Planning

Useful feedback in the Ampersand parser

Maarten Baertsoen and Daniel S. C. Schiavini

Open Universiteit Nederland, faculteit Informatica
T61327 - Afstudeerproject bachelor informatica

November 3, 2014

Version 1.0

Contents

List of Figures

List of Tables

1 Introduction

1.1 Identification

This document contains the planning for the execution of the graduation project 'Useful feedback in the Ampersand parser'. This planning gives the high-level requirements, the risks and a timeline for the project. It is the milestone product of the project phase 2. As such, the planning provides the steps for reaching the project objectives, and provides criteria that are used to validate and accept the results of the graduation.

This document is part of the graduation project of the computer science bachelor at the Open Universiteit Nederland. The project 'Useful feedback in the Ampersand parser' is assigned to the students Daniel Schiavini and Maarten Baertsoen, with support of the supervisor Dr. Bastiaan Heeren and examiner Marko van Eekelen. The assignment is given by professor Stef Joosten, who researches how to further automate the design of business processes and information systems by the development of the Ampersand project.

1.2 Goal of this document

The main goal of this document is to capture the taken decisions and agreements around the execution, the management and the control of the project. In order to make the targets clear, allowing the project team to put the right focus, the project context is also depicted in the document.

The document describes the current situation and the issues it presents, making clear why the project has been started. The purpose is thus to describe the management approach and the describe the aimed solution in high-level, as well as changes, risks and problems that might occur.

1.3 Document overview

An introduction is given is this chapter. Afterwards, a general description of the project is given in section 2. Then, in section 3, strategies are proposed for the acquisition of knowledge. In section 4 the project approach is explained and the management strategy is given in section 5.

Assumptions and constraints are given in section 6. Strategies for risk management are then proposed in section 7, possible interferences with other projects and an overview of known risks, including the mitigation approach, at the beginning of the project.

Before the actual realization approach is described, a clear view on the project requirements is set in section 8.

Details of the development techniques, including design, implementation and testing, are given in the project realization strategies of section 9. Issue management is explained in section 10. The strategy for integration of the released software is given in section 11, while the documentation strategy is given section 12 and the used tools and methodologies are given in section 13.

Finally, in the appendix, a glossary of terms, definitions and abbreviations is given, just as a list of references.

2 Project description

2.1 The Ampersand project

In November 2003, the Business Rules Manifesto[1] was written, with the main purpose of declaring independence for business rules in the world of requirements. The manifesto supports the vision of business rules as equivalent to requirements. This is considered a radical change on how people see the world of business architecture.

In December 2010, Stef Joosten, Lex Wedemeijer and Gerard Michels published the paper 'Rule Based Design', presenting the Ampersand approach. The approach puts the rules in the center, using them to define the business processes. Ampersand is named after the & symbol with the desire of realizing results for both business and IT, in an efficient and effective way.

In 2011, the Ampersand compiler was created as an open source project. Since then, the compiler has been improved and applied in both business and academic contexts. The Ampersand end-users write business rules in a specific language (ADL), and compile that specification into functional specification, documentation and working software prototypes. These rules are based on agreements between the different stakeholders.

The theory behind Ampersand has been throughly studied, and is based on mathematical concepts, e.g. Relational algebra and Tarski's axioms. Using this compiler, users write the requirements in ADL and generate all the system specification independent of the platform. The main advantage is that the requirements consistency and traceability are always correct (and even provable), from the lowest level up to the front-end. The requirements are presented to stakeholders in natural language, guaranteeing that any business expert who knows the context can validate the requirements. Figure 1 depicts the artifacts generated by the Ampersand compiler.

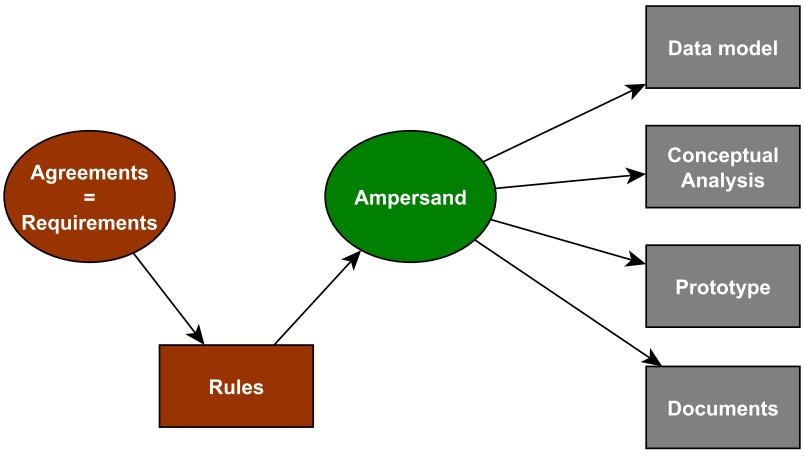

Figure 1: The Ampersand approach generates different artifacts based on the business rules

2.2 Project architecture and components

The compiler developed for the Ampersand research project runs in several steps. Thence the Ampersand compiler is also divided in several subcomponents:

Parser: This component receives the ADL code as input, and parses that code into a parse-tree (also known as P-structure).

Type checker: The Ampersand type checker receives the P-structure as input and converts it into a relational algebra format, suitable for manipulation (also known as A-structure or ADL-structure). The semantics of ampersand are expressed in terms of the A-structure.

Calc: The Calc component receives the A-structure as input, and manipulates it according to the research rules, generating the functional structure (also known as F-structure). The F-structure contains all design artifacts needed to write a specification and generate the output.

Output components: All design artifacts present in the F-structure are ready to be rendered. Several components use this data structure to generate the wished output. The output components currently implemented (and their output formats) are the following:

- Atlas (HTML interface);
- Revert (Haskell source);
- Query (prototype generation);
- Documentation Generator (Pandoc structure).

The complete architecture is depicted in Figure 2. The part of this architecture relevant for this project is depicted in Figure 3.

2.3 Current situation

The end-to-end process of the ampersand project, from compiling towards the generated artifacts is correct, however there is a major improvement topic identified in the first step, the parsing of the input scripts.

One of the main complaints from users is the quality of the errors generated by the Ampersand parser making it though for the end users to correct faulty ADL statements. Since the beginning of the project, the parser subcomponent never received special attention, and it has not been analyzed for improvements.

In order to generate better, useful and to the point error messages, it is assumed that a complete refactoring of the parser will be necessary. The main challenge is to choose the correct kind of architecture and libraries in order to generate the most user-friendly messages possible.

Besides the main project task of improving the parser's feedback, a list of user wishes has accumulated over the years. The main customer and other users would appreciate if as many wishes as possible could be fulfilled.

