be made, the ordinal scale also allows for sorting. It is, however, not possible to quantify the difference between the sorted elements. Examples of items on this scale are healthy and sick. Intuitively, we can classify “healthy” to be better than “sick”, but it is not possible to say by how much.

**Interval** Adding additional powers of expressiveness, the interval scale makes it possible to quantify the difference between two measurements. It is not possible to argue the ratio of difference due to a lack of an absolute zero point. The differences,
VariableA
Derivable
DimensionA
Ordinal
VariableB
Observable
DimensionB
Ratio
+
-

Figure 4: Causal relations

However, can have a ratio. Temperature in degrees Celsius is a variable measured on this scale: we can say the difference between 10 and 20 degrees Celsius is 10, but we cannot say that in absolute terms the second is twice as hot as the first.

**Ratio** The ratio scale, compared to the interval scale, adds the notion of a non-arbitrary zero point. This allows us to say how many times the unit of measurement can fit in the measured value: it allows for multiplication. Examples of such variables are the temperature in Kelvin and length in meters. Neither of these can have a negative value and as such have a non-arbitrary zero point.

5.2.4 *Causal relation*

As most information models, DWDMs consist of model elements that represent real-world elements (the variables and their values, as we discussed earlier) and relations between such elements. In DWDMs, these relations are causal relations. Causal relations in DWDMs have two properties. We say a causal relation between X and Y exists, if a change in X results in a change to Y.

The first property of a relation is its *direction*. Causal relations have a source and a target which indicate which variable (the target) is affected by the change in value of the other variable (the source), i.e. if the source’s value changes, the target’s changes too. An arrow from variable A to variable B indicates that if A changes, B changes, i.e. the partial derivative of B to A does not equal zero. We use the term “causality” because this is common when working with CLDs.

The second relation property is the *polarity* of a relationship. The polarity expresses whether a causal relation is monotonic or anti-monotonic. These types of causal relationships are also called positive or negative causal relations. A positive causal relation from source variable A to target variable B indicates that an increase (or
Figure 5: A simple causal path

By combining one or more causal relations, we can create a path. Two variables are linked by a path, if we can follow a sequence of causal arrows from one to the other. A path is called a loop if it contains a variable twice; otherwise, it is called a simple path. A path, as per the direction of the relations, connects its start and ending variables indirectly. Because of the nature of causal relations, this allows us to reason about indirect effects. We can perform this reasoning in two directions. Observe the picture in Figure 5. If we measure variable A and we see an increase in A, we as a result, B will increase as well, and that C will decrease. We call this process forward reasoning, as it allows us to say something on the effects of variables down the path of A. On the other hand, if we measure variable C and see a decrease in it, we can reason that this decrease was caused by an increase in variable B, which in turn is caused by an increase in variable A. This process is called reverse reasoning, which allows us to backtrack the path of changes that has lead to the value change. If a variable has multiple incoming relations, a change in its value can have multiple causes. In both examples, we must assume that the diagram represents all causal relations among the three variables. If there were additional, invisible relations, the effect of changing A might have be different, and the cause of a change in C might be different.

In addition to paths, we can create loops. A loop, as noted, contains a variable twice. In Figure 6, we see an example causal graph with two loops.

The first loop we can identify, is the path A-B-C-A. Looking at the polarity of the causal relations, we see that these are all positive causalities. As such, we can reason that an increase in A, causes an increase in B, which causes an increase in C, which in turn causes
an increase in A. This is called a *reinforcing loop* and it can represent uninhibited growth. In the DWDM language, reinforcing loops cannot be bound by setting a limit to the loop. The only way the effect of a reinforcing loop can be limited, is through the influence of other relations. We find such a situation in the second loop, which consists of the path A-B-A. In this loop we see both a positive and a negative causal relation, for example, an increase in A causes an increase in B, which in turn results in a decrease in A. Unlike the first type loop, this loop can model a stable situation. These loops are called *balancing loops*.

5.2.5 Norms

A *norm* is defined by Merriam-Webster [64] as “a set standard of development or achievement usually derived from the average or median achievement of a large group.” In the context of well-being, we see a norm as a range of values a variable may have that can be considered normal or healthy for the general population. What is normal or healthy on the average in a population, however, might not be so for the individual and vice versa. For example, an elite athlete might have a resting heart rate as low as 40 – 50 BPM, whereas the average for the general public lies between 60 – 100 BPM. In DWDMs, norms have a minimum and a maximum value to capture this interval, and are expressed in terms of the dimension of the variable they are associated with. These values can be obtained from medical literature and through interviews with domain experts.

A variable can only have one norm and a norm can only be associated with one variable. We visualize the association between a norm and a variable using a dashed line, making a clear distinction from an association between two variables. Norms allow us to capture additional static information about the domain. This information will
improve the degree to which we can reason over a DWDM. For example, a person’s heart rate is trivially quantified as the number of beats per minute, and validated boundaries exist. Using the boundaries identified, we can reason whether the variable’s value is still within a normal range. Shown in Figure 7, the normal range for a person’s heart rate is within $60 – 220$ beats per minute. If a person’s heart rate is measured, and the measured value is not in this range, something has to be done to increase or decrease it. Taking personal circumstances into account is important here, as an athlete, as noted earlier, will have a different normal heart rate than the general population.

5.3 DISCUSSION

The DWDM language was constructed with a number of scenarios in mind. This section discusses these application domains and briefly considers the usage of the language outside the area of well-being.

5.3.1 Well-being domain

The domain of well-being is highly diverse, containing information at different levels of abstraction. We can capture this information in variables, norms, and causal relations. By doing so, it is possible to make hidden information and relations visible, allowing for discussing among experts.

Traditional software engineering models, such as the UML, capture classes of objects, and object with variables models real world entities. On the other hand, DWDMs only capture variables and their properties. Relations between variables in DWDMs indicate
a change relationship, i.e. causalities, which allow for reasoning over the model, relationships between classes in the UML modeling a structural relationship. Furthermore, DWDMs model static and dynamic properties of the modeled domain in a single diagram, instead of relying on the combination of models which is required in the UML: variables and norms represent static information, visualizing which elements are present in the domain, and which values these elements must take. Causal relations model both static and dynamic information: the presence of a relation between two variables models both that the variables are related, and how they affect each other. This allows for dynamic reasoning over a static model. Combined with knowledge regarding norms, it is possible to formulate interventions if variable values are at risk of becoming outside the normal range.

In section 5.2.5 we described the usage of norms in the DWDM language. We presented norms as having two values: a lower- and an upper bound. To allow for personalization, we could add a “desired” value, which could capture the norm value that would be appropriate for the specific user of the application. However, we argue that personalization is something that should be done at application run-time. Because the DWDM models are design time models representing the well-being context that can be expected at run-time, we think the addition of such a value would interfere with the separation of abstraction levels. Another element that may be added or improved at application run-time, is the “strength” of positive and negative causal relations: although these relations should hold for the general population, they might be stronger or weaker for specific individuals. Discovering such relationship strengths at run-time would allow the application to move to the set goal more effectively. Machine learning can play an important part in this adaptive application behavior: the default boundaries for the application are set by the information included in the model, but can be personalized by the discovery of better norms or the strength of causal relations by the application as it operates.

5.3.2 Usage in other domains

Although the DWDM language was created with the well-being domain in mind, it is possible to represent any structured context,
with elements that affect each other, in its grammar. Variables and their properties do not restrict such use and neither do causal relationships. We find literature applying causal loop diagrams in the domains of, among others, business and management [105] and biology [125]. In these fields, causal diagrams are used to model and simulate large systems in order to gain understanding of the effects of changes in variables.

Well-being is not the only domain where context-aware systems are of interest or importance. As application scenarios in these other domains are also primarily aimed at the collection of data about context entities, the processing of this data and the usage of the resulting information in the process of making run-time decisions, we argue that causal modeling as the one presented in this chapter can also be of use there. For example, the context of control systems could be modeled using the DWDM language, describing how valve positions affect the flow of chemicals, and the result on chemical reactions taking place in reactors. Our addition of norms and dimensions to the CLD notation would allow us to model such systems. Also, we see a trend in the automotive industry where cars are being equipped with an increasing number of sensors and reasoning systems to control everything from the workings of the engine to driving the car itself. Causal models created in our DWDM language could be used to describe the context in and around that can be observed through sensors, relating elements of this context. Again, our additions to the CLD language allow for this, as they are aimed at sensor-driven systems.

When designing for other domains, it should be kept in mind that hardware and software logic, such as sensors and architecture elements, should not be modeled, rather, the focus should be on the domain properties. For example, if used for developing a temperature control system, the elements modeled are not the temperature sensor, the thermostat, and the heater, but the temperature itself, the preferred temperature, and the position of the heater’s control valve.

The use of the concept of norms might also be appropriate in other domains, providing an indication for the values which are deemed normal. This way the support system can indicate those values that are outside the normal range.
5.4 CONCLUSION

In this chapter we introduced the Dynamic Well-being Domain Model (DWDM) language. This domain specific language was designed with the specifics and requirements of the well-being domain in mind. Rather than being focused on solving problems regarding well-being and aiming to capture the technical aspects of these solutions, the language is primarily suited to detail knowledge regarding the domain elements, their relations, and their properties. It is, however, still possible to capture technical information. This information can then be used for the development of applications that are to support the user in the context depicted in the constructed DWDM. Rather than electing requirements for the system under development, DWDMs allow for capturing what is objectively good for the future user of the system in a structured, visual way.

The modeling language we described in this chapter allows us to partly move toward answering RQ2.3: design a method for developing context-aware well-being systems. The models expressed in the DWDM language are able to capture domain knowledge, but still be readable by domain experts (requirement R6). Because the models consist of elements and relations between them, they are similar to models usually used for software engineering (such as UML), and because of this are readable by software designers as well (requirement R1).
In the previous chapter we introduced the Dynamic Well-being Domain Model language (DWDM). The DWDM language was designed with the domain of well-being in mind. This chapter discusses means of creating domain models for specific applications within this domain. We can construct such a model in either of two ways: creating a new model from scratch (section 6.2.1), or by instantiating a larger reference model (section 6.2.2). We provide a user-centric and a technology centric approach for instantiating a model. We work toward answering the following research question in this chapter:

**RQ2.3 [DP]** Design a method for developing context-aware well-being systems that improves current methods with respect to the requirements specified in chapter 3.

Besides the methods of creating models we also discuss a reference well-being model that was created for the domain considered by the COMMIT/SWELL project (section 6.1). We apply the model creation process to the step counter case (section 6.3) to further illustrate the proposed methods. Finally, we discuss the model creation and refinement process (section 6.4).

### 6.1 A GENERIC WELL-BEING DOMAIN MODEL

In this section we describe the construction of the COMMIT/SWELL reference domain model. We constructed the model as a general reference for well-being knowledge in the COMMIT/SWELL project. The model covers two of the three well-being areas identified in section 4.3: physical and mental well-being were analyzed, modeled and related, social well-being was not regarded. The choice for this
focus was based on the scope of the commit/swell project. We split the model into two parts for depiction here. The first part contains the elements related to physical well-being, which can be found in Figure 8, the second part contains the variables related to mental well-being, which is shown in Figure 9.

The generic well-being domain model covers the domain knowledge as provided by two experts in the fields of physical and mental well-being. These experts were researchers taking part in the commit/swell project. Both of them had their education in their respective fields of expertise, after which they engaged in research in persuasive technologies for physical well-being and stress monitoring respectively. These experts provided information that they obtained from literature on, among others, the dangers of a sedentary life [127], goal setting [61] and feedback [3, 122] in the case of physical well-being, and cited sources regarding causes of stress [65, 72] and management of stress and burn-out prevention [52] in the field of mental well-being.

The information captured in the domain model was elicited during three co-creation sessions. During the first session, an initial model was constructed using the “Creating a model from scratch” construction method explained in section 6.2.1. In the second and third sessions, this model was refined. The experts substantiated their claims using published literature. The validity of this model is tested in chapter 7.

6.1.1 Physical well-being

The primary focus of physical well-being in the reference model, is the question of how to make people more active, i.e. increase physical activity (the degree to which the user is physically active). The rationale is that physical activity is good for a person’s physical well-being, as it contributes to the change of several physical aspects that have a direct positive causal relation to physical activity. The associated variables are cardiovascular fitness (how well the cardiovascular system works) and body composition (the percentage of body fat, muscle, water, and bone).

However, before a person will start undertaking physical activity, s/he will need to be motivated and have the feeling that s/he can indeed successfully perform the activity. This last concept is called
6.1 A GENERIC WELL-BEING DOMAIN MODEL

Figure 8: Reference commit/swell domain model: Physical well-being

self-efficacy. A person cannot be expected to be motivated without a certain incentive: if everything in his/her life is exactly as s/he wants it to be, s/he will not start to undertake more physical activity. For this, an intention to change is needed. Such intention can
have either of two causes: an internal cause, resulting from a decreasing feeling of well-being and a subsequent realization that changes need to be made, or through external factors. An external factor can be feedback, which can be given by parties such as friends, family, health care professionals, or well-being applications. By increasing the feedback a person is receiving, his/her awareness regarding the situation will increase too. This awareness then results in an intention to change, motivation, and eventually in increased physical activity.

If a person starts undertaking increased physical activity, his/her heart rate, at that moment, will increase, his/her cardiovascular fitness and body composition will improve. A better body composition will result in improved physical well-being. However, with a person’s overall well-being improving, his/her intention to change decreases, eventually reducing the physical activity again. This is a balancing loop.

6.1.2 Mental well-being

The field of mental well-being is concerned with the subjective experience people have regarding the world around them. An important variable here is energy, which we can see as an internal battery for a person: if s/he experiences increased degrees of mental stress, the battery drains and eventually depletes. This variable has a direct positive causal relation toward mental well-being. By undertaking leisure activities, the person can “recharge” his/her battery. To capture this, the energy variable has two incoming relations: a positive relation from the recovery variable, and a negative causal relation coming from experienced stress. A person can increase his/her degree of recovery by undertaking activities such as hobbies and resting. The amount of stress a person experiences is based on a person’s personality. Not only is it dependent on perceived stressors, which may not be the same for each individual, but also on the way a person can deal with the stressors, their coping mechanisms. Some people are better at coping with stress, reducing the degree to which stress is experienced. The perceived stressors are also subjective: they are affected by the actual, objectively existing stressors, the degree to which a person is confident about their ability to perform their job, i.e. work efficacy, and their feeling of being in control of the situation, i.e. the locus of control. Increased work efficacy and locus of control
have a positive effect for the person: they have a negative causal relation toward the perceived stressor variable: if the work efficacy and locus of control increase, less stressors are experienced.

When observing the stressors variable, we find that this is affected through positive causalities, by among others distractions and task load: if a person is distracted from his or her work, this will cause increased stress. The same holds for the number of tasks that are

Figure 9: Reference commit/swell domain model: Mental well-being
to be done. It is possible to mitigate these stressors through, for example, *information support*, which reduces stressors by preventing them from occurring: if a knowledge worker is supported in his or her search for information, this reduces the amount of work to be undertaken by him/her self.

Another aspect deemed positive for mental well-being, is *social interaction*. In the context of knowledge workers, this interaction can come from discussion with colleagues who provide feedback. As we discussed, social interaction and relations may have additional positive and negative effects on a person’s mental well-being, however, we will not go into detail regarding this.

### 6.2 Construction Process

The creation of a DWDM requires both domain knowledge to elicit the relevant domain variables, and technical knowledge to identify the possibilities using the current state of the art regarding sensor technology. Because of this, the expertise of both application domain and technology experts is required when constructing domain models.

We recognize two ways of building a DWDM for an application. We discuss both in this section.

#### 6.2.1 Creating a model from scratch

The choice to construct a new model from scratch, rather than basing the application context model on an existing reference model, can be based on the absence of a reference model for the application domain. The process of constructing a new model from scratch is summarized in Table 9. We discuss the steps of the process below.

1. The construction process starts by discovering the primary variable of the system under development: what element in the context should the system be supporting or influencing? In case of the reference well-being domain, this would be “well-being;” the system’s highest goal is to improve the well-being of its user. We designate this variable to be the *goal variable*.
1. Identify context-aware system goal.
2. Decompose goal.
3. Identify variables affecting the (sub)goal(s).
4. Decompose variables and identify variables affecting/affected by discovered variables until scope is reached.
5. Identify additional causal relations among variables.
6. Determine variable measurability, dimension and measurement scale, and processing type.
7. Identify norms related to the variables.

### Table 9: DWDM construction: creation of a model from scratch

<table>
<thead>
<tr>
<th>Feeling of well-being</th>
<th>Observable</th>
<th>No dimension</th>
<th>Ordinal</th>
</tr>
</thead>
</table>

2. Next, we start to decompose this goal, i.e. identify which sub-goals should be satisfied for the overall goal to be satisfied. By looking at the “well-being” variable, we see that there are sub-elements related to it: well-being consists of physical, mental, and social aspects, as we discussed in section 4.3. All of these aspects are interrelated. To model this, we add variables to the model with positive causal relations to the first variable; in the well-being domain model, we add “physical well-being” and “mental well-being,” adding positive causal relations to the “well-being” variable. When we created this model in the context of the commit/swell project, social well-being was omitted from it.

3. For the variables currently in the model, we should identify how these are affected. For example, the “mental well-being” of a person is dependent on the levels of perceived “energy.” This is a positive causal relation going from “energy” to “mental well-being”. Energy, in turn, is affected by two domain
factors: the way a person perceives the stress that is present in the domain, and how s/he recovers from these stressors. This results in the addition of two variables. The first is “experienced stress,” which has a negative causal relation to “energy,” and “recovery,” which has a positive relation with “energy.” We can think of this construct as a battery: stress and work cause the battery to drain, whereas relaxation and spending free time recharges the battery. If the battery is empty, i.e. the person has no more energy left, s/he will experience a decrease in his/her well-being.

4. We continue the process of decomposing variables and identifying how these are affected until new variables are outside the scope of the application, or when arriving at a variable that is not influenced by other variables. Variables not measured or influenced by the application are not modeled, but their existence should be kept in mind. Depending on the expected impact of the relation, we may need to model even apparently unrelated variables. Variables outside the scope of the system, i.e. those not measured by or influenced by the system, may impact the workings of the system to the same extent as internal variables.

5. After a variable has been added to the model, we should verify whether this variable has causal relations to other variables in
the model. If this is the case, we add these relations.

6. If we add a variable, we document the dimension of it as well. Furthermore, technology experts must be consulted. They help in the process to identify if, for each of the variables present, we can observe the variable through sensors, whether we have to derive the variable, or if we can control the variable or directly influenced by the system.

7. When we add variables to the model, we assign norms to them, if applicable. We can obtain these norm values from medical literature. For example, a variable “heart rate” has the dimension “beats per minute,” and a normal range for adults of 60 – 70 when in rest.

The model now contains information regarding the relevant domain elements, the relations between them, and the norm values for variables. We used this process to construct the reference model for well-being for the commit/swell project, as depicted in Figu-
A WELL-BEING DOMAIN MODEL

User centric
1. Identify application goal.
2. Identify means toward the goal.
3. Identify variables that allow goal value derivation (either through causal reasoning or direct measurement).
4. Repeat until a path from means to goal is present.

Technology centric
1. Identify available sensor technology and the variables measured by them.
2. Perform breadth-first search on incoming variable relations to identify derivable variables.
3. Decide on goal based on found variables.

Table 10: DWDM construction: creation of a model derived from a reference model

res 8 and 9 on pages 77 and 79. These models are also shown in Appendix B, which includes a description per model element.

6.2.2 Derived model

The second method of constructing an application specific domain model is by starting from an existing reference model. The advantage of this method over creating an entirely new model is an increase in the speed of development because of the reuse and reliability of the input data, assuming that the reference model has already been validated. As such, this method should be preferred over the construction of a new model if a reference model is available.

For the derivation of an application specific model from a reference model, we can take either of two approaches. Firstly, we can reason about the application domain from a user centric perspective. Secondly, we can reason by putting the technology available to us at the center. Both methods were discussed by Soriano Perez [103].

A summary of both methods is given in Table 10.

User centric variable selection

As the well-being domain is highly personal, a user-centric approach to developing the application is preferred. This approach does not take the availability of hardware or software into account, so
the development of specific devices, i.e. a specific set of sensors and actuators, might be required to run the resulting application. This process is highly similar to the way a new DWDM is created, however, it is possible to base the application context model on an existing model, rather than having to reason about undiscovered variables and causal relationships.

1. The first step in this process, is deciding upon the goal of the application, i.e. in what way will it support its user? Is it to increase the value(s) of one or more domain properties, or is a decrease in a value something we are after? The variable representing these properties are modeled in the DWDM. An example goal could be to increase “Mental well-being.”

2. Secondly, we need to identify a means in order to achieve this goal. For example, in order to increase one’s “Mental well-being,” we can offer support by reducing factors that attribute to the stressed feelings (decrease the number of “Perceived stressors”).

3. Thirdly, we have to find those variables that will allow us to measure whether we are actually approaching our goal. This is done by selecting those variables that have relations (either incoming or outgoing) with the variables from the previous step. In our example, we may ask the user regarding his/her “Experienced stress,” or regarding their feeling of “Energy.”
The “expansion” of our graph is repeated until we find variables that are “observable.” Without observable variables in the domain model, it would not be possible to measure changes in the domain. If this is the case, the source model must be re-evaluated and expanded with the help of domain experts. This can be done using steps 4 through 7 of the method for creating a model from scratch. As we have multiple observable variables in the example model, no changes are needed.

The resulting domain model should have at least one path from *means* variables, i.e. variables that are to be affected by the application, to *goal* variable(s), i.e. variables that represent the application’s goal as identified in the first step of the process. An increased number of measurable variables is preferred, as this allows for multiple ways of obtaining context information so data inaccuracy can be dealt with, however, the scoping of the model should be kept into consideration. Increasing the number of sensors used by an application will also increase technical requirements to the hardware, such as a
faster processor, more storage, and increased energy consumption. Technical experts must be consulted to make a trade-off between increased volumes of data and hardware capabilities, and regarding the completeness of sensor data and the possibility to combine data to increase completeness.

**Technology centric variable selection**

Although preference should be given for the user-centric way of design, a technology point of view can also be adopted, developing an application for a certain technology platform, such as a specific smartphone or hardware device, with known sensors and capabilities. In contrast to starting off by looking at user needs, the process is structured around the possibilities of the available hardware. It should be noted that satisfying user needs still plays a vital role in the success of the application. We will look at the same example as before.

1. The first step in this selection process is different from the previous method. Rather than looking for an intended application goal, we first identify the possibilities of the hardware platform the application will have to run on. This results in a selection of several observable variables. Using these variables, we create a new DWDM that contains the relevant subset of variables for our application.

   In this first step, we select the “Experienced stress” variable as the one being observable through sensors. In this case, the sensor would be a small software program asking the user to score their current stress level.

<table>
<thead>
<tr>
<th>Experienced stress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
</tr>
<tr>
<td>No dimension</td>
</tr>
<tr>
<td>Ordinal</td>
</tr>
</tbody>
</table>

2. Secondly, we identify what can be deduced from the identified variables. We perform a breadth-first search of the causal graph, following incoming causal relations, adding the found variables and relations to our application specific DWDM.
This results in the following model:

3. Once a satisfactory goal has been identified, we can stop searching. We repeat this process for each observable node. If we encounter an already discovered node during the search, thus connecting the observable variable (indirectly) to the graph, we also stop our search for that variable.

In the resulting model we can see that by trying to reduce “Stressors” such as “Task load” and improving aids like “Information support,” it is possible to reduce the “Experienced stress” of a worker. What cannot be seen, which was visible in the user-centric model, is that this will benefit the user’s “Mental well-being.” Similarly, in the user-centric model, the specific stressors were not shown.

6.3 A STEP COUNTER APPLICATION DWDM

This section describes the creation of an application specific DWDM for the case discussed in section 3.6 on page 28. We describe how to construct the DWDM using a user centric, and through a technology centric method, as we discussed in section 6.2.2. This process
is based on the reference well-being domain model detailed in the previous section, shown in Appendix B.

**User centric variable selection**

The first step entails the selection of the application goals. For this, we select “Physical well-being,” the application’s goal being the increase of this variable. This gives us the following initial model.

<table>
<thead>
<tr>
<th>Physical well-being</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td></td>
</tr>
<tr>
<td>No dimension</td>
<td></td>
</tr>
<tr>
<td>Ordinal</td>
<td></td>
</tr>
</tbody>
</table>

Secondly, the means to attain this goal are identified. As the application is to improve the user’s physical well-being by showing the number of steps taken per day, the “Physical activity” variable will need to be observed. We select this variable as the means for the application, resulting in the following model.

<table>
<thead>
<tr>
<th>Physical well-being</th>
<th>Physical activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Observable</td>
</tr>
<tr>
<td>No dimension</td>
<td>IMA/day</td>
</tr>
<tr>
<td>Ordinal</td>
<td>Ratio</td>
</tr>
</tbody>
</table>

Thirdly, we explore incoming and outgoing relations to and from “Physical well-being.” We first find “Cardiovascular fitness” as having a positive causal relation to our goal variable. Next, we find an incoming relation from “Body composition.” We deem additional incoming causal relations, such as from “Recovery,” outside the scope of the application, and are therefore ignored.

Looking at the causal relations toward “Cardiovascular fitness,” we see that a positive relation from “Physical activity” exists. A similar relation exists between “Body composition” and “Physical activity.” As an observable variable has been found, we can conclude this process step, giving us this model:
The final step entails assertion that a path between the identified goal, “Physical well-being,” and the means, “Physical activity,” exists. Looking at the constructed DWDM, we see that two of such paths exist, concluding the construction process of the step counter DWDM.

Although the model now captures what is measurable using sensor technology or by asking the user, the idea behind this application is to keep it simple, and not disturb the user with questions. This entails the alteration of the measurability property of two variables: for both “Body composition” and “Physical well-being” a value may be obtained, either through complicated measurements, or by asking the user. However, the step counter will not be using these techniques of obtaining a value. Both variables are altered so they are now “derivable”. An application that is constructed from the resulting DWDM will base its view of the user’s physical well-being off the values obtained from a sensor that can measure physical activity, such as an accelerometer. The final version of the DWDM is shown here:
6.3 A STEP COUNTER APPLICATION DWDM

Technology centric variable selection

The first step of the technology centric variable selection method requires us to select the variables measurable by the hardware platform. If we assume that the hardware we have available only contains an accelerometer, the only variable we can observe is “Physical activity.” This will be the first variable of our DWDM.

Next, we perform multiple iterations of a breadth-first search over the incoming causal relations. After five iterations, we arrive at the following application specific DWDM. This DWDM includes the controllable variable “Feedback,” which should allow us to affect the amount of physical activity undertaken indirectly.
The DWDM we discovered now contains several variables that cannot be influenced directly or indirectly, namely “Self-efficacy,” “Feeling of well-being” and “Physical well-being.” As we are dealing with a technology centric way of creating the DWDM, we remove these variables from the model. The resulting model only contains variables that can (in)directly be affected by a context-aware system.
6.4 DISCUSSION

In current literature, we find systems that are used by health care professionals in order to classify the health of a patient. Examples of such classification systems are the International Classification of Functioning, Disability and Health (ICF) [135] and the International Statistical Classification of Diseases and Related Health Problems (ICD) [133], both of which are international standards created by the World Health Organization (WHO). Although these systems contain information about all currently known systems and functions of the human body (in case of the ICF), and the known diseases (ICD), no visual overview is provided and no relations between the elements documented. It is therefore not possible to easily reason over the captured knowledge or deduce what actions a person should undertake in order to improve his/her physical or mental well-being. This is also the reason why it is not possible to cross-check our DWDM model with the ICD and ICF: the classifications are structured in a tree-shape, with no relations existing between leaves. Furthermore, the information provided by these sources is not easily readable by application developers. This is something that is possible in the reference well-being model provided in this chapter.

