Method Management System: 
Rule-Based Method Enactment Using MediaWiki and Semantic MediaWiki

Master Thesis Information Science

Muhammad Deri Taufan
4012887, m.deritaufan@student.ru.nl

Supervisor (1st reviewer): Dr. S.J.B.A. (Stijn) Hoppenbrouwers
2nd reviewer: Prof.dr.ir. Th.P. (Theo) van der Weide

Research Number: 146 IK
# Table of Contents

1. **Introduction** 5  
   1.1 Context of the thesis 5  
   1.2 Problem statement 7  
   1.3 General software project management 8  
   1.4 Organization and content of the thesis 8  

2. **Method** 9  
   2.1 Research Questions 9  
   2.2 Methodology 10  

3. **Theoretical Frameworks** 11  
   3.1 Agile Service Development: rule-based method engineering 11  
   3.2 Goal-driven approach 13  
   3.3 Declarative workflow 14  
   3.4 The need for an incremental approach 14  
   3.5 Project management 15  
      3.5.1 Traditional project management 16  
      3.5.2 Agile project management 16  
   3.6 Integrated theoretical frameworks 18  

4. **The Method Management System** 20  
   4.1 Overview of the system 20  
   4.2 Architecture of the system 20  
      4.2.1 MediaWiki 21  
      4.2.2 Semantic MediaWiki 23  
   4.3 Requirements 25  
      4.3.1 Use case diagram 26  
      4.3.2 Use cases 27  
      4.3.3 Scenarios 35  
   4.4 Design 38  
      4.4.1 Information structure 38  
      4.4.2 Declarative workflows and temporal ordering 41  
   4.5 Implementation 42  
   4.6 System session example 43
5. Evaluation

5.1 Test cases
   5.1.1 HEMD project
   5.1.2 ResAna project

5.2 Result and findings
   5.2.1 HEMD Project
   5.2.2 ResAna Project

6. Conclusions

6.1 Conclusions
6.2 Future work

References

Appendix A: HEMD project execution
Appendix B: ResAna project execution
Appendix C: Wiki Markup Template Source Code
Appendix D: The HistoryOfPage Extension Source Code
Acknowledgements

I could not have finished this master thesis without the help of many people directly or indirectly. Firstly, I would like to extend my gratitude to Nuffic and Nuffic Neso Indonesia for granting and organizing the StuNed scholarship for me, without the scholarship I would have not been able to pursue my master degree in the Netherlands. I would also like to express my sincere appreciation to my supervisor Dr. S.J.B.A (Stijn) Hoppenbrouwers who has given his valuable input and guided me to the accomplishment of this thesis. He always has faith in me for completing every task that was assigned to me.

I want to extend my gratefulness to Prof.dr.ir. Th.P. van der Weide for taking care of all issues that I have faced even before I started the study. He helped me during my pre-master program that by the end brought me to the master program. He is a mediator between the (administrative) issues and me.

And finally, I want to thank my family, friends and in particular my parents and my beloved wife, Ainul, for their unconditional support during my study in the Netherlands.

- Muhammad Deri Taufan -
CHAPTER 1

Introduction

As the intricacy and variety of services and systems in Information System Development (ISD) are becoming larger and more complex, the need for appropriate, effective, and flexible service or information system project development methods has increased excessively. Adopting such methods in development of services or systems needs to take into account organizational, technical, and human-related constraints and requirements. Failing to adapt to changes is sometime what makes service development and ISD projects fail.

1.1 Context of the thesis

Service and system development still face many problems; ample surveys and research have been published so far describe failures of such service and Information System (IS) projects from which we could learn. The kinds of reasons of projects failure are business reasons, management reasons and technical reasons (McManus and Wood-Harper, 2007).

Table 1-1 provides the key reasons for the failure of 51 projects from the initial 214 projects studied\(^1\) for period of the 1998-2005 across the European Community.

<table>
<thead>
<tr>
<th>Business reasons (N=10)</th>
<th>Management reasons (N=27)</th>
<th>Technical reasons (N=14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business strategy superseded</td>
<td>Ability to adapt to new resource combinations</td>
<td>Inappropriate architecture</td>
</tr>
<tr>
<td>Business processes change (poor alignment)</td>
<td>Differences between management and client</td>
<td>Insufficient reuse of existing technical objects</td>
</tr>
<tr>
<td>Poor requirements management</td>
<td>Insufficient risk management</td>
<td>Inappropriate testing tools</td>
</tr>
<tr>
<td>Business benefits not clearly communicated or</td>
<td>Insufficient end-user management</td>
<td>Inappropriate coding language</td>
</tr>
<tr>
<td>overstated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure of parent company to deliver</td>
<td>Insufficient domain knowledge</td>
<td>Inappropriate technical methodologies</td>
</tr>
</tbody>
</table>

Table 1-1. Key reasons for the failure of 51 projects

\(^1\) The 214 projects were comprised of both public sector and private sector firms that included 63 projects from public sector and 151 projects from private sector.

\(^2\) The original waterfall model proposed by Royce (1970) makes provision for "feedback
Other research was conducted by the Standish Group for the period of 2000 to 2008 as can be seen in Figure 1. The report classifies the findings into three categories, namely, succeeded projects, failed projects and challenged projects. Succeeding projects are those projects which are delivered on time, on budget, and with required features and functions. Projects that are cancelled prior to completion or delivered and never used are listed under failed projects, whereas projects that are late, over budget, and/or delivering less than the required features and functions are classified as challenged projects. The picture (Figure 1-1) also shows a decrease in the project success rate (year: 2008) compared to the previous year’s rate (year: 2006), which has declined from thirty-five percent down to thirty-two percent. Accordingly, the project failure rate is increased within the period of 2006 to 2008, from nineteen percent to twenty-four percent.

![Figure 1-1. Project success rates from the Standish Group (2009)](image)

Statistics related to service and IS projects failure or success rate could be endless and there is no point to report them all here. The statistics referring to challenged and failed projects only shows us no ISD Methods exist that are best in all situations. Often, classic ISD methods are too generic and cannot be followed literally; they need to be adapted to the situation at hand. Situational Method Engineering (SME) aims to resolve the issues and to provide techniques allowing the construction of project-specific methods for a specific situation (Kumar and Welke, 1992; Brinkkemper, 1996).

However, employing situational methods does not mean that organizations could adapt **agilely** to external changes such as faster changing customers’ demands, competition, and political interference (regulation). Existing studies on Situational Method Engineering mostly focus on situational fit of the overall development process while still describing the actual method fragments in terms of procedures and their execution order to develop products (Brinkkemper et al., 1998).
1.2 Problem statement

The successfully applied agile software development process in the software development industry has made significant advances like decreasing time-to-market and addressing rapidly changing customer demands. Approaching this to more general domain, (Hoppenbrouwers et al., 2011) propose a solution to the aforesaid issue with SME by extending existing method engineering practices and defining a framework that supports a more agile way of working for the development of new services. By viewing agility in broader perspective, any business entity may apply the same concept of agility to tackle challenges in the business environment.

Before delving further, first we have to define agility. Qumer and Henderson-Sellers (2007) offer the definition of agility of any entity as “a persistent behavior or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment”. By definition, a business entity is considered agile if the business entity is able to create or adapt a business service efficiently and effectively when changes occur.

(Hoppenbrouwers et al., 2011) try to extend the situational method engineering approach regarding the assembly of method fragments by adding a second dimension, which is agility in operational execution. In addition, the work suggests a particular operationalization of the method engineering approach and process in terms of method fragments, situational factors and assembly rules. The idea behind this is to provide as much freedom as possible by means of declarative method specification that leads to minimal specification.

Likewise, the work introduces the ‘game metaphor’ as a consequence of the declarative approach. Methods can be considered as games, which have clear objectives and a set of rules that need to be complied to (Hoppenbrouwers, 2008, 2009). If we have objectives and rules, it means we also have goals. Indeed, it is goals that are the key concepts in this approach. Goals could be seen as the intentional part of the method specification. Whereas to realize goals we need activities that are the operational part of the method specification and enactment.

With that in mind, the aim of this thesis is to propose a tool or a system that supports method enactment (operationalization of method engineering) of the rule-based method engineering approach. The system should be able to address these high-level requirements:

1. The system has to support a goal-driven approach to method enactment.
2. The system has to operate in declarative manner which allows for minimal specification.
3. The system has to be able to track the progress of method specification and enactment—the intentional and the operational part.
1.3 General software project management

While the purpose of the system is specifically to support rule-base based method enactment, we also think that perhaps the system should also be used as a tool in more general project management for project planning, monitoring and control. What we mean by general projects is simply projects that run not strictly on the basis of the rule-based method engineering approach. In other words, general projects do not involve any method engineering process for Agile Service Development. Even if a project uses agile software development methodology without involving in any rule-based method engineering process, in our case the project is still to be categorized into general project management.

This vision does not shift our focus from our very first objective of the thesis. In fact any consideration of requirements will only be focused on such efforts.

1.4 Organization and content of the thesis

The thesis is structured in six chapters aside from the references and appendixes. The source code of the MediaWiki extension that was solely created to support the system, and Semantic MediaWiki markup templates and forms can be found in appendixes.

Chapter 1, which is this chapter, presents service and information system project statistics about the success and failure rates of projects in public and private sectors as context of the thesis. Later on, the chapter also introduces motivating statements about the problem at hand that needs to be solved.

A formulation of research questions will be included in chapter 2. The chapter is also devoted to the intermediary setup of the project, including an overview of the methodology applied.

Chapter 3 gives the elaborate theory underlying the system’s concepts. The chapter will start with detailed explanation about the rule-based method engineering approach. The remainder of chapter 3 will be allocated to discuss classic project management practices, goal-driven analysis, incremental approach, and declarative workflows.

Chapter 4 addresses architecture, requirements, design, implementation and sample sessions of the system. Chapter 4 forms the core of the thesis, a proof-of-concept prototype of the system for method enactment support.

Next, chapter 5 subjects the system to a user test. The given test cases mimic the real-life projects. There are two test cases. The first one is a general project that falls under general project management practices and the second is a project that takes into account Agile Service Development: the rule-based method engineering approach.

To conclude our work of this thesis, chapter 6 answers the research question and sub questions. Chapter 6 also offers suggestions for future research in the same field.
CHAPTER 2

Methods

Within the Design Science paradigm (Hevner et al., 2004), IT artifacts are broadly defined as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices) and instantiations (implemented and prototype systems). The main goal of our work as discussed before is to develop a system for supporting rule-based method enactment; hence this thesis falls the under design science paradigm.

Hevner et al. propose seven guidelines for design-science research:

1. The product of design-science research is a purposeful IT artifact
2. For a specific problem relevance domain
3. The artifact should have been thoroughly evaluated for its utility, quality and efficacy
4. Provide clear contributions in the areas of the design artifact
5. The artifact itself must be rigorously defined, formally represented and internally consistent
6. Incorporate a search process by means utilizing available means to discover an effective solution to a problem
7. The design-science result must be presented effectively both to a technical audience and to a managerial audience.

Our work follows the given guidelines in design science. It is certainly our aim to create a purposeful IT artifact that can be used to solve a problem within the domain of rule-based method engineering.

2.1 Research Questions

In order to achieve the thesis goal, a main question is addressed. The main research question is:

\textit{Is it possible to build a system supporting rule-based method enactment using MediaWiki (MW) and Semantic MediaWiki (SMW)?}
This question is further derived into more specific sub-questions, which are answered throughout this thesis. Those sub questions are as follows:

- What are rule-based method engineering and its characteristics?
- What are requirements resulting from rule-based method engineering?
- What is a system that fulfills the requirements?
- Why are MediaWiki and Semantic MediaWiki chosen in developing a system?

### 2.2 Methodology

In order to achieve the conclusion of this research, six main steps are taken:

1. Formulate the main research question
2. Study and collect relevant literature or theoretical framework of the system
3. Gather the requirements for the system
4. Design the system
5. Build the system
6. Test and evaluate the system

The thesis is carried out partially involving both literature study and design science. The starting point of the theoretical frameworks is the rule-based method engineering approach (Hoppenbrouwers et al. 2011) from which the research question emerges. From there, we try to escalate to the theory of declarative workflows (van der Aalst et al., 2009 and Voesten, 2009) and the goal-driven approach (van Lamsweerde, 2009).

Following the literature study, gathering the requirements of the system is the next important step. In order to provide a proper requirement specification, a use case driven approach (Kulak and Guiney, 2003) to requirements is employed. The design of the system is in line with the gathered requirements of the system and its information structure of the system is expressed in Object Role Modeling/ORM (Halpin, 1998).

The implementation of the system is done iteratively, which is also in line with the design science paradigm. In the process of developing the system, ideas may emerge along the way. Thereby refinement is inevitable in implementation and is done in iterations.

The last step is to test and evaluate the system by means of test cases. Tests to the system are conducted to obtain practical validation. We try to mimic realistic projects.
CHAPTER 3

Theoretical Frameworks

In this chapter we will develop theoretical frameworks underlying the system. The first theoretical framework is grounded in literature on rule-based method engineering in agile service development, and therefore prior to presenting the framework, we provide a brief overview of the literature on situational method engineering.

The next bit of theoretical framework concerns a goal-oriented or goal-driven approach. It has been mentioned in chapter 1 that one of our high-level requirements is to support goal-driven approach in method specification and enactment, so it would be desirable to have some background on the subject matter. Subsequently, declarative workflow, which supports minimal specification in rule-based method engineering, also becomes our focus.

The last thing that we want to include in our frameworks is ISD project management. This comprises of a classic software development methodology, e.g. waterfall methodology and a modern software development methodology, agile methodology. Those methodologies are the prominent methodologies used in information system development (ISD) projects. They will be involved in the testing of our prototype.

3.1 Agile Service Development: rule-based method engineering

Prior to elaborating more on rule-based method engineering, it is good to have some background knowledge of method engineering, in particular situational method engineering. The center of interest in method engineering (ME) and situational method engineering (SME) is on formalizing the use of methods for system development. Brinkkemper (1996) gives a definition of method engineering as a discipline of developing, customizing, and/or adapting methods, techniques and tools for system development. A major component of method engineering, situational method engineering, covers developing methods for a specific situation by excluding topics such as comparing methods and method knowledge infrastructure (Henderson-Sellers and Ralyté, 2010).

The construction is accomplished by selecting pieces of method—some scholars call it method fragments e.g. (Harmsen et al., 1994, Brinkkemper 1996) and others call it method chunks e.g. (Rolland and Prakash, 1996), and (Plihon et al., 1998)—that have
been created and stored in some method base. Hoppenbrouwers et al. (2011) uses the term method fragment.

The term method fragment was coined by Harmsen et al. (1994) and is defined as “coherent pieces of information system development methods” (Brinkkemper, 1996) or can be regarded as atomic elements of a method. Method fragments once established and derived should be stored in a method base. Figure 3-1 shows process in situational method engineering when method fragments are taken out of method base for a specific project execution, project characteristics must be taken into account in order to help determine which fragments are appropriate or not appropriate given the current requirements.

![Figure 3-1. Engineering a situational method from the elements in the methodbase, taken from Henderson-Sellers and Ralyté (2010).](image)

The work of Hoppenbrouwers et al. (2011), the rule-based method engineering approach, is what becomes the theoretical basis to our work in developing the system. The work introduces the notion that ‘situational’ is not synonymous to ‘agile’. A method is considered to be agile if situational factors (e.g. project characteristics) can be applied to on the fly changes in the method by providing quick responses to novel situational information and short feedback loops applying to the method. The approach follows the configuration process for situational method engineering proposed by Brinkkemper (1996) with an addition of agility in operational execution.