2.4 Goals of the project

The main objective for the graduation project is to implement useful feedback in the Ampersand parser. In order to achieve this goal, the following research & analysis activities will take place:

- Analysis of user-friendly messages in compilers;

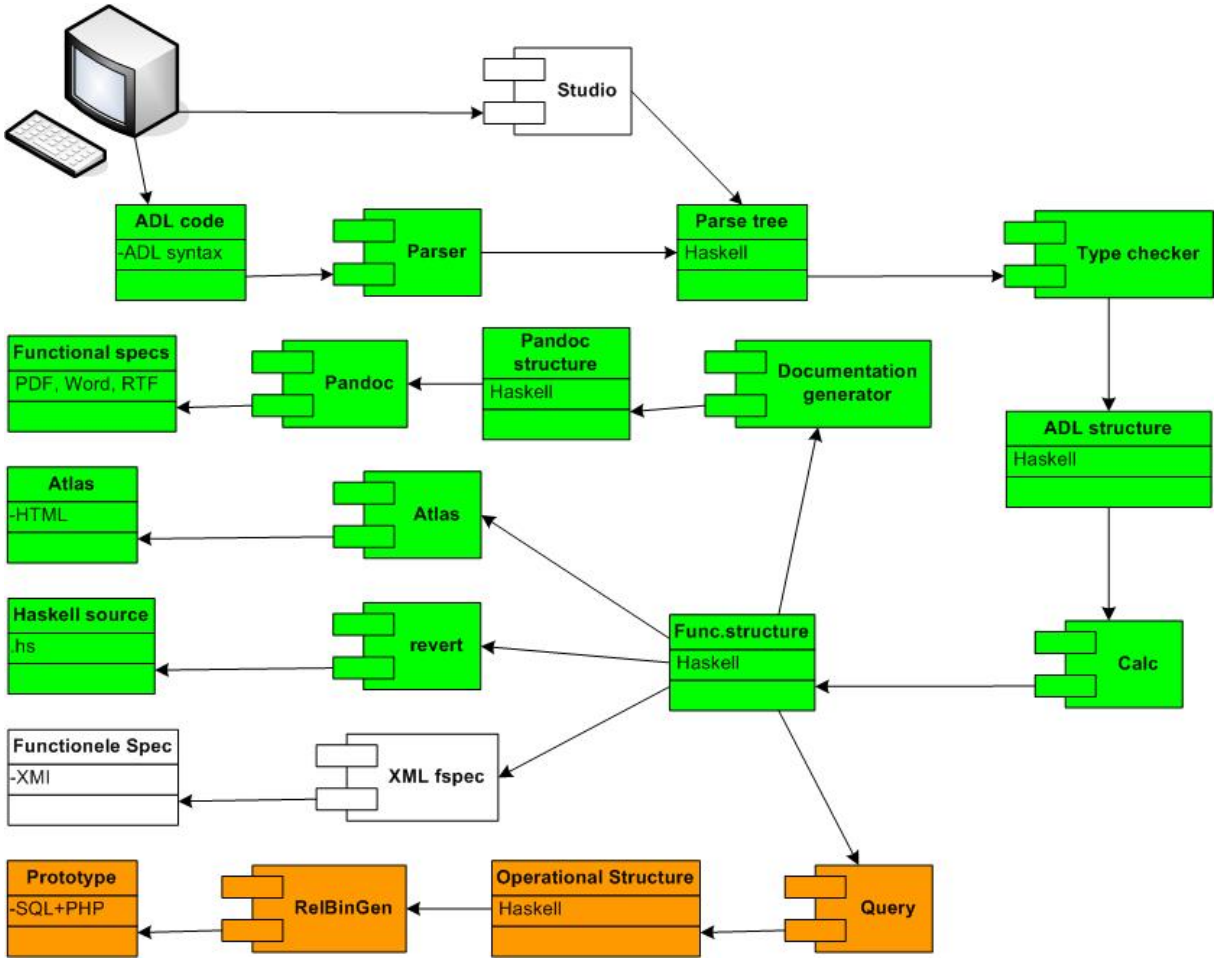

Figure 2: Architecture of the project, showing where the parser fits in the Ampersand system

The components in green background are part of the Ampersand compiler. Components in orange are part of the Ampersand Prototype compiler. Finally, components in white background are future components, not yet implemented.

- Comparison of different Haskell parsing libraries (also for pretty-printing);

Additionally, the following activities may also take place:

- Researching tools and techniques in Haskell for improving the software quality (e.g. testing and error messages);
- Analysis of the current development environment in relation with software engineering principles such as continuous delivery/integration;
- Recommending improvements for the overall software quality;

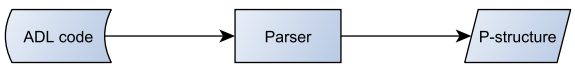

Figure 3: Relevant data flow for the Ampersand parsing component

In case the new parser is successfully implemented and accepted, while the project members still have time budget available, the list of open user wishes issues can be addressed. Some of these wishes are substantial, so that most of them cannot be fulfilled during the graduation project. The current list of open issues has been provided[2], although it must be clear that the issues are strictly seen as lower priority. See also section 8 for the list of high-level requirements.

2.5 Project environment

The Ampersand project is used in the following environments, for different users:

Research: The Ampersand project is part of a research domain on the use of business rules for software design;

Academic: Ampersand is used as main tool in the course 'Ontwerpen met bedrijfsregels' (code T18321) from the Open Universiteit Nederland;

Business: The compiler is used in business environments to design and develop real world business software.

2.6 Critical success factors

The following factors are critical for the project's successful completion. Each critical success factor is covered by one or more measures outlined in this project plan:

Maintainability: the code shall at least be as maintainable as the current Ampersand code. It is known however that maintainability is hard to measure (see also section 7).

Production code: the implemented functionalities shall be integrated into the master branch for production use;

Users context: in order to provide useful feedback, the user context wherein the Ampersand compiler is used needs to be well understood, and the improvements must have extra user value. Just like maintainability, useful feedback is a hard to measure topic, this risk is listed in section 7 and will be addressed in phase 3c (research context).

Communication: Independent on how good the project results are, if the solution is put into production without proper communication and documentation leading to confused and demotivated users, the project will be perceived as a failure. Although the project aims to add the feedback in the ampersand parser as transparent as possible, the final customer and user perception remains a critical success factor of this project;

The syntax of the Ampersand grammar is written in the the EBNF notation. Any changes to the syntax must be documented according with this notation. The notation can also be added as comment in the source code, in order to make clear that all the grammar is implemented completely and correctly.

2.7 Objectives and commitments

Besides the project goals described in subsection 2.4 and the customer goals described in subsection 2.6, the project members declare herewith to have the following objectives and commitments fulfilled by the end of this graduation project:

Customer: The primary objective is to deliver a well-working and maintainable piece of software that will help the customer.

Users: Another important goal is to make the software more user friendly, and so improve the usability.

Knowledge: Building up knowledge is the main reason why one starts a bachelor study. As such it is important do learn more about functional programming, Haskell, compilers, business rules and research in general.

University: Hopefully the final thesis will be of use for the university and other students.

