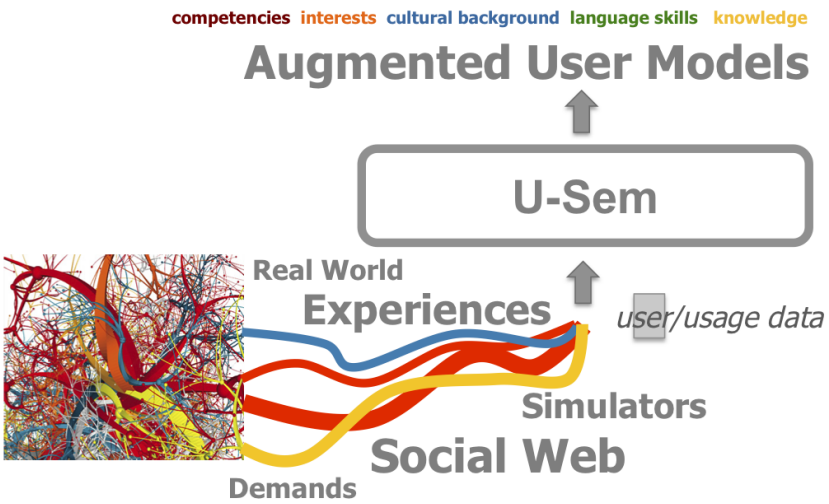
U-Sem, a framework for augmented user and context modeling

Master's Thesis



Borislav Todorov

U-Sem, a framework for augmented user and context modeling

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Borislav Todorov
born in BIRTHPLACE

Web Information Systems
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://wis.ewi.tudelft.nl

U-Sem, a framework for augmented user and context modeling

Author: Borislav Todorov
Student id: 123456789
Email: any@email.com

Abstract

This document describes the standard thesis style for the Software Engineering department at Delft University of Technology. The document and it's source are an example of the use of the standard LaTeX style file. In addition the final appendix to this document contains a number of requirements and guidelines for writing a Software Engineering MSc thesis.

Your thesis should either employ this style or follow it closely.

Thesis Committee:

Chair:	Prof. dr. ir. A.B.C. Een, Faculty EEMCS, TUDelft
University supervisor:	Dr. ir. A.B.C. Twee, Faculty EEMCS, TUDelft
Committee Member:	Ir. A.B.C. Drie, Faculty EEMCS, TUDelft

Preface

A place to put some remarks of a personal nature.

Borislav Todorov
Delft, the Netherlands
November 13, 2012

Contents

Preface **iii**

Contents **v**

List of Figures **vii**

1 Introduction **1**
1.1 Motivation 1

2 System Requirements **3**
2.1 Requirements gathering 3
2.2 Feature prioritization 4

Bibliography **9**

3 Plug-in Environment **11**
3.1 Requirements 11
3.2 Background 12
3.3 Design 12
3.4 Implementation 12
3.5 Validation 12

Bibliography **15**

4 Conclusions and Future Work **17**
4.1 Contributions 17
4.2 Conclusions 17
4.3 Discussion/Reflection 17
4.4 Future work 17

A Glossary **19**

B Requirements and Guidelines **21**
B.1 Requirements 21
B.2 Guidelines 22

C Prioritization questionnaire **23**

List of Figures

1.1 Flow chart defining the sequence of actions taken in order to complete the project. 2

2.1 The value distribution of the 5 requirements in the U-Sem project. 6
2.2 The cost distribution of the 5 requirements in the U-Sem project. 7
2.3 This diagram shows the comparison of the value/cost ratio of the requirements. 8

3.1 The environment in which the system will operate. The components coloured in red build up the Plug-in feature. 12
3.2 Business model describing the process for adding new plug-in to U-Sem . 13
3.3 Business model describing the process for adding existing plug-in from repository to U-Sem 13

Chapter 1

Introduction

1.1 Motivation

1.1.1 Research questions

The main research question is the following:

How to facilitate the work of scientists and organizations interested in user modeling and analysis?

The following sub-questions articulate the problem:

1. **How to enable users to add custom functionality and change system's behaviour?** (plug-ins)
2. **Is it possible to develop a component that enables users to store and retrieve information in arbitrary data formats?** (Universal datastore)
3. **How to enable users to maintain their results up-to-date?** (Scheduling)
4. **How to enable multiple users to use the system simultaneously without interfering with each other and keeping their work and results protected?** (Multi User and privacy)
5. **How can real-world use cases benefit from this system?**

1.1.2 Contributions

1.1.3 Organization of the work

There are many possible ways to approach that. One can use the waterflow model, incremental, ...