The generic well-being domain model discussed in this chapter captures mental and physical well-being aspects considered by the COMMIT/SWELL project. For future purposes, however, it may be needed to add detail to the model. Detail can be added to the model in two ways. The first technique is already used in the current generic model: a variable that adds detail is added with a causal relation to the more general variable. An example of this is the “stressors” variable, which has incoming relations of several specific stressors, such as “task load” and “distractions.” The last two provide us with more information than the first. The second way of adding detail is through substitution. Using this method, the general variable is replaced by one or more variables with increased detail. This may be beneficial if the meaning of the new variables is clear without the original variable: if the “stressor” variable was replaced by the “task load” and “distractions” variables, information would have been lost.

The model discussed in this dissertation was created in a general way: it may be the case that influences from outside the modeled context affect causal relations in the context. An example of a
well-being application that implements additional causal relations is the Roessingh Research and Development Activity Coach [77], which aims to provide the user with feedback messages that increase the user’s self-efficacy regarding the physical activity undertaken. In our reference well-being model, only the person’s awareness is deemed influenceable by providing feedback messages. The Activity Coach adds a positive causal relation from “feedback” to “self-efficacy” (Appendix G, page 262, Figure 40). As this is specific to the application (all feedback messages increase awareness, but not all enhance self-efficacy), these changes are not adopted in the reference domain model.

Although it was our intention to be correct when formulating the domain model, we did not claim that it is complete. Model completeness is challenging to claim and prove for systems with clear scoping, let alone the broad domain of well-being. The only claim we can make regarding the model is that the knowledge captured in it represents the knowledge of those that created it, with the scoping of their own expertise, at the time of writing. This does not mean it has little value. It is a starting point for discussion with other domain experts to inquire their view on physical and mental well-being. Furthermore, it can be used as a starting point for the creation of application specific domain models. Regular reiterations and validations should be made to guarantee that the model still represents the current state of the art regarding well-being knowledge, and with respect to the technical aspects captured in the models, such as the measurability of the variables.

6.5 Conclusion

Using variables and causal relations to capture the knowledge about the well-being domain can help clarify how concepts are related. This chapter describes how the DWDM language can be used to model an overall view of the well-being domain. Although scoped by the vision and focus of the COMMIT/SWELL project, this reference model can be used as a starting point for additional expert discussion, and for the development of context-aware well-being applications. More discussion among and with experts may result in alterations in the reference domain model, whereas the creation of application specific models may require the addition or alteration of causal re-
lations. Such changes are only intended for the specific application domain and are not meant for the reference domain model. We provided two methods for the creation of DWDM models: creating the model from scratch, or basing it off a reference model.

In this chapter, we provide answers to research question RQ2.3, i.e. designing a development method that improves current methods with regard to the requirements posed in earlier chapters. The method for creating domain models presented here improves current methods by reducing the time required by the domain expert (R7) and by reducing the chance for software errors (R9): domain experts are only needed to create the domain model, which can then be used in the rest of the development process, as we show in part III.

The reference model presented in this chapter contains domain knowledge that links the mental and the physical well-being domains. When using one of the two model derivation methods, i.e. user centric or technology centric, we can create application specific models. Because these application specific models use the knowledge provided by our experts, the applications based on them will use this knowledge as well, reducing the chance of incorrect assumptions regarding the domain.
In chapter 5, a modeling language for the representation of the well-being domain was introduced. In chapter 6, this language was used to create a reference model of well-being. This model, however, was based on the expert views of two people; one of them active in the domain of physical well-being, the other operating in the field of mental well-being. It should be verified that their views and opinions also reflect those of their peers.

In this chapter we discuss the validity of both the DWDM language we introduced in chapter 5 and the reference domain model we described in chapter 6 at the same time. First we discuss the research we designed to answer this problem (section 7.1). Next, we describe the execution of the research (section 7.2), and detail the results (section 7.3) and discuss them (section 7.4). Finally, we draw conclusions.

We work toward answering the following question:

**RQ3.1 [kq]** Do domain experts think the modeling language is suitable to capture the domain of well-being?

### 7.1 Research Design

In the research presented in this chapter, we aim to establish the validity of the domain modeling language and the generic domain model proposed in chapters 5 and 6. We will not ask domain experts whether they think our DWDM language is suitable, as this is unlikely to yield satisfactory answers: the domain experts do not have an expertise in modeling languages, but in the domain of well-being. We ask the experts about their opinion on the generic domain model instead, and deduce the suitability of the modeling language from their answers.
This section discusses the selected objects of study, treatment design, measurement design, and possible threats to the validity of the research design.

7.1.1 Objects of study

We constructed the reference well-being domain model with the help of two experts. We assumed that the knowledge shared by them, and represented in the domain model, was based on both knowledge of current and past literature, and experiences in practice. To validate this, we presented the model to the experts’ peers, i.e. other experts in the physical and metal well-being domains, and asked them whether they agree with the relations in the model.

We distinguished two groups of experts: those in the physical well-being field, and those with knowledge of mental well-being. Although some experts had knowledge on some of the subjects of the other field, their opinions was not be asked for these subjects.

Our object of study was the knowledge an expert has on physical or mental well-being, our population being the set of all experts in these fields. We applied generalization by analogy, assuming that the knowledge and experience of the interviewed experts is similar to the extent that much of their expert knowledge should be the same across experts. Experts from the COMMIT/SWELL project and researchers from the University of Twente were asked to participate. Those taking part in the questionnaire or interviews did so out of free will and without provided incentives. All experts had multiple years of research and/or practical experience in their fields.

7.1.2 Measurement design

The reference model of well-being captures information regarding both mental and physical aspects. However, asking an expert whether the model as a whole is correct would unlikely yield a satisfactory answer: the model is too big and contains too much information to analyze at once by a single expert. Instead, we split the model into sub-models. The sum of these encompassed the total domain model, including some overlapping variables or causal relations. The sub-models were small enough to be described in a single sentence. The sentences were used to ask the experts about
their opinions in interviews. By describing the sub-models in a single sentence, we mitigated the risk of the participants not being able to read the models correctly. Additionally, the sub-models and accompanying sentences were used to create an online questionnaire which could be filled out without the need for an interviewer to be present. Here, the participants were able to choose either “Agree,” “Do not know,” or “Disagree.” In case of disagreement, the questionnaire asked for an explanation. Again, participants were asked to respond to the sentences, rather than to the model. We counted the number of times each answer was given per model. Experts filling out the questionnaire could select “Do not know” for questions about elements that were not in their domain of expertise.

If an expert indicated that the information represented by the model was faulty, his/her argument was analyzed and judged. If multiple experts disagreed, their arguments were compared to see whether they were in agreement with each other. If the majority of arguments agreed, this was adopted as the new truth.

For the questionnaires, experts answered in a fixed format, with the option to give free-format explanations. During face-to-face interviews with experts, free-format answers were given. The interviews were recorded, interpreted, and the answers provided mapped to the fixed format answers of the questionnaire. Experts were not questioned on sub-models that covered elements not in their domain of expertise.

Because the answers given by the experts had to be interpreted. It is therefore possible that we interpreted the answers in a biased way to favor our desired outcome, causing a threat to the internal validity of our experiment. Although not completely preventable, we aimed to mitigate this risk by asking the interviewed experts to either agree or disagree with the statement, after which they could support it with claims.

A possible threat to our conclusion validity arises from splitting the model. We run the risk of reasoning with fallacy of composition [8]: not all “wholes” are equal to the sum of their parts. The context of the variables might for example influence the validity. We did not specifically address this issue in the research, as we thought that the method used was the only practical way of conducting the interviews/questionnaires. Even if the entire model would have been shown to the experts, they would only comment about the elements relating to their own field, resulting in similar composition
1. An increase in task load will result in an increase in computer activity, which causes the number of stressors to increase.

2. An increase in information support will decrease the number of stressors.

3. Distractions cause an increase in stressors.

4. An increased perceived stress is caused by an increase in the number of stressors. If a person has a higher locus of control, the experienced stress will be reduced.

5. The experienced stress can be reduced by teaching a person to cope with stress.

6. If somebody has the feeling s/he is good at his/her job, this will result in stress being experienced to a lesser degree.

7. Hobbies aid recovery.

8. An increase in the experienced stress will result in a decrease in energy.

9. Department feedback is beneficial for social interaction. This will result in increased mental well-being.

10. Increased energy will increase mental well-being.

11. If a person’s self-efficacy increases, this will result in an increase in the physical activity being undertaken.

12. Resting aids recovery, which will result in a person having an increased feeling of physical well-being.

13. If the feeling of well-being decreases, this will cause a person to undertake actions to change his/her current situation.

14. By giving feedback about the undertaken activity, the person will gain awareness in his/her situation.

15. Increased awareness will result in motivation, which will cause the person to be more physically active.

16. If a person increases his/her physical well-being, this will have a positive effect on his/her overall feeling of well-being.

17. Physical activity will result in improved body composition.

18. Improved cardiovascular fitness will result in a decreased resting heart rate.

19. Increased cardiovascular fitness will result in an improved feeling of physical well-being.

20. Improved body composition will increase the feeling of physical well-being.

Table 11: Questions asked to establish reference well-being model validity

issues. A full validation of the model would then only be possible when asking people who are experts in both the field of physical and of mental well-being. However, the risk of this issue invalidating our results regarding the reference model are deemed limited, as the interviews conducted provided more information than a binary “yes” or “no” with regard to the validity of sub-models: the context of variables and sub-models was also discussed, and nuances were recorded.

The statements used are listed in Table 11. Some of these were asked as questions in the interviews. The sub-models accompanying these questions are listed in Appendix C.

As the participants taking part in the experiment were known through the COMMIT/SWELL project, or employees of the University
of Twente, it is possible that the answers provided were “desired,” i.e. what the participants thought the interviewer wanted to hear. This risk is more prevalent in the face-to-face interviews. This caused a threat to our external validity. To mitigate this risk, the experts interviewed were always asked to support their opinion with arguments, preferably citing literature.

7.2 RESEARCH EXECUTION

This section reports on the execution process of the research specified in the previous section.

7.2.1 Subjects

In the experiment executed, we collected data in two different ways. First, we interviewed experts, either in person or through phone calls or Skype. All interviews were recorded for processing at a later time. Second, we distributed questionnaires for experts to fill out at their own discretion.

The experts had varying levels of experience. The least experienced subjects were Ph.D. candidates with over three years of professional experience. These subjects did have a relevant study background, so they have additional previous experience in the field. The most experienced subjects were practitioners who had been working in the field for decades.

A total of twelve experts were questioned: five experts were interviewed, seven responded to our questionnaire. The experts that answered the questionnaire were self-selected, responding to an e-mail request. This e-mail was sent to members of the COMMIT/SWELL project. Of the twenty people active in this mailing list, seven participants responded.

We summarized the information about the participants in Table 12.

7.2.2 Case

The first step in the execution of the experiment entailed the creation of a questionnaire. This questionnaire was based on sub-models of the reference domain model. We decided that the sub-
models were to be as small as possible. We evaluated each causal relation in the reference model if it would be understandable if only this relation and the related two variables, i.e. the source and target of the relation, were taken as a sub-model. This was the case for relations such as $\text{Self-efficacy} \rightarrow \text{Physical activity}$. Other relations, however, required additional causal relations and variables to be added in order to be understandable, such as $\text{Rest} \rightarrow \text{Recovery} \rightarrow \text{Physical well-being}$.

When all sub-models were created, we made sentences to describe the information modeled. These sentences were in the form of statements.

Using the questionnaire functionality of Google Drive\(^1\), we made forms that adhered to the following template:

- A depiction of the sub-model.
- Sentence describing the information in the diagram.
- Options: “Agree,” “Do not know,” and “Other.”.

In the questionnaire description, we explained that the “other” option should be regarded as “disagree.” This choice, however, was

\(1\) https://drive.google.com

Table 12: Model validation experiment participants

<table>
<thead>
<tr>
<th>Method</th>
<th>Age</th>
<th>Field of expertise</th>
<th>Experience (years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questionnaire</td>
<td>25–30</td>
<td>Software</td>
<td>0–5</td>
</tr>
<tr>
<td></td>
<td>35–40</td>
<td>Behavioral change</td>
<td>0–5</td>
</tr>
<tr>
<td></td>
<td>25–30</td>
<td>Health care</td>
<td>0–5</td>
</tr>
<tr>
<td></td>
<td>40–45</td>
<td>Software</td>
<td>15+</td>
</tr>
<tr>
<td></td>
<td>40–45</td>
<td>Occupational Safety</td>
<td>15+</td>
</tr>
<tr>
<td></td>
<td>25–30</td>
<td>Software</td>
<td>0–5</td>
</tr>
<tr>
<td></td>
<td>30–35</td>
<td>Software</td>
<td>10–15</td>
</tr>
<tr>
<td>Interview</td>
<td>40–45</td>
<td>Behavioral change</td>
<td>15+</td>
</tr>
<tr>
<td></td>
<td>25–30</td>
<td>Health care</td>
<td>0–5</td>
</tr>
<tr>
<td></td>
<td>45–50</td>
<td>Health care/Software</td>
<td>15+</td>
</tr>
<tr>
<td></td>
<td>35–40</td>
<td>Behavioral change</td>
<td>15+</td>
</tr>
<tr>
<td></td>
<td>25–30</td>
<td>Behavioral change</td>
<td>5–10</td>
</tr>
</tbody>
</table>
not named as such, because of limitations in the questionnaire software: only the option named "other" allowed an additional input field asking for an explanation.

The interview questions consisted of the same sentences used for the questionnaire, this time formulated as questions, rather than statements. The goal of this change was to get more elaborate answers from the experts than a simple agree/disagree.

7.3 RESULTS

The results obtained are twofold. Firstly, we have the results gathered through the questionnaire. Secondly, there are the results gathered by interviewing the experts. We will discuss the results obtained per question number, as listed in Table 11. We have indicated the parts of the model the questions are about in Figures 10 and 11.

1. An increase in task load will result in an increase in computer activity, which causes the number of stressors to increase.

   QUESTIONNAIRE Only two participants agreed with this statement, five others disagreed. They all agreed on the fact that it is not necessarily computer activity increasing if the task load increases, this depends on the type of task.

   INTERVIEWS The experts agreed that this could indeed be the case, noting that an increase in computer activity can be part of the result of increased task load, not all work is done on the computer.

   CONCLUSION Even though it appears the two groups of experts are in disagreement, looking at their reason for their answer makes it clear they do agree with each other. The consensus over the statement is agreement: if there is more computer activity, this can be caused by, and as such imply that there is, an increase of task load.

2. An increase in support to manage information will decrease the number of stressors.

   QUESTIONNAIRE Of the seven participants, six agreed with this statement, one disagreed. The disagreeing participant mentioned
that good information support could help, but distracting information support would only add to the problem.
7.3 RESULTS

Figure 11: Elements the mental well-being model referenced to in the questionnaire

INTERVIEWS Noting that information support will likely prove the most beneficial for those who are not competent themselves in finding information, the experts agreed with the statement. However, they also noted that if people are used to finding information themselves, offering them support may result in the opposite, i.e. it may become a hindrance.
CONCLUSION Both groups of experts agree with this statement, so no changes are needed in the model.

3. Distractions cause an increase in stressors.

QUESTIONNAIRE Three people disagreed with this statement. The first mentioned that as distractions are a type of stressor, there is no causal relation between the two: it is inherently true that an increase/decrease in distractions results in the number of stressors being increased/decreased, as distractions are stressors. The second participant noted that distractions can be ways of introducing relaxation, while the third participant mentioned that distractions are no stressors if there is no fixed goal or pressure. Two participants agreed with the statement, two others did not know.

INTERVIEWS The question asked states “distractions.” The experts agreed, however, that some distractions can be beneficial to the person. If s/he is looking for some distractions in the form of drinking coffee with colleagues, this can be a good thing. However, distractions may also be a hindrance, which will be stressors.

CONCLUSION Both groups agreed that the term “distractions” does not fully cover what is intended in the model, being negative distractions. A variable that models positive distractions should be added.

4. An increased perceived stress is caused by an increase in the number of stressors. If a person has a higher locus of control, the experienced stress will be reduced.

QUESTIONNAIRE Of the seven participants, three indicated that they agree, with one of them noting that there is no such thing as “high” or “low” locus of control, only internal and external. Four participants indicated they did not know, with one also mentioning the discrepancy in the “high” vs “low” locus of control description.

INTERVIEWS The experts agreed with this statement, one referring to the self-determination theory [28, 85]. This theory states that the concepts of “autonomy,” “competence,” and “relatedness” are
positive factors for mental well-being. This question regards autonomy.

**Conclusion** Although he questionnaire results are inconclusive, the interviewed experts were in agreement with each other, citing sources to back up their statements. Because of this, we conclude that this relation does not need changing.

5. **The experienced stress can be reduced by teaching a person to cope with stress.**

**Questionnaire** Again three participants agreed, while four did not have enough knowledge to answer the question.

**Interviews** This statement was confirmed by the experts.

**Conclusion** With no expert disagreeing with this statement, the model will not need changing with regard to this relation.

6. **If somebody has the feeling s/he is good at his/her job, this will result in stress being experienced to a lesser degree.**

**Questionnaire** Five participants thought this statement was true, two did not know the answer.

**Interviews** One of the experts agreed with the posed statement, again citing the self-determination theory, this time relating the statement to the “competence” aspect. Another expert noted that although some people know they are good at their job, their experienced stress still increases; people suffering from this have the feeling that because they are good at their job, they should perform better than possible, i.e. good is never good enough. This will increase stress.

**Conclusion** Although one interviewed expert disagreed with the statement, we chose not to alter the model: the group of people for which the statement “an increase in feeling good at ones job will increase the level of stress” is a minority, and may be suffering from
psychological issues. As the domain model is aimed at the general population, such a minority cannot be taken into account.

7. **Hobbies aid recovery.**

**Questionnaire** One of the participants disagreed with this statement, indicating that not all hobbies aided recovery, but that some hobbies could be stressors as well. No further examples were provided. Four other participants agreed with the statement, while two did not know.

**Interviews** Hobbies may aid recovery, experts agreed, but there were reservations. Hobbies should, for example, be experienced as free time, rather than yet another obligation: not all hobby activities are experienced similarly, they may become a chore after a while. Also, there is a certain optimum with regard to free time: too much free time will result in a person no longer being motivated, causing a lower feeling of fulfillment. This has a negative impact on the feeling of mental well-being.

**Conclusion** The experts agreed that the statement holds in general, with some remarks that the real world is more subtle. No alterations are made to the model.

8. **An increase in the experienced stress will result in a decrease in energy.**

**Questionnaire** Two participants noted that stress can also result in (short term) increased energy, mentioning a certain level of stress is always required for people to be productive. In the long run, however, stress would reduce energy. Three participants agreed with the statement, two did not know enough regarding the subject.

**Interviews** According to the experts, increased stress will result in a decreased feeling of energy. One expert noted that it is therefore important to have a balanced work-personal life.

**Conclusion** With all experts in agreement, the model remains unchanged for this relation.
9. Department feedback is beneficial for social interaction. This will result in increased mental well-being.

**Questionnaire** Four of the responses to this statement were in agreement, however, three of them disagreed. Two of these disagreements agreed on that the effect of feedback is related to whether it entails positive or negative feedback. One participant noted that it depends on the personality of the person receiving the feedback: s/he might not want social interaction.

**Interviews** In general, the experts agreed, noting that it does matter what the contents of the meeting is for it to be benefit a positive experience. This statement could be described by the “relatedness” aspect of the self-determination theory.

**Conclusion** Overall, both groups of experts agreed with the statement, with both groups noting that this is dependent on the type of interaction. The reason for the difference in level of agreement between the two groups was likely caused by the possibility of expressing the opinion: those filling out the questionnaire were bound by the strict rules of the questionnaire forms, whereas the interviewees were more free in expressing their opinion.

10. Increased energy will increase mental well-being.

**Questionnaire** One person did not have enough knowledge to answer this question. Five participants agreed, while one participant noted that energy can be seen as a part of mental well-being.

**Interviews** The experts all agreed with this statement.

**Conclusion** The general consensus was one of agreement, so no changes to the model will be made.

11. If a person’s self-efficacy increases, this will result in an increase in the physical activity being undertaken.

**Questionnaire** With three people agreeing with this statement, and three people not having enough knowledge, one person dis-
agreed. They remarked that the person also needs to have the intention to undertake physical activity, not just knowing they are able to perform the activity correctly. Furthermore experts remarked that the effect “is probably not linear and has an upper limit.”

**Interviews** Although agreeing with this statement, one expert noted that the undertaking of activity is dependent on the activity people believe they are good at: if they think they will be good at walking, they will not suddenly start running, although this might be the case in the future.

**Conclusion** With the experts in agreement, no change is made to the model with regard to this statement.

12. **Resting aids recovery, which will result in a person having an increased feeling of physical well-being.**

**Questionnaire** Four of the participants agreed with this statement. One participant, however, noted that “too much rest is not beneficial.” Two participants neither agreed, nor disagreed.

**Interviews** Experts agreed that resting aids recovery, which is good for the feeling of well-being, but noted that the duration of the rest is important too. Resting too long may result in them becoming too passive: they may be over-compensating for the activity undertaken.

**Conclusion** Both groups of experts agreed with the statement, both making note that too much rest is not beneficial. As such, the model will be adapted, adding an upper limit to the “Rest” variable.

13. **If the feeling of well-being decreases, this will cause a person to undertake actions to change his/her current situation.**

**Questionnaire** Five participants in total agreed with this statement, one noting they did not have enough knowledge to answer. One participant disagreed, mentioning that an increase in the “feeling of well-being could increase the intention to change in another
domain.” What other domain was intended here, was not mentioned.

**Interviews** The experts partly agreed on this, noting that it is highly dependent on the person. Some people indeed undertake actions to change the situation, whereas others might not see the problem with the situation, or accept the situation as it is. One expert noted that, depending on the feedback presented, people might also want to change the current situation even though they already are happy; changing the situation may result in further improvement of how a person is feeling, regardless of how happy they currently are.

**Conclusion** Although not without making mention of subtle implications of the statement, the experts agreed with the statement in general. The model will not be altered for this causal relation.

14. **By giving feedback about the undertaken activity, the person will gain awareness in his/her situation.**

**Questionnaire** One person did not know the answer to this question, three agreed. Of three disagreeing, one mentioned that this was a too simplistic view of reality.

**Interviews** The experts agreed that people will likely become more aware, one expert noting that this has not been fully established in literature and this is a current active field of research.

**Conclusion** The experts filling out the questionnaire and those being interviewed disagreed with each other in this statement. The primary difference between the groups might again be in the way answers are provided, the interviewees being able to express their answers in a free format, rather than only a choice from three options.

15. **Increased awareness will result in motivation, which will cause the person to be more physically active.**
QUESTIONNAIRE Three people did not agree with this statement. A comment made was that awareness ("I am aware that smoking is bad.") does not always lead to change (quitting smoking). Three others agreed, and one person did not have enough knowledge on the subject.

INTERVIEWS As with question 13, the experts noted that this will be highly dependent on the person; as an example, smoking was mentioned: even though people know smoking is bad for their health, they do not quit their habit. The right motivation has to be found to persuade people to alter their lifestyle.

CONCLUSION The statement did not receive either agreements or disagreements from both groups of experts. The experts did agree that how a person copes with increased awareness will differ for each individual. As neither agreement, nor disagreement was found, we have chosen not to alter the model: when following the line of reasoning made by the experts, either a positive change can happen, or no change is made, i.e. the strength of the relation may differ. A different type of causality is highly unlikely. For example if a person is made aware smoking is bad, this might result in a change: s/he might reduce his/her smoking, or remain at the same level, but an increase is unlikely.

16. **If a person increases his/her physical well-being, this will have a positive effect on his/her overall feeling of well-being.**

QUESTIONNAIRE All participants agreed that an increase in physical well-being would increase the overall feeling of well-being. In the questionnaire, this question was posed as "If physical and/or mental well-being improve, this results in an overall improved feeling of well-being, and vice versa." Two participants noted that this "vice versa" relation was not realistic. As this would result in an unrealistic situation of ever improvement, the question was altered for the conducted interviews. The result from this questionnaire statement, was used when making the alterations to the reference domain model.

INTERVIEWS All experts agreed on this.
CONCLUSION As stated, the original domain model is to be altered such that there no longer is a relation from “Feeling of well-being” to both “Physical well-being” and “Mental well-being.”

17. Physical activity will result in improved body composition.

QUESTIONNAIRE A small majority of the participants agreed with this statement, three noting they did not know.

INTERVIEWS As with the previous question, all experts agreed with this statement, one mentioning that there is, however, an optimum to this; the expert was of the opinion that some professional level athletes are no longer healthy. During the interviews, the concept of “body composition” had to be explained to the experts.

CONCLUSION Body composition was found to be a term not generally used by the experts. The model should be altered to be more clear.

18. Improved cardiovascular fitness will result in a decreased resting heart rate.

QUESTIONNAIRE Although this question is based on basic physiological knowledge, four participants did not feel comfortable agreeing or disagreeing. Three participants agreed with the statement.

INTERVIEWS Without discussion, all experts agreed with this question.

CONCLUSION As experts did not disagree, the model will not change with regard to this relation.

19. Increased cardiovascular fitness will result in an improved feeling of physical well-being.

QUESTIONNAIRE Of all the participants, four agreed with this statement. No participants disagreed, while three did not know.
VALIDATING THE GENERIC DOMAIN MODEL

Figure 12: Results of the well-being domain model questionnaire

INTERVIEWS Again, no discussion or arguments were posed, all experts agreeing.

CONCLUSION With only agreement, we assume this relation too is modeled correctly.

20. **Improved body composition will increase the feeling of physical well-being.**

QUESTIONNAIRE None of the participants disagreed with this statement. Three participants agreed, while four indicated they had no knowledge on the subject.

INTERVIEWS This question was answered in agreement as well.

CONCLUSION With only agreement by the experts, we again assume this relation to be correct.

The results are summarized in Figure 12.
7.4 DISCUSSION

Based on the results obtained through the questionnaire and the interviews, we can draw conclusions about the use of the DWDM modeling language and about the validity of the reference commit/swell domain model.

7.4.1 Results compared

When we compare the results obtained from the questionnaire and the expert interviews, we find that both groups have similar views on the well-being domain. Even though some questionnaire participants indicated for some questions that they did not agree with a statement, from their answer it became apparent that they agreed, but had some reservations regarding the rigor of the statements. The experts interviewed phrased such answers as “Yes, but . . .”. Due to the format of answering questions, i.e. fixed versus free format, answers seemed different at first glance, but were similar after further inspection.