Graduation: As this is a graduation project, it is natural to have the graduation as an important objective.

3 Knowledge acquisition

3.1 Domain & techniques

In the project phase 3a, the project members shall investigate the domain and the techniques available. Each project member will do a separate analysis, and will present a separate document with their findings. The planning of the investigation is presented a section per project member.

3.1.1 Haskell parsing libraries & User-friendly error messages

This part of the research is going to be executed and documented by Daniel Schiavini. It is divided in two interrelated parts:

Haskell parsing libraries: The current Ampersand parser is created with the Utrecht University parser combinator library (UU.Parsing). This library uses monads and combinators and has quite some good features. There are, however, other libraries that can provide other feature sets.

The purpose of this research is to choose the library best suited to the development of the new Ampersand parser. Some of the libraries available are:

- Helium
- Parsec
- Polyparse
- Happy parser generator

There are also advantages and disadvantages on using monads and combinators, which must be researched beforehand. The result of this research could also be that the UU.Parsing library is the best choice. One big advantage for this library is that the current code base can be very helpful.

The baseline approach for researching the libraries will be to check whether they have all required features and their corresponding effectiveness. Also, existing tools built with the parsing library will be checked, specially the maintainability of the code and quality of the error messages.

User-friendly error messages The most important feature of the parser that will be built, is that it should generate user-friendly error messages. To understand what kinds of messages can be (and should be) generated, a research will be done on what good errors are and how to generate them.

This is also related to the choice of the parsing library, since the chosen library should support the generation of good errors without extensive effort. Coincidently, a good source of knowledge are the papers of the supervisor, Bastiaan Heeren, who has done his PhD thesis on the generation of top quality type error messages[4].

The results of this research will be reported in a separate document, and be added as an appendix of the thesis.

3.1.2 Business Rules & Ampersand Users

This part of the research is going to be executed and documented by Maarten Baertsoen.

The goal of the Ampersand project is to provide a methodology of defining business requirements using natural language. This research will analyze following topics:

The Ampersand methodology The Ampersand methodology provides a way to transform rules from business stakeholders into consistent functional specifications and a working software prototype. Within this section, the theoretical approach of the Ampersand Methodology will be investigated:

- Goals of the methodology
- The approach and rules
- How to use the methodology
- Usability, when to use the methodology
- Status and future road map
- Tools supporting the use of the ampersand methodology and their corresponding artifacts

The practical use of the Ampersand tools After a theoretical research of the domain, the practical use, including the tools, will be highlighted.

This topic will analyze how effective and useful the Ampersand Methodology is when correctly applied. An overview will be given regarding the way it is used:

- Getting started with the Ampersand methodology
- Requirement gathering, documentation and validation: the Ampersand way
- How to use the automatically generated artifacts
- Correctness and efficiency
- Ease of use
- The benefits and disadvantages
- Potential enhancement areas

3.2 Research context

After understanding more about the domain and the techniques involved with the project, the next step is to understand the context in which the project has been requested. Besides, the relation of this project with the main research should be made clear. In the project phase 3b, the focus is on the the use of business rules for defining IT systems.

Since the relevance of this project in the research can be only well defined later on, only a high-level planning can be given. This project part will be done in three steps:

- Understanding the research, by reading existing papers and other literature.
- Consulting, by formulating questions that can then be asked to the customer (Stef Joosten).
- Reporting the findings, in a separate section of the thesis.

3.3 Knowledge documentation

The project phases 3a and 3b will result in three reports. Each report should have around 5 pages and will be added to the bachelor thesis. The specific deliverables are described in section 12.

4 Project approach

Just like any other project, this project needs a fit-for-use approach. This section 'project approach' summarizes the project methodology that will be applied together with an exhaustive overview of the project planning, milestones and corresponding deliverables.

4.1 Project methodology

The main drivers to determine our project methodology is based on the kind of deliverables that needs to be produced as a result of this project, more specific, the actual technical realization of the project.

The technical realization can be grouped into 2 categories:

1. The introduction of a new error message approach to provide clear and useful error messages and warnings towards the end-users
2. Stand-alone modification points toward the Ampersand Parser based on:
   a) Reported issues
   b) Enhancements
   c) Extensions

Given the stand-alone aspect of the modification points, the use of an iterative approach is quite obvious as a backlog of topics is already available and the topics can be taken up independently of each other.

The error messages need somewhat more attention to guarantee that a solid error message architecture is built up. The project team will therefore realize a Proof-of-Concept to demonstrate the correct approach and base architecture. After validation of this PoC,

the project team will realize the error message architecture and actual implementation in an iterative approach. More information on the approach is described in section 9.

Within this project methodology, the following aspects are identified as important to guarantee a qualitative project within time, budget en corresponding to the agreed on project scope:

1. Project management
2. Knowledge acquisition
3. Project realization approach
4. Integration & release
5. Testing & validation
6. Communication
7. Documentation

All these approach topics are fully documented within this project plan.

4.2 Project phases

The project activities are grouped in several project phases. Each project phase has it own milestone, but each phase can start whenever there is sufficient information available from the previous preceding phase. This means that the team doesn't need to wait until a previous phase is finished.

Phase 2 - Task allocation and Project plan

The key foundation to guarantee a successful project is a well thought and appropriate project plan and corresponding project management techniques. The project plan will document the full approach that will be applied to manage the project to a successful delivery and satisfied customer.

Based on the individual learning goals of each project team member, specific tasks and roles will be assigned to them, allowing the development of these learning goals during the project. A detailed view on the allocated roles and tasks, will clarify the expectations towards the team members and will draw a clear view towards the project environment regarding who is doing what.

Phase 3a – Domains & Techniques

Project success has a direct interdependency with the actual knowledge of the project team regarding the system itself, the context and most importantly, the business drivers behind the system. This phase will allow to the project team members to acquire the necessary knowledge to fully understand the system, more precisely by investigation of the following topics:

1. The business goals of the system: the actual added value the system promises towards the end-users
2. Academic, business and system knowledge in the specialism to understand the way the system is used and to be able to use the system as it would be used in a real life situation

3. Specific terminology and professional language of the specialism
4. Relations towards other specialisms, organizations and systems
5. Important and relevant theoretical and practical evolutions in the specialism

This phase will be processed by the project team members individually.

Phase 3b – Research context

After gaining a thorough knowledge of the specialism and the current system, the project team will further analyze the specific drivers behind the project requirements. The higher the understanding of the goal of the project in relation to the specialism, including the current evolutions, and the way this project can contribute towards this evolution, the better the project team can focus on the topics that matter.

This phase will form the basis of the actual design and implementation phase and therefore, this phase be initiated before the start of the design & implementation phase but will continue simultaneously as additional topics will need to be investigated based on moving insights.

Phase 3c – Design & implementation

In phase 3c, the project team will start the in-depth analysis of the requirements and create a design for the new or changed functionality based on the research context fundamentals. The actual implementation and careful testing of the coding are part of this phase.