In this work we will use the incremental model because of the many advantages it brings. We design and implement one feature at a time, providing a working system after each iteration.

The last thing that has to be considered before launching the next phases of the project is to decide on the order in which the features will be implemented. This issue is addressed in the next section.

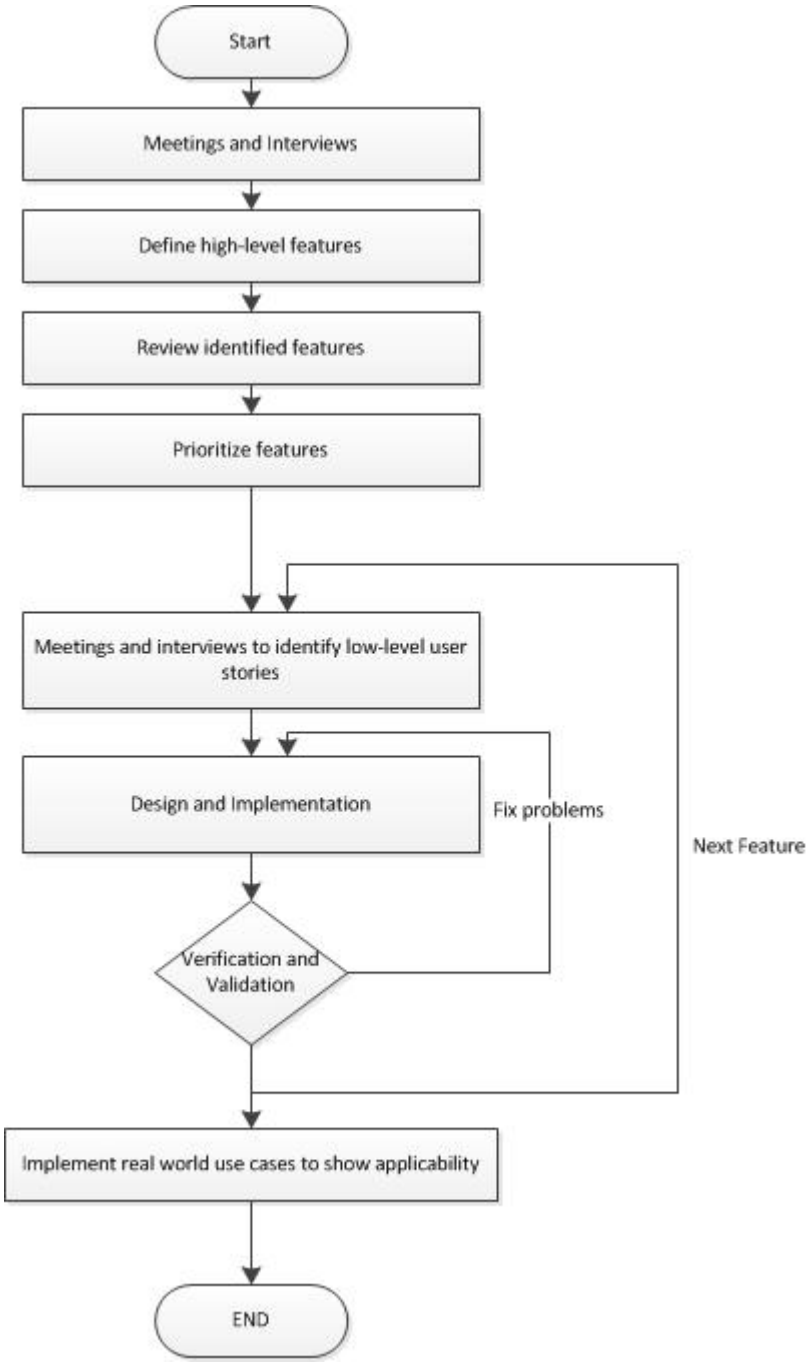


Figure 1.1: Flow chart defining the sequence of actions taken in order to complete the project.

1.1.4 Organization of the thesis

Chapter 2

System Requirements

This chapter describes the process of identifying the main needs of the stakeholders and defining the scope of the project. **TODO**

2.1 Requirements gathering

In this section we aim to elicit the needs of the users and structure them into high-level system features that will later be designed and implemented.