The most recent practice of SME declares that changes in the environment will result in replacing complete method fragments, but Hoppenbrouwers et al. argue that “changes in the environment will not always lead to changes in the executed method but can still influence the operational execution of a specific method fragment”. A particular operationalization of the method engineering approach and process in terms of the selection process of method fragments, situational factors and assembly rules are proposed to realize the idea. Figure 3-2 depicts the idea of this approach.
Method fragments description and identification

Hoppenbrouwers et al. (2011) choose a rule-based, \textit{declarative} approach to the description of method fragments in which declarative description allows for \textit{minimal specification}. The introduction of declarative approach leads to the association of methods with games (Hoppenbrouwers 2008, 2009). When we are playing a game, there are specific rules and objectives that need to be followed. Goals are a central concept in this approach. Goals are to be considered as the \textit{intentional} part of the method enactment.

On the other hand, goals need to be realized. This is when activities come in handy to achieve such realization. Activities are the \textit{operational} part of the method enactment that can be temporally ordered. The goals and activities become our primary focus in developing a system for the method enactment, leading us to the next theoretical framework viz. goal-driven approach and declarative workflows, which will be covered in the following section.

3.2 Goal-driven approach

Van Lamsweerde (2009) explains the central role of goals in the requirement engineering process; we can adopt the same idea as the base of the system’s goal-oriented approach. A goal is defined as a prescriptive statement of \textit{intent} (van Lamsweerde, 2009), which is in line with Hoppenbrouwers et al. (2011) who consider a goal as the intentional part of the method enactment. As implied by the wording ‘prescriptive statement’, goals are declarative. Moreover, a goal can also be classified into types depending on the context of the use and/or classification systems if any.

According to van Lamsweerde, a goal could also have a sub-goal that contributes to it. When the notion of sub-goals is introduced, at the same time we admit the existence of parent (super) goals. Sub-goals and parent goals could be identified by asking two kinds of questions of HOW and WHAT. For instance, let G be a goal already identified. Sub-goals of G are found by asking question ‘how can G be satisfied?’ Whereas parent goals of G are found by asking question ‘why should G be satisfied?’
In order for goals to be achieved, we need goal operationalization. Operationalization refers to the process of mapping leaf goals (sub goals) to operations (activities) ensuring them (van Lamsweerde, 2009). An activity may operationalize one or several (sub) goals underlying it.

The goal-driven approach presented here will be our guideline to implement the system. A further implementation will be presented in Chapter 4.

3.3 Declarative workflows

The rule-based method engineering approach as proposed by Hoppenbrouwers et al. (2011) suggests that a declarative approach is preferred over an imperative approach because a declarative description, which gives as much freedom as possible, allows for minimal specification: planning only what is necessarily to be planned and leaving the rest to a team’s powers of self-organization. Traditional approaches (e.g. imperative approach) tend to use procedural process models to explicitly specify the execution process. According to van der Aalst et al. (2009), the biggest problem with imperative (procedural) approach is flexibility. If processes are defined beforehand and then confronted with changes, those processes tend to be rigid and unable to adapt to changes. This is not what agile service development wants.

We want the system to support a declarative way of working. As previously mentioned, goals are declarative, but what about activities? In this regard, we also want that the operational part (activities) of the method enactment should be declarative. The good example of this is the work of Voesten (2009). Voesten says that declarative means “…declaring what needs to be done in the process (the activities or tasks) and declaring which rules or constraints…” The control-flow constraints between activities can be seen as pre and post conditions for an activity. This is also in line with what van Lamsweerde (2009) suggests by characterizing every operation (activity) by a pair of conditions called the domain pre-condition and the domain post-condition.

To this end both the concept of declarative and control-flow (temporal ordering) will be taken into account in developing the system. A more elaborate description of the two concepts will be presented later in section 4.4.2.

3.4 The need for an incremental approach

The incremental approach to method engineering has been subject research by several scholars e.g. Krzanik and Simila (1997), and Tolvanen (1998). But it seems no clear definition had been published until van de Weerd (2009) defined “a method increment as: a method adaptation, in order to improve the overall performance of a method.” Adaptation can apply to the insertion, editing or removing of method fragments. Hence, the method increment plays an important part also in the rule-based method engineering and we need to address this specifically for the system that we develop.
The work of van Lamsweerde (2009) advocates operationalization as an incremental process. Van Lamsweerde says, “…a new goal might appear while building an operation model…” and “The operationalization of this new goal may require new associated operations or a new associated set of rules…” With that in mind, adding incremental capability to the system is a paramount requirement, realized by means of enabling and tracking changes (adding, deleting, and editing goals/activities) for each of increments throughout project execution.

The need for an incremental approach leads us to the need for iteration. Iteration involves re-adding, deleting and editing of goals/activities within the system throughout the project execution. Hoppenbrouwers (2009) distinguishes three types of iteration:

- Triggered: an event causes a new iteration
- Ad hoc: user instantiated, e.g. by change of ideas
- Planned: iteration scheduled in the temporal plan

In the light of the above, the system to be built has to be able to cope with the increments of method engineering and the iteration process. Fortunately, MediaWiki with its extension Semantic MediaWiki is able to support this by means of its built-in functionalities and a custom made extension that is tailored for our incremental need.

3.5 Project management

In order to understand project management, we first must begin with the definition of a project. Kerzner (2003) considers a project as any series of activities and tasks that:

- Have a specific objective to be completed within certain specifications
- Have defined start and end dates
- Have funding limits (if applicable)
- Consume human and nonhuman resource (e.g. money, people, equipment)
- Are multifunctional (i.e., cut across several functional lines)

On the other hand, project management involved project planning and project monitoring and includes such items as (Kerzner, 2003):

- Project planning:
  - Definition of work requirements
  - Definition or resource needed
- Project monitoring:
  - Tracking progress
  - Making adjustments

With that in mind, we believe that the system could be also used for project management activities in ISD development. The following two sections are allocated to
a brief discussion of traditional project management versus agile project management in ISD. This theoretical framework is later useful to gain insight into test cases in chapter 5.

3.5.1 Traditional project management

According to Hass (2007), traditional project management involves:

- Very disciplined and deliberate planning and control methods.
- Distinct project life cycle phases.
- Project tasks completed one after another in an orderly sequence.

There are times when the requirements for a problem are well understood—when work flows from communication through deployment in a reasonably linear fashion. The waterfall model, the classic life cycle, is perhaps the first and most well known approach to software development of traditional project management. The waterfall model divides the project development into consequent phases that have to be executed sequentially one after the other.

This model assumes that the process of software development can be divided in five phases viz. communication (customer specification of requirements), planning, modeling, construction, and deployment (Pressman, 2010). The waterfall life cycle is summarized in Figure 3-3. This theoretical framework, waterfall model of traditional project management, gives us a better understanding of the classical methodology, which provides valuable insight into practical evaluation of the system.

![Figure 3-3. Waterfall life cycle](image)

3.5.2 Agile project management

Traditional project management is not good enough when is faced with fast changing external environments that impact on objective and potential solutions of a project. Sometime traditional project management can be perceived as unnecessary slow and bureaucratic. To overcome these issues, this is where agile project management comes

---

2 The original waterfall model proposed by Royce (1970) makes provision for "feedback loops", but most majority organizations that use this model consider it as if it were strictly linear.
in handy. Agile project management allows iterative and incremental approaches to project implementation.

Hass (2007) describes conditions in which agile methods are used, and these conditions are as follows:

- Project value is clear;
- The customer actively participates throughout the project;
- The customer, designers and developer are co-located;
- Incremental feature-driven development is possible; and
- Visual documentation (cards on the wall vs. formal documentation) is acceptable.

The “Manifesto for Agile Software Development”\(^3\) states:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following plan

That is, while there is value in the items on the right, we value the items on the left more.

Scrum (Takeuchi and Nonaka, 1986; DeGrace and Stahl, 1990; Sutherland, 2004) is one of methods that is referred to as agile methodology. Pressman (2010) says “Scrum principles are consistent with the agile manifesto and are used to guide development activities within a process that incorporates the following framework activities: requirements, analysis, design, evolution and delivery.” Work tasks that occur within a process pattern are called a *sprint*. The work conducted within a sprint and the number of sprints required will vary depending on the project complexity. Figure 3-4 depicts the Scrum model.

---

\(^3\) Agile Alliance, http://www.agilemanifesto.org
3.6 Integrated theoretical frameworks

We have described so far the underlying frameworks of the system. In order to have a better insight into the relevant inter-relationships between the concepts, we present those inter-connected concepts in Object Role Modeling/ORM (Nijssen and Halpin, 1989; Halpin, 1998) diagram (Figure 3-5).

This ORM diagram does not represent how the information structure of the system is depicted (for the information structure of the system see section 4.4.1). The system in the diagram is directly or indirectly connected to the aforesaid theoretical frameworks. We argue that there are two types of ISD project that can be supported by the system. The first type of the projects, which becomes our main objective, is the method enactment type of project. This project employs the method engineering process in its execution. Moreover, this type of project as the result of the suggested declarative workflows in the rule-based method engineering should be accomplished in the incremental approach, which leads to iterations.

The other type of the projects is general ISD projects in which it does not employ any method engineering process or whatsoever. This last type of the projects uses classical ISD methodology (i.e., waterfall life cycle) or agile development methodology (i.e., Scrum) to accomplish its objective.

Goals are to be considered as the intentional part of the method enactment. In order for goals to be realized, they need activity. Within the concept of the rule-based method engineering (Hoppenbrouwers et al., 2011), activities, which realize goals, are to be considered as the operational part of the method enactment.

There is one thing that needs to be noted; we choose not to use any uniqueness constraints in the ORM diagram because it is irrelevant to the context of our universe of

---

4 Taken from http://en.wikipedia.org/wiki/Scrum_(development)
The total role constraints in the diagram mean that an entity must fulfill a particular role, e.g., the system must be able to support the goal-driven approach of method enactment (as this is our primary objective) and may be able to support the general ISD projects.

**Figure 3-5.** An ORM diagram of the relationships between concepts and the system.
Chapter 3 should give us a rich profusion of conceptualization to proceed with the development of the system, which we dubbed the Method Management System (MMS). Within chapter 4, we elaborate the details of the system. It begins with a brief overview, followed by the architecture of the system, system requirements, design, and implementation, and ends with example sessions of the working system.

4.1 Overview of the system

The Method Management System aims to provide support for rule-based method enactment (see chapter 3). Rule-based method engineering employs a goal-driven approach to the selection of method fragments. In order for goals to be realized, they need activities. Goals are considered to reside in the intentional part of rule-based method enactment, while activities reside in the operational part.

Furthermore, the system is mandated to be able to keep up with every incremental change during project execution. These incremental changes may be associated with method increments (van de Weerd, 2009) or with realization of goals (i.e., goals are achieved).

4.2 Architecture of the system

Within the research sector and industry, the notion of software/system architecture has become a generally recognized concept. According to Bosch (2004), better control over design, development and evolution of large and increasingly dynamic software systems spring from the importance of stressing the components and their connectors of a software system. Thereby, in this section we will explain the underlying architecture that supports the method management system.

The method management system is built on the top of MediaWiki and Semantic MediaWiki (figure 4-1). Each of these components we will explain concisely in the following sections.
4.2.1 MediaWiki

MediaWiki is web-based wiki software, which is developed by the Wikimedia Foundation⁵ and used to run all of its projects: Wikipedia⁶, Wiktionary⁷, and Wikinews⁸. In 2002, the first version of the software was deployed to serve the needs of the free content encyclopedia. Many companies have deployed it since then as a content management system for internal knowledge management⁹.

MediaWiki provides a rich core feature set and a mechanism to attach extensions to provide additional functionality beside its function for collaboration on the web. The extensions and page display/manipulation are two of many centerpieces of MediaWiki that will be elaborated here.

Extensions

A MediaWiki extension allows MediaWiki to be made more advanced and tailored according to purposes. The extensions can vary greatly in complexity. The MediaWiki extensions are categorized into parser functions, academic and encyclopedia-related

---

⁵ For more information http://wikimediafoundation.org/
⁶ http://www.wikipedia.org/
⁷ http://www.wiktionary.org/
⁸ http://www.wikinews.org/
data display, integration, combating spam, and searches/queries/data processing/aggregation. The one that becomes our focus here is parser functions of extensions.

Among the most used extensions is a parser function extension. A parser function extension allows different content to be rendered based on the result of conditional statements. These conditional statements can perform functions such as evaluating whether a parameter is empty, comparing two strings, evaluating mathematical expressions and returning string values according to particular parameters. The prominent examples of this type of extension are ParserFunctions, StringFunctions and VariablesExtension.

Page display and manipulation

MediaWiki uses an extensible lightweight wiki markup designed to be easier to use and learn than HTML. A consensus seems to have been reached that Wikicode requires context sensitive grammar rules (Curry, Trotman and Albert, 2010), as a result efforts to create a formal markup spec is incomplete. The following table (Table 4-1) illustrates the differences between wiki markup and HTML with the same rendered output.

<table>
<thead>
<tr>
<th>MediaWiki syntax</th>
<th>Equivalent HTML</th>
<th>Rendered output</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Take some more [[tea]],&quot; the March Hare said to Alice, very earnestly.</td>
<td>&lt;p&gt;&quot;Take some more &lt;a href=&quot;/wiki/Tea&quot; title=&quot;Tea&quot;&gt;tea&lt;/a&gt;,&quot; the March Hare said to Alice, very earnestly.&lt;/p&gt;</td>
<td>&quot;Take some more tea,&quot; the March Hare said to Alice, very earnestly.</td>
</tr>
<tr>
<td>&quot;I've had nothing yet,&quot; Alice replied in an offended tone: &quot;so I can't take 'more'.&quot;</td>
<td>&lt;p&gt;&quot;I've had nothing yet,&quot; Alice replied in an offended tone: &quot;so I can't take &lt;b&gt;more&lt;/b&gt;.&quot;&lt;/p&gt;</td>
<td>&quot;I've had nothing yet,&quot; Alice replied in an offended tone: &quot;so I can't &lt;b&gt;take&lt;/b&gt; more.&quot;</td>
</tr>
</tbody>
</table>

Table 4-1. Side-by-side comparison between wiki markup and HTML.\(^\text{10}\)

\(^\text{10}\) Quotation is taken from Alice's Adventures in Wonderland by Lewis Carroll.
Another really nice feature that MediaWiki has is a Recent Changes feature. This feature provides a list of recent edits to the wiki. The list contains basic information about those edits such as the edit summary, the page edited, as well as the modification dates. But there is a trade-off, in a very active wiki; sometimes it is hard to track changes manually, because so many edits occur.

4.2.2 Semantic MediaWiki

While MediaWiki has become a popular tool for collaboration on the web and as we said previously many vibrant companies employ it to exchange knowledge, MediaWiki alone lacks in query capabilities. Using MediaWiki per se means reading articles and gathering information distributed across multiple articles. This may require a huge amount of time if processes are manual, like in the case of requesting a list of cities in Indonesia that have a population more than 250,000. Even though its data is quite structured for humans, there is no way that such data would be understood by machines. That is why the Semantic MediaWiki (SMW) came in the first place.

SMW is a semantically enhanced wiki engine that enables users to annotate the wiki's contents with explicit machine-readable information (Krötzsch et al., 2006). Using this explicit machine-readable information, Semantic MediaWiki addresses core issues of today's wikis:

- **Consistency of content**: same information often occurs on different pages. Changing a master page should also reflect in the other pages.
- **Accessing knowledge**: in wikis, which have thousands of pages, finding and comparing information from different pages is like finding a needle in a haystack, hard but not impossible.
- **Reusing knowledge**: information in many wikis is only accessible in a browser or some similar applications.