Special attention will be given to support both functional as non-functional requirements such as correct appliance of coding conventions and the creation of maintainable, durable and future-proof code.

Phase 3d – Project documentation

Although often neglected, or under-appreciated, clear and target group oriented documentation is crucial towards short and long term satisfaction of the customer. On a short term basis, the customer can have a wrong perception of the actual realization of the requirements within the project. On a long term, if each future system modification or design is hampered due to inconsistent and incomplete documentation, chances are low that the customer will embark again on a new project with the previous project team members, or in real life, with the company that delivered the project.

Several documentation deliverables, targeting a specific audience, are therefore listed in section 12 to assure the short and long-term customer happiness.

Phase 4 – Project closure and final project presentation

The final closure of the project will assure that all closing activities are correctly addressed and that all deliverables are handed over and accepted.

This phase will be closed by a thesis, containing the project overview, a retrospection of the project realization and execution together with the lessons learned and conclusions.

The formal closure is embodied by a final presentation.

4.3 Project planning, milestones and corresponding deliverables

The detail project planning is based on the project milestones as depicted in this section.

A commitment is given by the project team to adhere to the following milestone planning, officiated by the delivery of all their corresponding deliverables, grouped by their corresponding project phase:

1. Phase 2 - Task allocation and Project plan
   a) Delivery of the detailed project plan - 13/11/2014
   b) Detailed allocation of tasks within the project team - 13/11/2014
2. Phase 3a - Domains & Techniques
   a) Delivery of scientific article by Daniel Schiavini - 03/12/2014
   b) Delivery of scientific article by Maarten Baertsoen - 03/12/2014
3. Phase 3b - Research context
   a) Detailed report of the 'research context' - 29/04/2014
4. Phase 3c - Design & implementation
   a) Analysis & design documents - 30/03/2014
   b) Test report - 20/03/2014
   c) Source code - 27/03/2014
   d) The actual release of the realized product - 20/04/2014
   e) IT documentation - 20/03/2014
   f) User documentation & user manual - 20/03/2014
5. Phase 3d - Project documentation
   a) Project documentation document - 04/05/2014
6. Phase 4 - Project closure and final project presentation
   a) Project essay - 20/05/2014
   b) Project presentation - date to be jointly determined, final due date 29/05/2014

The project activities will be managed in a detailed project planning by the project team, and the status will be shared during project alignment meetings. Deviations, changes and issues within the project plan will be managed by the project team itself, in full openness towards the project stakeholders.

As long as the changes to the detailed project plan aren't compromising the milestone plan, the project is considered to be 'in control', the status will be green. If there are issues that can impact the qualitative and timely delivery of the milestones, but which are in control of the project team, the project status will be orange. If there is an impact (on scope, timing and quality) not in control anymore by the project team members, the project is considered to be 'at risk' and the status will be red. In this case, additional support from outside the project is needed.

Any deviation of the project plan (due to risks, delays, issues,...) that can have an impact on the project milestone planning will be reported even before they really hit the project plan. This way, all project stakeholders will be informed and can cooperate in the corrective actions.

5 Project management

This sections describes the project management approach and used techniques to guarantee project success. As the project team consists only out of 2 team members, special attention is given to make the project management approach holistic but light. Holistic to assure that all important aspects with regards to the management of this project are properly covered and lightweight to avoid that the project team loses time in handling unnecessary tasks which only have added value in larger project teams.

5.1 Project governance / roles & responsibilities

We distinguish 8 different project parties or roles in the project environment, each of them are allocated with specific tasks a result in a corresponding responsibility.

The project team The project team consists out of Daniel Schiavini and Maarten Baertsoen. This team takes the full responsibility for the day to day project management as well as the actual delivery of all project work products consisting out of

1. The actual Haskell program code in full conformity with the coding conventions;
2. Exhaustive code documentation;
3. All necessary, final and intermediate, deliverables to ensure project's success such as: project planning, action/issues/risks lists and intermediate status reports based on the project methodology.

The full list of all work products with their corresponding description is included in section 4.

The technical lead The technical lead will have the final word and responsibility regarding the following aspects:

1. Product selection: selection of the tools and libraries that will be used to realize the project;
2. Coding conventions: define, adjust and support correct appliance of the coding conventions;
3. Lead over the technical architecture;
4. Quality supervision over the technical aspects of the project;
5. Correct integration of the refactored parser into the Ampersand environment.

This role is assigned to Daniel Schiavini.

The project lead The project leader will monitor and control the project management activities in the project and as documented in this project plan, such as budget usage, communication, risks and issue resolution. The project lead will as well organize and facilitate meeting with the stakeholders where needed. Within the project, quite some tools and accelerators are used; the project lead will assure that these are used correctly and act in case some revisions or corrective actions are needed.

This role is assigned to Maarten Baertsoen.

The business lead The business lead within the project will assure that the solution addresses the business requirements in a correct way. This means that the business lead will monitor the effectiveness of the solutions and will ensure that the requirements are correctly captured and documented. Both Maarten and Daniel will share this role. To make it clear to the stakeholders who is responsible for which business aspect, each group of related requirements will be assigned to either Daniel or Maarten.

This assignment will be further analyzed and documented in phase 3c.

The project supervisor The project supervisor is Bastiaan Heerlen. He is the main contact person for the project team. It is the role of the project supervisor to monitor the project team on a regular basis as well as to provide support and guidance to the project team members both on content and approach. It is the responsibility of the project team to send project status updates towards the project supervisor; these status updates will form the basis for the regular alignment meetings.

During project execution, it might occur that the effort needed to meet the customer expectations is way over the foreseen budget of the project (measured in hours). Should this situation occur, the supervisor will facilitate customer alignments to adjust the customer's expectations.

All efforts spent are reported on a regular basis to the supervisor who will track the spent effort to make sure that every aspect of the project receives the proper attention it deserves.

The customer The project will deliver an actual product with all corresponding documentation in a clear and controlled way. The customer's satisfaction is one of the key success factors of this project.

Although this project is conducted in an educational environment, the customer shall be treated just like he would be in a professional environment.

End-Users The Ampersand Parser is actively used by several users groups. Although these user groups are not addressed directly, they will be affected when the project team releases new topics in the Ampersand Parser. All necessary communication and change management techniques applied to support the introduction of new functionality are described within this document.

All topics in which the end user groups are involved need to be carefully aligned with the customer as it is the customers responsibility to keep the Ampersand Parser up and running and in good shape. Introduction of bugs, unclear functionality or even the total unavailability will harm the customer reputation.

The examiner Last but not least, the examiner will evaluate all aspects of the project to which he will grant a score. These sub-score will be used to determine the final and individual course result. It's obvious that the customer's satisfaction is one of the main benchmarks, however the examiner will also evaluate the used project approach, academic skills and personal investigation areas.

5.2 Communication

Regular alignment and communication sessions are foreseen within the project. The communication plan described below identifies the way the project team will align and

communicate with the involved project parties.