2.1.1 Stakeholders

Stakeholders are the people that are have some kind interest in the project. They are the ones that will be affected by the project and thus they are the people that will be the source for the characteristics of the system we have to build. We identified the following as the main stakeholders of the project:

- Scientists - people that develop algorithms and approaches for user modeling and analysis and allow other scientits and users to access them.
- ImReal - system that want to use the services provided by the scientists and has its own requirements
- RDFGears - provide the workflow engine and want to reuse some of the things.

2.1.2 Interviews

Once we had identified the stakeholders of the project we started to think how to extract the requirements from them. Literature suggest a wide variety of possible approaches []. However, there is one technique that proves to be really effective and it is the semi structured interviews [1]. In order to perform it we prepared some important questions that we wanted to know and then we did discussions **TODO**

2.1.3 Identified features

We analysed carefully all the raw information that we gather from the interviews and we identified several high-level features. We presented them to the stakeholders and

after some discussions we ended up with the following final features that the system should provide:

- **Multiuser support and Access control** The system should allow access to multiple users simultaneously. It should also provide access control mechanism that deals with the following issues:
  User types and authorization - What anonymous users are able to do and what registered user are able to do?
  Information privacy - Users should be able to protect private or sensitive information.
  Sharing and collaboration - How can users work together and reuse each other's work?

- **Plug-in environment** The system should enable scientists to extend it by plugging in custom logic such as RDFGears functions and other functional components. Users should be able to manage(add/update/remove) this custom logic runtime(without restarting the system). This process should not affect the work of other users.

- **Scheduling** The system should provide functionality that enables users to schedule and monitor the execution of workflows.

  TODO: What should be made clear is what events are available in scheduling. For example, configure to run on a specified date/time/interval. Or based on the completion of another component or workflow. Or based on the outcome (true/false) of a component, such that you can trigger it based on whether or not you found something in a crawl.

- **Universal data storage**

  Many of the workflows need to store various types of data(e.g. intermediate and final results). Therefore, the system should provide a mechanism that enables storage and retrieval of information in arbitrary data formats.

- **Integration with Hadoop** The amounts of information that have to be processed in the system can sometimes be huge. It is critical that this information is processed efficiently and Hadoop is often the solution for that. Therefore, the system should provide functionality that enables easy integration with Hadoop.

2.2 Feature prioritization

Having already defined the requirements we have to define the order in which we are going to design and implement them. Requirements(features) prioritization is the process of determining the order in which candidate requirements or features of a software product should be implemented. The criteria to order the requirements can vary a lot, for example one can consider the requirements' importance, value, cost, risks or views of stakeholders etc. Scientists and other systems(ImReal) that depend on U-Sem already need the functionality and thus we are mainly concentrated to deliver the most essential functionality as early as possible. Having this in mind, the criteria

we choose for the prioritization is the value(importance) of each functionality as well as the time required for implementation(cost).

Requirements prioritization is a relatively old research topic and there are numerous approaches that are available [4]. The most popular include Quality Function Deployment (QFD) the Analytical Hierarchy Process(AHP), the cost-value approach proposed by Karlsson, Wiegers' method, as well as a variety of industrial practices such as team voting, etc. However, literature also suggests that there is no perfect solution for this problem and the applicability of each approach depends heavily on the particular situation it is used.

For our project, we choose the Karlsson's Cost-Value approach that makes use of the analytic hierarchy process [1]. This decision was based on several factors. Firstly, it uses the exact criteria that we are interested in(value and cost). Secondly, it is especially suitable for prioritizing a small number of requirements[1], which is our case. And last but not least, it is a proven and widely used [3].

2.2.1 Requirements prioritization using the Cost-Value Approach

In this section we will describe step by step the process of prioritizing requirements using the Cost-Value approach. The process consists of three distinct steps. The following sub-sections will address these steps.

Value assessment

In the Requirements gathering section we identified 5 high-level requirements(features) that cover the main functionality of the system. In this step we are using the AHP's pairwise comparison method in order to assess the relative value of the candidate requirements. We asked a group of four experienced project members to represent customers views. We instructed them on the process and asked them to perform pairwise comparisons of the candidate requirements based on their value(importance). For the comparison criteria we used the one defined in []. Fig.. Apendix 1 shows the form that they were asked to fill.