Semantic MediaWiki is available as an extension of MediaWiki. The interaction between SMW and MediaWiki is based on MediaWiki's extension method: Semantic MediaWiki is registered for certain events or requests, later on MediaWiki calls SMW functions when needed.

**Inline queries**

Semantic MediaWiki incorporates a query language that allows access to the wiki's database. There are two options in the way to use this query language: (1) by directly querying the wiki and (2) by adding the answer to a page. The latter is what we call an inline query. An inline query enables us to add dynamically created lists or tables to a page; therefore it delivers up-to-date information for readers. The basic way of writing an inline query is to use parser function `#ask.`
The syntax of Semantic MediaWiki’s query language is closely associated to wiki text, where on the other hand its semantics resemble certain class expressions in the Web Ontology Language Description Logic (OWL DL)\textsuperscript{11}. Any query condition is encoded as a query atom whose syntax is similar to that of SMW’s annotations: for example, \texttt{[[City located in::Indonesia]]}. SMW’s inline queries not only work in single fixed values, but also in ranges of values and even specify nested query expressions.

SMW uses some specific symbols to control the structure of queries: || as disjunction operator (like OR in SQL), + as empty condition which matches everything, and <, >, != respectively. The following is an example of an inline query: \{\{#ask: [[Category:City]] [[Located in::Indonesia || Netherlands]] \}\}. The aforesaid query gives the result of all cities that are located in Indonesia or the Netherlands.

Like OWL DL, Semantic MediaWiki’s query language forbids cross-references between parts of queries due to unsupported explicit variables. For example, it is not possible to query for all people who died in the city they were born in (Krötzsch et al., 2006). The constraint makes query answering tractable which Krötzsch et al. (2006) believe essential for Semantic MediaWiki’s usage in very large wikis.

Properties

Users can add necessary semantic data into MediaWiki by adding annotations to the wiki text using a special markup. Every article corresponds to exactly one ontological element and every annotation in an article describes some statements vis-à-vis this single element. If information is reused in different places, users must still be able to recognize where the information originated, this is the cruciality of locality. All annotations refer to the abstract concept represented by a page, not to the HTML document.

Expressing binary relationships between one semantic entity—as represented by a wiki page—and other such entities or data values is achieved by using properties in Semantic MediaWiki. Properties are like variables in programming languages, so depending on topics of interest SMW’s users have full control over the set of available properties.

MediaWiki does not have a general mechanism for assigning property values to pages. Hyperlinks are a prominent example of binary relations in the wiki text. Each hyperlink establishes a relationship between two pages, without specifying the kind of relationship. Semantic MediaWiki allows hyperlinks to be characterized by properties, for example the target link becomes the value of a user-provided property. But SMW does not only take other wiki pages as values, there are other available types of properties that can be used.

\textsuperscript{11} http://www.w3.org/TR/owl-guide/
Data types

Semantic MediaWiki allows us to use a number of data types that can be used with properties. Each type has its characteristic in processing user input and displaying data as the output. Like properties, (data) types have dedicated pages within the wiki by creating a link to the particular page in every type declaration. Following is the list of data types in SMW\textsuperscript{12}:

- Page: link to a page in MediaWiki (default)
- String: text strings that are not longer than 255 characters
- Number: integer and decimal number
- Boolean: giving the value to a property to true/false, also yes/no or 1/0
- Date: specifies particular points in time
- Text: similar to the String type with unlimited length
- Code: similar to the Text type but with additional precautions to preserve special formatting as used for technical text
- Temperature: variation of Number type that supports units of temperature
- Telephone number: validates and stores international telephone number based on the RFC 3966\textsuperscript{13}
- Record: used for compound property values
- URL: used to display an external link to its URL object
- Email: display an e-mail address with \texttt{mailto:} link
- Annotation URI: like URL type but with some technical differences in Semantic MediaWiki’s RDF (Resource Description Framework) export

As previously mentioned, in evaluating properties data types are very crucial. The chief reasons are (a) data types determines how systems should handle the given values, (b) in order to understand which values have the same meaning data types are employed, and (c) data types have some specific behavior as described above.

4.3 Requirements

The requirement process is one of the most important activities in software development. Practices have shown us that one of the success criteria when developing computer software is the completeness of requirements (the Standish Group, 1994). Therefore, within this section we describe the requirements of the system.

The requirements explained here are functional requirements based on a use-case-driven approach. What we mean by functional requirements is what users need for the system to work and indeed, functional requirements are solely functions and features of the system (Kulak and Guiney, 2003). We adopt use cases to the

\textsuperscript{12} \url{http://semantic-mediawiki.org/wiki/Help:Properties_and_types}
\textsuperscript{13} \url{http://www.ietf.org/rfc/rfc3966.txt}
requirement process because use cases are acknowledged as a suitable approach that enhances the quality of the deliverables resulting from the requirement engineering process (Jacobson and Christerson, 1995).

4.3.1 Use case diagram

The use case diagram of the method management system below (Figure 4-2) depicts the relationship between an actor and use cases, and between a use case and another use case. The actor as described earlier (see section 4.1 “Overview of the system”) is obviously a project manager.

The system consists of thirteen different use cases with the majority of the tasks involving creating, editing and viewing. There are four use cases coping with the creation of project, goal, activity and person: Set Project, Set Goal, Set Activity and Set Person respectively. Other four use cases are employed for altering: Edit Project, Edit Goal, Edit Activity and Edit Person, while viewing a project, goal and an activity is handled by View Project, View Goal and View Activity respectively. The last two use cases, View Goal Tree and Set Temporal Rule, are used for viewing a goal tree within a particular project and adding temporal rules respectively.

In the beginning we thought Set Project use case was not necessary, since we are going to test the system and mimic one project only. Later we decided to have the use case Set Project, because we need to have a mechanism for tagging goals and activities that belong to a particular project when searching for goals. Some readers perhaps may wonder about a use case for handling deletion of goals, activities, etc., where is the use case for such interaction between the actor and the system? The answer is we do not need to stipulate explicitly the deletion process in a form of use case, in view of the fact that the Method Management System is built on the top of MediaWiki that has the feature for deletion. This is also the same reason why we do not include the log-on process to the use case diagram. For further detailed description of all use cases, we explain it on the next section (see section 4.3.2 “Use cases”).
Figure 4-2. A Use Case Diagram describing the interaction between “Project Manager” and the system

4.3.2 Use cases

This section explains above use cases using a detailed template. The template is called a use case template. Although use cases are part of Unified Modeling Language (UML), which was the result of collaborations between three famous object-oriented methodologists Booch, Rumbaugh, and Jacobson (1999), there is no standard template for writing use cases. Some scholars have proposed their own standard for a use case template, such as Cockburn (1998), Coleman (1998), and Kulak and Guiney (2003). For the purpose of this thesis, we use the use case template as proposed by Kulak and Guiney (2003) with minor modification to it.
**Use Case Name:** Set Project

**Description:** Create a project along with its details

**Actors**
- Project manager

**Triggers:**
There is a new project, which is going to be executed.

**Basic Course of Events:**
1. The actor navigates to the create project area.
2. The actor enters the information for the upcoming project, as follows:
   - Project’s name
   - Description
   - Planned start date
   - Planned finish date
   - Purpose
   - Objectives
   - Success criteria
3. When satisfied with the data entry, the actor commits the changes.
4. The system validates data according to the data entry rules, for instance: validating dates.
5. The system stores the data.

**Alternative Paths:** None

**Exception Paths:** None

**Table 4-2. Set Project use case.**

---

**Use Case Name:** Set Goal

**Actors**
- Project manager

**Triggers:**
The actor wants to add a goal in order to accomplish a project.

**Description:**
A goal that relate to a specific project needs to be recorded. The goal means for the project to be succeeded.

**Basic Course of Events:**
1. The actor navigates to the create goal area.
2. The actor enters the following identifying details of the goal:
a. Goal name  
b. Description  
c. Project’s name  
d. Top goal indicator <yes, no>  
e. Urgency <required (default), nice-to-have>  
f. Goal type  
g. Super goal(s)  
h. Sister goals(s)  
i. Goal status <not achieved (default), pending, done>  
j. Achievement criterion  
k. Deadline

3. When satisfied with the data, the actor commits changes.  
4. The system validates the data entered according to data entry rules.  
5. The system stores the data entered.

<table>
<thead>
<tr>
<th>Alternative Paths:</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception Paths:</td>
<td>4. If the actor left the top goal indicator not selected, the system alerts the actor that the field cannot be blank.</td>
</tr>
</tbody>
</table>

**Table 4-3.** Set Goal use case.

<table>
<thead>
<tr>
<th>Use Case Name:</th>
<th>Set Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>Project manager</td>
</tr>
<tr>
<td><strong>Triggers:</strong></td>
<td>The actor wants to specify an activity that corresponds to a particular goal.</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>A goal is entered that can belong to one or more goals.</td>
</tr>
</tbody>
</table>
| **Basic Course of Events:** | 1. The actor navigates to the create activity area.  
2. The actor enters the following information:  
   a. Activity name  
   b. Description  
   c. Goal(s)  
   d. Date and due time  
   e. Location |
The actor commits changes.
4. The system validates the entered information.
5. The system stores changes.

<table>
<thead>
<tr>
<th>Alternative Paths:</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception Paths:</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 4-4. *Set Activity* use case.

<table>
<thead>
<tr>
<th>Use Case Name:</th>
<th>Set Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Project manager</td>
</tr>
<tr>
<td>Triggers:</td>
<td>The actor needs to allocate people to the project.</td>
</tr>
<tr>
<td>Description:</td>
<td>A person involved in a project is recorded.</td>
</tr>
<tr>
<td>Basic Course of Events:</td>
<td></td>
</tr>
<tr>
<td>1. The actor navigates to the create person area.</td>
<td></td>
</tr>
<tr>
<td>2. The actor enters the following details:</td>
<td></td>
</tr>
<tr>
<td>a. Person name</td>
<td></td>
</tr>
<tr>
<td>b. Role in organization</td>
<td></td>
</tr>
<tr>
<td>c. Email address</td>
<td></td>
</tr>
<tr>
<td>d. Phone number(s)</td>
<td></td>
</tr>
<tr>
<td>3. When satisfied with the data entry, the actor commits the changes.</td>
<td></td>
</tr>
<tr>
<td>4. The system validates the information entered.</td>
<td></td>
</tr>
<tr>
<td>5. The system stores the information.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alternative Paths:</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception Paths:</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 4-5. *Set Person* use case.

<table>
<thead>
<tr>
<th>Use Case Name:</th>
<th>Edit Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Project manager</td>
</tr>
<tr>
<td>Triggers:</td>
<td>The actor due to change of thought or condition</td>
</tr>
</tbody>
</table>
wants to alter a project information.

**Description:**
The actor changes project information.

**Basic Course of Events:**
1. The actor navigates to the edit project area.
2. The actor changes the already inserted information.
3. The actor commits changes.
4. The system validates changes.
5. The System stores changes.

**Alternative Paths:**
3. The actor could commit changes with the option: choosing a minor edit if the actor does not want the changes added to a log record for history purpose.

**Exception Paths:**
None

**Table 4-6. Edit Project use case.**

**Use Case Name:**
Edit Goal

**Actors:**
Project Manager

**Triggers:**
The actor due to change of thought or condition wants to alter a goal information.

**Description:**
The actor changes goal information.

**Basic Course of Events:**
1. The actor navigates to the edit goal area.
2. The actor changes the already inserted information.
3. The actor commits changes.
4. The system validates changes.
5. The System stores changes.

**Alternative Paths:**
3. The actor could commit changes with the option: choosing a minor edit if the actor does not want the changes added to a log record for history purpose.

**Exception Paths:**
4. If the actor altered the goal field blank, the system alerts the actor that the field cannot be blank.

**Table 4-7. Edit Goal use case.**
### Use Case Name: Edit Activity

**Actors:** Project manager

**Triggers:** The actor is in need to change some information regarding an activity.

**Description:** The actor makes changes to activity information.

**Basic Course of Events:**
1. The actor navigates to the edit activity area.
2. The actor changes the already inserted information.
3. The actor commits changes.
4. The system validates changes.
5. The System stores changes.

**Alternative Paths:**
3. The actor could commit changes with the option: choosing a minor edit if the actor does not want the changes added to a log record for history purpose.

**Exception Paths:** None

Table 4-8. *Edit Activity* use case.

---

### Use Case Name: Edit Person

**Actors:** Project manager

**Triggers:** The actor wants to change some information on a person involved in a project.

**Description:** The actor changes person information due to a particular condition or circumstance.

**Basic Course of Events:**
1. The actor navigates to the edit person area.
2. The actor changes the already inserted information.
3. The actor commits changes.
4. The system validates changes.
5. The System stores changes.

**Alternative Paths:**
3. The actor could commit changes with the option: choosing a minor edit if the actor does not want the changes added to a log record for history purpose.
**Table 4-9. Edit Person use case.**

| Exception Paths: | None |

Use Case Name: View Goal Tree  
Actors: Project manager  
Triggers: The actor is in need to view a goal representation of a tree like directed graph.  
Description: Render a goal tree for a particular project.  
Basic Course of Events:  
1. The actor navigates to the view goal tree area.  
2. The actor specifies a project name of which goals exist  
3. The system responds by displaying a goal tree.  
Alternative Paths: None  
Exception Paths: None

**Table 4-10. View Goal Tree use case.**

| Use Case Name: | View Goal |

Actors: Project manager  
Triggers: The actor would like to view a goal  
Description: Displaying entered information for a specific goal.  
Basic Course of Events:  
1. The actor navigates to the main area.  
2. The actor chooses a project, which contains a goal.  
3. The system displays project’s information.  
4. The actor selects a goal.  
5. The system displays goal’s details.  
Alternative Paths:  
1. The actor navigates to the search area.  
Exception Paths: None

**Table 4-11. View Goal use case.**

| Alternative Paths: | None |

Exception Paths: None
**Table 4-12. View Project use case.**

<table>
<thead>
<tr>
<th>Use Case Name:</th>
<th>View Project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>Project manager</td>
</tr>
<tr>
<td><strong>Triggers:</strong></td>
<td>The actor wants to view information from a project.</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>Viewing the information of a project.</td>
</tr>
</tbody>
</table>
| **Basic Course of Events:** | 1. The actor navigates to the main area.  
2. The actor chooses a project.  
3. The system displays the project information along with goals that belong to the project with a tree like representation. |
| **Alternative Paths:** | None |
| **Exception Paths:** | None |

**Table 4-13. View Activity use case.**

<table>
<thead>
<tr>
<th>Use Case Name:</th>
<th>View Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>Project manager</td>
</tr>
<tr>
<td><strong>Triggers:</strong></td>
<td>The actor wants to check the information of an activity.</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>Viewing the information of the activity.</td>
</tr>
</tbody>
</table>
| **Basic Course of Events:** | 1. The actor navigates to the main area.  
2. The actor chooses a project, which contains a goal.  
3. The system displays project’s information.  
4. The actor selects an activity within a project timeline.  
5. The system displays activity’s details. |
| **Alternative Paths:** | None |
| **Exception Paths:** | None |
### Use Case Name: Set Temporal Rule

**Actors:** Project manager

**Triggers:** The actor wants to add temporal rules to an activity.

**Description:** Adding temporal rules of the activity.

**Basic Course of Events:**

1. The actor navigates to the set temporal rule area.
2. The actor chooses the following information:
   a. First activity
   b. Temporal rule
   c. Second activity
3. When satisfied with the data entry, the actor commits the changes.
4. The system stores the rule.