5.2.1 Internal

The project team members, Daniel Schiavini and Maarten Baertsoen will align on a weekly basis. During this meeting, following topics will be discussed:

1. Tasks done
2. Tasks due
3. Alignment on and assignment of the upcoming project tasks
4. Actions
5. Issues & Risks

Given the project size of the team, Daniel and Maarten will communicate informally on a regular basis besides this recurrent weekly meeting.

5.2.2 OU

A recurrent meeting between the project supervisor and the project team is planned every 3 weeks. This meeting is an official feedback meeting, from team to supervisor, providing the full status of the project encompassing all important project aspects such as progress, issues and risks.

During these meetings the supervisor will monitor and steer the project progress and will check the proper project management by consulting the project's statistics.

All documents will be delivered to the project supervisor at least 24 hours before the actual meeting to allow the project supervisor to perform an in depth review.

5.2.3 Customer

The sessions with he customer will be planned carefully, both on timing and content, to make sure that the customer is not disturbed too much with project questions not relevant for them as a customer, but without losing the insight in the customer's needs.

The main principle to distinguish between the questions that can be posed to the customer or not, is by reflecting this project to a professional business project. Every question or issue that should be handled by a professional project team, based on knowledge and experience, will be handled internally within this project as well.

All questions, issues and status updates towards the customer will be listed by the project team and will be revised with the project supervisor to assure to customer meetings are efficient, clear and most importantly, that these meetings provide the customer a trustful feeling regarding the project.

5.3 Time keeping

The project team members will record every effort they have spent on the project an overview over:

1. Date

2. Phase
3. Domain
4. Member

This overview will be shared with the project supervisor on a regular basis and whenever requested.

5.4 Project reporting

The project team will share periodic status updates during the project, this reporting will cover following project aspects:

1. General project status
2. Timing
3. Results achieved
4. Budget (spent hours)
5. Risks and issues
6. Quality
7. Focus for the upcoming weeks

5.5 Quality assurance

The customer's perception of the project quality will have a tremendous impact on the overall rating of the project deliverables and therefore, it is for the project team of utmost importance to demonstrate the measures taken to guarantee project success.

Having correct quality assurance processes is not sufficient when these are not used on a day-to-day basis. The project team will present quality reports towards the project supervisor and the customer to gain their confidence that the project team will deliver the promised quality.

Following topics will be monitored and reported:

1. Issue handling
2. Follow up of actions
3. Release procedures to put project coding into the production environment
4. Communication & change management
5. Test management
6. Tool effectiveness
7. Documentation management

5.5.1 Process quality & monitoring

The used project management processes must be both light and effective. The project team will monitor the effectiveness of these processes on a regular basis and carry through useful changes after alignment with the project supervisor.

5.5.2 Deliverables quality & monitoring

Each project deliverable will be reviewed by the project team member's counterpart within the project team.

Where needed, several revision iterations will be foreseen until the project team comes to an agreement regarding the sufficient quality level of the deliverable. Deliverables approved within the project team will be shared with the project supervisor who can perform an additional quality check before the deliverables are shared with the customer and the project examiner.

Each delivery towards the customer can be followed by a subsequent revision phase to perform a final fine-tuning to assure customer satisfaction.

Especially for the work items that consist out of programmed code, a thorough review will be done to assure that the programmed code is in line with the coding conventions as delivered by the customer (see subsection 9.1).

6 Assumptions and constraints

The following assumptions and constraints are considered in this project plan:

- For the main objectives, the majority of the changes are concentrated in the parser. It is assumed that the changes in other parts of the software will be very limited.
- The project members will need to spend time building up knowledge on various involved parts: business rules, Ampersand, Haskell, LaTeX, etc.
- The architecture as described by the customer and documented on the Ampersand Wiki is assumed to be correct and up-to-date.
- The delivered software will be reviewed and tested by the relevant parties within a week.
- The customer and supervisor will be able to answer questions, give support, feedback and advice without much delay.
- The assumption is made that two project members will be available for an average of 12 hours a week. This can be influenced by the members' jobs or private lives.
- Regression testing can be done using the available Sentinel Server
- Given the major impact on the parser part, we assume that this project team has the lead over the parser code during the project. Any fixes are changes on the parser outside the project team will be aligned first on impact and coherence with the ongoing parser changes.

7 Risk management

Risk management is one of the measure taken within the project to guarantee the process quality and hence project success.

The risk management approach consist out of 5 steps

1. Risk identification
2. Risk qualification
3. Determination of risk mitigation measures
4. Monitoring and control
5. Evaluation and fine-tuning

7.1 Risk identification

Every identified risk within the full life-cycle of the project will be followed up by the project team. The definition of a risk in this project is 'every event that can influence the project in a tangible negative way'. Of course there can be risks with a positive effect, but these will be merely considered as a 'lucky coincidence'.

All risks will be kept together in a central repository, the risk register, in which a summary with the main characteristics is provided. The following information is registered for each risk:

- Risk ID
- Status
- Domain
- Identification date
- Short description
- Additional information
- Mitigation actions
- responsible
- date next review
- Impact
- Likelihood
- Risk evolution (status quo, increased or decreased probability/evolution)

7.2 Risk qualification

Not every identified risk will have the same importance, and the focus will be on the risks that are worth paying attention to. This means that the focus will be on the medium or heavy risks which, if they occur, will have a bad or sometimes a catastrophic effect on he project.

The exact qualification of the risk type is based on the risk likelihood (what is the chance of the risk to become a fact), and the risk impact (how hard will this risk hit the project if it happens).

Table 1 identifies the risks qualification based on these 2 parameters:

	Likelihood		
Impact	Low	Medium	High
High	M	H	H
Medium	L	M	H
Low	L	L	M

Table 1: Risk qualification

Based on the identified risk type, the risk will be managed accordingly:

L - Low Risk: Low risks will be monitored on a regular basis, but no specific actions will be initiated with regards to risk mitigation and/or risk avoidance.

M - Medium Risk: Medium risk will be managed in a more proactive way. Risk mitigation options will be documented, and these will be taken into account during the risk's life-cycle. It is up to the project team to decide whether the risk is managed actively or merely monitored.

H-High risks: The project team will act on high risks with a high sense of urgency. High risks will be investigated in detail, a risk mitigation approach will be installed and the risk will be monitored intensively throughout the risk's life-cycle.

7.2.1 Determination of risk mitigation measures

The possible risk management strategies options to cope with each distinct risk are the following:

Risk avoidance Actions are undertaken to avoid the risk, meaning that the risk will disappear or become less likely due to the taken actions.

Risk reduction The risk itself cannot be avoided, but measures are taken to limit the negative consequences in case the risk materializes.

Risk sharing This kind of risk cannot be coped with by the project team on its own, therefore the risk mitigation responsibility is shared with other resources or project teams to identify a joint risk approach.