We let the participants to work alone, defining their own pace. We also allowed them to choose the order of the pair's comparison. Discussions were also allowed. When all participant finished the pairwise comparison, we started to calculated the value distribution. However, first we calculated the the consistency indices of the pairwise comparisons. According to [**?**] values lower than 0.10 are considered acceptable and according to [1] values around .12 are commonly achieved in the industry and can also be considered acceptable. The calculation showed that two of the participants has indeces higher than .23 which indicates serious inconsistencies. Therefore, we asked them to revise their answers and the results were around 0.12 **TODO**

	Stakeholder 1	Stakeholder 2	Stakeholder 3	Stakeholder 4
Consistency ratio	0.04	0.23	0.13	0.26

Table 2.1: The initial consistency ratios for each of the stakeholders.

	Stakeholder 1	Stakeholder 2	Stakeholder 3	Stakeholder 4
Consistency ratio	0.04	0.12	0.13	0.11

Table 2.2: Consistency ratios for each of the stakeholders after refinement.

Once we had achieved satisfying results we calculated the distributions. We outlined the candidate requirements in a diagram and presented the results to the project members. Each requirement's determined value is relative and based on a ratio scale. Therefore, a requirement whose value is calculated as 0.20 is twice as valuable as a requirement with a value of 0.10. Additionally, the sum of the values for all requirements equals 1. This means that a requirement with a value of 0.10 provides 10 percent of the value of all the requirements. You can see that the "plug-in environment" requirements is the most valuable. It is followed by the "data-store" and "Access control". At the bottom, providing considerably less value, are the "scheduling" and "hadoop" requirements.

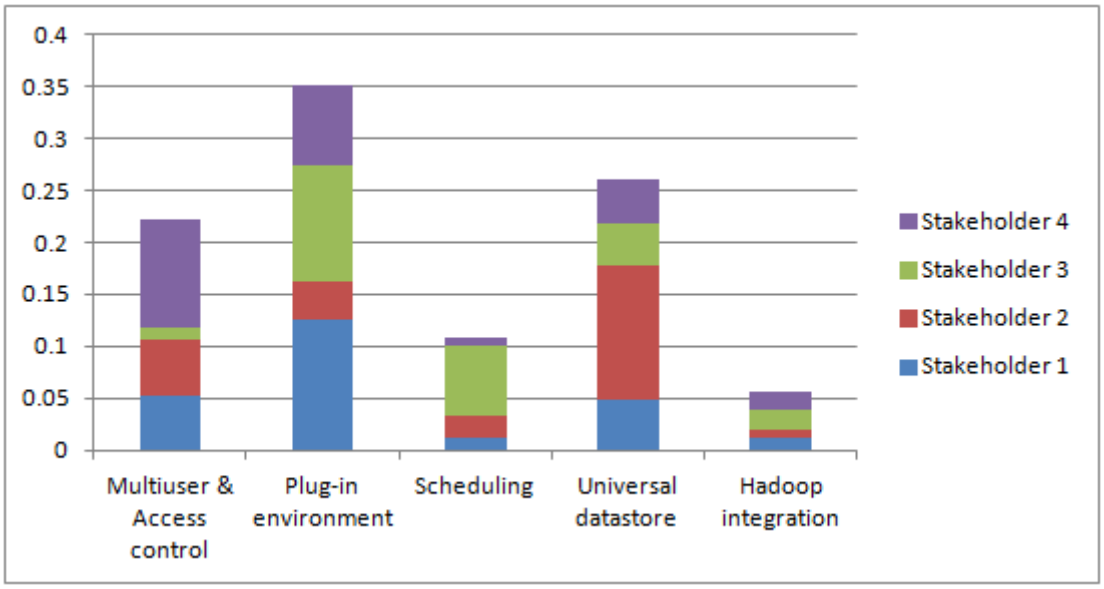


Figure 2.1: The value distribution of the 5 requirements in the U-Sem project.

Cost assessment

Value on its own is not enough to estimate the priority of the requirements. The problem is that a requirement with high value may also be costly to implement and also take a lot of time. Our aim is to provide the most value as quick as possible. Thus, requirements that provide a bit less value but on the other hand are easy and quicker to implement might be the better choice. In this step, we measure the distribution of cost between the requirements. In order to do that, we asked the software engineer responsible to build the system to perform AHP's pairwise comparison to estimate the cost of implementing each of the candidate requirements. The process absolutely the

same as the one described in the previous section. However, this time the requirements are compared based on their cost rather than their value.

Once the comparison was finished, we measured the consistency index of the answers. The result was **0.029** which is completely acceptable and there was no need for further refinement.

Finally, we used the AHP technique to calculate the cost distribution of the requirements. We outlined the results in a digram FIG. Again, the determined values are relative and based on a ratio scale.