During the expert interviews, we only asked questions relevant to their field of work. This was not the case for the participants of the questionnaire, who were asked all questions and were provided the option not to answer. We therefore expected that for each question a consistent number of participants would indicate they did not know the answer to the question. This was not the case. From this, we can conclude that there have been participants answering questions outside their specific field of expertise, or that they considered their knowledge of the other domain sufficient to answer these questions. As this reduces the value of the answers provided (we cannot assume that all answers were only provided by true experts in the particular field), we value the answers and opinions provided during the interviews higher.

7.4.2 Domain model

Looking at the answers given by those participating in the questionnaire and by the experts interviewed, we find that in general both groups agreed with the information captured in the model and with
each other. Additions and alterations to be made to the model are as follows:

- Altering the “distractions” variable, renaming it to “hindrances,” i.e. a negative, unwanted distraction, and adding a variable “diversions”, i.e. a positive, desired distraction, with a negative causal relation to “experienced stress.”

- Removing the causal relations from “Feeling of well-being” to “Physical well-being” and “Mental well-being.”

- An upper limit to “Rest” should be added.

- Adding more detail by removing the “Body composition” variable, and adding variables such as “Body fat percentage” and “Lean body mass,” as it was found that the term “Body composition” was too abstract to properly reason over.

- Removing the variable “Heart rate,” as this caused confusion. As a result, “Resting heart rate” and “Active heart rate” remain and should be altered to be observable.

The improvements proposed here have been applied to the reference commit/swell domain model. The new model is shown in Figures 13 and 14.

7.4.3 Modeling language

While evaluating the domain model, we also obtained information that lets us evaluate the modeling language in which the model was created. Shortcomings can be identified by finding statements made by the experts or questionnaire participants that cannot be modeled using the modeling language.

Looking at the questionnaire and interview responses, it becomes clear that personalization is a highly important aspect of the well-being domain. Many answers given were in the form of “Yes, but …”. Such nuanced views cannot be modeled in the DWDM language at the moment.

Quantification of data is also a potential problem. Whereas “observable” variables inherently have a value and dimension, this is not the case for abstract concepts such as “work efficacy,” or “cardiovascular fitness.” Allowing to quantify these would ease reasoning over them.
The DWDM language has three levels of measurability. Many of the variables in the mental well-being field can, strictly speaking, not be captured using any of these classifications: users should provide these. These variables are currently denoted as observable, as we can regard the questionnaire used to obtain the value as a measurement device. We could add a new measurability level such as “askable” to increase the expressiveness of the models.

Another addition to the modeling language, would be the ability to model temporal aspects. Although not mentioned by those interviewed, the ability to model time, e.g. how long until an effect is at its maximum, how long will it last et cetera, would greatly improve the expressiveness. We did not consider this feature when developing the language. Experts did mention that personalization is of great importance when modeling the domain of well-being; the ad-
validating the generic domain model

Figure 14: Improved reference COMMIT/SWELL domain model: Mental well-being

dition of temporal aspects would improve the ability to personalize the models.
In chapters 5 and 6 we proposed a modeling language to capture the well-being domain and a reference well-being domain model. Through a questionnaire and expert interviews, we aimed to validate both in this chapter, answering research question RQ3.1: do the domain experts think the modeling language is suitable to capture the domain of well-being. We used the same questions in the questionnaire and the interviews, which were derived from the causal relations captured in the reference model.

Results from the questionnaires and the interviews indicate that the current model, in general, is a correct representation of the knowledge available in the domain. What also became apparent, is that personalization is highly important: basic physiological processes will be the same for everybody, but the way people interpret feedback and adhere to advice will be different from person to person, and even from day to day.

The DWDM language seemed, judging by the expressiveness required by the additional statements made by the experts, suited to capture the well-being domain. We summarized additions and changes to it to improve this expressiveness. As such, we have to answer research question RQ3.1 with a tentative yes: although the modeling language appears to be suited to capture the domain of well-being, some improvements should be made to enhance it.

In the application development process, as we discuss in part III from page 123 onward, we have used the original domain model instead of the improved model discussed in this chapter. The model was perceived useful by the participants of the development experiment described in chapter 9. As such, we argue that the improved version of the model will be perceived equally useful.
Part III

MODEL-DRIVEN DEVELOPMENT OF CONTEXT-AWARE WELL-BEING SYSTEMS
In the previous part, we used the DWDM language to capture well-being domain information. In this part, we use the resulting models to develop applications. This chapter discusses a model-driven method as an application development approach. The process consists of two parts. The first phase of application development consists of converting the information depicted in the DWDM into models which are more traditionally used in software engineering. The second phase entails the generation of application code from these models. This model-driven process was derived from general principles of Model-Driven Engineering. We created this method to improve development speed and reduce effort of developing context-aware well-being applications.

This chapter first discusses the concept of modeling and model-driven development (section 8.1), after which we present the modeling levels, as recognized in the Model-Driven Architecture (section 8.2). Continuing, the development process is detailed (section 8.3), and means of automation are discussed (section 8.4). We work toward answering the following research question:

**RQ2.3 [DP]** Design a method for developing context-aware well-being systems that improves current methods with respect to the requirements specified in chapter 3.

### 8.1 Model-Driven Development

According to Kurtev [58] “A model represents a part of the reality [...] and is expressed in a modeling language. A model provides knowledge for a certain purpose.” Models allow us to capture knowledge about the real world, and represent it in an abstract form. The
Model-driven development using DWDMS

The act of converting real world knowledge into such an abstract form required is called modeling. Modeling has become common practice in nearly every field, from (software) engineering to business.

We can identify two broad groups of modeling languages. The first are the general purpose modeling languages. The best known example of such a language is the Unified Modeling Language (UML). Languages like these can be used to model a broad range of domains and are primarily used in software development: while modeling, the software engineer defines objects and relations between them, which model the subject domain, i.e. the domain in which the software will operate. These objects are, when writing the software, translated into software objects. The idea is that, when executing the software, the objects that exist at run-time represent real world objects that exist in the domain of interest.

Despite the claim of these languages that they are general purpose, the need may arise to create models that are aimed at a particular field of interest. Such models are created in a Domain Specific Language (DSL). Rather than allowing any domain to be modeled, the language prescribes the syntax of the models more strictly, using a terminology common to the subject domain. For example, in a domain where people are modeled, rather than having an object “Person” of which the “name” attribute should be instantiated at run-time, a DSL can have elements that directly represent individuals, identified by their names. DSLs have the ability to lower the level of abstraction, which allows the experts in the domain to express their knowledge using the terminology they are accustomed to; this is not always possible when presenting them a model created in a general purpose modeling language such as UML, as the level of abstraction might be too high for them to instinctively work with.

To define a modeling language, either general purpose or domain specific, we need a language to write the definition. This second language is usually defined so that it can be used to define many modeling languages, including itself. These principles are described by the Meta-Object Facility (MOF) [76], which is a standard of the Object Management Group (OMG). The MOF specifies how modeling languages can be defined by using four modeling levels. We list these levels in Table 13.

The use of metamodels and models as a primary development artifact is common in Model-Driven Engineering [54]. In MDE, mo-
### 8.2 Modeling Levels

The models used in a model-driven development process can reside at different levels of abstraction and detail. We use the three levels introduced by the Object Management Group in the Model-Driven Architecture (MDA) [68] to classify these modeling levels.

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M3</td>
<td>Metametamodel</td>
<td>Metametamodels specify a language to describe metamodels. A model at M3 is an instance of itself. OMG’s MOF and Eclipse’s Ecore are examples of metamodeling languages.</td>
</tr>
<tr>
<td>M2</td>
<td>Metamodel</td>
<td>Language for describing models, which is an instance of the metamodel. Domain specific languages are defined at this level. Modeling languages as the Business Process Modeling Notation (BPMN) and the Unified Modeling Language (UML) are examples of entities residing at this level.</td>
</tr>
<tr>
<td>M1</td>
<td>Model</td>
<td>Instances of metamodels. Most modeling tools allow users to work at this level. A Class diagram is an example of an M1 entity.</td>
</tr>
<tr>
<td>M0</td>
<td>Model instances</td>
<td>Runtime instances of the models of the M1 level reside at this level. Entities at level M0 should not be regarded as models. An example of an M0 entity is a Java class file.</td>
</tr>
</tbody>
</table>

Table 13: Levels of the OMG Meta-Object Facility [76]

dels are the important design artifacts. By establishing standards for working with these models, such as having all models in the same process adhering to the same metamodel or metametamodel, it is possible to achieve interoperability between models at different stages of the development process. The Model-Driven Architecture [55, 68] is such a standard for MDE developed by the Object Management Group (OMG). Model transformations can be part of this process. In a model transformation, an input model, which is an instance of a certain metamodel, and a model transformation definition are fed to a model transformation engine which produces an output model that in turn is an instance of a metamodel; this second metamodel can be the same or a different metamodel from the input metamodel. The model transformation definition uses the structure of both the input and the output metamodels.
8.2.1 Computational Independent Models

The highest level of abstraction for models, is that of the Computational Independent Model (CIM). Models at this level do not contain any information regarding the use of implementation technology or means of processing this information, they do not provide a specification of functionality. Models created in the DWDM language are at this level: elements and their relations are discussed, but how these should be used in the resulting software system is not considered.

8.2.2 Platform Independent Models

Models at the Platform Independent Model (PIM) level of detail include the same information as CIMs, i.e. elements in the domain and the relations among them, but add additional information regarding computations. This information is described in a way that does not include technology platform specific methods. We use the Unified Modeling Language (UML) for expressing models at this level. We distinguish between two types of models: those that represent the structural aspects of the system under development, and those dealing with behavior. In the process detailed in this chapter, we use UML Class diagrams to capture the software’s components and their relations, and use UML Activity Diagrams to depict the behavior of these specific components.

8.2.3 Platform Specific Models

Models at the Platform Specific Model (PSM) level include, in addition to the information present in the PIM level models, details about the implementation platform, such as specific method and API calls. We use UML diagrams in the process described in the next sections. From these models, we are be able to generate application code.

8.3 Development Process

The process of creating a context-aware well-being system based on DWDMs consists of two phases. The first phase entails adding
8.3 Development Process

Figure 15: Overview of the partially automated development process.

detail to models, moving from the CIM level to the PIM level, and finally to the PSM level. The second phase regards the generation of code from these models, and the assembly of this code to form a compilable application. We show the structure of this process in Figure 15. In this section, we describe the steps a developer should take to move from one model to the next. Parts of this process can be automated, which we show in section 8.4.

8.3.1 CIM to PIM

After we created a CIM level model, i.e. an application specific domain model, using the method discussed in section 6.2, we used the information captured in this model to define both static and dynamic application structures of a PIM level model.

Static structure

We based the basic structure of the platform independent models on the general architecture for well-being systems as proposed by Bosems et al. [20]. This architecture is shown in Figure 16. The architecture was constructed during a workshop session with commit/swell members; during this workshop, the Class-Responsibilities-Collaboration (CRC) method as described by Beck and Cunningham [10] was used to determine the required components and the way they interact. It would be possible to use
a different reference architecture, as long as the basic components such as sensors and actuators were available. Changing this underlying architecture would not alter the model-driven method itself, the steps would remain the same. However, the CIM to PIM model transformation would need to be altered to reflect the structure of the new architecture, and the architecture of the resulting application would change, altering qualities such as modularity and maintainability.

We transform the DWDM models into UML Class diagrams, so software engineers can work with them during the software development process. The names used for the classes in the architecture diagram differ from terms used in the DWDM. This choice was made, because Class diagrams are aimed at creating a solution (which consists of (software) sensors, actuators and storage), whereas DWDMs are there to capture the problem domain.

The architecture consists of several classes, with Storage (representing means of storing data), Sensor (representing means of measuring data) and Concept (representing data that should be reasoned about by combining information measured through sensors) being abstract, i.e. these cannot be instantiated directly, but should be extended by other classes. The Storage and Sensor classes have methods that allow for the querying of the value of the variable represented by the class, whereas instances of the Concept class have methods to influence the represented variables value.

The components shown in the overall architecture have predictable ways of interacting with each other. We recognize three ope-
8.3 Development Process

Figure 17: Component interaction when obtaining data

...rations the application will carry out: (i) obtain data and information, (ii) perform reasoning, and (iii) perform interventions. When obtaining data and calculating information about concepts, the interaction between components is as shown in Figure 17. The interactions are as follows:

1. The DataInterpretation component requests new sensor data from the Sensor component.
2. The Sensor component returns the most recently read value.
3. The DataInterpretation component queries the Concept component to calculate a new value.
4. The Concept component returns the calculated value.
5. The DataInterpretation component stores the obtained and calculated data in the Storage.

We should note that this flow depends on the actual components used. Some sensors might be event driven, sending data as events occur rather than waiting for a query. Furthermore, not all applications have Concept components, so steps 3 and 4 might not always be needed.

The second type of interaction, performing reasoning, consists of the PlanAction component retrieving information from the Storage component, and using internal algorithms to decide whether to pose an intervention. If an intervention is required, the component interaction is as we show in Figure 18.

1. The PlanAction component retrieves data from the Storage component
2. The Storage component returns the requested data.
Now that we have an overview of the relations between components, we can start defining the process of creating an application. As a first step in this process, we create extensions to the abstract classes. We do this by looking at the measurability property of the variables in the domain model. If the variable’s measurability property is “observable,” a new class that extends the Sensor class is created in the PIM diagram. The name of this new class is the name of the variable, appended with the phrase “Sensor.” To allow for causal reasoning, we store information regarding the DWDM variable in a class property (measuredVariable), and a way of retrieving this information (getVariable). Additionally, the class has a method to initiate the data collection from the hardware sensor (collectData()). For “derivable” variables, we add classes that extend the Concept, which also has the variable information and means retrieval, and a method for getting which sensors could indirectly provide information regarding the direction of change of the variable’s value (getSensors()). Variables which are “controllable” result in the creation of classes that extend the Actuator class. Again, such classes include information regarding the original DWDM variable, but also include a method to retrieve the current state of the actuator (getStatus()), and a method that can set the status of the actuator (setStatus()). If no “controllable” variables exist in the DWDM, it is not possible to instantiate actuators. In such a case, the software developer should manually implement a user interface that can convince the user to change his/her behavior such that the relevant elements identified in the DWDM are altered. An example of such a
persuasive UI, is that of the Activity Coach [77], which is discussed in more detail in Appendix G.

We can apply this process to our running example of the step counter application. We start by looking at the elements in the DWDM diagram. Here we find one observable variable, being “Physical activity”. For this variable, a class extending the Sensor class is generated; this class is called “PhysicalActivitySensor”. The other variables are derivable, causing the generation of three classes extending the Concept class: “BodyCompositionConcept,” “PhysicalWellbeingConcept,” and “CardiovascularFitnessConcept.” As no controllable variables exist, a UserInterface class, extending Actuator, is generated. The result is shown in Figure 19.

Dynamic structure

Where the static application aspects are dependent on the measurability of domain variables, the dynamic structure is determined by four DWDM aspects: the dimension of variables, the causal relations between variables, and norms.

Dimension From the dimension of a variable, we can learn whether a single sensor value can be used to determine its value, or multiple data points have to be consulted. For example, if a heart rate sensor stores a data point for every heart beat, we have to count these for the last minute to provide the correct variable value, i.e. the number of beats per minute. This calculation requires the sensor information from the heart rate sensor, and the time. Alternatively, we can count the number of beats in the last few seconds and calculate the beats per minute based on the taken time frame.
Whether calculations based on a smaller time frame are possible depends on the resolution of the sensor that is used. An example of sensor data collection from a sensor that requires only a single value can be found in Figure 20a, while the calculation of a value based on multiple data points is shown in Figure 20b.

**Causal relations** The properties of the variables dictate how the variable values are calculated. However, not all variables in DWDMs are immediately readable through sensors: the value of some variables has to be derived through reasoning. This can be done by employing *causal reasoning*. We discussed this process in section 5.2.4 on page 69. To translate this process to the application, all classes that extend the `Sensor` or the `Concept` class are given a `getValue()` method. If a variable is derivable, i.e. in the application it is an implementing class of `Concept`, the `getValue()` method calls the method with the same name from the classes corresponding to DWDM variables that have outgoing causal relations to the class. We clarify this process in the steps below. Observe the following
In this DWDM, the variables “Recovery” and “Coping” are observable, the variables “Energy” and “Experienced stress” are derivable. Because of these properties, the values of “Recovery” and “Coping” can be determined directly, for example through a questionnaire, but the values of “Energy” and “Experienced stress” have to be reasoned about. If we call the getValue() method of each of these classes, the values of them would be obtained in the following way:

- **Recovery**: direct
- **Energy**: Recovery.getValue() - ExperiencedStress.getValue()
- **ExperiencedStress**: -Coping.getValue()
- **Coping**: direct

The way derivable variables are quantified in this example is simplified when compared to calculations that may be required in practice. “Energy” has to be calculated by determining the value of “Recovery”, the value of “Experienced Stress”, and as such, the value of “Coping”. A derivable variable will cause calls to the getValue() method of other derivable variables, until an observable variable has been found of which the actual value can be used. This is summarized in Listing 1. From this code it becomes apparent that a DWDM without observable variables cannot yield a system that is context-aware.

The polarity of the causal relations should be taken into account when quantifying derivable variable values. In this example, the only method of this quantification is by adding/subtracting the values of the variables that have outgoing causal relations to the derivable variable.

Causal reasoning is also used to determine how variable values should be affected. Every class representing a variable is equipped with an increase() and a decrease() method. If either of these methods is called, reverse causal reasoning is used: we want to identify
which variables should be affected such that the value of the variable that should increase/decrease changes. To do so, the `increase()` or `decrease()` methods of the classes representing variables that have outgoing causal relations to the variable that the application intends to affected are called similarly to the `getValue()` method discussed earlier, i.e. the class can alter the value directly, or calls the method of another class. Instead of performing this method execution until a `Sensor` class is reached, the application executes the method until a class derived from a “controllable” variable, i.e. a class extending the `Actuator` class, is encountered. We show the pseudo-code for this in Listing 2.

Listing 1: Obtaining the value of a derivable variable

Listing 2: Increasing a variable’s value
The final aspect captured in DWDMs that can affect the system behavior, are the norms that can be added to the variables. Where the behavior described until now concerns obtaining variable values, norms dictate what interventions should be posed and when these interventions should occur. We can perform causal reasoning to determine long-term plans, but if a measured variable’s values are currently outside the scope that is deemed normal, the system based on the information captured in the DWDM has to take actions more quickly. For the system to be able to take these actions, an if-then-else construct that is called every time a sensor value is obtained is generated for each variable-norm pair. For example, we can calculate the average maximum heart rate as $206.9 - (0.69 \times \text{age})$. This value can be set as the upper bound norm for the heart rate variable. If a heart rate is measured and the value is greater than this norm, the system might alert the user to slow down his/her physical activity and relax immediately. The example is shown in Figure 21.

When generating the dynamic structure of the step counter application, several models are created. Firstly, there are those modeling the way the value of the represented variable is to be obtained. For the PhysicalActivitySensor class, this model is identical to the one shown in Figure 20b, as physical activity is an observable variable of which the value has to be calculated over a period of time. Additionally, the physical activity variable has a norm associated with it,
causing the generation of an additional behavioral model. We show this in Figure 22. The dynamic structure generated for the class representing the physical well-being variable is shown in Figure 23, the diagrams for the classes representing the cardiovascular fitness and body composition variables being similar. A difference between Figures 22 and 23 is that the first includes decision nodes, which are absent in the second diagram. This difference is caused by the presence of norms: if a variable does not have an associated norm, the application cannot alert the user if a certain variable value is calculated, as it has no knowledge about “right” or “wrong” variable values.

8.3.2 PIM to PSM

Platform Specific Models add detail to the Platform Independent Model level by introducing technology and implementation information. If the modeling language used for the PIM level model has enough expressiveness, it can be reused to create the PSM level model. If UML Class and Activity Diagrams are used for the PIM level
models, it is possible to reuse them for the PSM level models. Because of dependencies on the actual implementation platform, we will not discuss the specific platform mapping details, but will give an overview of the steps to be taken instead. Programmers familiar with the implementation platform should be able to create the PSM level models using these steps.

We distinguish four areas where technology details should be added. We do so through the following steps.

1. We have to create platform specific mappings between the Sensor classes and the sensor interfaces offered by the underlying platform. The most convenient way of doing so is by using Application Programming Interfaces (APIs) supplied by the platform. If no APIs or other hardware abstraction is offered, we have to manually create bindings to the hardware.

2. Make platform mappings that concern output. As is the case with the mappings of the input devices, the exact code and method calls depend on the implementation platform of choice. Every operating system or platform that offers a graphical user interface to the user allows for the use of graphi-
cal frameworks to render the application interface. We should take care to adhere to UI design standards of the intended platform; offering an application with the _look and feel_ of an iOS app on Windows Phone will cause confusion with the user. As we deem UI development outside the scope of this dissertation, we will not elaborate on this.

3. We make mappings to classes to affect other domain variables indicated as "controllable." These might, however, not all be approachable through programming interfaces. Again, this would require direct hardware bindings to be created.

4. We should construct mappings regarding the storage method. There is a wide selection of technological solutions to store data obtained from sensors, the inferred information, reasoning steps performed, and other data. For example, limited, but fast internal memory can be used for temporary storage, the data residing here being sent to a slower, but more sizable method of storage, such as internal hard disks, flash storage, or external storage providers, if it is deemed no longer of immediate need. The choice for a storage medium or method, such as files in the operating system file system, or relational or NoSQL database, is highly dependent on the performance requirements and system infrastructure.

We can apply this PIM to PSM process to our step counter example. Looking at the class diagram, there is only one sensor in our design, namely the `PhysicalActivitySensor`. We can bind this sensor to the accelerometer API of the implementation platform, like the `Sensors.Accelerometer` on Windows, the `CMMotionManager` on iOS, or the `TYPE_ACCELEROMETER` on Android.

Next, we bind the actuators to the platform. Our step counter only contains one actuator, being the user interface. Current modern operating systems allow programmers to create application user interfaces by specifying the layout of screens using XML. No specific bindings are required, only the layout description. We will not go into detail on the subject of user interface design.

The last addition we do to the PSM model are means to store data obtained through the sensors. This can either be done on the device itself, or (a summarized version of the data) can be stored on an external (cloud) platform such as Microsoft HealthVault [67],
CommonSense [94], or Apple HealthKit [7]. By doing so, the need for storage space on the smartphone is eliminated. Additionally, the data is accessible by other applications. The downside is, however, that a network connection should be used in order to transfer data to the server, affecting battery life negatively.

For our step counter, we opt to store the data on the smartphone device itself in a small SQLite\(^1\) database. We add a class that will perform the database operations to the design. The final class diagram for the application is shown in Figure 24.

### 8.3.3 PSM to code

Up to this point, we focused only on the creation and manipulation of models. We should translate these models, after all information needed to reason about the execution of the system is added, into programming code. This is under the assumption that the execution platform of choice does not operate through the interpretation of PSM level models.

As per the previous steps, we have two types of models. The first, the structural models defined a UML Class diagrams, can, for most object-oriented languages, immediately be translated to code. For example, when moving to the Java platform, a new file is created per diagram class, and attributes and methods are added. As this is common practice in object oriented software design, we will not elaborate on this further.

We should translate the behavioral models created in the previous phase as well. We created Activity diagrams for the methods dealing with the gathering of values or the affecting of them in all

---

\(^1\) [https://www.sqlite.org/](https://www.sqlite.org/)
classes extending Sensor, Concept, and Actuator. We can directly translate these Activity diagrams to code which can be inserted in the code files created from the Class diagrams.

We performed the PSM to application code for the step counter example, translating the UML diagrams into Java code. This code was executable, although we had to use a fake activity sensor: instead of using real hardware, we opted to read previously collected activity sensor information from a file.

8.4 Process Automation

After following the process of mapping elements from one model to the next, a software developer will have obtained a platform specific model of the context-aware application. This process is highly structured which makes it a candidate for (partial) automation. In this section, we discuss which parts of the process can be automated, which can ease the process of software development. We propose to automate the process using technologies offered by the Eclipse project [112].

The first technology we need, is one for representing the models at different levels of abstraction. We selected the Eclipse Modeling Framework (EMF) [35] for this. To start, we described the DWDM modeling language in a metamodel in the Ecore metamodeling language [113]. We showed this metamodel in Figure 2 on page 63. Only a language specification does not allow for the user friendly creation of models, so we constructed an editor for the DWDM language using the Eclipse Graphical Modeling Framework (GMF)
This set of Eclipse plug-ins enabled us to create a graphical editor for the DWDM language based on the DWDM metamodel. We describe this editor in more detail in Appendix A. For the representation and editing of UML models, we integrated the Eclipse UML2Tools plug-in in this editor. This structure is depicted in Figure 25.

Secondly, the automation of the process requires the use of techniques that allow for the manipulation of the models created. For this, general purpose programming languages, such as Java, may be used, or we can use a domain specific language. In this case, the domain is the manipulation of models represented in a language that adheres to a certain metamodel. We have opted for the use of a domain specific language, choosing the Atlas Transformation Language (ATL) [12, 111], as this language has shown to be the fastest performing among current model transformation languages [118]. Additionally, ATL has an active development community of users and is well maintained.

Thirdly, after constructing models at the PSM level, we should translate them into code. As with the model-to-model transformations, we have to make a choice between a general purpose and a domain specific language. We opted for the use of the Epsilon Generation Language (EGL) [114].

Finally, we should combine everything on the technology platform of choice. We have opted for the Java platform for the first development iteration of the system. As we cannot derive all logic from DWDMs, we have created a general framework that allows us to coordinate the collection of sensor data and the control of actuators. This framework follows the static application structure discussed in section 8.3.1. In addition, this framework takes care of the storage of information and provides extensibility for reasoning algorithms. The creation of algorithms that derive common behavior from data and provide feedback based on these is outside the scope of this work. It also offers a textual interface to observe the ongoing processes, allowing developers to inspect the collection of data and the results produced by the reasoning engine. This interface should be replaced by a user-friendly GUI if the application is provided to users.

If we apply these techniques to the development process described in section 8.3, we obtain a partially automated process, such as shown in Figure 26. In this figure, we can identify four model
transformations, all of which we can perform automatically. T1 and T2 map different aspects of the input domain model (represented using the DWDM language) to structural and behavioral models respectively (both represented in UML, the structural model in a Class diagram, the behavioral model in Activity diagrams), following the pattern described in section 8.3.1. The structure of such a model transformation is shown in Figure 27. This structure was inspired by the work performed by Kurtev [58].

We can summarize model transformation T1 using the code shown in Listing 3 on page 224. T2 is shown in Listing 4 on page 226. Note that we have written both these listings in a pseudo code, providing a high level overview of the working of the transformation. The transformations programmed in ATL are written declaratively, working on low level UML2 metamodel elements.
In Figure 26 we have not shown the platform mapping that should be performed to obtain PSM level models, as we discussed in section 8.3.2 and shown in Figure 15. We currently only automatically refine the structural- and behavioral models for a subset of a specific platform to move from a PIM to a PSM level. As a proof of concept, we only implemented the transformation for the ActivitySensor, translating it to a platform mapping to Microsoft Windows. We show this transformation in section A.2.1 on page 227. If we want this model transformation to work fully, we have to create model transformation rules to all specific implementation platforms, binding sensors which are to be recognized by name in the PIM to platform specific sensor and API calls.