Risk acceptance The project team decides to just accept the consequences of the risk. The team identifies the risk impact and foresee time budget to handle with the negative consequences in case the risk materializes.

The risk management strategy option will be documented for each distinct risk in the risk register.

7.2.2 Monitoring and control

The risks are monitored on a weekly basis and the necessary re-qualifications and new/ongoing mitigation actions are discussed when appropriate.

The risk register, together with a status overview, will be discussed with the project supervisor on a regular basis. The most important highlights and ongoing mitigation actions or results will be explained in more detail.

7.2.3 Evaluation and fine-tuning

The evaluation of the risk management approach is not planned in a formal way, but each time an improvement point is identified, the team will evaluate fine-tuning activities and implement these whenever considered useful.

7.3 Known risks at the beginning of the project

Risk management is a continuous process during the full project life cycle and the team will react quickly on new risks and will monitor the open risks with care. At the beginning of the project, some risks are already identified and documented in the risk register. Given the holistic goal of this project plan document, the identified high risks are summarized below:

ID 1

Likelihood Medium

Impact High

Qualification High

Domain Scope

Risk Some requirements are unclear and/or ambiguous

Possible impact We might build the features incorrectly

Mitigation type Avoidance

Mitigation approach For each requirement, a detailed description of the need including the solution approach will be documented and presented towards the customer to ensure that the delivered functionality meets expectations.

ID 2

Likelihood Medium

Impact High

Qualification High

Domain Scope

Risk Requirements are not ordered by priority.

Possible impact We might build the lower priority features before the high priority features.

Mitigation type Avoidance

Mitigation approach The project team will draw an overview of the perceived priorities and will align with the customer to reach an agreement regarding the priorities.

ID 3

Likelihood High

Impact High

Qualification High

Domain Scope

Risk We don't know what the problems of the current parser are.

Possible impact We might deliver a parser with the same problems.

Mitigation type Avoidance

Mitigation approach A meeting with the customer will be organized to gather knowledge of the existing system, especially regarding the current issues, pain points and pitfalls.

ID 4

Likelihood High

Impact High

Qualification High

Domain Scope

Risk The Haskell code, grammar description, code comments and examples are not consistent.

Possible impact We don't know which one is leading.

Mitigation type Avoidance

Mitigation approach The team will document the differences in the current code and plan a meeting with the customer to decide on the correct coding conventions.

ID 5

Likelihood High

Impact High

Qualification High

Domain Quality

Risk Useful feedback' is a subjective term, if the project team and the project customer have a different opinion regarding this term, the team could end up delivering a project according to spec but of which the quality is judged as 'insufficient'.

Possible impact unsatisfied customer, non acceptance of the project.

Mitigation type Avoidance

Mitigation approach A formal error message overview ill be created to establish a clear and traceable overview regarding which input error types will result in which error

ID 6

Likelihood High

Impact Medium

Qualification High

Domain Budget

Risk It already clear that the full list of current enhancement points cannot be treated within the current budget (hrs).

Possible impact Unsatisfied customer, non acceptance of the project.

Mitigation type reduction

Mitigation approach Within the project plan, it is clearly stated that the feedback mechanism in the parser receives full priority. The additional requirements will only be considered when the project team is able to integrate them without budget overrun. This will be discussed with the customer.

8 High-level requirements

In this section, a list of known requirements is given. Please note that because the project plan is made in the beginning of the project, only the highest-level requirements can be described. The requirements will be further clarified in phase 3c based on the stated scope 'useful feedback in the Ampersand parser'. Also note that the project will be executed in an iterative/agile way, so the requirements will change, and when they do this document might be not updated.

In this section, the abbreviation NAP is used, with the meaning 'new Ampersand Parser'.

8.1 Capability requirements

`PL-FUN-010` **Command-line interface**
The NAP shall only have a command-line interface.

8.2 Required states and modes

`PL-RSM-010` **Forward parsing**
The NAP shall provide forward parsing, from ADL to P-structure.

`PL-RSM-020` **Backwards parsing**
The NAP shall provide backwards parsing, from P-structure to ADL.

8.3 External interface requirements

`PL-EIF-010` **File interface**
The NAP shall read input text files in ADL format.

`PL-EIF-020` **File encoding**
The NAP shall read input files with Windows-1252 (also known as ANSI) encoding.

8.4 Internal interface requirements

`PL-IIF-010` **Interface with Type Checker**
The NAP code shall interface with the Type Checker via the P-structure.

8.5 Internal data requirements

`PL-IDR-010` **Generate P-structure**
The NAP code shall generate the P-structure in the same manner as the old Ampersand parser currently does. This implies that the P-structure created by the old and new parser will be a perfect match.

8.6 Adaptation requirements

No such requirements have been identified.

8.7 Safety requirements

Ampersand is not a safety critical application. Therefore, no such requirements have been identified.

8.8 Security and privacy requirements

Ampersand is not a security or privacy critical application. Therefore, no such requirements have been identified.

8.9 Environment requirements

`PL-ENV-010` **Windows environment**
The NAP shall run in the Microsoft Windows 7 operational system.

8.10 Computer resource requirements

8.10.1 Hardware requirements

No such requirements have been identified.

8.10.2 Hardware resource utilization requirements

No such requirements have been identified.

8.10.3 Software requirements

PL-CSW-010 **Haskell language**
The NAP shall be written in the Haskell programming language.

PL-CSW-020 **Haskell compiler**
The NAP shall be compiled with GHC version 7.8.3.

8.10.4 Communications requirements

No such requirements have been identified.

8.11 Software quality factors

PL-SQF-010 **User-friendly errors**
The NAP shall give user-friendly error messages for the most common parsing errors.
Note: Discovering which errors are the most common and what user-friendly messages consist of, is the most important part of the assignment. The strategies for achieving the required results are described in section 3.

PL-SQF-020 **Code annotation**
The NAP source code shall be documented in Haddock format.

8.12 Design and implementation constraints

PL-DIC-010 **Ampersand code base**
The NAP shall be implemented in the Ampersand code base.
Note: This also means that NAP shall be implemented in the Haskell programming language, and will use the same compiler as Ampersand does. The existing coding conventions shall be respected.

8.13 Personnel-related requirements

PL-PER-010 **Project members availability**
The project members shall be available for at least an average of 10 hours a week.

PL-PER-020 **Peer reviews**
The project members shall review each other's code and documentation before delivery.

8.14 Training-related requirements

No such requirements have been identified.

8.15 Logistics-related requirements

All deliveries are done online, therefore no logistics requirements have been identified.

8.16 Other requirements

No such requirements have been identified.

8.17 Packaging requirements

`PL-PAK-010` **Git repository**
The NAP code shall be managed in the 'AmpersandTarski/ampersand' GitHub repository, in the 'ABI_Parser' branch.

8.18 Precedence and criticality of requirements

`PL-PAC-010` **Customer wishes**
The implementation of the NAP shall have higher priority than any new features or customer wishes

9 Project realization

During the project phase 3d, the software solution for the customer is designed, developed and tested.