**Alternative Paths:** None

**Exception Paths:** None

---

**Table 4-14. Set Temporal Rule use case.**

---

### 4.3.3 Scenarios

Scenarios are required as a tool to focus on the detailed interactions, between an actor and a system. The reason is that use cases cannot tell the whole story and they are not highly detailed (Kulak and Guiney, 2003). A scenario is an instance of a use case and represents a single path through the use case. Ergo, we can construct a scenario for the main flow through the use case, and other scenarios for each possible variation of flow in the use case (for instance: alternative paths and exception paths).

We will not describe scenarios for each of the use cases here; instead we will describe mandatory use cases only for thirteen use cases within the system in order to test the validity of the use cases early in the lifecycle. We start with the Set Project use case accompanied with a narrative to portray production data.

---

**Narrative:** A group of five people in whom one of them is a project manager has been assigned the task to develop a scientific program called “ResAna”. The team decided to employ Agile software development methodology to achieve its objective. There will be three sprints through out the project.
The project is planned to start in 1 September 2010 and ended in 30 January 2011. The objective of the project is to build a prototype program of Java loop-bound function inference and integrate it to an Eclipse’s plugin.

Basic course of events:

1. The project manager navigates to the create project area.
2. The project manager enters the details of the project:
   a. Project’s name: ResAna
   b. Description: a tool to infer loop-bound functions for java programs
   c. Planned start date: 01/09/2010
   d. Planned finish date: 30/01/2011
   e. Purpose: -
   f. Objectives: to build a prototype program of Java loop-bound function inference and integrate it to an Eclipse’s plugin
   g. Success criteria: -
3. After looking through the entered data, the project manager is satisfied and saves it.
4. The system responds by checking that the date entered is correct and according to the default format “dd/mm/yyyy”.
5. The system stores the entered data.

Table 4-15. Set Project use case scenario.

Narrative: The project manager wants to add a goal called “Sprints are achieved”. He urges the team that the deadline for this goal is 7 September 2010 and classifies the goal as “method goal”. The goal is not a top goal and does not have any super goal or sister goal at the moment. The purpose of this goal is to hold an activities related to the initialization of the project, such as kick off meeting, etc.

Basic course of events:

1. The project manager navigates to the create goal area.
2. The project manager enters the following information regarding the goal:
   a. Goal name: Sprint are achieved
   b. Description: there are 3 sprints through out the project
   c. Project’s name: ResAna
   d. Top goal indicator: no
   e. Urgency: required
   f. Goal type: method goal
   g. Super goal(s): -
   h. Sister goal(s): -
i. Goal status: not achieved
j. Achievement criterion: -
k. Deadline: 07/09/2010
l. Dependency: -

3. The project manager takes a look at the data entered and commits changes.
4. The system validates the entered data and starts with checking the mandatory field like: top goal indicator, which should not be blank, and the date in the dateline field, should comply with the default format that is “dd/mm/yyyy”.
5. The system saves the data.

Exception paths:

4. The project manager forgot to enter the top goal indicator. The system displays a warning and informing the project manager that the top goal indicator should not be blank.

Table 4-16. Set Goal use case scenario.

Table 4-17. Set Activity use case scenario.
The three tables above (see Table 4-15, Table 4-16 and Table 4-17) provide a further illustration how the system interacts with the actor and vice versa with the help of the aforesaid narratives.

4.4 Design

The next thing that is vital to the software development is a design process. According to Pressman (2010) design is where stakeholder requirements, business needs and technical considerations all come together in the formulation of a product or system. The creator of Lotus 1-2-3, Mitch Kapor, presented a “software design manifesto” in Dr. Dobbs Journal. He said:

“What is design? It’s where you stand with a foot in two worlds—the world of technology and the world of people and human purposes—and you try to bring the two together...”

Following the explicit specification of requirements in the section 4.3, we present within this section a design consideration, which projects to the foregoing requirements. Of course we might not be able to describe all design considerations, thereby we choose to elaborate on the more noteworthy design decisions.

4.4.1 Information structure

The underlying information structure in the system is easy to describe using a conceptual model, to be more specific conceptual data modeling. The conceptual data model that we use is based on object role modeling/ORM (Nijssen and Halpin, 1989; Halpin, 1998). The complete overview of the information structure is depicted in Figure 4-4.

The most fundamental structures of the system consist of a goal structure and an activity structure. We will describe those structures in more detail in the following sections.

Goal structure

A goal has a description, goal type (e.g. method goal), goal status (not-achieved, pending or done), deadline, indicator of top goal (yes or no), achievement criteria if any, activity, sister goal and super goal. All properties are to be set by users.

A goal can play a role as a super goal and/or sister goal of other goals, together forming a hierarchy of leveled goals which can be portrayed as a tree. As to the case of super goal and sister goal, a goal has ring constraints: irreflexive (ir) and symmetric (sym) respectively. The former means that the roles cannot be filled by the same instance of an object, for instance a super goal of a goal cannot be the goal itself. The symmetric relation means if a relationship exists between two instances of an object,
the same relationship exists in the opposite direction (e.g. if goal A has a sister goal: goal B, then goal B has goal A as its sister goal too).

If users specify important properties of goals (e.g. a super goal, sister goals), the system is designed to represent those relationships within a form of a tree, or in this case it is similar to a directed graph. To illustrate how it works, let's assume we have 'goal A' as a top goal for 'goal B' and 'goal C'. We also have 'goal D' which has 'goal C' as its super goal. Furthermore, 'goal B' and 'goal C' are having a symmetric relationship of sister goal. These relationships are best illustrated it using a tree like directed graph (see Figure 4-3).

![Figure 4-3. A tree representation of goals.](image)

The picture above (Figure 4-3) shows the tree like directed graph of the example. A solid arrow means a connection from one goal (tail) to another goal (head) indicating that the tail has a super goal, the head. In the other hand, a dotted arrow means a symmetric relationship which in this case having a sister goal.

**Activity structure**

In the other hand, an activity has a description, a goal to which it is allocated, location, person responsible to accomplish the activity, due date/time, tool and information used. Also to comprise the notion of repeated activities, an activity could be set to recur at some interval (daily, weekly, or monthly).

In principle, one activity is linked to exactly one goal, but this would introduce doubtful obligation to define a specific goal for each and every activity and this does not seem to work in practice: it should be possible to plan activities without making their
goal explicit. So in the base of that, we allow the linking of goals and activities on an \( n:m \) basis.

In order to avoid possible confusion between a goal and an activity, van Lamsweerde (2009) suggests the use of the past participle of a suggestive verb for the goal’s name to refer to target condition that the goals prescribes, for instance ‘SprintsAchieved’. On
the other hand, he suggests the use infinitive tense for the name of corresponding activity, for instance ‘AchieveSprints’.

While this terminology tip is good for avoiding confusion, however it is not suffice in our case. We propose using passive mode of present tense to express the goal’s name, for example ‘Sprints are achieved’, which literally says that the desired condition is for sprints to be achieved. In addition, we prefer for the activity’s name not to be concatenated into a single word, for instance ‘Do sprint 1’.

4.4.2 Declarative workflows and temporal ordering

So far we have explained the concept of declarative workflows (van der Aalst et al., 2009) by incorporating temporal ordering. Temporal ordering allows us to express ordering relationships between activities such as “Do activity A before activity B”. Incorporating temporal ordering relationships in the system, which tries to show the operational part of rule-based method enactment, is of paramount importance. According to Voesten (2009), the basic set of temporal operators consists of “before”, “after” and “immediately before”. The third operator results in a very rare ordering, but for the purpose of completeness inclusion of the third rule is necessary to help gain insight into when testing the system.

Note that the introduction of constraints or operators does not enforce any actions within the flow of the system. It is purely information. Admittedly, the temporal ordering might suggest otherwise, but the idea behind temporal ordering is to describe the flow of activities within a project, in practice not to steer users in predetermined directions. Figure 4-5 expresses temporal ordering in ORM diagram. The codes, ‘bf’, ‘af’ and ‘ib’, represent “before”, “after” and “immediately before” respectively.

![Figure 4-5. Temporal ordering expressed in ORM diagram.](image)
4.5 Implementation

As first presented in section 4.2: architecture of the system, the challenge is to incorporate MediaWiki and Semantic MediaWiki combined with the requirements to have a working system for method management. But given that the nature of editing pages in MediaWiki is that it is done by using a single text box containing the editable text of the current page, the given requirements in section 4.3 require the system to have forms so that users would easily input relevant information without having to learn about wiki markup or semantic annotations.

In order to do so, we used some third-party extensions and also made a specific extension to support the purposes. Those principal extensions are Semantic Result Formats\(^{14}\), Semantic Forms\(^{15}\) and its additional input types called Semantic Forms Inputs\(^{16}\), while the self-made extension is called History of Page\(^{17}\). We will try to briefly explain the use of those extensions in the next section.

Extensions

Let us have a look to the first extension on the list, Semantic Result Format. In a nutshell, Semantic Result Format gives the possibility for us to have further result formats for Semantic MediaWiki’s inline queries or in other words, it provides alternative renderings/visualizations for SMW’s queries besides the plain table or list layout. The formats that are being used in our system are graph and timeline.

Semantic Forms on the other hand allow users to add, edit, and query data using forms. They are heavily tied in with Semantic MediaWiki and are meant to be used for structured data that has semantic markup. There is something that needs to be noted, the Semantic Forms enforce the use of templates in creating semantic data and does not support direct semantic markup in data pages. All semantic markups are intended to be stored indirectly through templates. Additionally, Semantic Forms Inputs enable us to collect further input types, but in this case we use this extension for the date type of input.

Lastly, the History of Page extension serves our purpose of tracking changes to specific pages in MediaWiki. MediaWiki does provide recent changes functionality, but the system needs to have an easiest way to look for changes for a specific page without having to go Special:RecentChanges\(^{18}\) page first. Furthermore, the extension provides users a date and link to a particular change and it only records the changes marked as a major change. This is helpful especially when it comes to an event when users made

\(^{14}\) http://www.mediawiki.org/wiki/Extension:Semantic_Result_Formats
\(^{15}\) http://www.mediawiki.org/wiki/Extension:Semantic_Forms
\(^{16}\) http://www.mediawiki.org/wiki/Extension:Semantic_Forms_Inputs
\(^{17}\) http://www.mediawiki.org/wiki/Extension:HistoryOfPage
\(^{18}\) Special pages are pages that are created by the MediaWiki to perform a specific function.
some a typographical error of a page and want to correct that error without wanting for the system to record that change or unimportant changes.

**Goal tree**

The goal tree as presented in section 4.4 posed the biggest challenge in providing an implementation at first. It is hard to implement it within the layered infrastructure, especially when you have to comply to the MediaWiki and Semantic MediaWiki programming environment.

Luckily, there is an external application that can be used in accordance with MediaWiki in providing such function. Graphviz\(^{19}\) is open source graph visualization program that takes descriptions of graphs in a simple text language and makes diagrams in useful formats. Using this program accompanied with a modified version of the Semantic Results Formats extension, we managed to implement the goal tree as previously presented. The example of a goal tree in section 4.4 is successfully implemented as shown in Figure 4-6. With this approach to the goal tree, the implementation produces undesirable outcomes expressly when dealing with a complex project with so many goals, which could lead to overly complex layout (e.g. unreadable text). This could be a good reason to make an improvement in the future for a goal tree representation.

![Figure 4-6. A tree like graph produced by the system.](image)

**4.6 System session example**

Next, we will present an example session. This example session is meant to illustrate the use of the system in a practical scenario. For that purpose, consider a simple scenario that has been narratively explained in section 4.3.3, but this time we will

\(^{19}\) [http://www.graphviz.org/](http://www.graphviz.org/)
consider different goals and activities. The project is dubbed the *Charter Project*. The objective of the project is to build a prototype program of Java loop-bound function inference and integrate it to an Eclipse’s plugin.

There are seven goals that have been identified for that purpose. These goals will be described one-by-one and linked with screenshots from the system. The system does not point to what steps should be taken first. As it is a case of project management, we should create a project entry first. This is presented by Figure 4-7. We fill in “Charter Project” as the name of the project and hit the submit button. This leads us to the next screen (Figure 4-8) where we can input the necessary information to the system as shown on the picture.

We fill in “Build ResAna prototype for loop-bound function inference” as the description of the project with “02/05/2011” as planned start date and “27/05/2011” as planned finish date. Later we also fill in the project objective field with “Build a prototype program of Java loop-bound function inference and integrate it to an Eclipse’s plugin”. Those fields are not mandatory fields, but it is always good to have some texts that describe a project. After filling in the necessary fields for the project, we will add some goals to the project for the initial increment or T0 phase.

![Figure 4-7. A create project form.](image-url)
The seven goals mentioned early are: “ResAna tool is developed” as a top goal of all goals, “Sprints are executed”, “Kick-off is carried out”, “Weekly meetings are held”, “Sprint presentations are given”, “Prototype is developed”, and “Eclipse plugin is developed”. Some of these goals have activities designed for them, namely “Do sprint 1”, “Do sprint 2”, “Do sprint 3”, “Book a room for meetings”, “Organize kick-off meeting”, and “Prepare presentation slides”. We begin by creating the entry for the top goal “ResAna project is developed”, fill in the information as shown on Figure 4-9. At the beginning of a goal creation the goal status is always not achieved, but as soon as a goal is successfully accomplished, the goal status could be changed to done. Users change the goal status from one to another status manually.

We add the rest of the goals that have a super goal pointing to the “ResAna tool is developed” goal. After completing this, we will have a tree representation of goals (Figure 4-10) by running a tree representation query specific for the project.

We also have some activities that need to be filled in. As is the case with goals, we also add some information for each of the activities. Figure 4-11 shows forms that need to be filled in.

Figure 4-8. Forms for detailed project information.
Figure 4-9. Forms for detailed goal information.

**Run query: Tree**

```
ResAna tool is developed
  Prototype is developed
  Eclipse plugin is developed
  Weekly meetings are held
  Sprint presentations are given

Kick-off is carried out
Sprints are executed
```

Figure 4-10. A tree representation of goals created.
We do the same to the rest of the activities to complete the process. To indicate that we should do “Do sprint 1” first then do “Do sprint 2”, we add the rule to fulfill the first activity before the second (Figure 4-12). Once we completed filling in all goals and activities to the system, we have a screen that looks like in Figure 4-13 and Figure 4-14.

**Figure 4-11.** Forms for detailed activity information.

**Figure 4-12.** Adding a temporal rule.
As previously mentioned, this is the initial increment or T0. Let’s say the Project Manager of Charter Project would like to delete one of the goals and any activity that it contains, and perceive these changes as the first increment or T1 phase. Would the system be able to track these changes? Yes, the system does track incremental changes as long as users do not check “This is the minor edit” checkbox. To show how it works, we will delete the goal “Sprint presentations are given” and its activity. Moving to the “Recent changes” tab, we will see that there are two major changes that are recorded by the system (Figure 4-15).
Clicking the recent link given by the system gives us the next screen that shows what changes have been made to the project compared to the previous one (see Figure 4-16). We can obviously observe that one goal and one activity were deleted from the current phase, which is T1 phase.
**Figure 4-16.** Difference between revisions.
CHAPTER 5

Evaluation

In the last chapter, we revise the requirements, based on the design made on the basis of the initial requirements and the subsequently implementation of the system. Our work does not just stop there. As part of design science approach, the artifact (the method management system) has to be rigorously evaluated. Pressman (2001) states that software needs to go through verification and validation. Verification denotes to a set of activities which ensure that software correctly implements a specific function, while validation denotes to a different set of activities that guarantee the software that has been built is traceable to requirements. Boehm (1981) puts it another way:

Verification: “Are we building the system right?”