Using model-to-text transformation T3, we obtain application code. Transformation T4 generates initialization code from the DWDM: based on the DWDM variables, their measurability, and causal relations between them, this code will perform the steps re-
quired to map the structure of the causal model to an internal system representation which can be used for reasoning.

The implementation of the model transformations described here is reusable for context-aware well-being applications for which the CIM level model is represented in the DWDM language: the transformations operate on the basis of the metamodels of the input and output models, instead of being tailored to specific model instances. The output of the transformations is a skeleton: the underlying reasoning functionality is offered, but no means of interacting with the user are provided.

After we have executed the model transformation steps, we can integrate the generated code. Currently, this has to be done by hand, but is limited to copying files to the appropriate folders in the structure of the framework. The result is an application that can collect and store data, and reason which interventions should be posed, such as increase variable A’s value such that B’s value decreases. The application presents this in a textual interface. We can now extend the code to create a graphical application that offers user friendly messages.

8.5 Discussion

This chapter we briefly mention means of quantifying derivable variables by adding or subtracting the values of variables with positive or negative causal relations to the derivable variable. Although this may be enough to serve as a placeholder calculation, it will in most real-world situations be an approach that is too simplistic. We should therefore look at means of quantification for each situation individually, with domain experts providing the way of calculating these values. If we cannot create such improved calculations, the simplified calculations suggested here can be used to derive a direction of change of the variable value, but the calculated number should be treated as being on an ordinal scale.

Although it may be argued that the model-driven process discussed in this chapter does not further the field of model-driven engineering, as it is a specific implementation of a development process that is discussed in more general terms in the MDA Guide [68], we argue that the use of causal domain models for well-being in such a process is indeed novel. By using such models, it is possible to
8.5 Discussion

capture the domain in a more detailed way than is the case with general purpose modeling languages. The advantages of the DWDM language over general purpose languages have been discussed in chapter 5.

It is currently possible to execute part of the process we discussed automatically: the automatic generation of structural and behavioral models, and of application code from these is possible. We mentioned that our current tool implementation does not support the automatic mapping from PIMs to PSMs, as this would be a time consuming task. Instead of offering such automatic mappings, however, the development tool could supply the developer with a list of PIM methods that should be enhanced with platform information, and aid him/her in the process of selecting the proper platform bindings. Although more extensible, as adding a new platform would only require a new list of bindings, rather than a full new implementation of the model transformation rules, the initial cost of implementing such tooling would be higher. Such an investment would only be sensible if a multitude of target platforms is expected, or rapid changes in these platforms is anticipated.

Another step we did not automate, is the combination of multiple generated pieces of code. However, if the code is annotated properly, which can be done during the generation process, search and replace algorithms could provide for rapid code combination.

The application we generated in this chapter roughly consists of four modules: (i) data collection, (ii) data processing and reasoning, (iii) feedback, and (iv) data storage. It is, however, not necessary that all these modules are located on the same physical device. Historical data could, for example, be stored on a remote location that is accessible through a network connection, or have data be processed by remote servers and only displaying the result on the end-user device. Different physical distribution strategies will have advantages (smaller storage requirements on the device) and disadvantages (longer processing times because data has to be downloaded) which should be taken into consideration.

A current trend in software engineering is the use of agile [39] or rapid development methods such as Scrum, as described by Schwaber and Sutherland [93]. In such methods the result of the process is more important than the development process itself: rather than going through the phases of requirements engineering, development, testing and release in a linear way, small iterative steps are
taken to add functionality to the software under development with the goal to have a potentially releasable product at the end of each iteration. Desired new features are provided by a product owner who is not part of the development team; the product owner should have the wishes and demands of the end-users in mind when requesting new features. An agile way of working can increase the speed at which a product is developed. In this dissertation, however, we focus on the top-down method of software development. We argued earlier that users are unable to specify requirements regarding the product up front and are only able to do so while using it. As users cannot formulate their requirements and feature requests, the product owner cannot do this either. As a result, it might require several iterations of development and testing to formulate feature requests in such a way that they align with user in-situ requirements. We argue that the development of a DWDM in the first process iteration, and basing the software on this model, will result in an increase of development speed, also when used in an agile software development process. While developing software in an agile way, A/B testing is often used to provide two sets of users with two different application iterations to increase development speed even more: the feedback from both groups of users can be used to improve the application in the next development iteration. This too can be done using our modeling language and development method: the developer should create two different models, generating the A/B versions of the application from them.

In the design process we describe in this chapter it is possible to model a domain in which a future context-aware application may operate to aid its users, and move from such a DWDM to an application. Designers and developers of these applications, even when supported by model-transformations, will need to make certain design decisions. The formulation of a rationale for such decisions made during the development phase is currently not explicitly captured in the development method we describe here. This might, however, be required when developing medical applications and the developer has liability regarding the operation of the application. Looije et al. [62] describe a methodology they call situated Design Rationale (sDR). Using this method, designers can formulate objectives, methods used to achieve these, functions that serve the methods and bring about effects, and the instruments to measure effects. These are similar to what we called goals and means
to achieve these goals. What the authors call instruments are observable variables in our DWDM language. In order to improve the DWDM language, annotations could be added to capture the design rationale in a way similar to the sDR method.

8.6 conclusion

The development of context-aware well-being systems is challenging. In this chapter, we provided answers to research question RQ2.3: design a method for developing context-aware well-being systems that improves current methods with regard to the requirements specified in earlier chapters. We have described a model-driven development process. The process uses causal domain models, represented in our DWDM language, as input. The process follows the levels as described in OMG’s Model-Driven Architecture, moving the detail of models from only capturing domain information, to models that describe the system independent of implementation techniques, and models that describe all details of the system implemented using given technology. After this, code can be created from the models. This highly structured approach satisfies our requirement R5 (clear relation between requirements and domain information, and software design) and R2 (clearly structured process), but still follows the regular process steps, allowing for the use of existing process management methods (R8).

We have shown that, because of well defined mappings between modeling levels, it is possible to automate this process through model-to-model and model-to-text transformations. This reduces the chance for software errors (R9), as the human element is removed from the process. As parts of the process require human creativity, not everything can be generated automatically. The transformations, however, do allow for increased productivity. Furthermore, because the process is based on existing tools (EMF, UML, ATL and EGL), existing tools to support the designer can be used (R3 and R4).
“There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.”

– C.A.R. Hoare (computer scientist)

In chapter 8 we discuss how we can use DWDMs to design and develop context-aware well-being systems in a model-driven fashion. The claims made are verified in this chapter. In the experiment we performed for this verification, Computer Science Master students executed the development method discussed earlier in this work. The participants filled out a questionnaire to give their opinions regarding the way of working. In addition, we analyzed the work produced according to a set of metrics, looking at the requirements specified by the participants and the structure of the models produced. We describe the research problem in section 9.1 and the research design in section 9.3. We discuss the results obtained from the experiment in sections 9.4 and 9.5. The method used in this chapter was inspired by Vriezekolk et al. [124].

9.1 RESEARCH PROBLEM

The design and development method described in chapter 8 is based on the assumption that the information regarding the context required to develop context-aware well-being systems can be captured in DWDM models, and if a domain model is created for a specific application domain, that this model can be used throughout the development process. The process uses techniques that have been used throughout other domains in the past, but are yet to be utilized to move from abstract well-being domain models to concrete
applications that are to support the well-being parameters depicted in these models.

Our interests in this experiment are twofold. Firstly, we aim to verify that the use of DWDMs as artifacts driving a model-driven development process is possible in principle. Secondly, if this is indeed possible, is such a process better than a similar process being executed without DWDMs being used.

We answer the following research question:

RQ3.2 [kQ] To which extent does the MDD method satisfy the requirements?

9.2 Research Design

This section describes the research designed for the validation of the model-driven development method for context-aware well-being systems. First, we identify the objects of study, second we discuss the treatment performed, third we detail means of quantifying results, and finally, we elaborate the threats to the validity of the results.

9.2.1 Objects of study

To answer the research problems, we decided to test our development method through a controlled lab experiment, in which students of the Master program in Computer Science at the University of Twente fulfilled the role of system developers. To ensure participants are acquainted with the concepts of (meta) modeling and model transformations in general, and the use of the ATL transformation language specifically, we decided to recruit participants among students who had taken a course in Model-Driven Engineering.

Even though the use of professional software developers to test the method proposed has preference over using students for this purpose, using students does not invalidate our experiment and it does not pose a threat to our external validity. Although results obtained from students can not be mapped directly to industry settings, lessons learned regarding the usage of the development process by students will be similar to the usage by industry professionals, as view and correctness of task execution is similar between both groups [44, 101, 108, 115].
9.2.2 Treatment design

The development of context-aware well-being systems goes through a number of phases, as we discussed in chapter 8. In this validation experiment, we are primarily interested in the model-driven process, such as depicted in Figure 15 on page 127. Because of this focus, subjects will not be asked to design an application specific DWDM, or create application code. The subjects will not be creating a DWDM. During a tryout execution of this experiment, four participants were unable to construct DWDMs; we think this might be due to their focus on implementing software, rather than describing the problem domain first.

During the experiment, we also did not ask participants to derive application code from Class and Activity Diagrams, as this is already common practice in software development. Instead, the focus is on the transformation of the given domain model to structural and behavioral models.

The resulting experiment requires the evaluation of the description of a specific case, the creation of requirements for this system, the analysis of the domain model, and the generation and verification of structural and behavioral software models. This leads to three threats to the validity of the experiment.

1. Subjects might not understand the model-driven method they are about to use or how Dynamic Well-being Domain Models should be interpreted or used.

   To make sure the subjects understood the intention of the experiment and were able to read the DWDM model provided, a short presentation was given to provide an overview of the process and the structure and grammar of the DWDM language. After this, participants were given a similar explanation in writing. Furthermore, participants are encouraged to ask questions regarding the model if at any time during the experiment anything is unclear.

2. Even if the participants understand what they will be doing during the experiment, lack of experience can cause the quality of the created artifacts to vary, and can cause a need for more time to finish the described tasks.
To ensure a basic level of participant experience, the subjects were all chosen from students who had successfully completed a 5 ECT course on Model-Driven Engineering.

3. If tasks, for any reason, take longer than anticipated during the design of the experiment, other tasks cannot be performed by the participant. If key questions are asked regarding these final tasks, they cannot be answered.

Time requirements per task were given so participants could keep track of time. This would not eliminate the risk of them running out of time, but allowed for planning in advance.

We provided the participants with descriptions of the DWDM language and of the case in which they were to apply to development method. We included this material in Appendix D. The text of the assignment consisted of the following:

- A short (5 minutes) presentation about the process that is to be followed and an introduction to the DWDM language.

- A written description of the DWDM language:

  “The notation of the Dynamic Well-being Domain Model (DWDM) is primarily inspired by the Causal loop diagram (CLD) notation. CLDs consist of boxes (modeling variables) and annotated arrows (modeling causalities). A causality may be either positive or negative: a positive causality from A to B (denoted as $A \rightarrow B$) indicates that an increase in the variable value of A (eventually) causes an increase in the value of variable B. A negative causality from A to B (written as $A \rightarrow B$) indicates that an increase in A causes a decrease in B.

  If a variable can be directly read through sensors, we say it is observable. If the application can directly influence the variable’s value, it is controllable. If the application should reason about the variable’s value using the value of other variables, it is derivable.

  Variables can also have a unit of measurement (“km/h” for speed, for example).”
A written description about the case to which the participants will apply the method:

“In this experiment, you will be designing an “Activity Coach” [77]. It is a smartphone application that is to help the user in managing his/her activity throughout the day. This coaching/management can go both ways: if a user is too active during the morning, the application should warn the user that this might leave him/her tired in the afternoon, if the user is not active enough, s/he should be encouraged to be more active.

The application uses sensors (tri-axial accelerometers) to monitor the user throughout the day. These sensors measure acceleration in three directions (x, y, z). For a full day, the desired amount of activity measured in IMA (sum of all the acceleration measured) should be 10,000. The user should be able to see his progress toward this goal at every time, preferably through a graph showing his/her activity and the desired norm.

We now have to identify the other aspects of the app. Firstly, the application should run on a mobile device, so battery power and storage capacity are limited. Because of this, recalculating and reevaluating the user’s goal progress every time a new sensor value is read might not be the best approach, just like saving all data for all time. Secondly, the application might be part of a bigger system where data storage and even the calculations are performed by other devices (cloud storage and computing). The app designed here should be able to cope with such future changes, the adaptations required to allow for these should be limited.”

A list of tasks. The participants were asked to perform the following:

1. Comment on an existing DWDM model, discussing whether the domain model depicted is complete.
2. Elicit a set of requirements based on the domain model and the case description.

3. Generate a UML Class diagrams using the provided model transformations.

4. Verify the UML Class diagram, and alter it if elements are missing or incorrect (based on the judgment of the developer).

5. Match the UML Class diagram and the requirements.

6. Generate a UML Activity diagram using the provided model transformations.

7. Match the UML Activity diagram and the requirements.

We did not ask the subjects to validate the Activity diagrams, as these models could not be presented in a visual format. We could only provide the participants with a textual representation of the UML models, which we deemed unreadable in this format.

- An editor for the DWDM language.

- Model transformations from the DWDM model to the structural and behavioral models.

- A DWDM model capturing the domain described in the case description. This model is shown in Figure 37 on page 251.

- A total of 90 minutes to complete all tasks, after which they were asked to fill out the questionnaire.

As the participants had previous experience with similar editors, we kept the explanation regarding the editor itself short.

Due to the complexity and sensitivity of the well-being domain, we designed the context-aware well-being application development method to use DWDMs as a process input. We therefore provided the participants of the experiment with a DWDM for the case described in the assignment. The risk here, however, is related to the complexity and sensitivity of the domain: if the case and the model did not contain enough complexity or sensitivity, the participants were essentially asked to create a regular application, rendering the use of the method void. Moreover, if the case used was too complex, the
9.2 RESEARCH DESIGN

1. How experienced are you with gathering requirements? (1–5)
2. How experienced are you with modeling application structures? (1–5)
3. How experienced are you with modeling application behavior? (1–5)
4. How experienced are you with model-driven engineering techniques? (1–5)
5. Are there steps in the design process that you would have done differently if you would not have had the supplied tools?
6. Did the Dynamic Well-being Domain Model provide you with information that you could not obtain from the case description, but was needed to land you with your current documentation/design?
7. Would you have liked to have more information? If so, what was unclear?
8. Did you have enough time to consider all of the exercise steps, or was more time needed?
9. Did you have to make alterations to the models that were generated? If so, to which models and what did you change?
10. Did using the DWDM editor and the model transformations ease the development?
11. Do you see room for improvement with regard to the DWDM editor?
12. Do you think that the design process could have been faster if you did not have used the DWDM editor and model transformations?
13. If you would be asked to redo the experiment, would you chose to use the DWDM editor again, or would you use tools of your own choice?

Table 14: Exit questionnaire model-driven process experiment

Allotted time for the experiment would have been insufficient. Both of these are threats to our construct validity.

The case we provided to the participants was a simplified version of an existing context-aware well-being system. As such, we considered it to be suitable both with regard to the complexity of context-awareness, and the sensitivity of well-being. Furthermore, due to the simplification, participants would find it easier to identify problems, than when confronted with the full case description.

9.2.3 Measurement design

As the primary focus of the experiment is the usage of the model-driven method, an exit questionnaire and our observations during the experiment are used for a qualitative assessment. The questions provided to the participants are listed in Table 14.

During the experiment, participants are asked after every process step whether they are satisfied with the models presented/generated, or some of the model’s content is to be changed. We specified
metrics for all tasks to be performed. The metrics are listed in Table 15. Starting the process with the gathering of requirements, the first four metrics deal with the number of requirements gathered, how many of these are defined in an ambiguous way, i.e. they do not capture concrete or measurable information such as “The system should use efficient algorithms”, how many are conflicting, the number that deals with the case description, and how many are “outside the box,” i.e. requirements not taken directly from the case description, but which are relevant to the development.

The metric five concerns structural models. This metric is used to compare the models submitted by the participants with the models that are generated from the original domain model. We measure the average Coupling Between Objects of the class diagram, which gives an indication to which extent the system components are related, and as such provides an indication of the maintainability of the system.

Using the last three metrics, we measure coverage of models. First, we relate the classes in the diagram to the requirements created by the participant: if we cannot find software components that contribute to satisfying a requirement, we deem the model incomplete. Second, the metric for the behavioral models also measures the coverage of the requirements by the Activity Diagrams. Third, we measure the coverage of the Class Diagrams by the Activity Dia-

<table>
<thead>
<tr>
<th>Metric name</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Number of requirements.</td>
<td>Count</td>
</tr>
<tr>
<td>2. Number of ambiguous requirements.</td>
<td>Count</td>
</tr>
<tr>
<td>3. Number of conflicting requirements.</td>
<td>Count</td>
</tr>
<tr>
<td>4. Number of requirements covering case description.</td>
<td>Count</td>
</tr>
<tr>
<td>5. Number of relevant requirements not dealing with case description.</td>
<td>Count</td>
</tr>
<tr>
<td>6. Coupling Between Objects</td>
<td>#relations / #classes</td>
</tr>
<tr>
<td>7. Coverage of requirements by Class Diagram</td>
<td>% of requirements</td>
</tr>
<tr>
<td>8. Coverage of requirements by Activity Diagram</td>
<td>% of requirements</td>
</tr>
<tr>
<td>9. Coverage of Class Diagram by Activity Diagram</td>
<td>% of classes</td>
</tr>
</tbody>
</table>

Table 15: Model metrics for evaluating changes made to models by participants
grams: are behavioral models created for the elements of the Class diagrams that are dynamic in nature, such as methods.

Similarly to the experiment performed in chapter 7, it is possible that the answers provided by the participants were biased, favoring those the participants deemed better for the researcher, posing a threat to our external validity. As the explicit opinions of the participants regarding the way of working was asked, this threat to the validity of the results cannot be mitigated. However, another part of this experiment was to gather objective data about the way of working to determine if it is more efficient with regard to the requirements posed earlier. This objective data, consisting of models and time requirements, is not subject to bias on the part of the participants.

9.3 RESEARCH EXECUTION

During a single experiment session, the research design discussed in the previous section was implemented and performed by a group of subjects. This section discusses this execution process.

9.3.1 Subjects

The students who were to take part in our experiment, were recruited among those who had participated in a course on Model-Driven Engineering six months prior to the experiment. The group of participants was self-selected, rather than randomly. As a motivation, the students were promised a compensation of €20,-. In total, eight students participated in the experiment. Due to the small number of participants, no statistical inference can take place. Any generalization is therefore based on reasoning by analogy.

9.3.2 Case

The case to which the participants applied our development method, was a simplified version of the RRD Activity Coach, as described in Appendix G. The concepts of “Feedback,” “Self-efficacy,” and “Body composition” were left out from this model. The purpose of the application was to motivate people to attain a certain level of activity, rather than focusing on different types of motivational
feedback. The participants were told that the level of activity could be measured through accelerometers, and that the result should be displayed in a graph.

**Environment during application** Participants working on the case were all in a single university classroom. During the experiment they did not communicate, either through speech or otherwise, as to make sure all submissions were indeed individual pieces of work.

### 9.4 Results

The execution of the experiment took participants anywhere between 58 minutes, and 1 hour and 36 minutes, the mean being 1 hour 15 minutes. We obtained three types of results through the experiment:

1. The answers to the exit questionnaire (Table 14), which we can use for a qualitative assessment.
2. An analysis of the artifacts created in the experiment using the defined metrics (Table 15), and the comments made can be used for additional qualitative analysis.
3. Remarks made by the participants regarding the tasks performed.

#### 9.4.1 Exit questionnaire

The first four questions of the exit questionnaire were concerned with the level of experience of participants on four subjects related to design and development. The answers provided are summarized in Table 16.

1. **Experience gathering requirements** With all participants being master students in computer science, we expected that each of them would indicate to have at least some experience regarding this subject, which would translate to a score of 3 out of 5. Two participants scored their experience lower than expected, giving themselves a 2 out of 5 score.
Table 16: Model-driven development experiment: exit questionnaire results, level of experience on a 5 point scale

2. **Experience modeling structures** Since their first year at the university, computer science students have been modeling software systems. As such, scores of 3 out of 5 or higher were expected here. Two of the participants scored themselves 2 out of 5 here, the rest feeling more confident. The participants seemed more confident about experience modeling application structures, than regarding their experience gathering requirements. This might be due to the computer science curriculum focusing more on modeling than on requirements analysis.

3. **Experience modeling behavior** Although modeling behavior is part of the participants’ education, experience shows that students tend to focus less on this aspect of software design. The scoring reflects this, with seven of the participants giving themselves a 3 out of 5.

4. **Experience with model-driven development** As the participants were likely to only have encountered MDD over a single course on the subject, scores here were expected to be low. On average, however, the scores were similar to the experience modeling behavior: two participants scored themselves 2 out of 5, and two scored themselves 4 out of 5. The other four gave scores of 3 out of 5.

Overall, none of the participants thought of himself as having no experience in the fields, one participant thinking of himself as an expert in requirements gathering. This is the only 5 out of 5 score in the questionnaire.
5. Are there steps in the design process that you would have done differently if you would not have had the supplied tools? Standard practice for software engineering entails the gathering of requirements, creation of structural models, creation of behavioral models, after which the code is implemented. During the education of students, this process is repeated to them. Surprisingly, almost half of the participants indicated that they would have started by creating a structural model, after which the code is to be produced; this skips two steps in the process that has been taught to the participants. This choice may be explained due to the case being relatively small: participants may have had the idea that a following a full development process would be unnecessary for the development of the activity coach. Two participants mentioned they would have followed the “traditional” process. Two participants noted they did not know how they would have proceeded, and one participant mentioned he would use the process as used in the experiment.

6. Did the Dynamic Well-being Domain Model provide you with information that you could not obtain from the case description, but was needed to get you to your current documentation/design? All participants responded positively to this question. Especially the ability to visualize causal relations between variables was deemed useful: seeing the way changes in variable values affect other variables in the domain was thought of as having added value over only a case description. One student mentioned that it is not always clear how values of derivable variables should be calculated.

7. Would you have liked to have more information? If so, what was unclear? Although the case description and the provided application specific DWDM appeared to be clear with all participants, most would have liked more technical information regarding the metamodels, models and model transformations. These details were purposely omitted in the experiment, as the inclusion of such an explanation would have required substantial additional reading time.

8. Did you have enough time to consider all of the exercise steps, or was more time needed? All but one participant indi-
cated they had enough time to complete the exercises. The participant claiming not having enough time mentioned he was only allowed one hour, rather than the hour and thirty minutes promised. An analysis of his self-reported start- and end time, however, showed he did use 10 minutes less than other participants, which was caused by him arriving too late.

9. **Did you have to make alterations to the models that were generated? If so, to which models and what did you change?** Half of the participants did not make alterations to the generated models, the others did change something. Of those who did not make changes, one participant did make notes regarding the changes he would have liked to make.

10. **Did using the DWDM editor and the model transformations ease the development?** All participants thought the use of the DWDM editor and the model transformations eased development. As primary reason they noted the availability of the reusable artifacts of the development method, such as the metamodel, the model transformations and the editor.

11. **Do you see room for improvement with regard to the DWDM editor?** As the DWDM editor and model transformations are at an early stage of development, we expected that the participants had feedback regarding the current working of the program. In the version of the tool used in the experiment, the visualization of Activity Diagrams is not yet available; this had been noted beforehand. Participants remarked that the visualization of these models would indeed benefit usability. Furthermore, people did not like the lay-outing of the models. Finally, remarks were made regarding the visualization of the DWDM models, suggesting additional coloring or graphical elements to emphasize the measurability of the variables and their measurement scale.

12. **Do you think that the design process could have been faster if you did not have used the DWDM editor and model transformations?** Although some participants argued that it is hard to say whether the process would have been faster for other cases (even though this was not asked), the general consensus was that our method is likely to be faster than using traditional methods. Some participants noted that using just UML
and Java for the development would be faster in the short run as this case only entails the creation of a small application. It should be noted that the answers given are based on the participants being provided with a ready to use DWDM and model transformations. If these were not given to them, answers may have differed. Also, the answers given may be influenced by what participants thought would be socially acceptable.

13. **If you would be asked to redo the experiment, would you chose to use the DWDM editor again, or would you use tools of your own choice?** Five participants noted they would re-use the DWDM editor and transformations. As with the previous question, these answers may have been given because they were thought to be socially acceptable. Three participants noted they would rather use their own tools, one out of curiosity regarding the difference in development effort, the other two because they were more accustomed to their own work flow.

9.4.2 **Artifacts**

We categorize the artifacts created by the research participants into four groups: DWDM application model, requirements, structural model, and behavioral model. Measures for these were defined in Table 15.

**DWDM application model**

Although a model for the domain of the envisioned application was provided to the participants, they were free to alter this model. Only one participant did so, adding the notions of “Encouragement” and “Feedback.” “Encouragement” can arguably be seen as a type of “Feedback”, however, the addition was interesting to see, indicating that, even without explicit instructions regarding the editor, the editing of the models was not too difficult.

**Requirements**

In the first tasks, participants were asked to create requirements based on the application DWDM. The requirements were specified on all levels of detail, ranging from high level, i.e. “The app should delete stored information as soon as it becomes useless,” to low level,
e.g. “The processing unit of the sensor should have a low-pass filter, so spikes in the measured data of the sensor is averaged out.” On average, we deemed 2–3 requirements to be ambiguous per submission, e.g. “It should cope with errors” or “The system should have server connectivity.” During the computer science curriculum, students are taught to specify requirements in the form of “The system shall . . .”. One of the participants specified the requirements in the form “The user should . . .”.

None of the requirements described by the participants were conflicting. This judgment was made by looking separately at the requirements selected by each participant. It is possible that requirements created by multiple participants were conflicting, but as there was no communication among participants, this was expected.

The “creativity” of the participants varied. On average, 4–5 of the requirements resulted directly from the exercise text, e.g. “Include a graph of activity of the day including goal”, an average of 3 requirements dealing with elements that were not explicitly stated in the case description, e.g. “Use modular setup to be able to swap storage and computing platform”. This varied between participants, with two participants only stating information from the provided case. The difference between participants may be a result of experience or interest in the domain at hand, or with the development of (context-aware) applications in general.

**Structural model**

The participants had to check the structural model generated from the application DWDM and alter it if they thought this was needed. To measure to what extent models were changed, we used the Coupling Between Objects (CBO) metric [25]. This metric is defined as the total number of relations between objects, divided by the total number of object classes. A relation between two classes will be incoming for one, and outgoing for the other class. Because of this, a single link between two classes should be counted as 2. We did not expect the metrics specified by Chidamber and Kemerer [25] to change by alterations made by the participants, so these were not taken into account. The CBO of the class diagram created from the provided DWDM was calculated at 2, so values different than this would indicate changes were made.
Only a single participant altered the class diagram such that the CBO’s value changed, going from 2 to 1.67, indicating a relation had been removed. Another participant added a class, but as this class was connected to existing classes using a single relation, this addition did not alter the initial CBO of 2.