9.1 Coding conventions

The coding conventions are described on the Ampersand wiki[3] and the project team will strictly adhere to these coding conventions. In order to guarantee consistency, the conventions will not be duplicated in this document.

9.2 Iterative approach

For this project, an agile iterative approach has been chosen for the following reasons:

- Offering fast feedback to the customer;
- Allowing adaptation during the project;
- Delivering software as often as possible;
- Often merging code with other ongoing projects (see section 7);
- Maintaining a sustainable pace of development;
- Allowing team self-organization;

Considering the member's experience with agile methods, some inspiration has been taken from Scrum. This project will be executed in a part-time way, thus the amount of work in each iteration will be very limited. That makes the use of Scrum very challenging, so the following remarks are to be considered:

- Team meetings with the supervisor are initially once per 3 weeks, so the iterations shall take three weeks.
- This meeting will work also as a review and planning meeting.
- The customer cannot be present in every team meeting; this will be done partially by distance, via e-mail and other means (see section 13).
- The project members shall get in contact as often as possible, for backlog refinement and for answering the three basic questions:
  - What have you done since yesterday?
  - What are you planning to do today?
  - Any impediments/stumbling blocks?

9.3 Validation

In order to validate that the right product is being built, the first step before the planning meeting is to define what is going to be built. This will be done by means of writing requirements and sharing them with the customer.

An important step on defining the requirements for the Ampersand parser will be defining which situations require improvement. This can be done by collecting user data, although this data might be hard to analyze afterward. The best and most efficient approach will be defined in cooperation with the customer.

Whenever applicable, additional documents, prototypes and examples will be shared with the customer or stakeholders. This will require a big amount of communication (see subsection 5.2) between the project members and other stakeholders. Even though the customer has limited time, the end product can only be as good as the information given to the project team.

The validation of the work products will be facilitated by the project team using a pre-defined validation approach:

- Upon the completion of a work product, the project team will do the verification and validation of the work product against the agreed on requirements;
- The verification and validation will be documented and handed over to the customer, the supervisor will be added in cc;
- After review by the customer's team, all remarks and issues of topics not corresponding to the agreed on scope will be delivered by the customer within a time frame of 1 week;
- The project team will treat and handle the remarks and issues within 1 week;
- The final work product is presented to the customer for acceptance.

9.4 Test approach

In order to verify the requirements, the approach is to automate as much as possible of the testing platform. Although this will take some extra effort at the beginning of the project, the automation guarantees that a lot of time is saved in regression testing.

See also section 13 for a list of tools that will be used for testing.

9.5 Test methodology

Automating tests for Haskell programs should always be possible. Indeed, the programs should be always deterministic. The Ampersand parser, specifically, can be used in a feedback way: the program code is parsed, and then converted back to source code. By parsing this code yet again, it can be verified that the result is the same.

9.6 Test documentation

All test scripts shall be added to source control (see section 13 for the used tools). If necessary, a simple test report will be delivered at the end of the sprint. Note that this report can be shared earlier with all stakeholders, as soon as a deliverable (backlog item) is ready.

9.7 Milestones

Considering the iterative approach, defining all backlog items beforehand is not possible. Defining which backlog items will be delivered in each sprint is thus also impossible, this will be done in a lean and agile way during phase 3c. However, some products are known beforehand and can be defined here. See Table 2 for the initial planned delivery dates.

Date	**Deliverable (s)**
2014-10-22	Project planning v0.1
2014-11-13	Project planning v1.0
2014-12-03	End of research: domain & techniques (phase 3a)
2014-12-24	No delivery (Christmas holidays)
2015-01-14	Delivery sprint 1
2015-02-04	Delivery sprint 2
2015-02-25	Delivery sprint 3
2015-03-18	Delivery sprint 4
2015-04-08	Delivery sprint 5
2015-04-29	Delivery research context (phase 3b)Final software delivery (phases 3c/3d)
2015-05-20	Final thesis deliveryPresentation

Table 2: Realization milestones

10 Issue management

During the project execution phase, mainly in phase 4, software and documentation issues will be found. Whenever the project members or other users encounter such issue, it should be logged and tracked. In section 13 a list of tools is given that will be used for

issue tracking.

If the issue is related to the scope of the project and if it is simple to solve, the project members shall take immediate action. This simplifies the process and removes some noise, so that the focus can be on the important issues.

Bigger or more complicated issues will be logged, prioritized and assigned, taking in consideration the experience and availability of the project members. For prioritizing issues, the following properties shall be considered:

- The expected effort to solve the issue;
- How often the issue occurs;
- What is the impact when the issue occurs;
- How many people are influenced by it;

If there is a risk the issue cannot be solved within the time constraints, a more extended conversation must be started. In case the issue is caused by changes in this project, canceling the regarding feature might be an option. Otherwise, escalation to other Ampersand developers can be considered. In all these cases, the project supervisor and the customer will be consulted as a sounding board.

Some issues can occur that are not related to the project's initial scope or that introduce a deviation from the initial scope. In this case, all these issue, and corresponding changes, will be logged in a Change Request list in which they will be evaluated based on project impact, cost (in hrs), added value and corresponding risk. Change request will be handled in full transparency with the customer and supervisor to avoid scope creep and the the risk that the focus drifts away from the initial project goal.

See also section 7 for risk management procedures.

11 Integration & release

The code changes of this project will be done in a separate branch (`ABI_Parser`) of the Ampersand GitHub repository (`AmpersandTarski/ampersand`). In the end of each sprint, the project members will evaluate whether the integration into the master branch is advised. If so, the following steps will be taken:

1. The code has already been merged and tested in the project branch.
2. The project manager shall send a notification via e-mail to the customer, including the test documentation and any other applicable information, such as the commit identifier (see subsection 9.6).
3. The customer should realize any complementary tests and accept the release.
4. When the release is accepted, a push request will be issued.

It is important to note that any code developed in a sprint should be usable and integrated into the master branch. If integration is not possible, the sprint can be considered failed.

12 Documentation

12.1 Documentation plan

Just like the project itself, the documentation will be managed as an iterative product. All project documentation templates, forming the basis for the documentation deliverables, will be created before the start of phase 3b (research context), and all analysis, design and coding documentation will be maintained on a weekly basis.

Each sprint delivery, the available set of documentation will be delivered informally towards the project supervisor and, where useful, towards the customer. The project team strives to finalize all intermediate documentation at least one week after the release corresponding deliverable.

12.2 Documentation deliverables

All documentation deliverables listed in section 4.3 (Project planning, milestones and corresponding deliverables) are explained in some more detail in this section.