Validation: “Are we building the right system?”

In respect of verification, we have fixed several bugs or errors that came up and did this in parallel with the implementation process. We will not elaborate more on the verification, because in information science we are more interested in how to build the right system. Thereby within this chapter, we validate the system and confront it with (practical) cases.

We aim for the test cases to give us the idea whether the conceptualization and implementation are sound or not.

5.1 Test cases

There are two test cases, which the system is confronted with. The first case is the Household Effect Measurement Device (HEMD) project. The second test case is the Resource Analysis (ResAna) project. First we present the overview of the test cases and later based on (practical) tests we present the result and findings of a particular test.

5.1.1 HEMD project

The first test case, the HEMD project, is dedicated to see how far the system supports rule-based method enactment. The project itself is categorized as the rule-based method enactment project, because there are goals that are produced by means of the
rule-based method engineering approach and employed in the project execution. We are interested to see how the system is able to give us an integrated status report of project execution of both the intentional part and the operational part.

The HEMD project is the project of a leading player on the Dutch insurance market. We do not run the system in the real HEMD project, but we do take the initial goals and activities produced by the initialization of the real project. These goals and activities are later to be regarded as the initial increment (T0).

We need to test the functionality of the system to be able to cope with rule-based method enactment and one of which is to be able to handle the incremental approach. Changes in the environment may influence the operational execution of a specific method fragment. This operational execution requires adding and/or deleting goals and activities in the system. That is why the system has to show that it is able to handle such tasks by providing the incremental approach to the changes. With respect to increments, we include seven increments, namely, T0, T1, T2, T3, T4, T5, and T6.

5.1.2 ResAna project

The ResAna project is a project that has been accomplished during the GiP House project management practice for Information Science students. The project was aimed to build the resource analysis loop-bound function inference prototype, a tool to infer loop-bound functions for java programs.

We aim to provide the system with this test in order to find out whether we can use the system to handle a project execution that is not based on rule-based method enactment. The ResAna project was managed within the agile development methodology from the beginning of the project until the project was completed, but for the purpose of research validation we try to simulate the project using agile methodology and later change the methodology to the waterfall methodology, which is discussed in chapter 3.

The result of this test case is not so important compared to the result of the first test case, since we do not take into account any requirement to build such a system for the purpose of general project management activities. However, the result of this test contributes to verifying whether the system can function as project management software.

5.2 Result and findings

5.2.1 HEMD Project

As previously mentioned the system was tested for its ability to deal with the HEMD project. The initial increment (T0) of the project was imported from the real scenario of the project. 35 goals and 14 activities were identified for this initial increment (for the
complete list of goals and activities for this increment, see Appendix A). T0 gives us the
goal tree as depicted in Figure 5-2.

For the T1 increment (refer to Appendix A for complete T1 increment), we made
up one goal (method goal) and its corresponding activity that were entered into the
system. We also changed the status of goal “Kick-off is held” from not achieved to
done. Since we have had two increments at this point, we were interested to know
whether the system really can show us the changes that have been made by comparing
these changes to the last increment (T0). Figure 5-3 displays the revision that has been
made comparing the T0 increment with the T1 increment. As we can see, the system
indeed shows the exact same changes as previously mentioned. The changes shown
are not in the form of standard wiki readable text, but in the form of wiki template
markup. This implies that users at the minimum need to know the wiki markup of
templates20 created for this system.

At this point, we have two increments viz. T0 and T1. The next increment is T2 in
which two goals were added, namely, ‘Functional testing is performed’ and ‘Non-
functional testing is performed’. These goals are categorized as method goals.
Furthermore, ‘User Stories are elicited’ was realized, thus the status changed. We
added these goals to the system using a different username. Prior adding the goals, we
already had a second username in the system that has its core on the top of MediaWiki.
The point that we want to make here is that building the system using MediaWiki
provides capability for collaborative approach. A collaborative manner of working is
implied in MediaWiki. Figure 5-1 shows the reversion of the project that has been edited
by two users viz. ‘Wikisysop’ and ‘Deritaufan’. This is evidence of using the system in
collaborative manner.

![HEMD](image)

(Comparison of revisions)

Revision as of 14:26, 13 May 2011 (edit)  Latest revision as of 14:35, 13 May 2011 (edit) (undo)

WikiSysop (Talk | contribs | block)  Deritaufan (Talk | contribs | block)  [rollback]

m ← Older edit

**Figure 5-1.** A revision shows two users have edited the HEMD project page.

The last three increments, namely, T3, T4 and T5 as can be seen in Appendix A
gave us a better understanding of how the recent changes featured in the system help
us tracking any changes from one increment to another increment. Figure 5-3 really
demonstrates how this recent change feature works in practice. Every change that we
made to the project in the system can be traced back to the previous increment. This

---

20 There are the project, goal, activity and dependency (temporal ordering) template within
the system. Each markup in the system begins with its corresponding name.
finding shows that the recent changes feature in the system really helps to track changes during project execution.

By using MediaWiki, we are able to provide a live document or dynamic document to track every change made in the intentional part and the operational part of rule-based method enactment. This live document may evolve through updates, be expanded as needed, and serve a different purpose over time. A combination of MediaWiki with Semantic MediaWiki indeed gives the ability for the system to provide a “living document” that evolves over time and helps us to gain insight into the project execution.

**Things to ponder**

As shown in the increment T4, we needed to delete the “Comprehensive regression tests are performed” goal from the system. The goal had one activity that operationalized it. Deleting a goal does not automatically mean deleting an activity it contains. Therefore in our test, we had to manually delete the “Perform regression test” activity too in order for the mutation to be completed. Although this is somewhat unfortunate in a project that consists of many goals and activities (which requires you to trace and delete goals and activity one-by-one), the practice gives much freedom and fits into a declarative manner of working.

The use of temporal rules for activities is easy to be performed and well understood. The temporal rules as Voesten (2009) suggested were well used (see T0 and T6 in Appendix A). At first we doubted that we would use the ‘immediately before’ rule in practice, but as soon as we added activities to the system and took the consideration as we were in the real situation of the project, it came apparent that this rule is useful, for instance: we do “Conduct Weekly Sprint Planning Meeting” immediately before we do “Execute Sprint 1”.

Another thing to be noted is that a goal tree is barely readable if the system renders more than 20 goals within a project. Although this is could be an unfortunate situation to a project with many goals, it does help people who involve in a project to see a big picture of how goals are inter-related one another. This could be a point for improvement in the future.
Figure 5.2: A goal tree resulted from the T0 increment.
Figure 5-3. Revision between T0 and T1
5.2.2 ResAna Project

This section reports the result of the ResAna project test. The result is used to validate whether the method management system can be used to manage general project management or as project management software.

The initial increment (T0) of the ResAna project as can be seen in Appendix B consists nine goals and nine activities with one temporal rule. Anyone who knows the fundamental concepts of Agile methodology would immediately know that the project used Agile methodology as its basis just by observing the goals, activities and temporal rules. This could be obvious because in naming the goals, we tend to name it to reflect the methodology that the project uses (e.g. ‘Sprint 1 is completed’). The ‘sprint’ term is mandatorily used within the Agile methodology (see section 2.5.2).

Another interesting finding from the ResAna project concerns adding activities to the system. Most project management applications tend to ask users about tasks’ start date and end date, which by the end will render a nice Gantt chart that is typical in project scheduling. We do not provide the start date field for activities because we do not want to break the rule of the declarative approach. Moreover, the end date of tasks that can be found in general project management software, here we call due date time. This declarative approach of project scheduling could be a burden to the general project management, especially when facing many activities that need to be (temporally) ordered. However, this is not the case for project execution of rule-based method enactment, since it employs the minimal specification approach, thus people within the project only specify their intentions without describing or planning it in detail.

Within the T2 increment, we experimented with the idea of changing the development methodology while the project is in progress. Doing this, we shifted from the Agile methodology to the waterfall methodology. Shifting from one methodology to another requires deleting some goals that reflect to the methodology first used and adding goals that indicate the new methodology used. The temporal rules also need to be adjusted, like in the case of waterfall life cycle, “gather requirements” before “do design”. Through this approach, anyone who has knowledge about the waterfall methodology (see section 3.5.1) and who would view the goals, activities and temporal rules of the project in the method management system and would immediately notice that the project now employs the waterfall methodology.

In the light of the above, the method management system may be used as project management software for general ISD projects (projects that do not employ the rule-based method engineering approach), but as the result of declarative approach that is employed by the system, it would be hard to schedule tasks (activities) in the sense of what most project management software does by means of Gantt charts by specifying the start and finish date of tasks (activities). After all, since the beginning we have not intended the system to function as project management software. In addition this
shows that the declarative approach really leads to minimal specification which is one of the characteristic of the rule-based method engineering.
CHAPTER 6

Conclusions

Up to this point, we have scrutinized the underlying theoretical framework of the method management system, designed and implemented it, even tested it as we have seen in the previous chapter. The work of Hoppenbrouwers et al. (2011), *Agile Service Development: A Rule-Based Method Engineering Approach*, is the starting point of this thesis in which we focus on implementing a prototype system for the purpose of rule-based method enactment. Conclusions are given in the next section, while the last section gives future prospects of expanding our work.

6.1 Conclusions

The conclusion of this thesis will be given by answering the main research question as stated in chapter 2. The main research question is:

*Is it possible to build a system supporting rule-based method enactment using MediaWiki (MW) and Semantic MediaWiki (SMW)?*

In order to answer the main research question, we split it up into four sub-questions that together answer the aforementioned question. In this conclusion the answers to the sub-questions will be collated from previous chapters.

The first sub-question asks about rule-based method engineering and its characteristics. A rule-based approach to method engineering defines a framework that conceptualizes a way of working for development of new services by taking into account agility. The approach follows the configuration process for situational method engineering—situational projects factors that influence the choice of method fragments—with an addition of agility in operational execution. Increased agility in operational execution is proposed by describing method fragments as sets of rules that determine context and constraints in which goals are to be reached.

Following are the characteristics of the rule-based method engineering:

- A rule-based, declarative approach to the description of method fragments
- Strong emphasis on goals in selection and structuring (goal-driven selection of method fragments and execution of process activities)
- Minimal specification: planning only what needs to be planned and leave the rest to project members' powers of self-organization
• Continuous change of practices and goals.

The second sub-question concerns drawing requirements from the rule-based method engineering and its characteristics. We identified three high-level requirements that form basis of design of the system. The first requirement is that the system has to support a goal-driven approach. Goals are a central point in the rule-based method engineering and are the intentional part of the method enactment. On the other hand, activities are the operational part, which realize goals. Furthermore, goals need to be structured as a tree representation for a visualization purpose. The next requirement is for the system to operate in a declarative manner which brings us to the declarative workflows. Three temporal operators (Voesten, 2009), namely, “before”, “after” and “immediately before” are used for the activity ordering. The last requirement is for the system to be able to track changes in a project execution. These changes results from continuous change of practices and goals that is one of the characteristics of the rule-based method engineering. This characteristic also brings us to the need of an incremental approach.

The third sub-question relates to a system that fulfills the requirements. The Method Management System or MMS is a prototype system that has a purpose to support rule-based method enactment. The system means supporting agility in project execution by allowing users to work with it in a declarative manner. The system is built on the top of MediaWiki and Semantic MediaWiki. For this matter, several community extensions and a custom made extension are used to fulfill the requirements (see section 4.5).

The last sub-question asks for reasons of choosing MediaWiki and Semantic MediaWiki. The main reasons for choosing Semantic MediaWiki are because of its capability in delivering consistency of contents and our need to have a form-based wiki system which employs Semantic Forms extension. Consequently, choosing to employ Semantic MediaWiki means that MediaWiki is also needed.

Following are advantages of using MediaWiki and Semantic MediaWiki:

• Less programming. We do not build the system from scratch. In fact, the only thing that needs to be programmed is the custom made extension (HistoryOfPage extension).
• Tracking edits. Among the features of MediaWiki to assist in tracking edits is a Recent Changes feature that provides a list of recent edits to the system. This feature in combination with HistoryOfPage extension fulfills the need for the incremental approach and a ‘living document’ that evolves over time that help us to gain insight into the project execution.
• A collaborative way of working. We have seen this in section 5.2.1 where two users have made changes to the HEMD project.
• Dynamics of the structured knowledge representation. Semantic MediaWiki provides the means to keep a flexible, structured data schema consisting of properties and classes.
• Inline queries. The biggest advantage of SMW, beside its flexible annotation paradigm, is the possibility to reuse data across the platform by querying it from other pages. These inline queries allow requesting sets of data or just single property values and displaying them on a page in various formats.

With this in mind, we have a rich profusion of answering the main research question. Therefore, we conclude that the prototype system for supporting rule-based method enactment can be built with the use of MediaWiki in combination with Semantic MediaWiki, but technically we have not so strongly shown it in the test cases as the increments in the test cases were arbitrarily made.

6.2 Future work

As with every research, some ends need further research and in our case one end that needs further research is the implementation. While the use of MediaWiki along with Semantic MediaWiki benefits to us in implementing the method management system, we are intrigued to also try in the near future to implement the system using a more declarative way of programming without having to describe any control flow. A functional programming language could be a means of achieving it. It sounds ambitious, but it is clearly giving a future prospect to the proposed workflow of the rule-based method engineering.
References


based methods. 5th Int. Conf. on Software Process (ICSP’98), Chicago, Illinois, USA.


Appendix A: The HEMD Project Execution

A. HEMD Increment T0

List of goals added (35 goals):

Name: HEMD service is up and running

| Description: The top goal on an HEMD project |
| Top goal: yes |
| Urgency: required |
| Goal type: - |
| Goal status: not achieved |
| Super goal: - |
| Deadline: - |
| Sister goal(s): - |
| Achievement criterion: - |

Name: Kick-off is held

| Description: A kick-off of the project |
| Top goal: no |
| Urgency: required |
| Goal type: Method goal, Event goal |
| Goal status: not achieved |
| Super goal: HEMD service is up and running |
| Deadline: - |
| Sister goal(s): Architectural meetings are held, Code reviews are performed, Sprint planning meetings are held, Standup meetings are performed, Retrospective meetings are performed, Sprints are accomplished |
| Achievement criterion: - |

Name: Architectural meetings are held

<p>| Description: Architectural meetings are need to be done |
| Top goal: no |
| Urgency: required |
| Goal type: Method goal, Event goal |
| Goal status: not achieved |
| Super goal: HEMD service is up and running |
| Deadline: - |
| Sister goal(s): Code reviews are performed, Sprint planning meetings are held, Standup meetings are performed, Retrospective meetings are performed, Sprints are accomplished |
| Achievement criterion: - |</p>
<table>
<thead>
<tr>
<th>Goal</th>
<th>Description</th>
<th>Top goal</th>
<th>Urgency</th>
<th>Goal type</th>
<th>Goal status</th>
<th>Super goal</th>
<th>Deadline</th>
<th>Sister goal(s)</th>
<th>Achievement criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Achievement criterion:</strong></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Achievement criterion:</strong></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Achievement criterion:</strong></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Achievement criterion:</strong></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Achievement criterion:</strong></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Top goal: no</td>
<td>Goal type: Method goal, Event goal</td>
<td>Goal status: not achieved</td>
<td>Super goal: HEMD service is up and running</td>
<td>Sister goal(s): Sprints are accomplished, Kick-off is held, Architectural meetings are held, Code reviews are performed, Sprint planning meetings are held, Standup meetings are performed</td>
<td>Achievement criterion: -</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>-----------------------------------</td>
<td>--------------------------</td>
<td>--------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>-----------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Top goal: no</th>
<th>Goal type: Method goal, Event goal</th>
<th>Goal status: not achieved</th>
<th>Super goal: HEMD service is up and running</th>
<th>Sister goal(s): Kick-off is held, Architectural meetings are held, Code reviews are performed, Sprint planning meetings are held, Standup meetings are performed, Retrospective meetings are performed</th>
<th>Achievement criterion: -</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Top goal: no</th>
<th>Goal type: Method goal, Deliverable goal</th>
<th>Goal status: not achieved</th>
<th>Super goal: HEMD service is up and running</th>
<th>Sister goal(s): Requirements are determined</th>
<th>Achievement criterion: -</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Top goal: no</th>
<th>Goal type: Method goal, Deliverable goal</th>
<th>Goal status: not achieved</th>
<th>Super goal: HEMD service is up and running</th>
<th>Sister goal(s): User Stories are elicited</th>
<th>Achievement criterion: -</th>
</tr>
</thead>
</table>
**Name:** Client involvement is needed

**Description:** Clients are only involved if insufficient knowledge is available at V&V about what clients need (by the Product Owner)

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Method goal, Communication goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: Requirements are determined</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Design decisions are discussed, Steps and changes are logged, Technical design are reviewed, Requirements are discussed, PB consent is addressed, Testmanagement acceptance is attained, Reporting to PB is performed</td>
<td></td>
</tr>
</tbody>
</table>

Achievement criterion: -

**Name:** Reporting to PB is performed

**Description:** Report to Project Board (PB) (every two week) (by the Product Owner)

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Method goal, Communication goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Design decisions are discussed, Steps and changes are logged, Technical design are reviewed, Requirements are discussed, PB consent is addressed, Testmanagement acceptance is attained, Client involvement is needed</td>
<td></td>
</tr>
</tbody>
</table>

Achievement criterion: -

**Name:** Design decisions are discussed

**Description:** Design decisions need to be discussed with and approved by architect

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Method goal, Communication goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Steps and changes are logged, Technical design are reviewed, Requirements are discussed, PB consent is addressed, Testmanagement acceptance is attained, Client involvement is needed, Reporting to PB is performed</td>
<td></td>
</tr>
</tbody>
</table>

Achievement criterion: -

**Name:** Steps and changes are logged

**Description:** PE logs every step and change in a process document

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type:</td>
<td>Goal status:</td>
</tr>
<tr>
<td>Super goal:</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s):</td>
<td>Design decisions are discussed, Steps and changes are logged, Technical design are reviewed, Requirements are discussed, PB consent is addressed, Testmanagement acceptance is attained, Client involvement is needed, Reporting to PB is performed</td>
</tr>
<tr>
<td>Goal type:</td>
<td>Method goal, Communication goal</td>
</tr>
<tr>
<td>Super goal:</td>
<td>HEMD service is up and running</td>
</tr>
<tr>
<td>Sister goal(s):</td>
<td>Design decisions are discussed, Technical design are reviewed, Requirements are discussed, PB consent is addressed, Testmanagement acceptance is attained, Client involvement is needed, Reporting to PB is performed</td>
</tr>
</tbody>
</table>

**Name:** Technical design are reviewed  
**Description:** The technical design needs to be reviewed in architectural meetings  
**Top goal:** no  
**Goal type:** Method goal, Communication goal  
**Super goal:** HEMD service is up and running  
**Deadline:** -  
**Sister goal(s):** Design decisions are discussed, Steps and changes are logged, Requirements are discussed, PB consent is addressed, Testmanagement acceptance is attained, Client involvement is needed, Reporting to PB is performed  
**Achievement criterion:** -  

**Name:** Requirements are discussed  
**Description:** Product owner has discussed requirements with expert on inboedelverzekeringen process. NOT contact with intermediaries or clients; but in bigger projects this would be the case (*exception on method rule?*)  
**Top goal:** no  
**Goal type:** Method goal, Communication goal  
**Super goal:** HEMD service is up and running  
**Deadline:** -  
**Sister goal(s):** Design decisions are discussed, Steps and changes are logged, Technical design are reviewed, PB consent is addressed, Testmanagement acceptance is attained, Client involvement is needed, Reporting to PB is performed  
**Achievement criterion:** -  

**Name:** PB consent is addressed  
**Description:** The Project Board (PB) as given its approval to the project  
**Top goal:** no  
**Goal type:** Method goal, Communication goal  
**Super goal:** HEMD service is up and running  
**Deadline:** -  
**Sister goal(s):** Design decisions are discussed, Steps and changes are logged, Technical design are reviewed, PB consent is addressed, Testmanagement acceptance is attained, Client involvement is needed, Reporting to PB is performed  
**Achievement criterion:** -
are reviewed, Requirements are discussed, Testmanagement acceptance is attained, Client involvement is needed, Reporting to PB is performed

<table>
<thead>
<tr>
<th>Achievement criterion: -</th>
</tr>
</thead>
</table>

**Name:** Testmanagement acceptance is attained  
**Description:** Testmanagement accords deployment  
**Top goal:** no  
**Urgency:** required  
**Goal type:** Method goal, Communication goal  
**Goal status:** not achieved  
**Super goal:** HEMD service is up and running  
**Deadline:** -  
**Sister goal(s):** Design decisions are discussed, Steps and changes are logged, Technical design are reviewed, Requirements are discussed, PB consent is addressed, Client involvement is needed, Reporting to PB is performed  

<table>
<thead>
<tr>
<th>Achievement criterion: -</th>
</tr>
</thead>
</table>

**Name:** Less paperwork is achieved  
**Description:** Less paperwork for intermediary is needed  
**Top goal:** no  
**Urgency:** required  
**Goal type:** Product goal  
**Goal status:** not achieved  
**Super goal:** HEMD service is up and running  
**Deadline:** -  
**Sister goal(s):** Acceptants time is saved  

<table>
<thead>
<tr>
<th>Achievement criterion: -</th>
</tr>
</thead>
</table>

**Name:** Acceptants time is saved  
**Description:** Less paperwork for intermediary is needed  
**Top goal:** no  
**Urgency:** required  
**Goal type:** Product goal  
**Goal status:** not achieved  
**Super goal:** HEMD service is up and running  
**Deadline:** -  
**Sister goal(s):** Less paperwork is achieved  

<table>
<thead>
<tr>
<th>Achievement criterion: -</th>
</tr>
</thead>
</table>

**Name:** Architectural constraints are conformed  
**Description:** There are goals and templates, stated by the architecture group, to which a project must conform. Three main components: -PBP (calculation rules), mostly handled by "product engineering" -VIA (process engine), mostly handled by "software development" -Model Engine
(user interface engine) (all three components are relevant for IWM)

<table>
<thead>
<tr>
<th>Top goal</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Product goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): -</td>
<td></td>
</tr>
</tbody>
</table>

**Achievement criterion:** -

<table>
<thead>
<tr>
<th><strong>Name:</strong> System registers filling-in</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> System registers filling in of IWM so that rappels can be sent</td>
</tr>
<tr>
<td>Top goal: no</td>
</tr>
<tr>
<td>Goal type: Product goal</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
</tr>
<tr>
<td>Sister goal(s): -</td>
</tr>
</tbody>
</table>

**Achievement criterion:** -

<table>
<thead>
<tr>
<th><strong>Name:</strong> HEMD e-forms can be filled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Intermediary can fill in HEMD e-forms</td>
</tr>
<tr>
<td>Top goal: no</td>
</tr>
<tr>
<td>Goal type: Product goal</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
</tr>
<tr>
<td>Sister goal(s): System registers filling-in, HEMD forms are sent automatically</td>
</tr>
</tbody>
</table>

**Achievement criterion:** -

<table>
<thead>
<tr>
<th><strong>Name:</strong> HEMD forms are sent automatically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Automatic sending of HEMD forms</td>
</tr>
<tr>
<td>Top goal: no</td>
</tr>
<tr>
<td>Goal type: Product goal</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
</tr>
<tr>
<td>Sister goal(s): System registers filling-in, HEMD e-forms can be filled</td>
</tr>
</tbody>
</table>

**Achievement criterion:** -

| **Name:** Test case template is used |
### Technical design template is used

**Description:** The preconceived template for technical design needs to be used

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Tool &amp; instrument goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Test case template is used, Backlog tool is used</td>
<td></td>
</tr>
<tr>
<td>Achievement criterion: -</td>
<td></td>
</tr>
</tbody>
</table>

### Backlog tool is used

**Description:** Backlog tool of MS TVS is used

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Tool &amp; instrument goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Test case template is used, Technical design template is used</td>
<td></td>
</tr>
<tr>
<td>Achievement criterion: -</td>
<td></td>
</tr>
</tbody>
</table>

### Zorgplicht is observed

**Description:** -

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Product goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Calculation rules are conformed</td>
<td></td>
</tr>
<tr>
<td>Achievement criterion: -</td>
<td></td>
</tr>
</tbody>
</table>

### Calculation rules are conformed

**Name:** Calculation rules are conformed
**Description:** Calculation rules typically conform to the rules stated by Verbond van Verzekeraars

<table>
<thead>
<tr>
<th>Top goal:</th>
<th>no</th>
<th>Urgency:</th>
<th>required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type:</td>
<td>Product goal</td>
<td>Goal status:</td>
<td>not achieved</td>
</tr>
<tr>
<td>Super goal:</td>
<td>HEMD service is up and running</td>
<td>Deadline:</td>
<td>-</td>
</tr>
<tr>
<td>Sister goal(s):</td>
<td>Zorgplicht is observed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Achievement criterion:</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Name:** Tests are performed

<table>
<thead>
<tr>
<th>Description:</th>
<th>Tests have taken place - Test report (by users? Are there types of test (technical/acceptance?))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top goal:</td>
<td>no</td>
</tr>
<tr>
<td>Goal type:</td>
<td>Method goal, Deliverable goal</td>
</tr>
<tr>
<td>Super goal:</td>
<td>HEMD service is up and running</td>
</tr>
<tr>
<td>Sister goal(s):</td>
<td>Comprehensive regression tests are performed</td>
</tr>
<tr>
<td>Achievement criterion:</td>
<td>-</td>
</tr>
</tbody>
</table>

**Name:** Comprehensive regression tests are performed

<table>
<thead>
<tr>
<th>Description:</th>
<th>Full (sufficient?) regression tests have to be performed after a change in the system has occurred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top goal:</td>
<td>no</td>
</tr>
<tr>
<td>Goal type:</td>
<td>Method goal, Deliverable goal</td>
</tr>
<tr>
<td>Super goal:</td>
<td>HEMD service is up and running</td>
</tr>
<tr>
<td>Sister goal(s):</td>
<td>Tests are performed</td>
</tr>
<tr>
<td>Achievement criterion:</td>
<td>-</td>
</tr>
</tbody>
</table>

**Name:** Design is delivered

<table>
<thead>
<tr>
<th>Description:</th>
<th>There will be some sort of design description (no further specification what this looks like?) (delivered by the engineers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top goal:</td>
<td>no</td>
</tr>
<tr>
<td>Goal type:</td>
<td>Method goal, Deliverable goal</td>
</tr>
<tr>
<td>Super goal:</td>
<td>HEMD service is up and running</td>
</tr>
<tr>
<td>Sister goal(s):</td>
<td>-</td>
</tr>
<tr>
<td>Achievement criterion:</td>
<td>-</td>
</tr>
</tbody>
</table>
Name: Time allocation is conformed

**Description:** Time spent per team member should not exceed 4 hours per week

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Method goal, Work planning goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Work items are planned, Division of work is achieved</td>
<td></td>
</tr>
<tr>
<td>Achievement criterion: -</td>
<td></td>
</tr>
</tbody>
</table>

Name: Work items are planned

**Description:** There are work items

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Method goal, Work planning goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Time allocation is conformed, Division of work is achieved</td>
<td></td>
</tr>
<tr>
<td>Achievement criterion: -</td>
<td></td>
</tr>
</tbody>
</table>

Name: Division of work is achieved

**Description:** Work load is distributed among the team members (to be done in a sprint planning meeting)

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Method goal, Work planning goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: HEMD service is up and running</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): Time allocation is conformed, Work items are planned</td>
<td></td>
</tr>
<tr>
<td>Achievement criterion: -</td>
<td></td>
</tr>
</tbody>
</table>

**List of activities added (14 activities):**

Name: Perform Kick-off meeting

<p>| Description: - |
| Goals: Code reviews are performed | Due date: - |
| Location: V&amp;V, Middelharnis | Instrument used: - |
| People involved: - | Information needed: - |
| Repeat activity: - |</p>
<table>
<thead>
<tr>
<th>Name: Conduct Peer Code Review</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Peer review meeting to find and fix mistakes overlooked in the source code</td>
</tr>
<tr>
<td><strong>Goals:</strong> Kick-off is held</td>
</tr>
<tr>
<td><strong>Location:</strong> V&amp;V, Middelharnis</td>
</tr>
<tr>
<td><strong>People involved:</strong> -</td>
</tr>
<tr>
<td><strong>Repeat activity:</strong> -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Execute Sprint 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Execution tasks for sprint 1</td>
</tr>
<tr>
<td><strong>Goals:</strong> Sprints are accomplished</td>
</tr>
<tr>
<td><strong>Location:</strong> V&amp;V, Middelharnis</td>
</tr>
<tr>
<td><strong>People involved:</strong> -</td>
</tr>
<tr>
<td><strong>Repeat activity:</strong> -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Execute Sprint 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Execution tasks for sprint 2</td>
</tr>
<tr>
<td><strong>Goals:</strong> Sprints are accomplished</td>
</tr>
<tr>
<td><strong>Location:</strong> V&amp;V, Middelharnis</td>
</tr>
<tr>
<td><strong>People involved:</strong> -</td>
</tr>
<tr>
<td><strong>Repeat activity:</strong> -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Execute Sprint 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Execution tasks for sprint 3</td>
</tr>
<tr>
<td><strong>Goals:</strong> Sprints are accomplished</td>
</tr>
<tr>
<td><strong>Location:</strong> V&amp;V, Middelharnis</td>
</tr>
<tr>
<td><strong>People involved:</strong> -</td>
</tr>
<tr>
<td><strong>Repeat activity:</strong> -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Execute Sprint 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Execution tasks for sprint 4</td>
</tr>
<tr>
<td><strong>Goals:</strong> Sprints are accomplished</td>
</tr>
</tbody>
</table>
Name: Conduct Weekly Sprint Planning Meeting

Description: Communication channel with product owner on what requirements to complete in a given sprint and the team doing some ground work on getting going with the sprint.

Goals: Sprint planning meetings are held

Due date: -

Location: V&V, Middelharnis

Instrument used: -

People involved: -

Information needed: -

Repeat activity: every 1 week

Name: Conduct Daily Stand-up Meeting

Description: Daily meeting to provide status update to the team members

Goals: Standup meetings are performed

Due date: -

Location: V&V, Middelharnis

Instrument used: -

People involved: -

Information needed: -

Repeat activity: every 1 day

Name: Conduct Architectural Vision Meeting

Description: Identify architectural vision meeting for the project

Goals: Architectural meetings are held

Due date: -

Location: V&V, Middelharnis

Instrument used: -

People involved: -

Information needed: -

Repeat activity: -

Name: Conduct Retrospective Meeting

Description: Retrospective meeting after each sprint is conducted.