When looking at the percentage of classes contributing to the requirements specified, we find that this number was highly diverse: the lowest percentage of requirements that could be fulfilled was 0%, the highest number 80%.

**Behavioral model**

The final model to be generated during the experiment was a behavioral model. The model generated would, by default, include method descriptions to calculate values for derivable variables. We chose not to have the participants create a user interface, or define any form of user interaction. This caused the metric regarding the coverage of requirements by the activity diagram to be low: the mean coverage was found to be 12%, with a standard deviation of 13%. The highest coverage percentage in the models generated by the participants was 33%.

The analysis of the behavioral diagrams showed that none of the participants had altered or added to the models. High percentages of coverage therefore indicate that the participant apparently had not included a lot of requirements regarding user interaction.

9.4.3 Remarks

In addition to filling out the exit questionnaire and creating artifacts by executing specific tasks, participants commented on the tasks performed.

Not all participants liked the structure of the DWDM, some suggesting specific changes to be made, such as adding causal relations, or changing polarities or measurability properties. The remarks made showed both insight into the domain, but also a technology minded view on the problem: the suggested changes included motivation on how to measure currently denoted “derivable” variables, and process flow-type structures. Rather than seeing the relations as causalities, they were treated as “first this, then that” structures.
When looking at the requirements elicited, we find that many of these deal with user interaction and storage of data. However, the class diagrams that were generated were not edited to facilitate these requirements: to allow for user interaction, we expected that a class UserInterface and classes extending Storage were added to the diagram. Only one participant added a class to the diagram, but this class was not intended for communication of information or other parties, or to facilitate storage of data.

Although not added to the models, some participants made remarks on what changes were intended. Here, user interfaces and means of storage were discussed. Furthermore, the addition of `get` and `set` methods for various classes were suggested.

The suggestions made by the participants will be used when developing the next version of the tool.

**9.5 DISCUSSION**

We tried to provide the participants with a case description that was realistic enough for them to see how the tools and techniques would be used in a real-life development process, yet small enough so the entire case could be covered in the limited time available. All participants but one indicated they had enough time to cover the described tasks; some of them noted they would have liked to have more (technical) information regarding the case and the implementation of the tooling used. The participants also remarked that the case described was quite minimalistic, which may have caused participants to underestimate the complexity of context-aware systems in general.

The tool used during the experiment was provided to the participants as a stand-alone Eclipse product with all the needed plug-ins readily installed. Because of their background, the participants had previous experience with the tooling environment, making it relatively easy for them to get started with the tasks. This might account for the high number of positive remarks regarding the tool.

The model transformations and the tool provided for use in the experiment were part of continuous research and development. As such, their level of maturity is not as high as commercially available software. For example, the DWDM-to-Activity diagram transformation generated models that were not yet compatible with the
generator tool for visualizing Activity diagrams. This resulted in participants having to verify the structure of these diagrams using the default Eclipse EMF model editor, which provides the user a hierarchical tree view of the model. Although the participants had experience with this way of inspecting models, this way of visualizing is not user friendly and can be confusing in case models have many interrelated elements. This may have caused the lack of changes or additions to the behavioral models. In contrast, the Class diagram editor was fully functioning, allowing the users to work with the models in a graphical way. This did not result in changes being made to the structural models. Other remarks made regarding the tool, and the visualization of DWDMs specifically, were expected: the tool used by the participants was generated using the Eclipse GMF without adding graphical details. Improving the layout of diagrams may make them easier to work with; adding redundant modeling information can be beneficial for understanding [13].

Although we attempted to formulate the exit questionnaire questions in a neutral way, not guiding the participants to a certain answer but asking an opinion supported by reasoning, it is possible that some questions may have been written in such a way that they indicated to the participants what answer would be the desired one. Especially those questions with regard to the usefulness of the domain model and the tool, questions 6 and 10 in Table 16, may have suffered from this, the unanimity of participant answers being an indicator of this. This does not, however, invalidate our experiment as the subjects of both questions is also covered by other evaluation questions.

The contents of question 6, inquiring about the usefulness of the DWDM provided, is also covered by questions 5 and 7, i.e. whether the participants would have done things differently without the tools and whether more information would have been useful. Answers provided for these two questions indicate that participants would have preferred detailed technical information and starting the application development as quickly as possible without considering the domain model first. Combining these and the answers given to question 6, we argue that although participants indicated the usefulness of the DWDM model, they would have preferred to start writing an application without considering the domain themselves; in a real-life scenario they would not have stopped to construct the
9.6 CONCLUSIONS

In chapter 8 of this dissertation, we proposed DWDMs as a primary development artifact: we suggested that they could be used to drive the development of context-aware well-being systems when used in an MDE process. We tested this theory in this chapter, answering research question RQ3.2: to which extent does the MDD method satisfy the requirements posed in earlier chapters?

To verify that DWDM driven development is a good way of creating context-aware well-being systems, six single case experiments were executed. Participants of this experiment were computer science master students, modeling professional software developers for our purpose of testing. The participants designed requirements, a static application structure and behavioral models, taking an average of 1 hour and 15 minutes. After this, an exit questionnaire was answered.

Participants remarked that the DWDMs provided them with more insight into the case, even though no more information was modeled than was provided in the case description. Especially the visualization of causal relations was considered useful.

Participants could generate structural and behavioral models from DWDMs using model transformations. They liked this feature, as it eased development. Especially when working on multiple, yet
similar projects, the participants thought that this automation could increase development speed.

Not all aspects of the used tool were liked. As the tool was still under development, some features were missing. Examples of such features included the ability of graphically explore Activity diagrams, and improved user interface elements for the visualization of DWDMs.

During the experiment, comprising both the trial run and the final experiment, we made several observations regarding the development of software using DWDMs and our model transformation driven process:

- During the trial experiment, participants were unable to successfully construct a DWDM. Rather than looking at the problem domain and trying to capture this, participants focused on constructing models similar to UML Class diagrams. The DWDM language, however, is not suited for this way of modeling, as relations between classes in a Class diagram have a different meaning than the relations between variables in a DWDM diagram. To mitigate this problem, participants of the final experiment where provided a ready to use DWDM.

We think that this difference in focus originates from the experience participants have when designing and developing software: students are often taught to develop software based on a given set of requirements, rather than have them gather requirements and look at the problem domain themselves. To prevent such issues from arising when using our development method in production environments, DWDMs should be constructed by people with experience in domain modeling.

- The participants could quickly work with the tool, and knew where they could find the required features, such as the model editors and how to start the transformation. These results, however, are biased, as all of our participants had earlier experience with using similar model editors and transformation tools. To allow developers without prior experience with the Eclipse Modeling Framework and the ATL model transformation language to work with our tool, visual aids can be added to make the process of modeling, transforming and editing more clear. The Eclipse Graphical Modeling Framework, for
9.6 CONCLUSIONS

example, includes a dashboard (Figure 28) that guides the developer through the steps of the development process.

- The tool provided to the participants was still a work in progress. As such, parts of the tool did not work as intended. One of these parts was the editor for UML Activity Diagrams, which was needed to model the dynamic structure of the application. For the tool to be useful when modeling behavior, this must be repaired.

The case provided to the participants was relatively small and had a clear, fixed scope. The choice for the scoping and size of the experiment was chosen to allow for timely completion of the task. The times required by the participants were comparable to each other. Real life situations, however, are never this clearly defined. To get a better understanding of the required time in total and per development step, an experiment should be undertaken where the scoping is less clear.

Looking at the results, we can positively answer our research question, namely that the MDD method satisfies our requirements by allowing software designers to reason over the domain information (R1), the process steps are clearly documented (R2), the software designer is supported by a tool which is similar to tools already known to the designers (R3 and R4), it relates domain information and software design (R5), and the chance for errors is minimized due to automation (R9).
Part IV

BEHAVIOR PREDICTION
“The best way to predict the future is to study the past, or prognosticate.”

– Robert Kiyosaki (businessman)

The DWDM language has been presented as a tool to capture the specifics of the well-being domain in general and the domain of specific systems, and to drive the design and development process of context-aware well-being applications.

As mentioned earlier, the models constructed in the DWDM language are also used to describe how a system could influence the context. In this chapter, we use this quality to predict whether this behavior is in line with the goals posed for the system by its designers. If this is not the case, the system may not be able to reach its goal. Minimizing this chance of software errors is requirement R9 of our method, as we discussed in chapter 3.

This chapter first details the analysis process which will allow us to predict the context behavior (section 10.1), after which details and limitations of this analysis method are discussed (section 10.2).

The following research question is answered in this chapter:

RQ2.4 [DP] Design an analysis method that reduces the chance of incorrect application scoping.

10.1 ANALYSIS PROCESS

The context behavior analysis method consists of twelve questions a designer should answer while developing the application. The application specific DWDM for the system under development should be used as an input for this process. If the developer cannot answer the questions in a satisfactory way, or unexpected answers are encountered, this could indicate that the application might not achieve

Parts of this chapter have been published by Bosems and van Sinderen [16].
the goals set for it, or that problems might arise during execution. The developer should alter the design of the system under development accordingly.

Developers and testers can also use the questions to find problems in existing applications: if an application is not achieving its goals, developers can use our analysis method to find the cause of this, and update the application’s functionality to repair this. Again, those answering the questions should base their answers on the application specific DWDM of the system they are validating.

The questions that should be answered by the application designer or tester are listed here. Developers should answer some of these questions at design time, however, as we created this method to be useful for existing applications too, these questions are asked again here.

1. **What is the goal of the application? Which variables are to be increased/decreased?** Every well-being application has a goal. Such a goal is realized by increasing or decreasing the value of one or more of the domain variables. By this requirement, we restrict the applicability of our analysis method to applications whose goals we can represent by the values of domain variables. By identifying this goal, the underlying application logic can better be explained.

2. **Which variables are to be affected by the application in order to satisfy the goal?** At design time, the means, i.e. the variables that are to be affected in order to attain the goal, are decided upon. Means may either be direct (“increase A to attain goal B”), or indirect (“alter A, such that C changes, which allows us to attain goal B, C being the indirect means”). The direct means are those variables which are directly influenced by the system, indirect means are affected by direct or other indirect means, such that the goal variable’s value changes. It should be possible to create a path from the means to the goal variable(s). Means for which this is not possible do not contribute to the goal.

3. **Which variables will be measured by the application?** In a DWDM, sensors can directly measure the value of observable variables. We must determine whether all the observable vari-
ables in the domain model are indeed monitored, or whether they serve another purpose for the application.

4. **Are goals and means directly related?** If goals and means variables are directly related through a single causal relationship, rather than through indirect relationships, achieving the goal will be easier for the application and potentially undesired side effects can be minimized. The same is true if the application can directly influence the goal, i.e. the goal is a means.

5. **Which variables will be directly affected by changes in the means?** If the application is introduced in the domain, it will affect context variables. By identifying which variables are directly affected by the application’s operation, we can predict potential side effects up front.

6. **Which variables are indirectly affected by the changes made?** As the domain model may contain more variables than those affected by the system, we want to know what other variables are influenced. To do so, we draw a tree by exploring the domain model in a Depth First Search fashion. Loops are not modeled, and if a variable already exists in the tree, it is captured again.

For example, observe the following DWDM:

![Diagram of DWDM]

If A were a means to achieving goal D, the tree of affected variables would be the following:
As we can see, the D variable appears twice in this tree, and no loops exist. In these trees, we are only interested in the causal relations, the variable properties are not modeled.

7. **Are there variables that are not affected?** Some variables might be part of the domain, influencing the context of the user. However, in some systems, not all of these are monitored or controlled. Not using these variables, however, may have effects on the user’s experience of the application.

8. **Are there conflicting effects?** Observing the DWDM provided in question number 6, we see that an increase in A will cause an increase in both B and C, however, this results in different changes for D, depending whether the causal relation B – D or C – D is regarded. Such situations are conflicting. The result of a conflicting situation is unpredictable: as no temporal aspects are modeled, we do not know which change will occur first and which second, or if they occur simultaneously. Depending on the order of effects, one of the causal relations may be prevalent over the other, or effects of either one are negated. In the case of the example mentioned here, the effect of increasing A may either be an increase in D, a decrease in D, or no change at all.

9. **Can conflicting effects be prevented by using other means?** If other variables are affected, but the goal still attained, can conflicts be prevented?

10. **Which loops can be identified?** Loops in DWDMs indicate exponential growth or a stable situation, modeled by reinforcing or balancing loops respectively. Exponential change is rarely desirable.
11. **Are reinforcing loops compensated by balancing loops?** As discussed, a reinforcing loop models exponential growth. Situations like this are both highly unlikely to exist and undesirable in the context of well-being systems. As such, attempts should be made to prevent them from occurring. One way of doing so is by having one or more of the reinforcing loop’s variables consisting in a balancing loop. This balances the reinforcing loop’s effect.

12. **Observations** After following these steps, we formulate an overall observation of the application DWDM, naming observations not captured by the questions.

When all steps have been taken, we can predict certain application behavior and identify potential pitfalls:

1. The application should not have too many goals, as this indicates a lack of application focus.

2. An increased number of means allows the application to achieve its goal through several different ways. This is a positive property.

3. As with the increased number of means, an increased number of measured variables indicates the possibility to chose from the data collected. If data collected through one source proves insufficient or unreliable, the application can use additional sources.

4. Goals and means that are directly related have a close connection. Increasing distance between means and goals may reduce the strength of causal relations between them.

5. By creating an overview of all variables affected by or affecting the system, we can identify which variables are not affected: the system does not influence them, but they are part of the system’s context. This might not be a problem, however, as these unaffected variables do affect the domain and are influenced through means not included in or measured by the system, the choice not to affect them might have negative consequences on the effect of the application.
6. Conflicting causal relations may cause unanticipated results due to the temporal effects of causal relations not being documented. Such resulting behavior may include changes caused by one relation immediately being negated by another, or being negated after a period of time.

10.2 DISCUSSION

The analysis and prediction of a context-aware application’s domain and influence through the use of Dynamic Well-being Domain Models is powerful, yet easy. By approaching the exploration of DWDMs in a top-down fashion, starting from a goal and set of means to achieve those goals, going into more detail, exploring graph structures, an in-depth view of the domain’s workings can be obtained before fully developing, testing, and implementing the application in said domain. However, some limitations have to be taken into account.

Firstly, we can only analyze those elements captured in the DWDM, even though other factors might also influence the domain. For example, the graphical user interface presented by the application, or other means of presenting information to the user and allowing for input from the user, cannot be captured by the domain model. However, the way in which the system interacts with the user is also related to the success of user-centric applications.

Secondly, the application specific DWDMs are based on norms found in the fields of medical and well-being research. However, these norms may not necessarily hold for individual users. For example, providing certain feedback may be aimed at improving a user’s self-efficacy, but because of interpretation by the user may cause the opposite. Such differences in interpretation may be caused by differences in background, education, beliefs, and other personal properties. For this user, the application will fail its goal as the means used are not suitable for the specific user. Run-time tailoring of feedback is then needed in order to realign the application’s interaction with the user.

Finally, developers or designers can perform the current process discussed by hand only if the input DWDM is not too large, and does not contain an abundance of cycles and variables that are affected through multiple paths. When constructing trees from com-
plex domain models, this process can become long and error prone. Automated graph exploration and analysis would be required in such cases.

10.3 CONCLUSION

In this chapter we have proposed a method to predict the effects a context-aware well-being application might have on the user’s context when implemented. This analysis process is based on the structured analysis of an application specific Dynamic Well-being Domain Model. By moving from high-level questions regarding the application’s goal to details such as the relations between domain elements and the way the system is to interact with these to achieve the identified goals, it is possible to identify what pitfalls the system could encounter when introduced into the envisioned context.

We provided an answer to research question RQ2.4 in this chapter: design an analysis method that reduces the chance of incorrect application scoping. By using the structured analysis method described, designers and developers can verify the scoping of the domain model. The application created based on this domain model will inherit these scoping properties. This satisfies our method requirement R9: minimize the chance for software errors.
Most software development processes are iterative, going back to the evaluation phase after the product deployment in order to improve the application. If developers find improvements, often based on user experience, they adapt or extend the application design, and revalidate and reimplement it, as in the engineering cycle proposed by Wieringa [131].

The developers and designers often evaluate software systems based on a comparison between services offered and requirements gathered at design time (adherence), or by gathering feedback from actual users. This is a time consuming task. We proposed to use DWDMs to predict the effects of context-aware well-being applications when introduced in the context in chapter 10. Developers can use this method at design time, reducing the chance for errors at run-time, and thus the need for additional evaluation iterations.

In this chapter, we use three applications to validate this analysis method, i.e. to test if the predictions regarding the behavior are accurate. We start by discussing the research problem (section 11.1) and design for a validation experiment (section 11.2), after which results are presented (section 11.4) and discussed (section 11.5).

11.1 Research Problem

We have already established that the development of context-aware systems is not straight forward: the use of multiple sensors to obtain information regarding different context elements may promise to

Parts of this chapter have been published by Bosems and van Sinderen [16].
provide a better overall picture about the system environment in theory, but will result in increasingly complex system architectures and processing algorithms in practice. If designers chose to use different context elements, this may cause conflicting situations in the reasoning algorithm used by the application. Additionally, the measurement and influencing of certain context variables may seem useful to attain the application’s goal, however, this may not always be the case. As we can see, the ability to predict run-time behavior of the application under development can help prevent future issues.

In this experiment, we are interested in the use of DWDMs to predict application behavior. We will not be able to exactly say what parts of the system should be changed, but we can pinpoint potential problems that can hinder the application’s adoption. We answer the following research question:

**RQ3.3 [KQ]** Which scoping mistakes can be prevented when using the method?

### 11.2 RESEARCH DESIGN

We discuss the planned validation process for the DWDM based application context behavior prediction method, which we described in chapter 10, in this section.

#### 11.2.1 Object of study

To test the analysis method proposed, we will use it to predict the behavior of existing, readily available applications. We selected three applications of which we obtained the documentation regarding the design rationale and end-user studies. These applications and end-user studies were all executed without our involvement. We applied the analysis method to this.

The object of study of this experiment is an instance in which the analysis method is used to predict the run-time behavior of a context-aware well-being application.

#### 11.2.2 Treatment design

We will verify the analysis method for the DWDM based prediction of run-time application behavior by analyzing currently existing ap-
applications that have already been evaluated by users. As the analysis results yield predictions regarding strong points of the system and potential problems, we should compare them against existing user experiences.

We do not intend to improve the design of the existing applications we research using our analysis method. Rather, we focus on the validation of the method, and provide pointers where the applications could have been improved if they had used our method during the initial development process.

First, we selected existing applications, and created DWDMs which we could analyze. For this we used the checklist described in chapter 10 on page 173. We answered all checklist steps, after which we draw a conclusion. These conclusions entail predictions regarding context-aware well-being application behavior and if we can expect problems to occur at run-time. Next, we gathered and analyzed the user studies that have previously been performed for the application, and we summarized the results. Finally, we compared the analysis results and the user studies. As we performed multiple validation experiments simultaneously, we based this experiment on the original reference DWDM model, not taking the suggested changes posed in section 7.4 into account. Big changes to this reference model could therefore pose threats to the construct validity of our experiment.

We used the following in the treatment process:

- Documentation of the selected application which contains requirements and design rationale.

- The analysis method.

- User evaluation documentation.

For the validation of the analysis method, we had to research suitable use cases. A case is suitable if it is of similar size and complexity as other applications in the same field: two applications tracking just physical activity are likely to be similar, however, if one of them is also tracking sleep patterns, its complexity will be higher.

Of the cases we selected, two of the systems focus on improving physical well-being (the mBeats application [119], and the Activity Coach [77]), one aiming at improving mental well-being (the Fishtualization [91]), focusing on both aspects of well-being. Not all
cases have similar complexity: while the mBeats application only operates on a smart-phone device, the Activity Coach is part of a larger tele-treatment infrastructure. To mitigate this issue, we only considered the Activity Coach itself in the analysis, disregarding the back-end processes. For all cases, the information we found in literature and design documentation was confirmed or supplemented by information obtained from informal interviews we conducted and presentations that were given by the application designers.

In this experiment, the same person performed the analysis method and the execution of this validation experiment. As a result, bias may exist regarding the execution of the validation process due to experience and due to a positive expectation by the creator of the method, causing threats to the external validity. To prevent such bias, we described the analysis method as a list of questions that will be answered similarly, regardless of the person performing the analysis; the questions are closed, no free-form answers are possible. We therefore expect the conclusions drawn from these answers to be similar to what other people would have answered.

11.2.3 Measurement design

The focus of the treatment is the accuracy of the analysis method. However, as the artifacts studied are only available in a textual representation, and we analyze the contents, rather than the shape or size of the text, presenting numerical measures for the accuracy is not possible. Instead, we count the number of potential problems or mistakes, i.e. the expected functionality of the application differs from the application goals, identified by the analysis, and compare this with the number of exhibited problems by the evaluated application. The difference between these provide an indication whether the analysis over- or underestimates potential issues.

We list the analysis questions to be answered for each of the cases in Table 17.

11.3 Research execution

We executed the experiment according to the research design. As we are not interested in time efficiency, we did not measure the required time.
11.3 RESEARCH EXECUTION

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Which variables are to be increased/decreased?</td>
</tr>
<tr>
<td>2.</td>
<td>Which variables will the application have an effect on?</td>
</tr>
<tr>
<td>3.</td>
<td>Which variables will be read by the application?</td>
</tr>
<tr>
<td>4.</td>
<td>Are goals and means directly related?</td>
</tr>
<tr>
<td>5.</td>
<td>What other variables will be directly affected by changes in the means?</td>
</tr>
<tr>
<td>6.</td>
<td>What other variables are indirectly affected by the changes made?</td>
</tr>
<tr>
<td>7.</td>
<td>Are there variables that are not affected?</td>
</tr>
<tr>
<td>8.</td>
<td>Are there conflicting effects?</td>
</tr>
<tr>
<td>9.</td>
<td>Can conflicting effects be prevented by using other means?</td>
</tr>
<tr>
<td>10.</td>
<td>Which cycles can be identified?</td>
</tr>
<tr>
<td>11.</td>
<td>Are reinforcing cycles compensated by balancing cycles?</td>
</tr>
</tbody>
</table>

Table 17: DWDM analysis questions to predict application behavior

11.3.1 Cases

We used three cases for the execution of this experiment:

**TNO/SWELL Fishualization** The TNO/SWELL Fishualization, as described by Schavemaker et al. [91], offers a way of providing visual feedback to a group of employees regarding their working habits. The tool consists of a small application that is installed on the user’s desktop computer. It tracks keyboard and mouse activity, the applications opened (Microsoft Word, PowerPoint, Outlook, et cetera) and how often switches are made between these applications. The system aggregates the data from all the installed client applications and visualizes it on a monitor at a central location, such as a coffee room. This allows colleagues to compare working style. The idea behind the system is to improve both personal insight in working habits, and to initiate conversations among colleagues about experienced work stress. We provide more information on the Fishualization in Appendix E.

**Philips/SWELL mBeats** Rather than focusing on the mental well-being of the user, the Philips/SWELL mBeats application, as discussed by van Dantzig et al. [119], wants to improve a user’s cardiovascular fitness. The system consists of two parts: a smartphone application and a heart rate sensor that the user should wear on his/her wrist at all times. The application can
display a target heart beat for the user to achieve in order to improve his/her fitness. If the user’s heart rate is within this range, the system rewards him/her points ("mBeats"). We describe more details about the mBeats application in Appendix F.

**RRD Activity Coach** Like the mBeats application, the Activity Coach aims to improve the user’s physical well-being, as op den Akker et al. [77] describe. In contrast, the Activity Coach is targeted at patients suffering from Chronic Obstructive Pulmonary Disease (COPD), which eventually causes a decrease in physical fitness due to a decrease in physical activity. The Activity Coach guides its user to spread out physical activity over the course of the day, teaching them not to be too active in the morning, as this would result in a lack of energy in the afternoon, but overall being increasingly active to improve endurance. To measure the user’s activity, the Activity Coach uses a tri-axial accelerometer, the sensor data of which is used by a smartphone application. Unlike the other two applications, the Activity Coach was designed to be part of a tele-treatment intervention, being provided and explained to the user by a health care professional [77]. Appendix G provides more information on the Activity Coach.

### 11.4 RESULTS

We first analyzed the selected applications using the method proposed in chapter 10. We started this analysis by creating a DWDM from the available design documentation. We carried out the analysis based on this DWDM and documentation. We compared the results with user evaluation studies.

#### 11.4.1 TNO/SWELL Fishualization

**Application analysis**

1. **Which variables are to be increased/decreased?** The application should increase the variable “Mental well-being.”
2. Which variables will the application have an effect on? The application will increase “Feedback” and “Department Feedback,” decreasing “Experienced stress” in the process.

3. Which variables will be read by the application? The system uses “Computer activity” as input.

4. Are goals and means directly related? No.

5. What other variables will be directly affected by changes in the means? By influencing “Feedback,” “Awareness” is directly affected. Changes in “Department feedback” influence “Social interaction” directly.

6. What other variables are indirectly affected by the changes made? We provide an analysis of these variables in Figure 29.

7. Are there variables that are not affected? No.

8. Are there conflicting effects? No.

9. Can conflicting effects be prevented by using other means? Not applicable.

10. Which cycles can be identified? The cycle “Mental well-being” – “Feeling of well-being.”


Observations

- The application does not deal with causes of stress.

- No attempt is made by the application to increase the user’s way of coping with stress, or with stress recovery. The tool will give insights, but nothing is done about the actual problem.

- The system uses two separate ways of improving the user’s well-being, increasing the likelihood of success.

- Potentials for application failure: (i) the provided feedback does not increase the locus of control and so does not help to reduce perceived stressors, (ii) department feedback does not lead to social interaction, (iii) the increase of feedback and awareness only increase experienced stress, e.g. the user gets confirmation that s/he indeed is busy.
Figure 29: Variables influenced by the TNO/SWELL Fishualization when introduced into the context

Analysis results
The Fishualization aims at decreasing the user’s experienced stress as to improve their mental well-being. In addition, it should improve the social interaction among coworkers by encouraging conversations about stress and how to prevent it. The current design of the application provides users with feedback about their work, providing a monitor regarding their computer behavior. The application recognizes stress caused by computer work, means of measuring this are shown in the domain model. However, the application currently does not aim to affect these stressors. The application design is based on the assumption that providing feedback regarding the current working conditions is enough to reduce the user’s ex-
11.4 RESULTS

experienced stress and result in conversation among colleagues. The application does not provide feedback on how to reduce the experienced stressors, and how to cope with stress, i.e. how to recover while at work or during free time. Because of this, the application might be interesting to the user at first due to the novelty, however, in the long run it might not be enough to keep the user engaged. Additionally, the application does not value the amount of work/computer input performed. As such, it only informs the user what the objective values of the perceived stressors are, not whether this is deemed high, low or average.

User study

The target of the Fishualization, was to increase both the user’s and their colleague’s insight in their well-being and levels of working pressure. However, a user study [57] among participants of a Dutch high-tech company showed that the social interaction that was aimed for, was lacking. Users did report an increase in their own insight of their energy levels, but the fish tank did not result in additional conversation and discussion regarding stress and stress prevention among colleagues. 14 employees of an independent Dutch research company indicated the same. These employees especially reported improved insights into personal working patterns and well-being at work. Group awareness and social interaction with colleagues, however, remained at the same level.