12.2.1 Detailed project plan

The detailed project plan (this document) covers at least the following topics:

1. Project description, context and goals
2. Project approach and project management
3. Planning
4. Risks and risk management approach
5. Quality approach and control
6. Project requirements, global architecture and the main component, including the environment, of the to be modified system
7. Test plan
8. Used tools, methodologies and documentation
9. knowledge gathering for proper project execution
10. Communication plan
11. Glossary

12.2.2 Detailed allocation of tasks within the project team

This project will be driven by 2 project team members who will work closely together. Based on the personal selection of the academic competencies, some highlights, responsibilities and corresponding roles will be assigned to each individual team member.

This document will clarify the assigned roles and tasks each team member will take care of.

The 'detailed allocation of tasks within the project team' is integrated in the deliverable 'Detailed project plan'.

12.2.3 Scientific article

Each individual project team member will deliver an in depth analysis of one or more topics regarding the environment in which the system is, or will be, used.

This deliverable will cover at least the following topics:

1. Problem definition
2. Motivation
3. Context
4. Used analysis techniques
5. Results
6. Conclusions
7. Glossary
8. Bibliography
9. Appendices, if useful

12.2.4 Detailed report of the 'research context'

The detailed report of the 'research context' will cover at least following topics:

1. Context
2. Conclusions of the researcher
3. Conclusions in regards to the project
4. Glossary
5. Bibliography
6. Appendices, if useful

12.2.5 Analysis & design

The 'analysis & design document' will cover at least following topics:

1. Design decisions and the reasoning behind them
2. Detailed solution design
3. Implementation guidelines & coding conventions
4. Tools & libraries used in the project

12.2.6 Test report

The 'Test report' will cover at least the following topics:

1. Overview of test scenario's
2. Test results
3. Conclusions

12.2.7 Source code

The source code will be documented in the code itself. It is of utmost importance that the code is documented in a clear and concise way. Topics needing a more elaborated documentation approach will be documented in the deliverable IT documentation.

12.2.8 IT documentation

Some coding structures, or the architectural design principles behind it, can require some additional documentation which could pollute the source code itself. The IT documentation deliverable will list complex coding structures and the clarification of them including the overall technical architecture, in accordance with the project scope and the technical design principles.

The IT documentation must provide the necessary support to our customer to fully understand the project coding and it needs to present a solid basis on which future design and implementation decisions can be taken.

12.2.9 Project documentation document

The project will be concluded in a general project documentation document. This deliverable will conclude all main deliverables, where possible, in a brief way.

The following additional topics will be documented to close the project:

1. Main realizations
2. Encountered issues & risks
3. Tools & methodologies effectiveness
4. Topics not delivered with the project
5. Budget analysis
6. Lessons learned
7. Conclusions

13 Tools, methodologies and accelerators

13.1 Collaboration

For the collaboration between the project members, the following tools will be used:

Dropbox For sharing time tracking and other reference documents;

GitHub For sharing git repositories of code, besides managing issues and documentation;

SourceForge For managing existing (old) issues;

Microsoft Office For writing internal documents, e.g. time tracking;

13.2 Documentation

For writing the documentation, the following tools will be used:

Haddock For annotating documentation on the code;

TeXworks For writing and compiling LaTeX documents;

Ampersand Wiki For keeping notes and information useful for users and other developers;

Railroad Diagram Generator For generating syntax diagrams based on EBNF notation.

13.3 Design

For designing the software and its architecture, the following tools will be used:

yEd For creating diagrams and graphs;

13.4 Development

For software development, the following tools will be used:

IDE No standard integrated development environment will be chosen: the project members are free to use any IDE, e.g. Eclipse, Leksah, Notepad++;

GHC The compiler GHC (Glasgow Haskell Compilation System), version 7.8.3, will be used;

Cabal For managing Haskell packages and compilation, cabal-install version 1.18.0.5 and Cabal library version 1.18.1.3.

13.5 Testing

Finally, the following tools will be used for testing the software:

Hpc might be used for checking, recording and displaying the code coverage of tests;

Sentinel server can be used for the integration tests;

QuickCheck for automating tests on the code and property based testing (e.g. pretty-print and reparsing, random code generation);

Glossary

A-structure

The ADL code generated by the Ampersand type checker, used as input for the calculator component.. 6

ADL

Ampersand Design Language. 6

ADL-structure

See A-structure.. 6

Agile

Group of software development methods in which requirements evolve through collaboration during the project's execution. 28

branching

Branching is the duplication of an object under revision control so that modifications can happen in parallel along both branches.. 31

Cabal

Library for managing Haskell builds and packages. 35

Dropbox

File hosting service that offers cloud storage, file synchronization, personal cloud, and client software. 35

EBNF

Extended Backus-Naur Form. 35

Extended Backus-Naur Form

Notation technique for documenting context-free grammars. 35

F-structure

The functional structure generated by the Ampersand calculator, used as input for the different output modules.. 6

GHC

Glasgow Haskell Compilation system. 35

Git

A distributed revision control and source code management. 35

GitHub

Git repository web-based hosting service which offers all of the distributed revision control and source code management (SCM) functionality of Git. 35

Haddock

A software documentation generator for the Haskell programming language. 35

Hpc

Library for checking, recording and displaying code coverage. 35

IDE

Integrated Development Environment. 35

Iteration

An iteration is the basic unit of development. In the end of each iteration, working software should be delivered. 28

LaTeX

Document preparation system and document markup language for the TeX typeset. 35

merging

The reintegration of a branch into the original code. 31

Microsoft Office

Office suite of desktop applications devleoped by Microsoft. 35

NAP

Ampersand Parser. 25

OU

Open Universiteit Nederland. 18

P-structure

The parse-tree generated by the Ampersand parser, used as input for the type checker.. 6

Phase 2

Planning phase of the project. 5

Phase 3a

Phase of the project when domains and techniques are investigated. 10

Phase 3b

Phase of the project when the research context is investigated. 12

Phase 3c

Phase of the project when the software is actually designed, developed and tested. See section 9. 28

PoC

Proof-of-concept. 12

QuickCheck

Library for testing Haskell code. 35

Railroad Diagram Generator

Tool for generating syntax diagrams based on EBNF notation. 35

RDG

Railroad Diagram Generator. 35

Sentinel

Test server for the Ampersand project. 35

TeXworks

Graphical user interface for editing and compiling LaTeX documents. 35

Wiki

Website that allows users to create web pages collaboratively. 28

yEd

Software for editing graphs. 35

References

[1] The Business Rules Manifesto
Version 2.0, November 1, 2003
Ronald G. Ross
http://www.businessrulesgroup.org/brmanifesto.htm

[2] Requirements for a parser of Ampersand
Stef Joosten
Version 8e27daf, September 25, 2014
https://github.com/AmpersandTarski/ampersand/commit/8e27daf

[3] Ampersand Coding Conventions
Stef Joosten & Han Joosten
Version 4, September 6, 2011
http://wiki.tarski.nl/index.php/Coding_conventions.

[4] Top Quality Type Error Messages
Bastiaan Heeren
ISBN 90-393-4005-6, September 20, 2005
www.open.ou.nl/bhr/phdthesis.