Goals: Retrospective meetings are performed

Due date: -

Location: V&V, Middelharnis

Instrument used: -

People involved: -

Information needed: -

Repeat activity: -
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Goals</th>
<th>Due date:</th>
<th>Location</th>
<th>Instrument used:</th>
<th>People involved:</th>
<th>Information needed:</th>
<th>Repeat activity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare progress report and slides</td>
<td>Prepare progress reports and presentation slides for the purpose of reporting to the Project Board</td>
<td>Reporting to PB is performed</td>
<td>-</td>
<td>V&amp;V, Middelharnis</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>every 2 weeks</td>
</tr>
<tr>
<td>Elicit User Stories</td>
<td>Elicit informal statements from users</td>
<td>User Stories are elicited</td>
<td>-</td>
<td>V&amp;V, Middelharnis</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Perform regression test</td>
<td>Perform regression tests following changes on the system</td>
<td>Comprehensive regression tests are performed</td>
<td>-</td>
<td>V&amp;V, Middelharnis</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Discuss Requirements with Expert</td>
<td>Product owner has discussed requirements with expert on inboedelverzekeringen process.</td>
<td>Requirements are discussed performed</td>
<td>-</td>
<td>V&amp;V, Middelharnis</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

List of temporal ordering:
Do "Execute Sprint 1" before "Execute Sprint 2"
Do "Execute Sprint 2" before "Execute Sprint 3"
Do "Execute Sprint 3" before "Execute Sprint 4"

B. HEMD Increment T1

List of goals added/edited:

Name: Sprint review meetings are conducted

Description: Review the work that was completed and not completed. Present the completed work to the stakeholders (a.k.a. “the demo”)

Top goal: no  
Urgency: required

Goal type: Method goal, Event goal  
Goal status: not achieved

Super goal: HEMD service is up and running  
Deadline: -

Sister goal(s): Architectural meetings are held, Code reviews are performed, Sprint planning meetings are held, Standup meetings are performed, Retrospective meetings are performed, Sprints are accomplished, Kick-off is held

Achievement criterion: -

Name: Kick-off is held

Goal status: done

List of activities added:

Name: Perform sprint review meetings

Description: -

Goals: Sprint review meetings are conducted  
Due date: -

Location: V&V, Middelharnis  
Instrument used: -

People involved: -  
Information needed: -

Repeat activity: -

C. HEMD Increment T2

List of goals added/edited:

Name: Functional testing is performed
**Description:** Test the system on its functionality  
**Top goal:** no  
**Urgency:** required  
**Goal type:** Method goal, Deliverable goal  
**Goal status:** not achieved  
**Super goal:** Tests are performed  
**Deadline:** -  
**Sister goal(s):** Non-functionality testing are performed  
**Achievement criterion:** -

**Name:** Non-functionality testing are performed  
**Description:** Non-functionality test of the system  
**Top goal:** no  
**Urgency:** required  
**Goal type:** Method goal, Deliverable goal  
**Goal status:** not achieved  
**Super goal:** Tests are performed  
**Deadline:** -  
**Sister goal(s):** Functional testing is performed  
**Achievement criterion:** -

**Name:** User Stories are elicited  
**Goal status:** done

### D. HEMD Increment T3

**List of goals added/edited:**

**Name:** Model storming session is performed  
**Description:** Typically impromptu events, one project team member will ask another to model with them, typically lasting for five to ten minutes  
**Top goal:** no  
**Urgency:** required  
**Goal type:** Method goal, Event goal  
**Goal status:** not achieved  
**Super goal:** HEMD service is up and running  
**Deadline:** -  
**Sister goal(s):** Architectural meetings are held, Code reviews are performed, Sprint planning meetings are held, Standup meetings are performed, Retrospective meetings are performed, Sprints are accomplished, Kick-off is held, Sprint review meetings are conducted  
**Achievement criterion:** -

### E. HEMD Increment T4

**List of goals added/edited:**
<table>
<thead>
<tr>
<th>Name: Architectural meetings are held</th>
<th>Goal status: done</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Requirements are determined</td>
<td>Goal status: done</td>
</tr>
<tr>
<td>Name: Client involvement is needed</td>
<td>Goal status: done</td>
</tr>
<tr>
<td>List of goals deleted:</td>
<td></td>
</tr>
<tr>
<td>Name: Comprehensive regression tests are performed</td>
<td></td>
</tr>
<tr>
<td>List of activities deleted:</td>
<td></td>
</tr>
<tr>
<td>Name: Perform regression test</td>
<td></td>
</tr>
</tbody>
</table>

**F. HEMD Increment T5**

List of goals added/edited:

<table>
<thead>
<tr>
<th>Name: Code reviews are performed</th>
<th>Goal status: done</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Sprint planning meetings are held</td>
<td>Goal status: done</td>
</tr>
<tr>
<td>Name: Standup meetings are performed</td>
<td>Goal status: done</td>
</tr>
<tr>
<td>Name: Retrospective meetings are performed</td>
<td>Goal status: done</td>
</tr>
<tr>
<td>Name: Sprints are accomplished</td>
<td></td>
</tr>
</tbody>
</table>
Goal status: done

G. HEMD Increment T6

List of temporal ordering added:

- Do "Conduct Weekly Sprint Planning Meeting" immediately before "Execute Sprint 1"
- Do "Conduct Weekly Sprint Planning Meeting" immediately before "Execute Sprint 2"
- Do "Conduct Weekly Sprint Planning Meeting" immediately before "Execute Sprint 3"
- Do "Conduct Weekly Sprint Planning Meeting" immediately before "Execute Sprint 4"
- Do "Conduct Retrospective Meeting" after "Execute Sprint 1"
- Do "Conduct Retrospective Meeting" after "Execute Sprint 2"
- Do "Conduct Retrospective Meeting" after "Execute Sprint 3"
- Do "Conduct Retrospective Meeting" after "Execute Sprint 4"
Appendix B: The ResAna Project Execution

A. ResAna Increment T0

List of goals added (9 goals):

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Top goal</th>
<th>Goal type</th>
<th>Goal status</th>
<th>Urgency</th>
<th>Super goal</th>
<th>Deadline</th>
<th>Sister goal(s)</th>
<th>Achievement criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResAna project is completed</td>
<td>The top goal on an HEMD project</td>
<td>yes</td>
<td>Top goal</td>
<td>not achieved</td>
<td>required</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sprint 1 is completed</td>
<td>Sprint 1 should be completed</td>
<td>no</td>
<td>Sprint goal</td>
<td>not achieved</td>
<td>required</td>
<td>ResAna project is completed</td>
<td>2010/10/22</td>
<td>Sprint 2 is completed, Sprint 3 is completed</td>
<td>-</td>
</tr>
<tr>
<td>Sprint 2 is completed</td>
<td>Sprint 2 should be completed</td>
<td>no</td>
<td>Sprint goal</td>
<td>not achieved</td>
<td>required</td>
<td>ResAna project is completed</td>
<td>2010/11/12</td>
<td>Sprint 3 is completed, Sprint 1 is completed</td>
<td>-</td>
</tr>
<tr>
<td>Name:</td>
<td>Sprint 3 is completed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>Sprint 3 should be completed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top goal:</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Urgency:</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal type:</td>
<td>Sprint goal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal status:</td>
<td>not achieved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Super goal:</td>
<td>ResAna project is completed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deadline:</td>
<td>2010/12/10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sister goal(s):</td>
<td>Sprint 1 is completed, Sprint 2 is completed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Achievement criterion:</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Name: | Reviews are conducted |
| Description: | Review at the end of each sprint should be conducted |
| Top goal: | no |
| Urgency: | required |
| Goal type: | Event goal |
| Goal status: | not achieved |
| Super goal: | ResAna project is completed |
| Deadline: | - |
| Sister goal(s): | Sprint presentations are given, Project kick-off is held |
| Achievement criterion: | - |

| Name: | Sprint presentations are given |
| Description: | At the end of each print, the presentation should be given |
| Top goal: | no |
| Urgency: | required |
| Goal type: | Event goal |
| Goal status: | not achieved |
| Super goal: | ResAna project is completed |
| Deadline: | 2011/01/12 |
| Sister goal(s): | Reviews are conducted, Project kick-off is held |
| Achievement criterion: | - |

| Name: | Documentation is completed |
| Description: | Documentation that accompanies the tool should be completed |
| Top goal: | no |
| Urgency: | required |
| Goal type: | Product goal |
| Goal status: | not achieved |
| Super goal: | ResAna project is completed |
| Deadline: | - |
| Sister goal(s): | Testing is conducted |
| Achievement criterion: | - |
### Name: Testing is conducted

**Description:** -

| Top goal: no | Urgency: required |
| Goal type: Product goal | Goal status: not achieved |
| Super goal: ResAna project is completed | Deadline: - |
| Sister goal(s): Documentation is completed |
| Achievement criterion: - |

### Name: Project kick-off is held

**Description:** -

| Top goal: no | Urgency: required |
| Goal type: Event goal | Goal status: not achieved |
| Super goal: ResAna project is completed | Deadline: - |
| Sister goal(s): Reviews are conducted, Sprint presentations are given |
| Achievement criterion: - |

### List of activities added (9 activities):

#### Name: Fix fields in the loop-condition

**Description:** -

| Goals: Sprint 1 is completed | Due date: 2010/10/20 00:00:00 |
| Location: GiP House | Instrument used: - |
| People involved: Joel Haasnoot | Information needed: - |
| Repeat activity: - |

#### Name: Add support for arrays

**Description:** -

| Goals: Sprint 1 is completed | Due date: 2010/10/20 00:00:00 |
| Location: GiP House | Instrument used: - |
| People involved: Ying Zeng | Information needed: - |
| Repeat activity: - |

#### Name: Add support for if-constructs
<table>
<thead>
<tr>
<th>Name: Add support for complete project files</th>
<th>Description: -</th>
<th>Goals: Sprint 1 is completed</th>
<th>Due date: 2010/10/20 00:00:00</th>
<th>Location: GiP House</th>
<th>Instrument used: -</th>
<th>People involved: Robin Oostrum</th>
<th>Information needed: -</th>
<th>Repeat activity: -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Add support for program slicing</td>
<td>Description: -</td>
<td>Goals: Sprint 2 is completed</td>
<td>Due date: 2010/11/10 00:00:00</td>
<td>Location: GiP House</td>
<td>Instrument used: -</td>
<td>People involved: Ying Zeng, Robin Oostrum</td>
<td>Information needed: -</td>
<td>Repeat activity: -</td>
</tr>
<tr>
<td>Name: Port prototype into Eclipse plugin</td>
<td>Description: -</td>
<td>Goals: Sprint 3 is completed</td>
<td>Due date: 2010/12/08 00:00:00</td>
<td>Location: GiP House</td>
<td>Instrument used: -</td>
<td>People involved: Joel Haasnoot, Liviu Varga, Robin Oostrum</td>
<td>Information needed: -</td>
<td>Repeat activity: -</td>
</tr>
<tr>
<td>Name: Do internal review</td>
<td>Description: Internal review between team member should be done</td>
<td>Goals: Reviews are conducted</td>
<td>Due date: -</td>
<td>Location: GiP House</td>
<td>Instrument used: -</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Name: Do external review

**Description:** External review between team and clients should be done

**Goals:** Reviews are conducted

**Due date:** -

**Location:** GiP House

**Instrument used:** -

**People involved:** -

**Information needed:** -

**Repeat activity:** -

Name: Add support for do-while loop

**Description:** -

**Goals:** Sprint 1 is completed

**Due date:** 2010/10/20 00:00:00

**Location:** GiP House

**Instrument used:** -

**People involved:** Liviu Varga

**Information needed:** -

**Repeat activity:** -

List of temporal ordering:

Do "Do internal review" immediately before "Do external review"

**B. ResAna Increment T1**

List of goals added/edited:

**Name:** Sprint 1 is completed

**Goal status:** done

List of activities added/edited:

**Name:** Fix fields in the loop-condition

**Goals:** Sprint 2 is completed

**Name:** Add support for arrays
**Goals:** Sprint 2 is completed

**Name:** Add support for if-constructs

**Goals:** Sprint 2 is completed

---

**C. ResAna Increment T2**

**List of goals added/edited:**

**Name:** Requirements are gathered

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Requirements pertaining the prototype are gathered</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Top goal:</strong></td>
<td>no</td>
</tr>
<tr>
<td><strong>Goal type:</strong></td>
<td>Communication goal</td>
</tr>
<tr>
<td><strong>Super goal:</strong></td>
<td>ResAna project is completed</td>
</tr>
<tr>
<td><strong>Sister goal(s):</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Achievement criterion:</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Urgency:</strong></td>
<td>required</td>
</tr>
<tr>
<td><strong>Goal status:</strong></td>
<td>not achieved</td>
</tr>
<tr>
<td><strong>Deadline:</strong></td>
<td>-</td>
</tr>
</tbody>
</table>

**Name:** Estimation works are prepared

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Sprint 2 should be completed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Top goal:</strong></td>
<td>no</td>
</tr>
<tr>
<td><strong>Goal type:</strong></td>
<td>Planning goal</td>
</tr>
<tr>
<td><strong>Super goal:</strong></td>
<td>ResAna project is completed</td>
</tr>
<tr>
<td><strong>Sister goal(s):</strong></td>
<td>Sprint 3 is completed, Sprint 1 is completed</td>
</tr>
<tr>
<td><strong>Achievement criterion:</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Urgency:</strong></td>
<td>required</td>
</tr>
<tr>
<td><strong>Goal status:</strong></td>
<td>not achieved</td>
</tr>
<tr>
<td><strong>Deadline:</strong></td>
<td>-</td>
</tr>
</tbody>
</table>

**Name:** Design is conducted

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Design document should be in place</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Top goal:</strong></td>
<td>no</td>
</tr>
<tr>
<td><strong>Goal type:</strong></td>
<td>Modeling goal</td>
</tr>
<tr>
<td><strong>Super goal:</strong></td>
<td>ResAna project is completed</td>
</tr>
<tr>
<td><strong>Sister goal(s):</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Achievement criterion:</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Urgency:</strong></td>
<td>required</td>
</tr>
<tr>
<td><strong>Goal status:</strong></td>
<td>not achieved</td>
</tr>
<tr>
<td><strong>Deadline:</strong></td>
<td>-</td>
</tr>
</tbody>
</table>

**Name:** Coding is finished
**Description:** At the end of each print, the presentation should be given

<table>
<thead>
<tr>
<th>Top goal: no</th>
<th>Urgency: required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal type: Construction goal</td>
<td>Goal status: not achieved</td>
</tr>
<tr>
<td>Super goal: ResAna project is completed</td>
<td>Deadline: -</td>
</tr>
<tr>
<td>Sister goal(s): -</td>
<td></td>
</tr>
<tr>
<td>Achievement criterion: -</td>
<td></td>
</tr>
</tbody>
</table>

**List of goals deleted:**

<table>
<thead>
<tr>
<th>Name: Sprint 1 is completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Sprint 2 is completed</td>
</tr>
<tr>
<td>Name: Sprint 3 is completed</td>
</tr>
<tr>
<td>Name: Reviews are conducted</td>
</tr>
<tr>
<td>Name: Sprint presentations are given</td>
</tr>
</tbody>
</table>

**List of activities added/edited:**

<table>
<thead>
<tr>
<th>Name: Gather requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: -</td>
</tr>
<tr>
<td>Goals: Requirements are gathered</td>
</tr>
<tr>
<td>Location: GiP House</td>
</tr>
<tr>
<td>People involved: -</td>
</tr>
<tr>
<td>Repeat activity: -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Do designing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: -</td>
</tr>
<tr>
<td>Goals: Design is conducted</td>
</tr>
<tr>
<td>Location: GiP House</td>
</tr>
<tr>
<td>People involved: -</td>
</tr>
<tr>
<td>Repeat activity: -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Do testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: -</td>
</tr>
<tr>
<td>Goals: Testing is conducted</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>Do documentation</td>
</tr>
<tr>
<td>Fix fields in the loop-condition</td>
</tr>
<tr>
<td>Add support for do-while loop</td>
</tr>
<tr>
<td>Add support for arrays</td>
</tr>
<tr>
<td>Add support for if-constructs</td>
</tr>
<tr>
<td>Add support for complete project files</td>
</tr>
<tr>
<td>Add support for program slicing</td>
</tr>
<tr>
<td>Port prototype into Eclipse plugin</td>
</tr>
</tbody>
</table>
List of activities deleted:

Name: Do internal review
Name: Do external review

List of temporal ordering added:

Do "Gather requirements" before "Do designing"
Do "Do documentation" after "Do testing"
Do "Do testing" after "Add support for arrays"
Do "Do testing" after "Add support for complete project files"

D. ResAna Increment T3

List of goals added/edited:

Name: Documentation is completed
Goal status: done

Name: Testing is conducted
Goal status: done

Name: Project kick-off is held
Goal status: done

Name: ResAna project is completed
Goal status: done

Name: Requirements are gathered
Goal status: done
**Name:** Estimation works are prepared

**Goal status:** done

**Name:** Design is conducted

**Goal status:** done

**Name:** Coding is finished

**Goal status:** done
Appendix C: Wiki Markup Template Source Code

WikiMedia markup template source code:

The wiki source of “Activity” template:

<noinclude>
This is the "Activity" template.