Analysis comparison

When we analyzed the Fishualization, we found that the application uses a dual mode of influencing the user: firstly, the application provides feedback to the user as an individual, secondly, the group of users is given overall feedback. The chance for the application to succeed in its goal was estimated to increase because of this, with the remark that department feedback might not lead to an increase in social interaction.

Comparing our predictions to the user experience reports, we find that users indeed did not engage in more talk about work stress, as we predicted through our analysis of the application specific DWDM. However, due to the duality of feedback, users did receive their own feedback on their work, which did result in a feeling of
better insight into their energy levels. We also predicted this increased likelihood of success through our analysis method.

Additionally, our analysis method showed that providing users with feedback regarding stressors, but not providing means of reducing these stressors, might actually increase the stress a user experiences. If the designers of the Fishualization had used our analysis method during the design of the application, they could have added feedback and means to prevent this.

We can conclude from this case analysis, that our method overestimates problems, rather than underestimating them, which can help designers to be more cautious when developing systems, i.e. keep problems in mind that might not occur.

11.4.2 Philips/SWELL mBeats

Application analysis

1. **Which variables are to be increased/decreased?** The goal of the application is to increase “Physical well-being.”

2. **Which variables will the application have an effect on?** The application will directly affect “Feedback.” This is done to increase “Active Heart Rate.”

3. **Which variables will be read by the application?** The application will measure the variable “Heart rate.”

4. **Are goals and means directly related?** No.

5. **What other variables will be directly affected by changes in the means?** By affecting “Feedback,” “Awareness” is changed.

6. **What other variables are indirectly affected by the changes made?** We show an analysis of these variables in Figure 30.

7. **Are there variables that are not affected?** The application does not affect the variable “Self-efficacy.”

8. **Are there conflicting effects?** Yes, an increase in “Physical activity” causes an increase in “Active heart rate” and a decrease in “Resting heart rate.” These both cause an increase in “Heart rate.”
9. **Can conflicting effects be prevented by using other means?**

No. We do note that the conflicting nature of this relationship is only caused by both types of heart rates being concepts, rather than measurable values: a “Heart rate” can be measured only by observing additional context elements can we determine whether the user was active or resting.

cal activity” – “Cardiovascular fitness” – “Physical well-being” – “Feeling of well-being.”

11. Are reinforcing cycles compensated by balancing cycles? Yes. Cycle (i) is a reinforcing cycle that is balanced by both (ii) and (iii).

Observations

- The application does not affect the user’s self-efficacy.
- We can only say something about the effectiveness of the application by measuring the heart rate.
- The application does not measure activity, so no guarantee can be given regarding the state of the heart rate, i.e. whether the user is active or resting.
- The application is to improve the feeling of well-being through both improvements in body composition and cardiovascular fitness, improving the chance of success.
- Potentials for application failure: (i) feedback is not received by the user or does not align with user ideas, (ii) the user does not recognize a problem (caused by previous point), (iii) the user’s heart rate measurement is inaccurate (user is active while measuring resting heart rate or vice versa).

Analysis results

The mBeats application aims to increase the user’s physical well-being. It will attempt to provide the user with feedback, as to make him/her be active more, increasing their heart rate. We expect both body composition and cardiovascular fitness to improve, which both have a positive effect on the user’s overall feeling of well-being. However, the application is only intended to provide the user feedback about the current and past levels of their heart rate; no feedback is given regarding the way this heart rate is achieved, i.e. the application does not influence the user’s self-efficacy. Because of this, users might not know whether they have performed the activity properly, which could result in a decline in motivation. Additionally, as the application does not measure the user’s activity (the focus of the application is solely on heart rate), the user might wonder how the cumulative heart rate was calculated, i.e. which activities were undertaken to achieve said heart rate.
User study

van Dantzig et al. [119] documents the requirements, design process, and user study and evaluation of the Philips/SWELL mBeats application. The authors primarily wanted to see the effect of providing the user with historical feedback, or solely providing them with their current heart rate; however, we can still draw conclusions about the overall working of the application from the user evaluations. Although participants mentioned they liked the idea of the application, some remarks were made.

Firstly, the users noted that only providing feedback regarding current heart rate was insufficient to keep them motivated; they would have liked more feedback on how to achieve the heart rate required for cardiovascular fitness improvement.

Secondly, users did not know which activities contributed to getting their heart rate to the right frequency, as the application did not perform activity tracking or recognition. Most users would have liked to be provided with an overall view of the undertaken activity throughout the day.

Thirdly, as the application did not aim at affecting the user’s self-efficacy through encouraging feedback, users lost confidence in their ability to meet the applications target throughout the test period. Users mentioned they would have liked motivational messages from the application.

Fourthly, due to connectivity issues between the measuring device (Mio Alpha heart rate sensor) and the storage device (smartphone), data was frequently lost. This annoyed the users.

Finally, half of the participants would have liked more feedback regarding their activity, such as “speed, distance and calories burned”, suggesting GPS tracking for this purpose.

Analysis comparison

When we analyzed the mBeats application, the primary problem we found was that the application did not affect self-efficacy. Looking at the user evaluation, we see that this was indeed the main reason for people not liking mBeats. Users wanted to be provided with feedback and encouraging messages regarding their activity, but did not receive these. If the designers had used our analysis method at design time, this issue could have been avoided.
Another potential problem we found in the application design was the lack of activity recognition and tracking. Users indicated this would be useful as it would provide additional insight into the way the application measured their activity. Designers could have prevented this problem if they had used our analysis method.

Our analysis did not anticipate that users would have certain specific requirements regarding user interface elements and proper connectivity between sensor and storage device. This is a weak point in our analysis method, which we can attribute to the fact that such non-functional requirements cannot be deduced from DWDMs. This also holds for user complaints regarding connectivity issues.

11.4.3 RRD Activity Coach

Application analysis

1. **Which variables are to be increased/decreased?** The goal of the Activity Coach is to increase “Physical well-being.”

2. **Which variables will the application have an effect on?** The application will increase “Feedback” in order to increase the user’s “Physical activity.”

3. **Which variables will be read by the application?** The application measures the variable “Physical activity.”

4. **Are goals and means directly related?** No.

5. **What other variables will be directly affected by changes in the means?** The application can affect “Awareness” and “Self-efficacy” through the “Feedback” variable.

6. **What other variables are indirectly affected by the changes made?** We show this analysis in Figure 31.

7. **Are there variables that are not affected?** No.

8. **Are there conflicting effects?** No.

9. **Can conflicting effects be prevented by using other means?** Not applicable.

10. **Which cycles can be identified?** We find three cycles, namely (i) “Feeling of well-being” – “Physical well-being,” (ii)
11.4 RESULTS

Figure 31: Variables influenced by the RRD Activity Coach when introduced into the context


11. Are reinforcing cycles compensated by balancing cycles? Yes, reinforcing cycle (i) is compensated by balancing cycles (ii) and (iii).

Observations

- No redundant variables exist in the domain model.
- We found a total of four ways through which the feeling of well-being is improved.

- Potentials for application failure: (i) feedback is not received by the user or does not align with user ideas, (ii) user does not recognize a problem (caused by previous point), (iii) the user cheats with the device (putting it down when s/he should not be active, shaking it to simulate activity).

**Analysis results**

The goal of the Activity Coach is similar to that of the mBeats application, namely to increase the user’s physical activity, in order to improve the user’s physical well-being. The means through which this is to be achieved, is by looking at the user’s activity (rather than heart beat), and by providing feedback regarding historical activity and improving the user’s self-efficacy. Due to this duality of feedback, the chance of the application succeeding to influence the user is increased. When we look at the variables that are influenced by the application, we find that there are no variables are unaffected. Chances of application success and longterm adherence are therefore high.

**Analysis comparison**

When analyzing the Activity Coach, we did not anticipate problems with the application, other than potential usability issues. Comparing our analysis with the various user studies performed, this prediction was confirmed. Users rated the application as highly usable, the feedback resulting in an increase of physical activity over the trial period.

Our analysis method predicted potential problems if a user would either not understand the feedback, or would decide to “cheat” the device. We did not find evidence of this occurring in the user studies. However, the usage of the Activity Coach did differ from the way the other applications were used: rather than being used by people who wanted to improve their own feeling of well-being or get insight, the Coach was tested for tele-treatment of COPD patients. Because of this, the problem was likely prevented or mitigated outside the application’s control.
11.4.4 Measurements

For each of the applications we identified a number of problems. When we compare these analysis results to the actual user studies, we obtain the following results.

We identified three potential problems for the Fishualization. Of these three, one was confirmed to be a problem ("department feedback not leading to social interaction"), one other ("providing feedback not increasing the locus of control and so not helping to reduce perceived stressors") was related: the users did report an increase of insight in their own energy levels, but no relation to perceived stressors was noted. No unanticipated problems were reported.

Three potential problems were also noted for the mBeats application. Of these, one problem was confirmed by the users ("feedback is not received by the user or does not align with user ideas"), whereas the other two ("user does not recognize a problem" and "user’s heart rate measurement is inaccurate") were present in a different form. The user not recognizing the problem was caused by a misunderstanding of the user interface: no specific feedback was provided, so the users thought they were not active enough (their heart rate did not increase past the set threshold). Inaccurate heart rate measurements were present in the form of communication errors between the smartphone and the sensor; rather than inaccurate sensor data, no sensor data was received for some periods of time. Users made two remarks regarding the user interface that were not anticipated.

The analysis of the Activity Coach yielded three potential problems for the application. None of these were identified as problems ("feedback is not received by the user or does not align with user ideas," "user does not recognize a problem", and "the user cheats with the device") during the user evaluations. No unanticipated problems were found.

11.5 Discussion

Using our method we were able to correctly identify potential problems in the applications studied, and over-estimated slightly; no issues were left unidentified. If our method would have been used while developing the applications discussed, special attention could
have been paid to the found issues, resulting in a better experience for the user.

Although we identified the behavior of the applications correctly, the analysis method does have certain limitations. These limitations are primarily caused by the input models used.

Firstly, it is not possible to predict the effect of different user interaction and feedback strategies. The method can identify which variables the feedback should concern, but the application developer should still define and evaluate the specific feedback messages and method.

Secondly, we cannot capture properties regarding quality in DWDMs and as such cannot reason about them using our analysis method. Reliability, efficiency and usability are all highly important to users of well-being applications and services: if these are not satisfied by the application to the levels desired by the user, s/he will stop using the application. We have seen reliability issues occurring in the Philips/SWELL mBeats user evaluation, causing issues in the user experience. Additionally, the user interface provided by the application should align with what the user is expecting, providing the user with the information s/he needs to perform the intervention suggested by the system.

Thirdly, as we do not capture temporal information in the DWDMs, we cannot make predictions regarding the timing of effects. Being able to do so would be beneficial when dealing with contradicting causal relations.

Finally, we cannot present security and privacy aspects of the application under development in DWDMs, so we will have to reason about these through other methods. As the domain considered consists of highly personal and private data, such requirements are important when aiming for large scale user adoption: users have to trust the application and related services, or they will not use them.

11.6 CONCLUSIONS

We started this experiment by analyzing design documentation for three applications that aim to improve well-being. Based on these documents and our reference well-being domain model, we created application specific domain models, which were subsequently analyzed using the analysis method we proposed in chapter 10. Based
on this analysis, we came to several conclusions per application, indicating potential problems. After this analysis, we consulted user study documentations in order to analyze whether anticipated issues arose as applications were used in practice.

Taking the limitations of DWDMs into account, we were able to create an analysis that accurately indicated potential problems and success factors in the design of the selected applications. The Fishualization application does manage to influence the user’s mental well-being, but only doing so because of its duality of feedback. The mBeats application did not provide self-efficacy feedback, resulting in negative user feedback. Additionally, users had a desire for activity feedback, which was also anticipated in our analysis. The Activity Coach application, being a more mature product than the other two applications, did not suffer problems, users indeed increasing their cardiovascular fitness and physical well-being.

When we count the number of issues identified through our analysis method, and the number of problems reported by users, we find that in all three cases our analysis either correctly identified the potential problems, or overestimated them: users did not experience one of the Fishualization issues, all mBeats problems were mentioned (all user interface problems together are counted as one here), and none of the Activity Coach issues were confirmed. Such overestimation can, when the analysis method is used in the development of applications, lead to more careful considerations of design choices.

While we performed the analysis, we found it was possible to find problems regarding functional user requirements, but not possible to identify lacking non-functional requirements. We can trace this issue back to the limitations of the DWDM language, and as such the models described in it: we can deduce (desired) functionality from these models, which is not possible for non-functional elements such as user interface design or methods of feedback. This does lead to a remark regarding our analysis method: if we find no problems when analyzing the application, this does not mean no problems exist. If the analysis does yield problems, however, they can certainly hinder the success of the application. Designers using the analysis method should therefore still check which non-functional requirements exist using other design methods.

In this chapter we answered research question RQ3.3: which scoping mistakes can be prevented when using the method. The mista-
kes we found regarded a lack of user feedback. Either the user was not engaged enough by the application (Fishualization), or the application did not influence self-efficacy properly (mBeats). We did find other reasons for potential application failure, but we could not confirm these based on the user studies we found. The majority of the found potentials for failure are about feedback, rather than incorrectly or redundantly measuring unrelated variables. We can conclude that if designers of context-aware well-being systems incorporate our analysis method in their development process, they can identify potential problems at design time, rather than when testing a prototype of the application.
Part V

CONTRIBUTIONS AND CONCLUSION
CONCLUSIONS AND FUTURE WORK

We started this dissertation by posing three overall research questions, which can be summarized as: how are context-aware well-being systems currently designed, how can this development process be improved, and are these improvements valid?

After introducing the Dynamic Well-being Domain Model language, we proposed that this language is suitable to capture the well-being domain, described a reference model of well-being, used DWDM models in a development process of a context-aware well-being system which was supported by a partially automated model-driven techniques, and predicted the behavior of context-aware well-being systems by analyzing their DWDMs. We tested the validity of these through a series of experiments.

In this last chapter of this dissertation we use the research results to answer the research questions we asked in the first chapter (section 12.1), suggest directions for future work (section 12.2), and provide overall concluding remarks (section 12.2).

12.1 ANSWERS TO RESEARCH QUESTIONS

Throughout this dissertation, we have been working toward answers for research questions which we asked in the first chapter. We provide these answers in this section. We start with the sub-questions, after which we answer the main research questions.

**RQ1.1 How are context-aware systems and well-being defined?**

Context-aware systems are systems that can obtain information about their real world and virtual surroundings through hard and software sensors, and can adopt their behavior to better support the user based on this information. Well-being is defined as “the state of being happy, healthy, or prosperous.” We regarded three different types of well-being: (i) physical, (ii) mental, and (iii) social. We answered this question in chapter 2.
RQ1.2 Which requirements engineering approaches have been proposed for context-aware systems?

In chapter 4, we looked at 11 methods for requirements engineering. Of these, only three authors each satisfied two of the nine requirements for a development method for context-aware well-being systems, as we specified in chapter 3. These authors were Seyff et al. [96, 97], Desmet et al. [29] and Jorgensen and Bossen [50]. Sawyer et al. [90] satisfied only one requirement.

RQ1.3 What challenges are specific for the design of context-aware systems and how are they addressed by current methods?

We looked into related work regarding the design and development of context-aware well-being systems in chapter 4. The primary challenge we encountered when working with these systems is how to deal with an ever changing context. All elements that together make up the setting in which a person can be, i.e. where s/he is, who s/he is with, the time, physical and physiological parameters et cetera, will change over the course of the operation of the system. Designers have to find ways of providing the user with feedback and the content of these feedback messages should be adapted to suit the context the user is in.

Current research efforts aim to adapt the traditional method of software engineering to the specific demands and challenges of context-aware system design: authors propose in-situ requirements engineering to be used, they suggest modeling languages to better capture the technical challenges of the system, and implement intelligent run-time adaptation of behavior and requirements. The focus, however, remains primarily on solving technical issues, rather than aiming to provide the user with the best possible experience, regardless of the encountered context. Those solutions that do aim to be user-centric do so by being goal, rather than requirement, oriented, allowing the application to adapt at run-time to continue satisfying the user’s goal.
RQ1 How are context-aware well-being systems currently designed and developed?

Context-aware systems are currently primarily developed in a technology centric way. When gathering requirements for these systems, authors mainly propose iterative approaches, or gather requirements in-situ. Most authors, however, assume the future context of operation is well known. Design and development methods for context-aware systems, both model-driven and "regular" are equally technology centric, aiming to improve the amount of context information gathered, or the quality of this context information. Additionally, authors propose methods to improve the degree to which a software system can react to changes in the environment.

12.1.1 RQ2.1 Define and motivate the requirements for developing context-aware well-being systems.

In chapter 3 we performed a stakeholder analysis for development methods for context-aware applications. Although these stakeholders are similar to those of regular applications, the added challenges of context-awareness does result in stricter requirements regarding the relation between requirements and domain information, and the design of the application. We elicited a total of nine requirements to which a development method should adhere.

12.1.2 RQ2.2 Which MDD methods for developing context-aware systems currently exist?

Also answered in chapter 4, we found that the current model-driven methods for context-aware application development are heavily based in technology. Rather than taking a user centric approach to development, taking what is needed by the user as a starting point, current methods primarily focus on new ways of capturing software and hardware components, and how the context can be measured using these.
RQ2.3 Design a method for developing context-aware well-being systems that improves current methods with respect to the requirements.

The basis of our solution direction was a domain modeling language that took the specifics of the well-being domain into account. We described it in chapter 5. This modeling language, called the Dynamic Well-being Domain Model (DWDM) language, was inspired by the Causal Loop Diagram (CLD) language, and allows stakeholders, domain experts, and designers to discuss, model, and reason about the domain and future context of an application under development. Designers can use DWDMs to capture variables, i.e. dynamic properties of the context, and causal relations, i.e. how changes in one variable affect other variables. We gave variables properties to indicate their measurability, their dimension, and their measurement scale. In addition, we included norm values that could be assigned to variables.

Using the DWDM language we created a reference model of the well-being domain through the help of experts from commit/swell. Our aim when creating this model was to relate both the mental and the physical aspects of the well-being domain. We constructed the model to serve as a starting point for the creation of domain models specific for applications under development. We discussed the creation process in chapter 6. We described two ways of creating an application specific domain model: from scratch, or based on a reference model. When using this second method, designers can take a user-centric approach, looking at what is needed by the user and basing the requirements on hardware on this, or a technology centric approach, where the designer takes the available technology platform as a starting point.

To streamline context-aware well-being application development, we designed a model-driven method with DWDMs being positioned as the input models. Through semi-automated steps, we created and generated structural and behavioral models, these models becoming more detailed throughout the process. The result of this process is an application that can support a specific part of the well-being domain. The process description can be found in Chapters 8. This process follows the Model-Driven Architecture steps of moving from a Computational Independent Model (CIM), to Platform Independent Model (PIM), to a Platform Specific Model (PSM). The PSM can then be used for code generation. We create our CIM level
models in the DWDM language, building the PIM and PSM level models in the UML. We use the Class diagram notation for the static application structure, and Activity diagrams to capture the dynamic application behavior.

**RQ2.4** Design an analysis method that reduces the chance of incorrect application scoping.

Predictability of well-being systems is important, but may be challenging if the system is context-aware. We can anticipate the behavior of the application, regardless of context by analyzing a DWDM created for a specific application, and comparing this analysis result with the intended, documented goals and means. We discussed this method in chapter 10.

**RQ2** Improve the design and development of context-aware well-being systems.

Based on the overview of current methods obtained, we proposed three directions for the improvement of the design and development process of context-aware well-being systems: (i) a reference model of the well-being domain, (ii) a model-driven process, and (iii) an analysis and behavior prediction method. Together, these improvements were to adhere to requirements we specified in chapter 3.

We created the reference model in a new modeling language we called the Dynamic Well-being Domain Model (DWDM) language. We used application specific derivations of this model in a model-driven process. Through the model-driven process, we refined the input model in two steps, moving from Computational Independent Models to Platform Independent Models, and finally to Platform Specific Models and code. Additionally, we created an analysis method for DWDMs to find possible shortcomings of applications based on them.

**RQ3.1** Do domain experts think the modeling language is suitable to capture the domain of well-being?

By interviewing a group of five experts in the domains of physical and mental well-being, we validated/evaluated the modeling lan-
guage and the reference domain model. In addition and with the same goal, we provided a group of experts with a questionnaire. Of this group, seven people responded. The questions we asked during the interviews and in the questionnaire encompassed the total of the constructed domain model, inquiring whether certain relations seemed valid to the expert. We summarized and compared the results of the interviewed experts and the questionnaire participants. We discussed these in detail in chapter 7.

In the evaluation, we found that overall the data represented in the model seemed valid. We proposed some alterations, consisting of the addition, removal or alteration of certain variables or causal relations. Similarly, the DWDM language was capable of modeling all required relations, but again we suggested some changes.

During the model evaluation, we put emphasis on verifying the correctness of information captured in the domain model. We did not check the modeling language directly, and we made no attempts to verify whether the domain model was complete. Information regarding completeness, however, is already known: as the focus during the creation of the model was on knowledge used in the commmit/swell project, the model can not be complete for all situations. When working on a specific application it is, however, possible to replace existing variables with additional, more detailed other variables if the specific domain calls for more detail in certain parts of the model.

Experts, during interviews, stressed numerous times that personalization is highly important is the application to become successful. They stated for several relations we included in the model that they may uphold in general, but might not work for individuals. We think that this is an important aspect to note: no matter how good the reference model may represent the knowledge in the domain, there will always be exceptions to established rules. This holds especially true for those elements focusing on the mental aspects of well-being. While physical aspects of the human body tend to adhere to fixed physiological principles, although even here exceptions exist, this is rarely the case for mental functions. As such, general statements and information known to be correct in general can be included in a reference or application well-being model, but there should always be room for additional interpretation, reasoning, and adaptation by the application.
In the discussion of the completeness of the modeling language DWDM in section 7.4.3, we noted that some improvements should be done to enhance the expressiveness of the language. One of these improvements would entail adding temporal information to the causal graph, indicating the average time for an effect to take place, when the effect is at its maximum strength et cetera. The main issue with modeling this information is that it is often highly personal or even unmeasurable. For physical or physiological interventions it is possible to measure or calculate this, but that may not be the case for interventions that are to change mental processes. For example, how would the time-to-maximum-effect of a feedback message be measured? If such values can be calculated, they will be different from person to person. They may even be different for the same person at different moments in time. So even though the addition of temporal aspects can be an improvement to the language, the use of them in practice will not be trivial.

RQ3.2 To which extent does our MDD method satisfy the requirements?

We provided the answers to this question in chapter 9.

We did not automate all of the steps of the described development process for context-aware well-being applications through tool support. The creation of the application specific DWDMs relied on the knowledge and expertise of domain experts. This was also the case for the creation of bindings to a specific implementation platform.

To test the process and the tools created to support the application development method, we provided them to eight subjects who had, as self reported, some experience with model-driven development. Although in the end the majority remarked they liked the tool and would use it again for similar experiments, the participants did pose remarks for improvement. Among these was the desire for more (technical) information, such as details regarding the construction of the input DWDM, the metamodels, and the model transformations. Furthermore, the participants suggested improvements regarding the tool and its interface. Finally, they mentioned that using “traditional” tooling or just writing application code would have likely been faster than using our DWDM tool in this specific case, using our tooling would probably be easier and faster when dealing with larger, more complex systems.
Although the alterations to the model editor proposed can be made with little development effort, larger changes are required to make the tool and model transformations ready for larger scale testing. These changes are primarily related to the way the model transformations currently function. Firstly, PSM level models are not yet generated through automated transformations. It is possible to add support for the creation/generation of such models by showing the developer classes taken from the PIM level model and a list of platform elements. A developer could add mappings manually this way: as the PIM classes have no fixed names, automatic matching will not be possible. Secondly, the code that is generated from PSM models has to be integrated by a developer: behavioral models and structural models provide for separate parts of the overall application. This integration can be done through model transformations, the transformations being defined as N-to-1 transformations (N input models, one output model). Thirdly, the code generated is not executable on its own. For this, we described a framework on page 144. For the build process of the application to be more efficient, the framework code should be added automatically.

3.3 Which scoping mistakes can be prevented when using the method?

For our validation, we used three applications for which user studies had already been performed. We analyzed the applications and compared the shortcomings identified by this analysis with the end-user evaluations. Of the three cases studied, the analysis method over-estimated the number of potential problems in all cases: for two of these the overestimation was one, for the other it was three. In none of the cases we found any unanticipated problems, i.e. not identified by the analysis method but present in the user evaluation results.

The evaluation showed a tendency of the analysis method to overestimate the number of potential problems. At the point in the development process where developers perform this analysis, however, we think this is not a problem: it can be seen as being too careful, rather than not careful enough (underestimating issues). As we argued that predictability is important, we think that having designers re-evaluate the design and the rationale behind it is a better
option in the long run than adding features based on faulty assumptions.

A shortcoming of the analysis method when applying it to applications that heavily rely on influencing and convincing users, is that the result of user feedback provided by the application cannot be anticipated: information regarding this is not included in the domain model. Although we cannot analyze this part of the system, this only concerns the exact feedback messages, which are inherently difficult to properly pose to the user: as we noted when discussing the domain model, we can only reason about fixed physiological with a degree of certainty; when involving subjective mental elements, we cannot be sure what the results of interventions will be.

We answered this question in chapter 11.

**RQ3 To which extent does the method satisfy the requirements?**

We validated the improvements suggested as solutions for RQ2.1 – RQ2.4 to be correct. Through a set of experiments, we tested the modeling language DWDM, the reference well-being domain model, the model-driven development process, and the behavior prediction analysis method. We detailed the process of validation and the results in Chapters 7, 9 and 11.

To create the commit/swell reference model, we organized sessions in which domain experts talked about and reasoned over the domain, and added to the model themselves, which validates requirement R6 (models should be readable by domain experts). We verified the model by interviewing additional experts. At the same time, we tested the expressiveness of the language: can we depict the suggested improvements to the model in the DWDM? We found the language to be expressive enough, and implemented the suggestions to the model. We then used application specific derived models in a model-driven development process to create a well-being application. The knowledge of the domain experts was not needed to derive these models, which satisfies requirement R7 (time required of experts should be minimized).

Using Computer Science master students, we tested the model-driven process, and found it to be helpful (validating requirement R1), but the participants suggested some improvements to the usability of the tooling. The process is structured similarly to ex-
Conclusions and Future Work

Existing model-driven processes, satisfying requirements R5 (well-documented relation between domain information and design), R8 (allow use of existing management methods), R9 (minimize the chance for software errors) and R2 (clearly documented steps), and uses tooling created in the Eclipse IDE. This satisfies our requirements R3 (the method should be tool supported) and R4 (the tools should be familiar to the developer).

We verified the behavior prediction analysis method by looking at three DWDM models of existing applications. This analysis showed potential shortcomings in these applications. We compared our findings with existing end-user studies, and found that our method had predicted (or overestimated) shortcomings. If our method had been used up front, these shortcomings could have been prevented. This too satisfies requirement R9 (minimize software errors).

12.2 Future Work

Based on the work proposed and validation experiments performed, we can formulate directions for future work. We do so in this section.