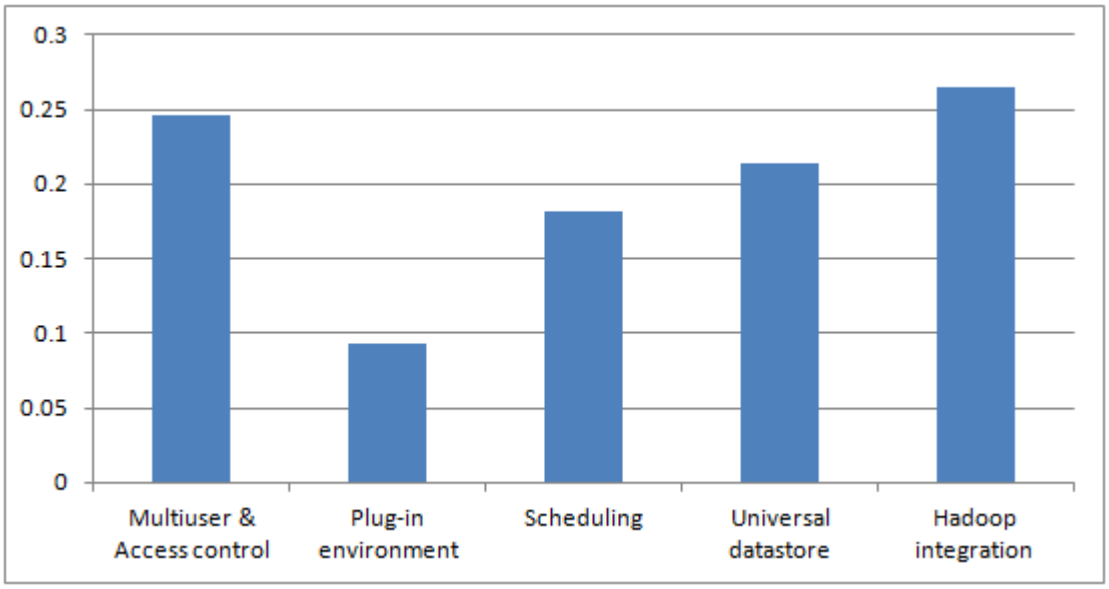


Figure 2.2: The cost distribution of the 5 requirements in the U-Sem project.

Cost-Value analysis

Once we have the value and cost value distribution of the requirements we can establish the order in which to implement the requirements. The decision is based on the value/cost ratio of each requirement. Figure[] shows the results. As you can see, the the plug-in environment is the definite winner and the hadoop integration provides very little benefit compared to the cost to implement it.

Discussion

Our observations about the stakeholders which were asked to perform the pairwise comparisons found the method intuitive and easy to understand. However, they also found performing the pairwise comparison to be a bit tedious. They sometimes got distracted and gave inconsistent results which was indicated by the consistency check. However, this was easily fixed by revising the answers and we reached a level of consistency that is considered acceptable.

Another disadvantage of this approach that affected our work is the fact that the method takes no account of interdependencies between requirements. However, this