Edit the page to see the template text.
</noinclude><includeonly>
{{#set_internal:Is activity in project
| Has activity name=({{1}})
| Has activity description=({{2}})
| Is allocated to goal#list=({{3}})
| Has due date=({{4}})
| Has location#list=({{5}})
| Has instrument used#list=({{6}})
| Has individual person#list=({{7}})
| Has information needed#list=({{8}})
| Has repeat period=({{9}})
| Has repeat unit=({{10}})
| Has repeat until=({{11}})
}}
</includeonly>

The wiki source of “Activity print” template:

<noinclude>
This is the "Activity print" template.

Edit the page to see the template text.
</noinclude><includeonly>
| class="wikitable collapsible collapsed"
| ! {{{1}}}
| Description
| ! {{{2}}}
|
| Goal(s) | {{3|}} |
| -- | -- |
| Due date | {{4|}} |
| Location | {{5|}} |
| Instrument used | {{6|}} |
| People involved | {{7|}} |
| Information needed | {{8|}} |
| Repeat activity | {{9|}} |
| Temporal rules | {{10|}} |

The wiki source of “Goal” template:

```
<includeonly>
{{#set_internal: Is goal in project
| Has goal name={{1|}}
| Has goal description={{2|}}
| Is top goal={{3|}}
| Has item urgency={{4|}}
| Has goal type#list={{5|}}
| Has supergoal={{6|}}
| Has sistergoal#list={{7|}}
| Has goal status={{8|}}
| Has achievement criterion={{9|}}
| Has goal deadline={{10|}}
}}
</includeonly>
```
The wiki source of "Goal print" template:

This is the "Goal print" template.

Edit the page to see the template text.

| Description |
| Top goal indicator |
| Urgency |
| Goal type |
| Super goal |
| Sister goal |
| Goal status |
| Achievement Criterion |
| Deadline |
| Subgoal(s) |
| Activity |

Subgoal(s):

Is goal in project:{{SUBJECTPAGENAME}}

Goal status:

Has supergoal:{{1}}

Activity:

Is activity in project:{{SUBJECTPAGENAME}}

Is allocated to goal:{{1}}
The wiki source of “Project” template:

<noinclude>
This is the "Project" template.
It should be called in the following format:
<pre>
{{Project
|Project description=
|Project planned start=
|Project planned finish=
|Project purpose=
|Project objective=
|Project success criteria=
}}
</pre>
Edit the page to see the template text.
</noinclude>

= Project =
{| class="wikitable"
<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>! Planned start date</td>
</tr>
<tr>
<td>! Planned finish date</td>
</tr>
<tr>
<td>! Purpose</td>
</tr>
<tr>
<td>! Objectives</td>
</tr>
<tr>
<td>! Success Criteria</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
= Goals =
{{#ask:{{Is goal in project::{{PAGENAME}}}}}}{{#if:{{Has goal status::done}}|format=count}}/{{#ask:{{Is goal in project::{{PAGENAME}}}}}}{{#if:{{Has goal name}}|?Has goal name}}|?Has goal description}}|?Is top goal}}|?Has item urgency}}
| Has goal type |
| Has supergoal |
| Has sistergoal |
| Has goal status |
| Has achievement criterion |
| Has goal deadline |

| mainlabel= |
| format=template|template=Goal print |

```
= Activity ({{#ask: [[[Is activity in project: ::{{PAGENAME}}}]]|format=count}} =
{{#ask: [[[Is activity in project: ::{{PAGENAME}}}]]
| Has activity name
| Has activity description
| Is allocated to goal
| Has due date
| Has location
| Has instrument used
| Has individual person
| Has information needed
| Has repeat period
| Has repeat unit
| Has repeat until
| mainlabel= |
| format=template|template=Activity print |
```

```
= Temporal Rules =
{{#ask: [[[Is dependency in project: ::{{PAGENAME}}}]]
| Has dependency
| Has dependency type
| Has dependency with
| mainlabel= |
| format=template|template=Dependency print |
```

```
= Recent changes =
{{#ifeq: {{REVISIONID}} | {{#mms: {{REVISIONID}}}} }}

<headertabs/>

[[Category:Projects]]
</includeonly>

The wiki source of “Dependency” template:
<noinclude>
This is the "Dependency" template.

Edit the page to see the template text.
</noinclude><includeonly>
The wiki source of “Dependency print” template:

```wikipedia
<noinclude>
This is the "Dependency print" template.

Edit the page to see the template text.
</noinclude>

{{#switch: {{{2}}}}
| before = "Do "{{{1}}}" 'before' "{{{3}}}"
| after = "Do "{{{1}}}" 'after' "{{{3}}}"
| immediately before = "Do "{{{1}}}" 'immediately before' "{{{3}}}"
}

</includeonly>
```

The wiki source of “Goal tree” template:

```wikipedia
<noinclude>
This is the "Goal tree" template.
It should be called in the following format:

</noinclude>

{{#vardefine: rownode|{{#ask: [[Is goal in project::{{{1}}}]][[Has goal name::+]][format=count]]}}
{{#ask: [[Is goal in project::{{{1}}}]][[Has goal name::+]]
| ?Has goal name
| ?Has supergoal=Has supergoal | ?Has sistergoal=Has sister goal(s)
| mainlabel=-
| format=graph
| graphcolor=Yes
| graphlink=No
| graphname=Goals
| graphlegend=Yes
| graphlabel=No
| graphsize=20,8
| rankdir=BT
| nodecount={{#var: rownode}}
}

</includeonly>
```
The wiki source of “Person” template:

```
<pre>
{{Person
|Has role organization=
|Has email address=
}}
</pre>
Edit the page to see the template text.
</noinclude><includeonly>
| class="wikitable"
| Role in organization
| |{{#arraymap:{{{Has role organization|}}}},{|x|[|Has role organization::x|]}}
|-
! Email address
| |{{{Has email::Has email address|}}}
|-
! Phone number
| |{{#ask:[|Is phone of person::|?Has phone|?Has phone type|mainlabel=|format=template|template=Phone print|]}}
|-
! Activity involved by person
| |{{#ask:[|Has individual
person::SUBJECTPAGENAME|]}}|format=list}{}
}[|Category:Persons]]
</includeonly>

The wiki source of “Project form” form:

```
<noinclude>
This is the "Project form" form.
To create a page with this form, enter the page name below;
if a page with that name already exists, you will be sent to a
form to edit that page.
{{#forminput:form=Project form}}
</noinclude><includeonly>
<div id="wikiPreview" style="display: none; padding-bottom: 25px;
margin-bottom: 25px; border-bottom: 1px solid #AAAAAA;">
= Project =
{{for template|Project}}
| class="formtable"
| Description:
| {{field|Project description|rows=2}}
|-
! Planned start date:
| {{field|Project planned start|input type=datepicker|show reset
```
button|disable input field}}

| ! Planned finish date:
| | {{{field|Project planned finish|input type=datepicker|show reset button|disable input field}}}
| |
| ! Project Purpose:
| | {{{field|Project purpose|rows=3}}}
| |
| ! Project Objectives:
| | {{{field|Project objective|rows=3}}}
| |
| ! Success Criteria:
| | {{{field|Project success criteria|rows=3}}}
| }
| {{{end template}}}

= Add Goal =
| {{{for template|Goal|label=Add goal|multiple}}}  
| | class="formtable"
| ! Name:
| | {{{field|1|mandatory}}}
| -
| ! Description:
| | {{{field|2|rows=2}}}
| -
| ! Top goal:
| | {{{field|3|mandatory}}}
| -
| ! Urgency:
| | {{{field|4|input type= radiobutton|mandatory|default=required}}}
| -
| ! Goal type:
| | {{{field|5}}}
| -
| ! Supergoal(s):
| | {{{field|6|input type=combobox|autocomplete on property=Has goal name}}}
| -
| ! Sister goal(s):
| | {{{field|7|autocomplete on property=Has goal name}}}
| -
| ! Goal status:
| | {{{field|8|mandatory|default=not achieved}}}
| -
| ! Achievement criterion:
| | {{{field|9}}}
| -
Deadline:

| ![field|10|input type=datepicker|disable input field|show reset button]|
|---

```
''Do this{first activity}:'

{field|1|input type=combo|autocomplete on property=Has activity name}

''ordering:'

{field|2|input type=dropdown}

{field|3|input type=combo|autocomplete on property=Has activity name}
```

```
{end template}
```

```
for template|Dependency|label=Temporal rules|multiple

'''Do this(first activity):'''

{field|1|input type=combo|autocomplete on property=Has activity name}

'''ordering:'

{field|2|input type=dropdown}

{field|3|input type=combo|autocomplete on property=Has activity name}
```

```
{end template}
```
'Free text:'

```{{standard input|free text|rows=10}}
<headertabs/>
```}

The wiki source of “Tree” form:

```<noinclude>
This is the "Tree" form.
</noinclude>
<includeonly>
<div id="wikiPreview" style="display: none; padding-bottom: 25px; margin-bottom: 25px; border-bottom: 1px solid #AAAAAA;">
```{for template|Goal tree}}

| Project: |
```
```field|1|input type=combobox|autocomplete on category=Projects}))
```end template
```</includeonly>
```

The wiki source of “Person form” form:

```<noinclude>
This is the "Person form" form.
To create a page with this form, enter the page name below; if a page with that name already exists, you will be sent to a form to edit that page.
```#forminput:form=Person form}}
```</noinclude>
<includeonly>
<div id="wikiPreview" style="display: none; padding-bottom: 25px; margin-bottom: 25px; border-bottom: 1px solid #AAAAAA;">
```Person = ```{for template|Person}}
```class="formtable"

| Role in organization:
| {{field|Has role organization}} |
|---|
!
Email:
| {{field|Has email address}} |

{{end template}}

{{for template|Phone|label=Phone number|multiple}}

''Phone number:'' {{field|1}}
'''Type:''' {{field|2|input type=dropdown}}
{{end template}}

= Free text =
'''Free text:'''

{{standard input|free text|rows=10}}

<headertabs/>

{{standard input|summary}}

{{standard input|minor edit}} {{standard input|watch}}

{{standard input|save}} {{standard input|preview}}
{{standard input|changes}} {{standard input|cancel}}
</includeonly>
Appendix D: The HistoryOfPage Extension Source Code

Source code of the HistoryOfPage extension:

```php
<?php
/**
 * @get the major history of a page
 * @return links
 * (c)2011 M. Deri Taufan
 */

if (!defined('MEDIAWIKI')) {
    die('This file is a MediaWiki extension, it is not a valid entry point');
}

$wgExtensionCredits['parserhook'][] = array('path' => __FILE__,
                                             'name' => 'HistoryOfPage',
                                             'description' => 'The extension that returns link(s) of non-minor changes to a page',
                                             'version' => 0.1,
                                             'author' => 'M. Deri Taufan',
                                             'url' => 'http://wiki.science.ru.nl/mms',
);

# Define a setup function
$wgHooks['ParserFirstCallInit'][] = 'hpParserFunction_Setup';
# Add a hook to initialise the magic word
$wgHooks['LanguageGetMagic'][] = 'hpParserFunction_Magic';

function hpParserFunction_Setup ( & $parser ) {
    $parser->setFunctionHook( 'mms', 'hpParserFunction_Render' );
    return true;
}

function hpParserFunction_Magic ( & $magicWords, $langCode ) {
    $magicWords['mms'] = array( 0, 'mms' );
    return true;
}
```
function hpParserFunction_Render($parser, $param1='') {
    $revPage = ''; $outputLink = ''; $pageTitle = ''; $timeband = 0;

    $protocol = $_SERVER['HTTPS'] == 'on' ? 'https' : 'http';
    $url = $protocol . '://' . $_SERVER['HTTP_HOST'];

    $dbr = wfGetDB(DB_SLAVE);
    $res = $dbr->selectRow('revision', array('rev_page'),
                           'rev_id=' . $param1, __METHOD__,
                           array('ORDER BY' => 'rev_id ASC'));
    $revPage = $res->rev_page;

    //query the page title based on page_id
    $res = $dbr->selectRow('page', array('page_id', 'page_title'),
                           'page_id=' . $revPage, __METHOD__,
                           array('ORDER BY' => 'page_id ASC'));
    $pageTitle = $res->page_title;

    //query timestamp
    $res = $dbr->select('revision', array('rev_timestamp', 'rev_id'),
                        'rev_page=' . $revPage . ' AND rev_minor_edit=0',
                        __METHOD__,
                        array('ORDER BY' => 'rev_id ASC'));

    $amboxStyle = 'ambox-content';
    foreach ($res as $row) {
        //if ($outputLink=='') {
        //    $outputLink = '<table class="ambox ambox-
        //    notice"><tr><td class="mbox-image"></td></tr></table>"
        //} else {
        if ($amboxStyle == 'ambox-style') {
            $amboxStyle = 'ambox-content';
        }
        else if ($amboxStyle == 'ambox-content') {
            $amboxStyle = 'ambox-style';
        }else {
            $outputLink = '<table class="ambox ambox-
            notice"><tr><td class="mbox-image"></td></tr></table>"
        }
        //
    }
}
} $outputLink = $outputLink."<table class="ambox
$amboxStyle">"."<tr><td class="mbox-image"><div><b>T$timeband</b></div></td><td class="mbox-text">"."$url="/mms2/index.php?title=$pageTitle&diff=prev&oldid=$row->rev_id (".$wfTimestamp(TS_RFC2822,$row->rev_timestamp).")</div></td></tr></table>"; /
$timeband = $timeband+1;
}

return $outputLink;
}