12.2.1 Dynamic Well-being Domain Model Language and Reference Domain Model

We constructed the DWDM language to capture the domain of well-being. According to the validation experiments conducted, it was well suited for this task. Future enhancements to this modeling language, however, can make it more useful.

Firstly, more types of measurability should be added, so direct user input, or answers provided by the user to questions asked may be properly classified. Currently, these are described as “observable,” with the observation device, arguably, the questions or questionnaire presented to the user. Additionally, it should be determined how such new classifications of measurability should be quantified.

Secondly, temporal elements should be added to the language to allow for the modeling of timed aspects. Timed aspects are of most use when adding them to causal relations to allow for the documentation of changes in variable values and their effects on other variables, e.g. the time required for the effect to be at its maximum. With
12.2 Future Work

this temporal data, enhanced reasoning may be performed which takes time into account.

Thirdly, means of modeling alternative causal relations should be added. During interviews with experts we found that some relations do not hold for all individuals. By adding an “alternative effect” construct to the language, such relations could be modeled as well, providing more information in the model.

Although the DWDM language is generic enough to be used in other domains than well-being, we did not test this. It would be interesting to see causal models be created for such other domains.

Even though the reference well-being model was generally deemed correct, we know it is not complete. Claiming full completeness for a reference model is too ambitious and any author doing so will be proved wrong at some point in time. However, we do see the possibility for the reference model to become more complete: by using the reference model as a starting point for the development of application specific domain models, adding detail regarding the specifics of the application domain, and adding these specifics back to the reference model, increased detail may be achieved. Care should be taken that causal relations altered by the application are not directly transferred to the reference model; only more detailed versions of variables should be included and application specific elements should be left out.

To create models in the DWDM language, we constructed a model editor using the Eclipse GMF framework. We described this in section 8.4 on page 140 and in section A.1.1 on page 221. Developers can only use this editor to model the DWDM; it is currently not possible to perform sanity or validity checks on the models created. Such checks would, however, improve the quality of the models created using the tool. Another check that could be added is one to verify the stability of the system described in the model, as discussed in section 5.2.4. A stable system would consist of only balancing loops, or have reinforcing loops balanced by other reinforcing loops. The tool can identify such loops by performing an analysis of the causal relations between the variables of the model. If such an analysis finds a situation which might indicate an unstable system, the tool can alert the developer of this.

In section 5.3.1, we argued that machine learning can play an important part in the operation and personalization of context-aware applications at run-time. Applications utilizing machine learning
algorithms require specific inputs, such as set boundaries for the adaptive behavior. Currently such boundaries cannot be explicitly modeled in DWDMs. Future research at the intersection of context-aware application development and machine learning algorithm development should evaluate whether the DWDM language is suited for this purpose or whether this would require additional language constructs.

12.2.2 Model-driven development using DWDMs

The model-driven process overall was well liked, however, some alterations can make it better, these changes are all related to the tooling used.

Firstly, the current model transformations could generate full Activity Diagrams, however, the diagrams generated cannot yet be visualized in an editor, significantly reducing their usefulness. Improving this transformation step would solve this issue.

Secondly, the PIM level diagrams cannot automatically be translated to PSM level models; this is to be fully done by hand. Tool support to aid this translation or even full automation, based on the execution platform of choice, would greatly increase the usefulness of the transformations.

Thirdly, the current models all include only parts of the overall system (static and dynamic elements), with no easy way of integrating them. Automatic assembly of these system parts would reduce developer effort required.

Fourthly, the model transformations provided allow for easy forward development of the desired application. It is, however, currently not possible to make automatic alterations to models that were created/generated earlier in the process if changes in later models are made, such as updating PIM level models from alterations PSM level models. As a result, models are no longer “in sync.” Some model transformation languages allow for bidirectional model transformations, but the language used in this dissertation, ATL, does not. Additional model transformation rules should therefore be written to allow for this.

Fifthly, we found that the editor for the DWDMs is currently only very basic and not visually appealing. Enhancements to the editor could include the ability to collapse unused diagram items, adding
12.2 Future Work

icons to variables to ease the identification of their measurability, and a real time simulator to visualize the results of changes in the value of a selected variable.

Sixthly, the software development process discussed in this dissertation is top-down oriented. As mentioned in chapter 8, current software development trends show an increase in the use of agile methods, in which the software product is developed in numerous small iterations, rather than through one large process. Although we argue that our method can also be used to improve such agile development processes, we have not tested this theory. We hypothesize that the development of a DWDM in an agile process, and its subsequent usage will not add significant additional burden to developers and could result in the benefits to the process and the product we have seen when used in a top-down development process.

Finally, we argued and referenced in chapter 4 that end-users are unable to properly formulate their demands regarding a context-aware application when they are not in the context at that moment. This was the basis for us to specify the modeling language we described in chapter 5: rather than basing an application on requirements provided by users in a workshop or other setting outside the context of use, we base it on the objective workings of the well-being domain as captured in a DWDM model. However, developers might still desire to include users in improvement iterations of the application. In the medical field, users will often include medical professionals using the context-aware application as a tele-treatment method. These professionals will be able to read the DWDM models, as these models are based on causal models, which are commonly used in the medical field. To improve readability even more, however, we could add annotations to the DWDMs to allow inclusion of design rationale, as we discussed in section 8.5.

12.2.3 Predicting application behavior using DWDMs

As the limitations identified for our behavior prediction method are caused by limitations to the DWDM language, future work on this part primarily depends on changes performed on the DWDM language. If such changes are made, additional questions should be added to cover those changes.
12.3 CONCLUSIONS

The field of well-being is highly diverse and complex. The support of people into obtaining and attaining a balanced life-style, as to improve their well-being, is becoming increasingly important, but also increasingly plausible with mobile sensor devices becoming the norm, often in the shape of a smartphone device.

The development of well-being systems that use the observed context to deliver the best possible service to their users is challenging and current methods primarily focus on solving technological problems. In this dissertation, we proposed user-centric methods for application development, allowing designers to capture and reason over the domain of well-being, develop the system while being supported by model transformations and tools, and predict the strengths and possible issues of the system under development.

We found that the proposed methods are a step in the right direction, allowing for more user focused design of context-aware well-being systems. However, we also found ways of improving our method, making it even more useful. If the DWDM language is extended by adding more types of measurability, the idea of temporal aspects, and additional types of causal relations, it would be even more expressive, which would enable developers to capture more information about the well-being domain. If developers model such additional information, they can add it back to the reference domain model to make it more complete.

The tooling we created for the model-transformation driven process we described should also be improved. Users of the tools currently cannot generate full visualizations of Activity Diagrams, or generate Platform Specific Models from Platform Independent Models. Also, future developers of the tools should add the possibility to generate full applications, rather than separate static and dynamic elements of an application. They should also look into reverse engineering/generation of models (moving from PSM to PIM models) and the visual presentation of the tool itself.
12.3 CONCLUSIONS

We only used the tools and development methods in the scope of the commit/swell project, only postulating that they could also be used in domains other than well-being. Future research into the DWDM and the tools surrounding it should be performed in such other domains.
Part VI

APPENDIX
In parts II and III, we discussed the DWDM language and its usage. A modeling language or model-driven technique, however, requires tooling to be of use to software designers. We describe the tool we developed throughout this research in this appendix. This tool has been used as part of the validation experiments performed in chapter 9. Comments regarding the tools made there thusly apply to the software described here.

A.1 TECHNOLOGIES USED

The DWDM modeling language was developed before the editor was created. The language syntax was described in an Ecore metamodel, as we discussed in chapter 5. This metamodel could then serve as the input for the creation of the model editor. We describe how this was done in this section.

A.1.1 Editor Development

We used the Graphical Modeling Framework (GMF) \[110\] to create the graphical editors for our DSL. GMF provides us with a set of Eclipse plug-ins, allowing for the structured development of graphical editors for Ecore-based models. Using the GMF Dashboard view, three models were derived from the DWDM Ecore metamodel:

- A domain generation model (.genmodel), which allowed the plug-in to generate basic tree editors. These editors are not user friendly, and as such were not used during the experiments.

- A tooling definition model (.gmftool), which allowed the customization of the tool menus.

---

Parts of this chapter have been published before in Bosems and van Sinderen [17].
The graphical definition model, in which the graphical representation of the model was to be specified such as the shapes sizes, and contents of the boxes and arrows to be drawn.

By combining the tooling definition model and the graphical definition model (this is done through a build in model transformation, which is driven by a third model: the mapping model (.gmfmap)), a diagram editor generator model is generated. Using this last model, it is possible to generate the graphical editor for the modeling language.

To be able to graphically edit UML models and perform model transformations, we integrated the UML2Tools and ATL plug-ins respectively when building the Eclipse-based editor. The building of the editor was done by exporting the project as “Eclipse product,” building it for Windows, Linux and Mac OS X. With all modern day computers running 64-bit operating systems, no 32-bit versions were build.

A.1.2 Core Framework

At the center of our executable application, we have a core framework implemented in Java. This framework is based on the architecture discussed in chapter 8. It contains components that are deemed reusable and common for context-aware well-being applications. The application development process discussed in previous chapters extends this core framework with components that are specific to a single application.

The basic functionality offered by the framework includes a graph traversal algorithm that allows for simplified reasoning over an internal representation of the DWDM graph. By identifying which variable the application should increase/decrease, i.e., what is the goal variable of the application, the algorithm traverses the graph, determining whether variables in the graph should be increased or decreased. As the means of making this change are known, and the sensors to measure certain variable values are known, the application can reason what should be done when certain sensor values are obtained; if these are desired values, the user should continue doing what they are doing, if the obtained values are undesirable, i.e. the direction of change is not the desired one, the user should be motivated into altering his/her activities.
It should be noted that only the reasoning over the graph is currently performed by the algorithm. The presentation of advice is not implemented, as this is deemed part of the user interface design.

A.2 Model Transformation

In our development process, we identify both model-to-model and model-to-text transformations. We have opted for two separate solutions for these. For the model-to-model transformation, we selected the Atlas Transformation Language (ATL) as the best suited candidate. The ATL editor and transformation engine are provided as an Eclipse plug-ins. ATL outperforms other model transformation engines and has an active community [118]. For the model-to-text transformations, we opted for the use of the Epsilon Generation Language (EGL). In ATL, two types of model transformations were created, namely the DWDM to UML Class diagram transformation, and the DWDM to UML Activity Diagram transformation. In EGL, a single DWDM to Java initialization code transformation was created.

We discussed the structure of the model transformation process in section 8.4; we labeled the model transformations T1 through T4. We use the same labels in this section.

DWDM to UML Class (T1)

The transformation of DWDM models to UML Class diagrams consists of several phases. The first phase consists of the generation of the reference architecture model elements, such as depicted in Figure 16 on page 128. These classes need to be generated first, as later, application specific classes modeling the sensors, actuators and concepts will relate to these classes. The relations between these classes are added as well.

The second phase entails the generation of classes. First there are those classes that extend the Concept class. These classes are generated from DWDM variables with the annotation derivable. The classes generated here will get a name, and getters and setters for their properties which include an object representation of the variable they represent and sensors that can directly observe to their value. We can derive these values from the DWDM as well, namely from the incoming causal relations from variables with an observable annotation.
tation. The classes for variables that are observable and controllable are generated in a similar way, extending the Sensor and Actuator classes respectively. Both contain slightly different operations due to their different natures. Sensor classes contain operations to gather data, whereas Actuators have operations to affect, for example, what is presented on the user interface, or the value of a context element.

In the third phase, all generated elements are related to the UML “Model” element. This is the top-level element in a UML2 diagram, other model elements being children of it. This concludes the generation of the PIM level UML Class Diagram. The pseudo code for this transformation is shown in Listing 3.

**DWDM to UML Activity (T2)**

The primary behavioral models generated from the DWDM diagrams show how variable values should be gathered and processed. As such, this process only holds for variables with the derivable annotation; observable variable values can directly obtained through sensors. First, the transformation generates elements that will be common for all Activity diagrams. These include a starting and ending node, and a fork and join element. We generate these last two to allow for parallel execution of the process: if the value of a derivable variable has to be calculated, the application does so based on the values of other variables affecting it. Because of this, the values of variables which have edges toward the derivable variable should be evaluated, which can be done at the same time. When all values
have been collected, the process executes a join, and can calculate the value.

Second, nodes are created, the sum of which representing the calculation of the value of an incoming variable. Which calculation is to be performed, depends on the dimension of the variable. For variables which have values that can be used on their own, such as blood pressure or weight, the value can be gathered using the getPreviousProcessedValue() function of the respective class. Variables for which the value is dependent on multiple sensor readings, such as heart rate or respiration frequency, require more calculations:
public UMLActivity DWDM2UMLActivity (UMLM dwdm) {
    UMLActivity res = new UMLActivity();

    // Transform derivable variables into getValue() flows
    foreach (Variable v in dwdm) {
        ActivityFlow act = new ActivityFlow();
        act.add(startNode);
        // First, transform if the variable is observable
        if (v.measureability = observable) {
            if (v.isTimeDepended) {
                act.add(node("calculateInterval"));
                act.add(node("getDataFromInterval"));
                act.add(node("sumData"));
            } else {
                act.add(node("getLastValue"));
            }
        } else {
            act.add(node("return result"));
        }
        act.add(endNode);
        res.add(act);
    }

    // Second, transform if the variable is derivable
    else if (v.measureability = derivable) {
        foreach (CausalRelation c in v.incomingRelations) {
            act.add(node(v.name + " getValue"));
        }
        act.add(node("sumValues"));
        act.add(node("return result"));
        act.add(endNode);
        res.add(act);
    }

    // Transform norms into if-then-else control flows
    // These control flows are placed in monitoring loops
    foreach (Norm n in dwdm) {
        ActivityFlow act = new ActivityFlow();
        act.add(decisionNode(
            sensorClass.currentMeasurement > norm.max, // condition
            sensorClass.decrease(), // true branch
            null // false branch
        ));
        act.add(decisionNode(
            sensorClass.currentMeasurement < norm.min, // condition
            sensorClass.increase(), // true branch
            null // false branch
        ));
        act.add(controlFlow(monitoringLoop));
        res.add(act);
    }
    return res;
}

Listing 4: DWDM to UML Activity diagram transformation
1. Get the dimension. This will tell us what over what period of time the sensor values should be calculated: beats per minute, kilometers per hour, et cetera.

2. Based on the dimension, the exact begin and end moment of the time slot for which we want to know the value is to be determined.

3. Once the time slot has been decided upon, sensor data over that period of time should be gathered.

4. Summing the sensor values that have been read in the time slot will yield the desired result.

Third, relations are created between the model elements generated in the first step, and the calculation paths created in the second step. This concludes the generation of the `getValue()` method for classes extending the `Concept` class.

The pseudo code of this transformation is shown in Listing 4.

A.2.1 PIM to PSM (ActivitySensor)

We cannot directly translate PIM level models into code. For this, we first need platform specific mappings. Our current solution consists of implementing a mapping per sensor, per platform. First, we refine the UML Class diagrams. Because we perform the transformation on a per sensor basis, we iterate all classes in the model, checking whether the class name matches that of a sensor class we support. We show an example in Listing 5. If this iteration finds a supported class by name, the transformation creates a new class with the same name, but adds attributes that hold a reference to the platform run-time class and the current value measured by the sensor. Once this process is done, the transformation adds the refined class to the model. If the found class is not of a supported type, it is simply copied into the new model.

After refining the static application structure, we do the same for the dynamic structure. Similar to the refinement of the UML Class diagrams, we iterate over the Activity Diagram nodes of the existing PSM level model looking for nodes that contain a certain string. In this example, we are looking at an ActivitySensor, which is based on an observable variable. Because of this, the method in the
Listing 5: PIM to PSM level static structure model for the ActivitySensor

PSM level Activity Diagram we are interested in is the `getValue()` method. When the transformation encounters a node that contains the text `getValue()`, it constructs a sub-flow. It gives this sub-flow the name of the original node. The sub-flow contains method calls to the platform specific methods to first obtain and store the current sensor value. It then process this value, which is stored in a platform specific way, to return a value in a format that suits the application. In the case of the ActivitySensor on a Microsoft Windows platform, this processing entails getting the different directions of movement from the sensor reading, which is obtained with the type `AccelerometerReading`, and returning the values of the X, Y and Z directions of movement in an array. When this processing is finished, the sub-flow is added to the overall flow. As was the case with the Class diagram refinement, if a node is encountered that does not match our search criteria, the transformation simply copies it to the new model. We show this process in Listing 6.

**DWDM to Java initialization (T4)**

Before it can be executed, we have to initialize the framework discussed in section A.1.2. We can generate part of this initialization code from the application specific DWDM. This is done through a model-to-text transformation.
public UMLActivity PIMtoPSMAct(UMLActivity PIMact)
{
    UMLActivity PSMact = new UMLActivity();
    ActivityFlow act = new ActivityFlow();

    foreach (origNode in PIMact.nodes) {

        /* If the node indicates a getValue action, refine it to fit the 
         * platform specific action of getting the value */
        if (origNode.name.includes(getValue)) {
            // Create a nested activity flow that includes the refinement
            ActivityFlow newFlow = new ActivityFlow();
            newFlow.name = origNode.name;
            // Add nodes to include the actual implementation
            newFlow.add(startNode);
            newFlow.add(node("this.currentValue = sensor.GetCurrentReading()"));
            newFlow.add(node("return [this.currentValue.AccelerationX,
                this.currentValue.AccelerationY,
                this.currentValue.AccelerationZ]");
            newFlow.add(endNode);
            act.add(newFlow);
        } else {
            PSMact.add(origNode);
        }
    }

    return PSMact;
}

Listing 6: PIM to PSM level dynamic structure model for the ActivitySensor

We initialize the framework code by translating the graph structure of the DWDM to a data structure that is used by the application to reason over the domain. For this, the transformation generates code that instantiates the Variable class. The instances contain all variable properties, such as name and measurement scale, but also information regarding norms associated with it; these norms are represented through upper and lower bound values for the variable to have. If the actual variable value is higher or lower respectively, actions have to be undertaken to decrease or increase the value.

After the variables have been translated to the application, causal relations are added between these variables. This is done by instantiating the Causality class. Instances of this class are provided with properties such as the polarity of the causality, and the relation's source and target.
The objects instantiated this way are added to the storage used by the application, allowing for retrieval of data when required, but also for the alteration of causal relations to facilitate personalization of the application’s behavior.
REFERENCE WELL-BEING DOMAIN MODEL

B.1 INITIAL MODEL

The model we present here is the original model version as we presented it to the domain experts, as described in chapter 7. We have divided the model into two pieces, the first containing the elements relevant to physical well-being, the second containing the variables found in the mental well-being domain. The part of the reference model regarding physical well-being can be found in Figure 32, the elements concerning mental well-being are depicted in Figure 32.

Definitions of variables - Physical well-being

ACTIVE HEART RATE A person’s heart rate while s/he is undertaking physical activity.

AWARENESS Having knowledge of the current context or situation.

BODY COMPOSITION The percentage of body fat, water, and bone in a person’s body.

CARDIOVASCULAR FITNESS The ability of the blood-cells, heart and lungs to supply the body with oxygen.

FEEDBACK “Helpful information or criticism that is given to someone to say what can be done to improve a performance.” Merriam-Webster [64]

FEELING OF WELL-BEING A feeling of “being happy, healthy, or prosperous.” Merriam-Webster [64]

HEART RATE The speed at which the heart beats, measured in beats per minute.

INTENTION TO CHANGE The degree to which a person is willing to alter his/her behavior.

MOTIVATION “The condition of being eager to act or work.” Merriam-Webster [64]
PHYSICAL ACTIVITY “Any bodily movement produced by skeletal muscles that requires energy expenditure.”

1 http://www.who.int/topics/physical_activity/en/
**physical well-being** The feeling of well-being regarding a person’s physical fitness.

**recovery** “The act or process of becoming healthy after an illness or injury.” Merriam-Webster [64]

**rest** Not undertaking physical activity.

**resting heart rate** A person’s heart rate when s/he is not undertaking physical activity.

**self-efficacy** The degree to which a person thinks s/he can successfully perform an activity.

---

*Definitions of variables - Mental well-being*

**computer activity** Work performed at a computer.

**coping** The ability to deal with (stressful) situations.

**deportment feedback** Feedback provided by colleagues.

**distractions** Hindrances that take a person’s attention off their work activities.

**energy** The ability and motivation of a person to undertake activities.

**experienced stress** Subjective feeling of “mental tension and worry caused by problems in your life, work, etc.” Merriam-Webster [64]

**feeling of well-being** A feeling of “being happy, healthy, or prosperous.” Merriam-Webster [64]

**heart rate** The speed at which the heart beats, measured in beats per minute.

**hobbies** Activities performed as a form of leisure.

**information support** Automated feedback, advice, and suggestions to aid a knowledge worker in his/her daily activities.

**locus of control** The degree to which a person feels s/he can exert control over events in his/her life.
Figure 33: Reference well-being model - Mental

**Mental well-being** The feeling of well-being regarding a person’s mental state.

**Perceived stressors** Subjective sources of stress.

**Recovery** “The act or process of becoming healthy after an illness or injury.” Merriam-Webster [64]
rest Not undertaking mentally challenging activities.

resting heart rate A person’s heart rate when s/he is not undertaking physical activity.

social interaction A relationship between two or more people.

stressors Sources of stress.

task load The amount of work a person has to perform during his/her job.

work efficacy The degree to which a person thinks s/he can successfully perform his/her work.
Figure 34: Improved reference well-being model - Physical

B.2 IMPROVED MODEL

Based on the improvements proposed in section 7.4.2, we have made alterations to the domain model. The changed variables are indicated as gray model elements. The improved model for physical well-being is shown in Figure 34, the elements concerning mental well-being are depicted in Figure 35.
B.2 improved model

![Diagram](image)

Figure 35: Improved reference well-being model - Mental
This appendix lists the questions and sub-models provided to the participants in the reference model validation experiment, which is discussed in chapter 7.

1. An increase in task load will result in an increase in computer activity, which causes the number of stressors to increase.

2. An increase in information support will decrease the number of stressors.
3. Distractions cause an increase in stressors.

4. An increased perceived stress is caused by an increase in the number of stressors. If a person has a higher locus of control, the experienced stress will be reduced.

5. The experienced stress can be reduced by teaching a person to cope with stress.
6. If somebody has the feeling s/he is good at his/her job, this will result in stress being experienced to a lesser degree.

7. Hobbies aid recovery.

8. An increase in the experienced stress will result in a decrease in energy.
9. Department feedback is beneficial for social interaction. This will result in increased mental well-being.

10. Increased energy will increase mental well-being.

11. If a person’s self-efficacy increases, this will result in an increase in the physical activity being undertaken.
12. Resting aids recovery, which will result in a person having an increased feeling of physical well-being.

13. If the feeling of well-being decreases, this will cause a person to undertake actions to change his/her current situation.

14. By giving feedback about the undertaken activity, the person will gain awareness in his/her situation.
15. Increased awareness will result in motivation, which will cause the person to be more physically active.

16. If a person increases his/her physical well-being, this will have a positive effect on his/her overall feeling of well-being.

17. Physical activity will result in improved body composition.
18. Improved cardiovascular fitness will result in a decreased resting heart rate.

19. Increased cardiovascular fitness will result in an improved feeling of physical well-being.

20. Improved body composition will increase the feeling of physical well-being.
SUBJECT MATERIAL APPLICATION DEVELOPMENT EXPERIMENT

The text in this appendix was provided to the participants of the experiment discussed in chapter 9.

D.1 INTRODUCTION

In this experiment, you will focus on the process of working toward a context-aware well-being application, moving from the specification of requirements, to the design of the application structure, and application behavior, not the actual programming of the application.

When designing the application, use the Eclipse application from http://1drv.ms/1DHe3Cz. Download the version appropriate for your platform (Windows, OS X or Linux)

1. extract it, download the workspace.zip and extract it to the same folder as where you extracted the Eclipse package. Use this workspace when performing the assignments. The editor will allow you to create Dynamic Well-being Domain Models (DWDM, more information regarding these later), perform model transformations from such a DWDM to UML class diagrams and UML activity diagrams. You may need to set the binary as executable when working with Linux or OS X

2. You have up to 1.5 hours for the entire exercise. After that time, please save all your work and send all your artifacts and a compressed file with your editor workspace to s.bosems@utwente.nl, mentioning the number written at the top of this assignment in the subject.

D.2 DYNAMIC WELL-BEING DOMAIN MODEL AND EDITOR

The notation of the Dynamic Well-being Domain Model (DWDM) is primarily inspired by the Causal loop diagram (CLD) notation. CLDs consist of boxes (modeling variables) and annotated arrows

1 Alas, only the Windows and Linux versions have been verified to work.
2 You may do so using chmod a+x eclipse, or by right clicking the eclipse file, going to Properties → Permissions → Allow executing file as program.
(modeling causalities). A causality may be either positive or negative: a positive causality from \( A \) to \( B \) (denoted as \( A \rightarrow^+ B \)) indicates that an increase in the variable value of \( A \) (eventually) causes an increase in the value of variable \( B \). A negative causality from \( A \) to \( B \) (written as \( A \rightarrow^- B \)) indicates that an increase in \( A \) causes a decrease in \( B \).

If a variable can be directly read through sensors, we say it is observable. If the application can directly influence the variable’s value, it is controllable. If the application should reason about the variable’s value using the value of other variables, it is derivable.

Variables can also have a unit of measurement (“km/h” for speed, for example).

D.3 Case description

In this experiment, you will be designing an “Activity Coach” [77]. It is a smartphone application that is to help the user in managing his/her activity throughout the day. This coaching/management can go both ways: if a user is too active during the morning, the application should warn the user that this might leave him/her tired in the afternoon, if the user is not active enough, s/he should be encouraged to be more active.

The application uses sensors (tri-axial accelerometers) to monitor the user throughout the day. These sensors measure acceleration in three directions \((x, y, z)\). For a full day, the desired amount of activity measured in IMA (sum of all the acceleration measured) should be 10,000. The user should be able to see his progress toward this goal at every time, preferably through a graph showing his/her activity and the desired norm. You can see an example in Figure 36.

We now have to identify the other aspects of the app. Firstly, the application should run on a mobile device, so battery power and storage capacity are limited. Because of this, recalculating and reevaluating the user’s goal progress every time a new sensor value is read might not be the best approach, just like saving all data for all time. Secondly, the application might be part of a bigger system where data storage and even the calculations are performed by other devices (cloud storage and computing). The app designed here should be able to cope with such future changes, the adaptations required to allow for these should be limited.
Figure 36: The Activity Coach interface

**REQUIREMENTS**

**Task 1** Have a look at the DWDM provided (it is located in the “input” folder of your workspace, the model is called `in.cld_diagram`). You can also find it on page 251. This domain model depicts the domain of the Activity Coach. Do you think all aspects of the domain have been captured? If not, what should be added, removed or otherwise changed?

**Task 2** Provide 5-10 requirements for the application discussed. Think of how the system should behave (in certain conditions), how it should manage uncertainty, how the UI should look like, what the storage requirements are et cetera.

**APPLICATION STRUCTURE**

**Task 3** Generate a static application structure for an application that is to satisfy your requirements. Do so by executing the `DDM2UMLClass` transformation (right clicking the appropriate transformation, selecting “Run As”, and clicking “ATL transformation”).
Generate the visual diagram by right clicking the generated model, and chose “Initialize Class Diagram.”