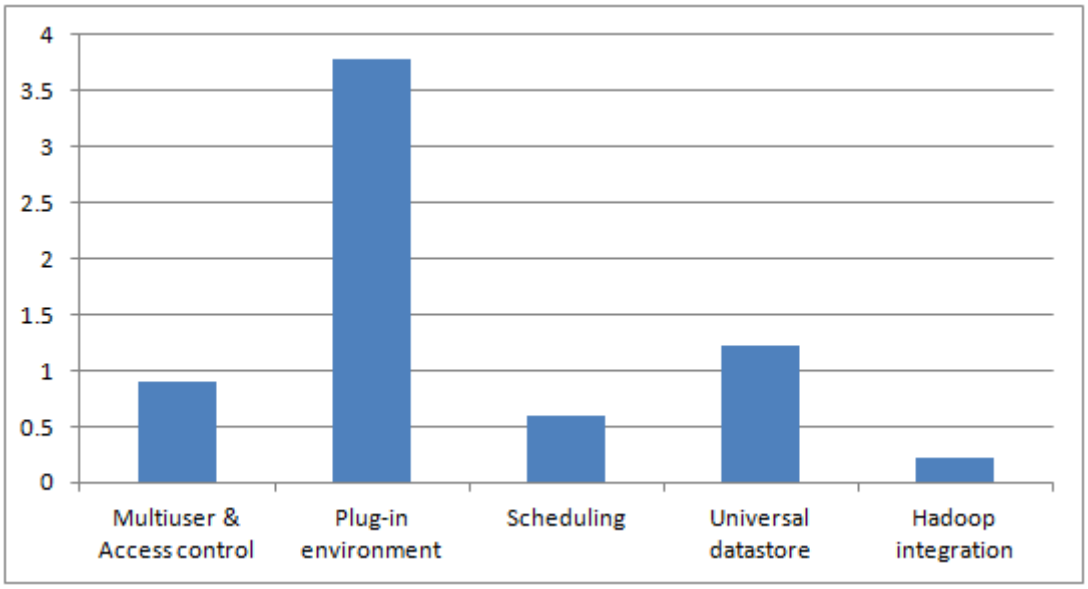


Figure 2.3: This diagram shows the comparison of the value/cost ratio of the requirements.

issue did not affect the project significantly because the project high-level features are loosely coupled and the only one that required some attentions was the "access control". **TODO**

Bibliography

[1] Oscar Dieste, Natalia Juristo, and Forrest Shull, Understanding the Customer: What Do We Know about Requirements Elicitation?

[2] Karlsson, J., Ryan, K. (1997). A Cost-Value Approach for Prioritizing Requirements, IEEE Software September/October 1997, 67-74.

[3] J Karlsson, C Wohlin, B Regnell - Information and Software Technology, 1998 - Elsevier

[4] Frank Moisiadis, THE FUNDAMENTALS OF PRIORITISING REQUIREMENTS

Chapter 3

Plug-in Environment

This chapter describes the process of design, implementation and validation of the "Plug-in Environment" feature.

3.1 Requirements

This section described the detailed requirements that regard this feature. In order to identify them we performed additional interviews with the stakeholders. Bellow you can find the requirements described as user stories.

TODO Example full usecase

MAYBE say that currently the main problem is that the functions are hard-coded in the workflow engine and cannot be modified without recompiling

3.1.1 (

What is plug-in?) custom functionality = Functions + workflows + ... **TODO add pictures how plug-ins work**

3.2 User stories

In this section we are formally expressing all the requirements into user stories so that it is more clear and easy to validate.

- Add

  As a user, I want to easily add custom functionality to U-Sem.

  As a user, I want to easily add custom functionality provided by other scientist to U-Sem.

  As a user, I don't want to be affected by other users adding new custom functionality.

  As a user, I don't want other users to be able to add, delete or modify my custom functionality.

- Management

As a user, I want to see the list of the custom functionality that is added U-Sem.

As a user, I want to delete custom functionality that has been added to U-Sem.

As a user, I want to be notified about any new versions of the custom functionality provided by other scientists.

As a user, I want to update existing custom functionality with the newest version provided by other scientists.

How to edit workflows(configuration) from plug-in?

- Usage

  As a user, I want to use the added custom functionality in the workflow engine.

  As a user, I don't want other users to see what is my custom functionality and use them.

3.3 Background

OSGI, Eclipse ....

3.4 Design

3.4.1 Environment

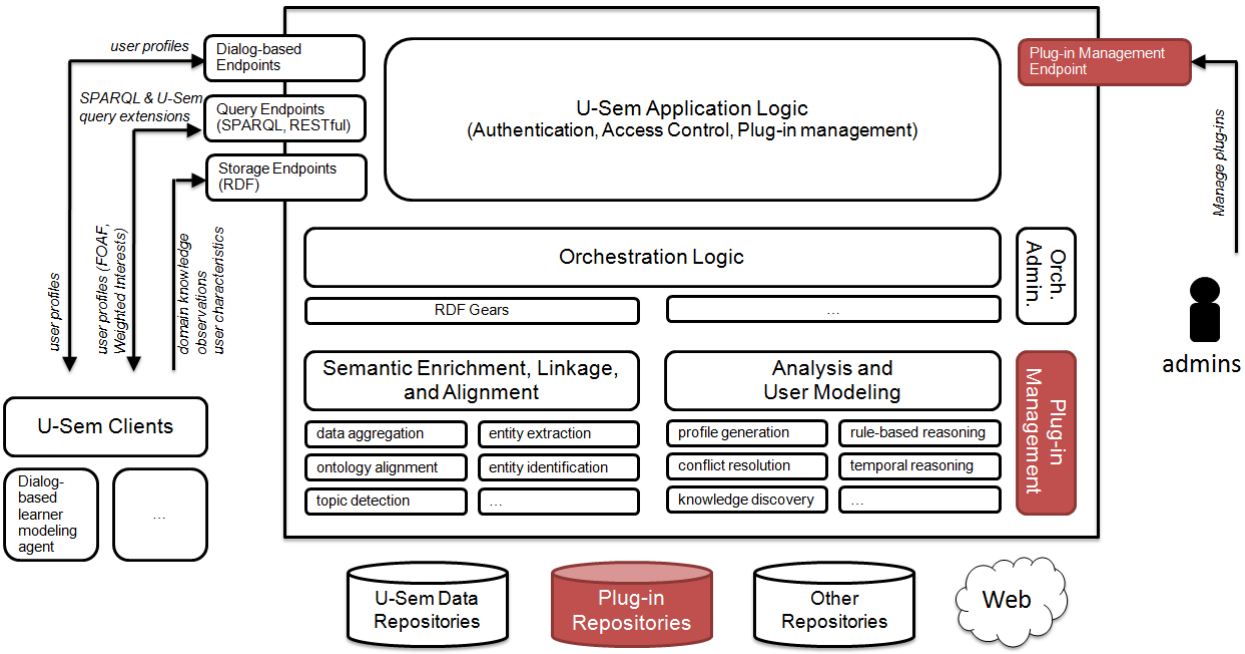


Figure 3.1: The environment in which the system will operate. The components coloured in red build up the Plug-in feature.

Plug-in Management

Plug-in Management endpoint

Plug-in Repositories

This section describes how the various component that build this feature interact between each other in order to satisfy the user requirements.

3.4.2 Process modelling

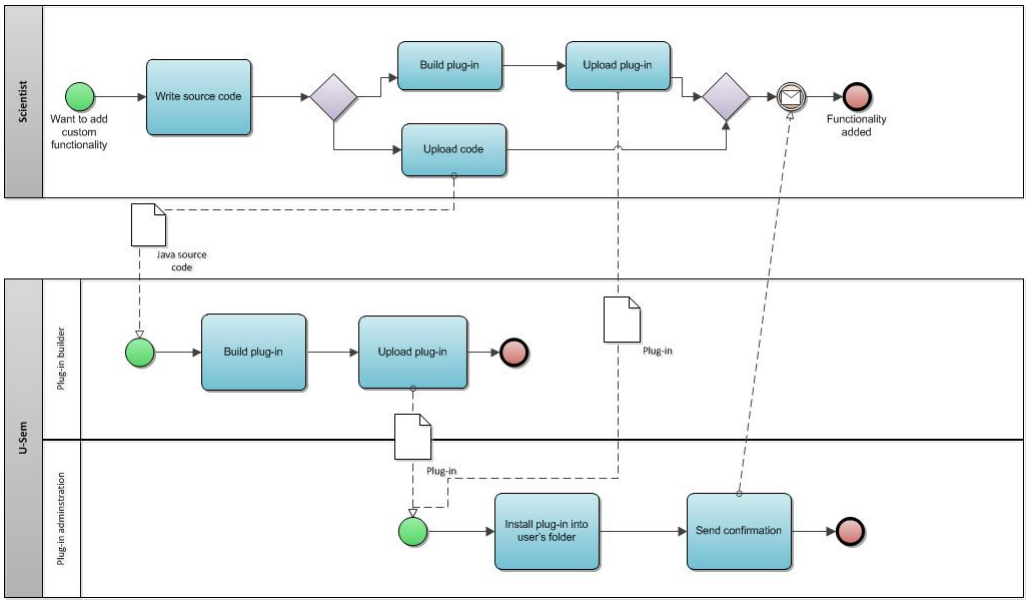


Figure 3.2: Business model describing the process for adding new plug-in to U-Sem

3.4.3 Functional viewpoint

3.4.4 Deployment viewpoint

3.5 Implementation

3.6 Validation

In order to make sure that the implemented feature satisfies the requirements we performed an experiment. We tried to extract outside as a plug-in a functionallity that is currently hardcoded into the workflow engine outside

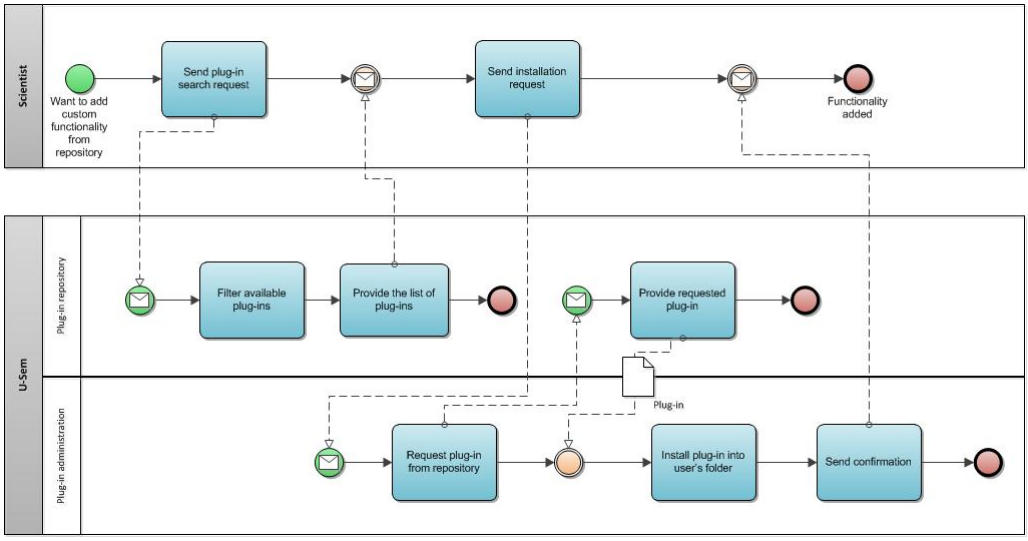


Figure 3.3: Business model describing the process for adding existing plug-in from repository to U-Sem

Bibliography

[1] Karlsson, J., Ryan, K. (1997). A Cost-Value Approach for Prioritizing Requirements, IEEE Software September/October 1997, 67-74.

Chapter 4

Conclusions and Future Work

This chapter gives an overview of the project's contributions.

4.1 Contributions

4.2 Conclusions

4.3 Discussion/Reflection

4.4 Future work

Appendix A

Glossary

In this appendix we give an overview of frequently used terms and abbreviations.

foo: ...

bar: ...

Appendix B

Requirements and Guidelines

This chapter details some requirements and guidelines for MSc theses submitted to the Software Engineering Research Group.

B.1 Requirements

B.1.1 Layout

- Your thesis should contain the formal title pages included in this document (the page with the TU Delft logo and the one that contains the abstract, student id and thesis committee). Usually there is also a cover page containing the thesis title and the author (this document has one) but this can be omitted if desired.
- Base font should be an 11 point serif font (such as Times, New Century Schoolbook or Computer Modern). Do not use sans-serif fonts such as Arial or Helvetica. *Sans-serif type is intrinsically less legible than seriffed type*
- The final thesis and drafts submitted for reviewing should be printed double-sided on A4 paper.

B.1.2 Content

- The thesis should contain the following chapters:
  - Introduction.
    Describes project context, goals and your research question(s). In addition it contains an overview of how (the remainder of) your thesis is structured.
  - One or (usually) more "main" chapters.
    Present your work, the experiments conducted, tool(s) developed, case study performed, etc.
  - Overview of Related Work
    Discusses scientific literature related to your work and describes how those approaches differ from what you did.
  - Discussion/Evaluation/Reflection
    What went well, what went less well, what can be improved?

- Conclusions, Contributions, and (Recommendations for) Future Work
- Bibliography

B.1.3 Bibliography

- Make sure you've included all required data such as journal, conference, publisher, editor and page-numbers. When you're using BibTeX, this means that it won't complain when running `bibtex your-main-tex-file`.
- Make sure you use proper bibliographic references. This especially means that you should avoid references that **only** point at a website and not at a printed publication.

  For example, it's OK to add a URL with the entry for an article describing a tool to point at its homepage, but it's not OK to just use the URL and not mention the article.

B.2 Guidelines

- The main chapters of a typical thesis contain approximately 50 pages.
- A typical thesis contains approximately 50 bibliographic references.
- Make sure your thesis structure is balanced (check this in the table of contents).

  Typically the main chapters should be of equal length. If they aren't, you might want to revise your structure by merging or splitting some chapters/sections.

  In addition, the (sub)section hierarchies with the chapters should typically be balanced and of similar depth. If one or more are much deeper nested than others in the same chapter this generally signals structuring problems.
- Whenever you submit a draft of your thesis to your supervisor for reviewing, make sure that you have checked the spelling and grammar. Moreover, *read it yourself at least once from start to end, before submitting to your supervisor*.

  Your supervisor is not a spelling/grammar checker!
- Whenever you submit a second draft, include a short text which describes the changes w.r.t. the previous version.

Appendix C

Prioritization questionnaire

In this appendix we provide the questionnaire used for the prioritization of the requirements.

TODO