**Task 4** Verify that the generated application structure fits your idea of a suitable structure for the application under development. If not, alter it.

**Task 5** Check if you are still satisfied with your requirements. If not, alter or add to them.

**APPLICATION BEHAVIOR**

**Task 6** Generate the application behavior. Use the DDM2UMLAct model transformation for this. Verify that the generated models satisfy your requirements. Due to the tool being pre-alpha, we can not generate the visual diagrams for these yet.

**Task 7** Verify whether your requirements still reflect your current ideas regarding the application. If not, improve them.

**D.4 ACTIVITY COACH DOMAIN MODEL**

We provided the participants of the experiment with a DWDM that was inspired by the DWDM we created for the RRD Activity Coach. To reduce the complexity of this model, we did not include the variables for “Self efficacy” and “Feedback.” Figure 37 shows this reduced DWDM.
Figure 37: The DWDM for the reduced Activity Coach case
The Fishualization [91] is a distributed application created by TNO in the commit/swell project. It consists of software installed on a user PC, a central database, and a website visualizing the information stored in the database.

E.1 OVERVIEW

The creators of the TNO/SWELL Fishualization recognize that there is an increase in work related stress among knowledge workers. Seeing that it may not be possible to conquer the causes of this stress, i.e. increasing work pressure, distractions, fragmented work et cetera, the idea arose to provide workers with insight into their own work, and that of their coworkers, allowing for group discussion about levels of work pressure.

E.2 GOAL

The goal of the Fishualization is to increase worker awareness of the causes of workplace stress. Additionally, the system aims to increase communication among colleagues, starting the discussion about stress. Both of these should help the worker cope with stress more easily, reducing the feeling of stress over time.

E.3 MEANS

To reach its goal, the Fishualization has two ways of influencing the user. Firstly, feedback is given directly to the individual, informing him/her about his/her current levels of work activity and energy. Secondly, department feedback is used. Rather than targeting a specific employee, an entire group is presented feedback.

For the group feedback, visualization of information was done through an virtual aquarium displayed on a television screen in a coffee corner. The aquarium is inhabited by fish, each fish represen-
ting an employee. As the information visualized can be personal in nature, the colors and patterns of the fish can be altered so the in-
dividual users know which fish is their own, but coworkers do not
know which fish belongs to which colleague. The way a fish swims
is caused by the way the employee uses his/her computer:

- The speed at which the fish swims is based on the speed of
  computer activity. Fast typing and mouse activity will cause
  rapid fish movement.
- The fish will change its direction of movement when the user
  switches his/her working context or task.
- The depth the fish swims at, is related to the energy level of
  the user: a fish swimming shallow, i.e. high on the screen,
  represents a user with high levels of energy.

In addition to the presence of fish in the aquarium, plants grow from
the bottom. Each plant represents a computer application, such as
Microsoft Outlook, Word, et cetera, its height being determined by
the total number of people using the application.

### E.4 SENSORS

To obtain the context information required to provide the user and
the group with feedback, two types of data collection are used. Fir-
ly, uLog [73] is used to collect information regarding key strokes,
mouse movements, and mouse button clicks. Furthermore, informa-
tion provided by the operating system is stored, such as the titles of
windows, which application is active, and when a switch between
applications is made. The data collected is aggregated on the user’s
own PC before it is sent to the aquarium display for visualization.

Secondly, data regarding the user’s subjective energy level is to be
collected. This is done through a small Java application called the
“Battery app.” The user can record his/her energy level here, the
application asking the user for this information in regular intervals.

### E.5 DOMAIN MODEL

A domain model in the DWDM language has been created for this
application. It can be found in Figure 38.
Figure 38: DWDM for the TNO/SWELL Fishualization
Philips/SWELL mBeats [119] is a smartphone application created by Philips in the context of the commit/swell project. It runs on Apple iPhone devices with version number 4 and higher.

F.1 Overview

Sartor et al. [89] describe that to prevent cardiovascular diseases, cardio-respiratory fitness, and so physical activity, is required. The authors discuss that cardiovascular diseases are increasingly becoming the primary cause of death in western countries. With increased physical activity contributing to a decrease in the risk of such diseases, the authors argue that, although the general population is aware of these problems, adherence to fitness regimes is low. Smartphone and smart watch applications are suggested as a solution direction.

F.2 Goal

The mBeats application aims to increase its user’s cardio-respiratory fitness (CRF) by motivating the person to have his/her heart rate within a “target heart rate zone,” i.e. within a personalized heart rate range, by engaging in physical activity. This zone is calculated in such a way that the CRF will increase over time if the user continues his/her adherence to the program.

F.3 Means

The application’s only way of influencing the user is by providing feedback and comments. This feedback is provided as:

- A “speedometer,” representing the full heart rate range of the user, including a coloring per zone to indicate which zones are good for CRF improvement, and an arrow to indicate the current heart rate.
- The current heart rate as a number.
- An “mBeats” score.

mBeats points are awarded when the user’s heart rate is within the calculated target zone. By giving the user points for “good behavior,” the application includes game elements, the rationale being that users will want to improve their own score.

### F.4 SENSORS

The application, as of writing, bases its reasoning logic on the input of a single sensor. The sensor used is the Mio Alpha\(^1\) heart rate sensor. This wrist worn device sends its data to a smartphone device running the mBeats application using Bluetooth Low Energy. The heart rate monitor uses an optical blood flow sensor to determine the heart rate. The sensor is not equipped with internal memory, so the smartphone device needs to be turned on, and within range to be able to store heart rate data.

Means of including other sensors in the application’s reasoning logic, such as accelerometers, are currently being researched.

### F.5 DOMAIN MODEL

A domain model in the DWDM language has been created for this application. It can be found in Figure 39.

---

\(^1\) https://www.mioglobal.com/
Figure 39: DWDM for the Philips/SWELL mBeats application
The Activity Coach [77], developed by Roessingh Research and Development, is a tele-treatment application that is to be used by patients while undergoing rehabilitation. This is different from the other two applications discussed, which can be used by people if they feel they should improve their own lifestyle.

G.1 OVERVIEW

Recognizing the need for reform in the field of rehabilitation treatment, Roessingh Research and Development created the Activity Coach as part of a larger overall tele-treatment platform [77], which included ways for health care professionals to observe patient progress remotely, for treatment plans to be better tailored to the patient, and for data storage. The application runs on smartphone devices with the Android operating system installed.

G.2 GOAL

The Activity Coach was specifically designed for the tele-treatment of patients suffering from COPD. For these patients, the management of energy expenditure throughout the day may be challenging. The goal of the system is to provide guidance to these people, preventing them from over-exertion in the morning so they do not have any energy left in the afternoon, but also avoiding patients becoming sedentary which will hurt their cardiovascular fitness in the long run.

G.3 MEANS

The Activity Coach has seen a number of development iterations, new versions including more elaborate ways of providing feedback. The primary way of showing a user how much energy s/he has
spent, is through a line graph. This graph depicts the total energy spent throughout the day; the x-axis showing the time of the day, the y-axis the cumulative energy expenditure. In addition, a reference line is shown. This is the cumulative expenditure users should aim for, preventing both over-exertion and a passive overall lifestyle.

Newer versions of the Activity Coach adapt the goal line, as to maintain a challenging, but attainable goal [21], and include persuasive feedback strategies [3].

G.4 SENSORS

The Activity Coach’s advice is based on measurements obtained from a tri-axial accelerometer and gyroscope. The device used for this purpose is the Inertia Technology ProMove 3D\(^1\). The user can clip the sensor to a belt, after which it sends the movement data through a Bluetooth connection to the smartphone running the Activity Coach.

G.5 DOMAIN MODEL

A domain model in the DWDM language has been created for this application. It can be found in Figure 40.

---

\(^1\) http://inertia-technology.com/promove-3d


Bibliography


Bibliography


Publications by the author


<table>
<thead>
<tr>
<th>No.</th>
<th>Author (Institute)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Botond Cseke (RUN)</td>
<td>Variational Algorithms for Bayesian Inference in Latent Gaussian Models</td>
</tr>
<tr>
<td>02</td>
<td>Nick Tinnemeier (UU)</td>
<td>Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language</td>
</tr>
<tr>
<td>03</td>
<td>Jan Martijn van der Werf (TUE)</td>
<td>Compositional Design and Verification of Component-Based Information Systems</td>
</tr>
<tr>
<td>04</td>
<td>Hado van Hasselt (UU)</td>
<td>Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference</td>
</tr>
<tr>
<td>05</td>
<td>Bas van der Raadt (VU)</td>
<td>Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.</td>
</tr>
<tr>
<td>06</td>
<td>Yiwen Wang (TUE)</td>
<td>Semantically-Enhanced Recommendations in Cultural Heritage</td>
</tr>
<tr>
<td>07</td>
<td>Yujia Cao (UT)</td>
<td>Multimodal Information Presentation for High Load Human Computer Interaction</td>
</tr>
<tr>
<td>08</td>
<td>Nieske Vergunst (UU)</td>
<td>BDI-based Generation of Robust Task-Oriented Dialogues</td>
</tr>
<tr>
<td>09</td>
<td>Tim de Jong (OU)</td>
<td>Contextualised Mobile Media for Learning</td>
</tr>
<tr>
<td>10</td>
<td>Bart Bogaert (UvT)</td>
<td>Cloud Content Contention</td>
</tr>
<tr>
<td>11</td>
<td>Dhaval Vyas (UT)</td>
<td>Designing for Awareness: An Experience-focused HCI Perspective</td>
</tr>
<tr>
<td>12</td>
<td>Carmen Bratosin (TUE)</td>
<td>Grid Architecture for Distributed Process Mining</td>
</tr>
<tr>
<td>13</td>
<td>Xiaoyu Mao (UvT)</td>
<td>Airport under Control. Multiagent Scheduling for Airport Ground Handling</td>
</tr>
<tr>
<td>14</td>
<td>Milan Lovric (EUR)</td>
<td>Behavioral Finance and Agent-Based Artificial Markets</td>
</tr>
<tr>
<td>15</td>
<td>Marijn Koolen (UvA)</td>
<td>The Meaning of Structure: the Value of Link Evidence for Information Retrieval</td>
</tr>
<tr>
<td>16</td>
<td>Maarten Schadd (UM)</td>
<td>Selective Search in Games of Different Complexity</td>
</tr>
<tr>
<td>17</td>
<td>Jiyin He (UVA)</td>
<td>Exploring Topic Structure: Coherence, Diversity and Relatedness</td>
</tr>
</tbody>
</table>
18 Mark Ponsen (UM), Strategic Decision-Making in complex games
19 Ellen Rusman (OU), The Mind’s Eye on Personal Profiles
20 Qing Gu (VU), Guiding service-oriented software engineering - A view-based approach
21 Linda Terlouw (TUD), Modularization and Specification of Service-Oriented Systems
22 Junte Zhang (UVA), System Evaluation of Archival Description and Access
23 Wouter Weerkamp (UVA), Finding People and their Utterances in Social Media
24 Herwin van Welbergen (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior
25 Syed Waqar ul Qounain Jaffry (VU), Analysis and Validation of Models for Trust Dynamics
26 Matthijs Aart Pontier (VU), Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots
27 Aniel Bhulai (VU), Dynamic website optimization through autonomous management of design patterns
28 Rianne Kaptein (UVA), Effective Focused Retrieval by Exploiting Query Context and Document Structure
29 Faisal Kamiran (TUE), Discrimination-aware Classification
30 Egon van den Broek (UT), Affective Signal Processing (ASP): Unraveling the mystery of emotions
31 Ludo Waltman (EUR), Computational and Game-Theoretic Approaches for Modeling Bounded Rationality
32 Nees-Jan van Eck (EUR), Methodological Advances in Bibliometric Mapping of Science
33 Tom van der Weide (UU), Arguing to Motivate Decisions
34 Paolo Turrini (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
35 Maaike Harbers (UU), Explaining Agent Behavior in Virtual Training
36 Erik van der Spek (UU), Experiments in serious game design: a cognitive approach
37 Adriana Burlutiu (RUN), Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference
38 Nyree Lemmens (UM), Bee-inspired Distributed Optimization
39  Joost Westra (UU), Organizing Adaptation using Agents in Serious Games
40  Viktor Clerc (VU), Architectural Knowledge Management in Global Software Development
41  Luan Ibraimi (UT), Cryptographically Enforced Distributed Data Access Control
42  Michal Sindlar (UU), Explaining Behavior through Mental State Attribution
43  Henk van der Schuur (UU), Process Improvement through Software Operation Knowledge
44  Boris Reuderink (UT), Robust Brain-Computer Interfaces
45  Herman Stehouwer (UvT), Statistical Language Models for Alternative Sequence Selection
46  Beibei Hu (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
47  Azizi Bin Ab Aziz (VU), Exploring Computational Models for Intelligent Support of Persons with Depression
48  Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
49  Andreea Niculescu (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012

01  Terry Kakeeto (UvT), Relationship Marketing for SMEs in Uganda
02  Muhammad Umair (VU), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
03  Adam Vanya (VU), Supporting Architecture Evolution by Mining Software Repositories
04  Jurriaan Souer (UU), Development of Content Management System-based Web Applications
05  Marijn Plomp (UU), Maturing Interorganisational Information Systems
06  Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks
07  Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
08 Gerben de Vries (UVA), Kernel Methods for Vessel Trajectories
09 Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms
10 David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment
11 J.C.B. Rantham Prabhakara (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics
12 Kees van der Sluijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems
13 Suleman Shahid (UvT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions
14 Evgeny Knutov (TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems
15 Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.
16 Fiemke Both (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
17 Amal Elgammal (UvT), Towards a Comprehensive Framework for Business Process Compliance
18 Eltjo Poort (VU), Improving Solution Architecting Practices
19 Helen Schonenberg (TUE), What’s Next? Operational Support for Business Process Execution
20 Ali Bahramisharif (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
21 Roberto Cornacchia (TUD), Querying Sparse Matrices for Information Retrieval
22 Thijs Vis (UvT), Intelligence, politie en veiligheidsdienst: vereenigbare grootheden?
23 Christian Muehl (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
24 Laurens van der Werff (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval
25 Silja Eckartz (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
26 Emile de Maat (UVA), Making Sense of Legal Text
27 Hayrettin Gurkok (UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games
28 Nancy Pascall (UvT), Engendering Technology Empowering Women
29 Almer Tigelaar (UT), Peer-to-Peer Information Retrieval
30 Alina Pommeranz (TUD), Designing Human-Centered Systems for Reflective Decision Making
31 Emily Bagarukayo (RUN), A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure
32 Wietske Visser (TUD), Qualitative multi-criteria preference representation and reasoning
33 Rory Sie (OUN), Coalitions in Cooperation Networks (CO-COON)
34 Pavol Jancura (RUN), Evolutionary analysis in PPI networks and applications
35 Evert Haasdijk (VU), Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics
36 Denis Ssebugwawo (RUN), Analysis and Evaluation of Collaborative Modeling Processes
37 Agnes Nakakawa (RUN), A Collaboration Process for Enterprise Architecture Creation
38 Selmar Smit (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms
39 Hassan Fatemi (UT), Risk-aware design of value and coordination networks
40 Agus Gunawan (UvT), Information Access for SMEs in Indonesia
41 Sebastian Kelle (OU), Game Design Patterns for Learning
42 Dominique Verpoorten (OU), Reflection Amplifiers in self-regulated Learning
43 Withdrawn
44 Anna Tordai (VU), On Combining Alignment Techniques
45 Benedikt Kratz (UvT), A Model and Language for Business-aware Transactions
46 Simon Carter (UVA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
47 Manos Tsagkias (UVA), Mining Social Media: Tracking Content and Predicting Behavior
48 Jorn Bakker (TUE), Handling Abrupt Changes in Evolving Time-series Data
49  Michael Kaisers (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
50  Steven van Kervel (TUD), Ontology driven Enterprise Information Systems Engineering
51  Jeroen de Jong (TUD), Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching

2013

01  Viorel Milea (EUR), News Analytics for Financial Decision Support
02  Erietta Liarou (CWI), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing
03  Szymon Klarman (VU), Reasoning with Contexts in Description Logics
04  Chetan Yadati (TUD), Coordinating autonomous planning and scheduling
05  Dulce Pumareja (UT), Groupware Requirements Evolutions Patterns
06  Romulo Goncalves (CWI), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
07  Giel van Lankveld (UvT), Quantifying Individual Player Differences
08  Robbert-Jan Merk (VU), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
09  Fabio Gori (RUN), Metagenomic Data Analysis: Computational Methods and Applications
10  Jeewanie Jayasinghe Arachchige (UvT), A Unified Modeling Framework for Service Design.
11  Evangelos Pournaras (TUD), Multi-level Reconfigurable Self-organization in Overlay Services
12  Marian Razavian (VU), Knowledge-driven Migration to Services
13  Mohammad Safiri (UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly
14  Jafar Tanha (UVA), Ensemble Approaches to Semi-Supervised Learning Learning
15  Daniel Hennes (UM), Multiagent Learning - Dynamic Games and Applications
Eric Kok (UU), Exploring the practical benefits of argumentation in multi-agent deliberation

Koen Kok (VU), The PowerMatcher: Smart Coordination for the Smart Electricity Grid

Jeroen Janssens (UvT), Outlier Selection and One-Class Classification

Renze Steenhuizen (TUD), Coordinated Multi-Agent Planning and Scheduling

Katja Hofmann (UvA), Fast and Reliable Online Learning to Rank for Information Retrieval

Sander Wubben (UvT), Text-to-text generation by monolingual machine translation

Tom Claassen (RUN), Causal Discovery and Logic

Patricio de Alencar Silva (UvT), Value Activity Monitoring

Haitham Bou Ammar (UM), Automated Transfer in Reinforcement Learning

Agnieszka Anna Latoszek-Berendsen (UM), Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System

Alireza Zarghami (UT), Architectural Support for Dynamic Homecare Service Provisioning

Mohammad Huq (UT), Inference-based Framework Managing Data Provenance

Frans van der Sluis (UT), When Complexity becomes Interesting: An Inquiry into the Information eXperience

Iwan de Kok (UT), Listening Heads

Joyce Nakatumba (TUE), Resource-Aware Business Process Management: Analysis and Support

Dinh Khoa Nguyen (UvT), Blueprint Model and Language for Engineering Cloud Applications

Kamakshi Rajagopal (OUN), Networking For Learning: The role of Networking in a Lifelong Learner’s Professional Development

Qi Gao (TUD), User Modeling and Personalization in the Microblogging Sphere

Kien Tjin-Kam-Jet (UT), Distributed Deep Web Search

Abdallah El Ali (UvA), Minimal Mobile Human Computer Interaction

Than Lam Hoang (TUE), Pattern Mining in Data Streams

Dirk Börner (OUN), Ambient Learning Displays
38 Eelco den Heijer (VU), Autonomous Evolutionary Art
39 Joop de Jong (TUD), A Method for Enterprise Ontology based Design of Enterprise Information Systems
40 Pim Nijssen (UM), Monte-Carlo Tree Search for Multi-Player Games
41 Jochem Liem (UVA), Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
42 Léon Planken (TUD), Algorithms for Simple Temporal Reasoning
43 Marc Bron (UVA), Exploration and Contextualization through Interaction and Concepts

2014

01 Nicola Barile (UU), Studies in Learning Monotone Models from Data
02 Fiona Tuliyano (RUN), Combining System Dynamics with a Domain Modeling Method
03 Sergio Raul Duarte Torres (UT), Information Retrieval for Children: Search Behavior and Solutions
04 Hanna Jochmann-Mannak (UT), Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation
05 Jurriaan van Reijsen (UU), Knowledge Perspectives on Advancing Dynamic Capability
06 Damian Tamburri (VU), Supporting Networked Software Development
07 Arya Adriansyah (TUE), Aligning Observed and Modeled Behavior
08 Samur Araujo (TUD), Data Integration over Distributed and Heterogeneous Data Endpoints
09 Philip Jackson (UvT), Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language
10 Ivan Salvador Razo Zapata (VU), Service Value Networks
11 Janneke van der Zwaan (TUD), An Empathic Virtual Buddy for Social Support
12 Willem van Willigen (VU), Look Ma, No Hands: Aspects of Autonomous Vehicle Control
13 Arlette van Wissen (VU), Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains
14 Yangyang Shi (TUD), Language Models With Meta-information
15 Natalya Mogles (VU), Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare
16 Krystyna Milian (VU), Supporting trial recruitment and design by automatically interpreting eligibility criteria
17 Kathrin Dentler (VU), Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability
18 Mattijs Ghijsen (UVA), Methods and Models for the Design and Study of Dynamic Agent Organizations
19 Vinicius Ramos (TUE), Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support
20 Mena Habib (UT), Named Entity Extraction and Disambiguation for Informal Text: The Missing Link
21 Kassidy Clark (TUD), Negotiation and Monitoring in Open Environments
22 Marieke Peeters (UU), Personalized Educational Games - Developing agent-supported scenario-based training
23 Eleftherios Sidirourgos (UvA/CWI), Space Efficient Indexes for the Big Data Era
24 Davide Ceolin (VU), Trusting Semi-structured Web Data
25 Martijn Lappenschaar (RUN), New network models for the analysis of disease interaction
26 Tim Baarslag (TUD), What to Bid and When to Stop
27 Rui Jorge Almeida (EUR), Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty
28 Anna Chmielowiec (VU), Decentralized k-Clique Matching
29 Jaap Kabbedijk (UU), Variability in Multi-Tenant Enterprise Software
30 Peter de Cock (UvT), Anticipating Criminal Behaviour
31 Leo van Moergestel (UU), Agent Technology in Agile Multiparallel Manufacturing and Product Support
32 Naser Ayat (UvA), On Entity Resolution in Probabilistic Data
33 Tesfa Tegegne (RUN), Service Discovery in eHealth
34 Christina Manteli (VU), The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.
35 Joost van Ooijen (UU), Cognitive Agents in Virtual Worlds: A Middleware Design Approach
36 Joos Buijs (TUE), Flexible Evolutionary Algorithms for Mining Structured Process Models
37 Maral Dadvar (UT), Experts and Machines United Against Cyberbullying
38 Danny Plass-Oude Bos (UT), Making brain-computer interfaces better: improving usability through post-processing.
39 Jasmina Maric (UvT), Web Communities, Immigration, and Social Capital
40 Walter Omona (RUN), A Framework for Knowledge Management Using ICT in Higher Education
41 Frederic Hogenboom (EUR), Automated Detection of Financial Events in News Text
42 Carsten Eijckhof (CWI/TUD), Contextual Multidimensional Relevance Models
43 Kevin Vlaanderen (UU), Supporting Process Improvement using Method Increments
44 Paulien Meesters (UvT), Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.
45 Birgit Schmitz (OUN), Mobile Games for Learning: A Pattern-Based Approach
46 Ke Tao (TUD), Social Web Data Analytics: Relevance, Redundancy, Diversity
47 Shangsong Liang (UVA), Fusion and Diversification in Information Retrieval

2015

01 Niels Netten (UvA), Machine Learning for Relevance of Information in Crisis Response
02 Faiza Bukhsh (UvT), Smart auditing: Innovative Compliance Checking in Customs Controls
03 Twan van Laarhoven (RUN), Machine learning for network data
04 Howard Spoelstra (OUN), Collaborations in Open Learning Environments
05 Christoph Bösch (UT), Cryptographically Enforced Search Pattern Hiding
06 Farideh Heidari (TUD), Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes

292
Maria-Hendrike Peetz (UvA), Time-Aware Online Reputation Analysis

Jie Jiang (TUD), Organizational Compliance: An agent-based model for designing and evaluating organizational interactions

Randy Klaassen (UT), HCI Perspectives on Behavior Change Support Systems

Henry Hermans (OUN), OpenU: design of an integrated system to support lifelong learning

Yongming Luo (TUE), Designing algorithms for big graph datasets: A study of computing bisimulation and joins

Julie M. Birkholz (VU), Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks

Giuseppe Procaccianti (VU), Energy-Efficient Software

Bart van Straalen (UT), A cognitive approach to modeling bad news conversations

Klaas Andries de Graaf (VU), Ontology-based Software Architecture Documentation

Changyun Wei (UT), Cognitive Coordination for Cooperative Multi-Robot Teamwork

André van Cleeff (UT), Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs

Holger Pirk (CWI), Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories

Bernardo Tabuenca (OUN), Ubiquitous Technology for Lifelong Learners

Lois Vanhée (UU), Using Culture and Values to Support Flexible Coordination

Sibren Fetter (OUN), Using Peer-Support to Expand and Stabilize Online Learning

Zhemin Zhu (UT), Co-occurrence Rate Networks

Luit Gazendam (VU), Cataloguer Support in Cultural Heritage

Richard Berendsen (UVA), Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation

Steven Woudenberg (UU), Bayesian Tools for Early Disease Detection

Alexander Hogenboom (EUR), Sentiment Analysis of Text Guided by Semantics and Structure

Sándor Héman (CWI), Updating compressed column stores

Janet Bagorogoza (TiU), Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO
29 Hendrik Baier (UM), Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains
30 Kiavash Bahreini (OU), Real-time Multimodal Emotion Recognition in E-Learning
31 Yakup Koç (TUD), On the robustness of Power Grids
32 Jerome Gard (UL), Corporate Venture Management in SMEs
33 Frederik Schadd (TUD), Ontology Mapping with Auxiliary Resources
34 Victor de Graaf (UT), Gesocial Recommender Systems

2016

01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
04 Laurens Rietveld (VU), Publishing and Consuming Linked Data
05 Evgeny Sherkhonov (UVA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
07 Jeroen de Man (VU), Measuring and modeling negative emotions for virtual training
08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
09 Archana Nottamkandath (VU), Trusting Crowdsourced Information on Cultural Artefacts
10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
11 Anne Schuth (UVA), Search Engines that Learn from Their Users
12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
13 Nana Baah Gyan (VU), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UVA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VU), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VU), Refining Statistical Data on the Web
19 Julia Efremova (Tu/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UVA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VU), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UVA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VU), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (UvT), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring
32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UVA), Single Sample Statistics, exercises in learning from just one example
34 Dennis Schunselaar (TUE), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UVA), Monitoring Social Media: Summarization, Classification and Recommendation
36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
40 Christian Detweiler (TUD), Accounting for Values in Design
41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
42 Spyros Martzoukos (UVA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
44 Thibault Sellam (UVA), Automatic Assistants for Database Exploration
45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
46 Jorge Gallego Perez (UT), Robots to Make you Happy
47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
48 Tanja Buttler (TUD), Collecting Lessons Learned
49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
50 Yan Wang (UVT), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

2017

01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UVA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VU), Detecting Interesting Differences: Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TUE), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (UV), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UVA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UVA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VU), Logics for causal inference under uncertainty
23 David Graus (UVA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VU), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People’s Preferences, Perceptions and Behaviors
28 John Klein (VU), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (UvT), From IT-Business Strategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT
30 Wilma Latuny (UvT), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VU), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TUE), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46  Jan Schneider (OU), Sensor-based Learning Support
47  Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48  Angel Suarez (OU), Collaborative inquiry-based learning

2018

01  Han van der Aa (VUA), Comparing and Aligning Process Representations
02  Felix Mannhardt (TUE), Multi-perspective Process Mining
03  Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction