Project Planning

Useful feedback in the Ampersand parser

Maarten Baertsoen and Daniel S. C. Schiavini

Open Universiteit Nederland, faculteit Informatica
T61327 - Afstudeerproject bachelor informatica

November 29, 2014

Version 2.0

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 Prof.dr. Marko C.J.D. van Eekelen. The assignment is given by Dr. 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 to propose a high-level solution, as well as describing 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. In section 3 the project approach is explained and the management strategy is given in section 4. Then, a clear view on the high-level project requirements is set in section 5. Then, in section 6, strategies are proposed for the acquisition of knowledge.

Details of the development techniques, including design, implementation and testing, are given in the project realization strategies in section 7. The strategy for integrating the released software is given in section 8. Strategies for risk management are then proposed in section 9, including an overview of known risks at the beginning of the project and their respective mitigation approaches.

Issue management is explained in section 10. The documentation strategy is given section 11 and the used tools and methodologies are given in section 12.

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 these rules 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 thoroughly 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 requirement's 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. 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.

2.2 Project architecture and components

The compiler developed for the Ampersand research project runs in several steps, hence 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 a 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.

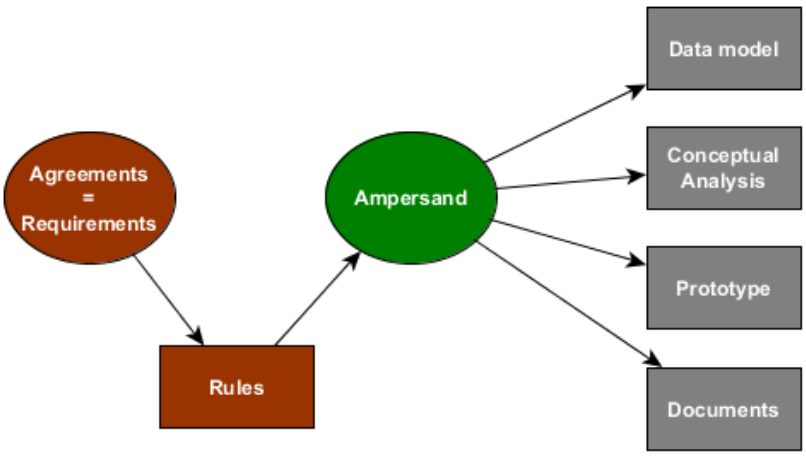

Figure 1: The Ampersand approach generates different artifacts based on the business rules (source: [2])

Calc: The Calc component receives an 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 hard 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.

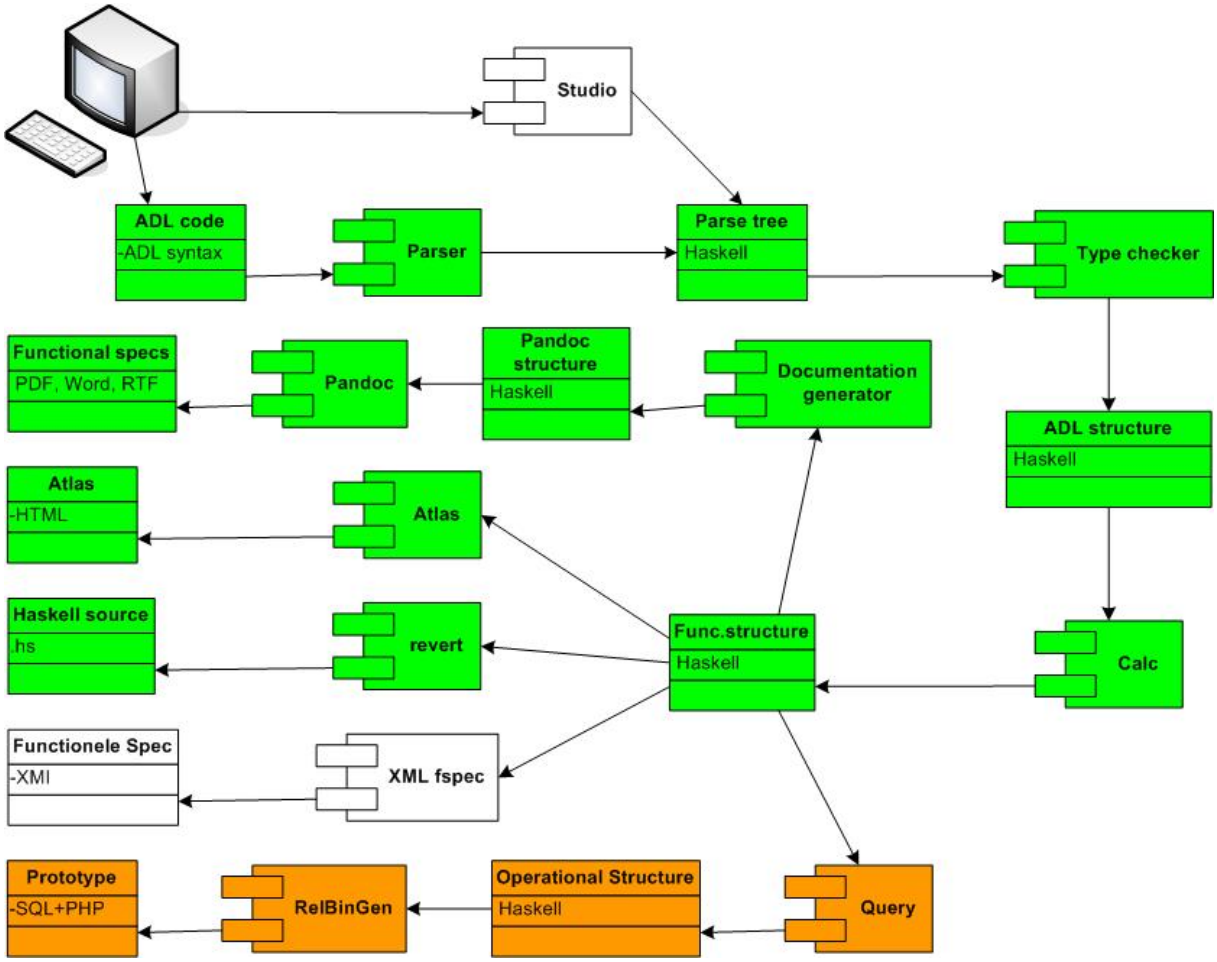

Figure 2: Architecture of the project, showing where the parser fits in the Ampersand system (source: [3]).The components in green background are part of the Ampersand compiler. Components in orange are part of the Ampersand Prototype compiler. Finally, components in white background are future components, not yet implemented.

Figure 3: Relevant data flow for the Ampersand parsing component

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 project members is to implement useful feedback in the Ampersand parser. This will be delivered as a well-working and maintainable piece of software, which needs to be user friendly in order to improve the usability. For achieving this goal, the following research & analysis activities will take place:

- Analysis of user-friendly messages in compilers;

- 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;

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 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 [4], although it must be clear that the issues are strictly seen as lower priority. See also section 5 for the list of high-level requirements.

On top of the project goals, the project members declare herewith to have the following personal objectives and commitments fulfilled by the end of this graduation project:

Knowledge: Building up knowledge is the main reason why one starts a bachelor study. As such it is important to 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.

2.5 Critical success factors

The following factors are critical for the customer in order to consider the project's completion successful. 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 9).

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 9 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 specified in 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 the complete grammar is implemented correctly.

2.6 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.
- It is assumed that the project members will need extra time to build up knowledge on various involved parts, e.g. 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 project team. If the customer needs additional testing before the code can be released into the production environment, it is assumed that the customer will test the new code within one week after delivery for acceptance. These specific releases will be communicated well in advance to make sure that the customer can plan these testing activities.
- 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, the assumption is made that this project team has the lead over the parser code during the project. Any fixes or changes on the parser outside the project team will be aligned first on impact and coherence with the ongoing parser changes.

3 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.

3.1 Project methodology

The main drivers to determine our project methodology are based on the kind of deliverables that need to be produced as a result of this project, more specifically, 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 ideal 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 7.

3.2 Project milestone planning

The project activities are grouped in several standard project phases as specified by the OU. Each project phase has it own milestone date, but they are not iterative, meaning that a specific phase doesn't need to wait until a previous phase is finished, but it can start once sufficient information is available from the previous phase.

More information regarding the actual content of the project phases can be found on the course site of the OU.

A commitment is given by the project team to adhere to the following milestone planning, officiated by the delivery of all their corresponding deliverables. To avoid the delivery of several separate documents, all subtopics within a specific phase are grouped together into a single document.

1. Phase 2 - Task allocation and Project plan – 13/11/2014
    a) Delivery of the detailed project plan
    b) Detailed allocation of tasks within the project team
2. Phase 3a - Domains & Techniques – 17/12/2014

   a) Delivery of scientific article by Daniel Schiavini
   b) Delivery of scientific article by Maarten Baertsoen

3. Phase 3b - Research context
   a) Detailed report of the 'research context' – 29/04/2015
4. Phase 3c - Design & implementation – Last iteration on 08/04/2015 (see below)
   a) Analysis & design documents
   b) Test report
   c) Source code
5. Phase 3d - Project documentation – 04/05/2015
   a) Project documentation document
6. Phase 4 - Project closure and final project presentation
   a) Project essay – 20/05/2015
   b) Project presentation – date to be jointly determined, final due date 29/05/2014

Considering the iterative approach, defining all backlog items, which will be realized in phase 3c, beforehand is not possible. Defining which backlog items will be delivered in each iteration is thus also impossible, this will be done in a lean and agile way.

However, the iterations and their corresponding delivery dates are listed below:

Date	**Deliverable (s)**
2015-02-04	Delivery iteration 1
2015-02-25	Delivery iteration 2
2015-03-18	Delivery iteration 3
2015-04-08	Delivery iteration 4

Table 1: Realization milestones

The project activities will be managed by the project team, and the status will be shared during project alignment meetings. Deviations, changes and issues within the project plan will be communicated within the 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' and 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 this project milestone planning will be reported towards the project supervisor even before they really hit the project plan. This way, all project stakeholders will be informed and can cooperate in the corrective actions.

4 Project management

This section describes the project management approach and used techniques to support the project's 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.

4.1 Project governance / roles & responsibilities

We distinguish 8 different stakeholders in the project, each of them allocated with a specific role and corresponding responsibilities.

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 delivery of all project work products, consisting out of:

1. The actual Haskell program code in conformity with the coding conventions;
2. Appropriate 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 work products with their corresponding description is included in section 3.

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 Heeren. 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.

The 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 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. The necessary communication and change management techniques applied to support the introduction of new functionality are described within this document.

Topics in which the end user groups are involved need to be carefully aligned with the customer, as it is the customer's responsibility to keep the Ampersand Parser up and running and in good shape. Introduction of bugs, unclear functionality or even the total unavailability of Ampersand will harm the customer's 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-scores will be used to determine the final and individual course result. 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.

4.2 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.

4.2.1 Internal

The project team members, Daniel Schiavini and Maarten Baertsoen will align on a weekly basis. During this meeting, the following topics will be discussed: Tasks done, tasks due, open actions, issues and risks.

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

4.2.2 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. Documents will be delivered to the project supervisor at least 24 hours before the actual meeting to allow the supervisor to perform a review.

4.2.3 Customer

The sessions with the customer will be planned carefully, both on timing and content, to make sure that the customer is not disturbed with irrelevant questions 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.

4.3 Time keeping

The project team members will track the effort they have spent on the project. This effort will be recorded together with date, task, project member and project phase. This overview will be used to monitor the spent effort over the project phases and will be shared with the project supervisor before the alignment meetings.

4.4 Quality assurance & Project reporting

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 share and discuss periodic status updates, quality

reports and identified improvement topics during the recurrent alignment meetings with the supervisor.

After each alignment session, a short summary will be sent towards the project supervisor and the customer, which will serve as official status updates.

4.4.1 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 then 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 7.1).

5 High-level requirements

In this section, a list of known functional and non-functional 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. Also note that the project will be executed in an iterative/agile way, so the requirements will change, and when they do this document may not be updated. However, a backlog of functionalities realized within the project will be kept by the project team.

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

5.1 Main requirement: useful and user friendly error messages

The main requirement is that the NAP shall give user-friendly error messages for the most common parsing errors.
Note that discovering which errors are the most common and what user-friendly messages consist of, is an important part of the assignment. At the beginning of the project, no list of undesirable error messages is available. It is up to the project team, as a part of the actual project, to identify undesirable error messages and requirements to implement the useful feedback mechanism in the ampersand parser.

The project team will investigate the current parser to be able to draw up these requirements. This input will be gathered by using the ampersand parser, investigating the Haskell code and by interviewing the students of the OU who are using the parser. This will allow the project team to discover the error messages to be improved.

The strategies for achieving the required results are described in section 6.

5.2 Additional functional and non-functional requirements

To assure that the listed requirements are as complete as possible, the project team used the standards ISO/IEC 9126 [5] and J-STD-016 [6] as requirement checklists. Since so little requirements have been identified, the requirements are listed in a light-weight format, instead of the formats required by the named standards. The completeness and correctness of the requirements has been checked with the customer, Stef Joosten.

Requirement	**Forward parsing**
ID	`PL-RSM-010`
Category	Required state/mode
Source	Current parser
Description	The NAP shall provide forward parsing, from ADL to P-structure

Requirement	**Backwards parsing**
ID	`PL-RSM-020`
Category	Required state/mode
Source	Current parser
Description	The NAP shall provide backwards parsing, from P-structure to ADL (also known as pretty printing)

Requirement	**File interface**
ID	`PL-EIF-010`
Category	External interface requirement
Source	Current parser
Description	The NAP shall read input text files in the existing ADL format

Requirement	**File encoding**
ID	`PL-EIF-020`
Category	External interface requirement
Source	Stef Joosten
Description	The NAP shall handle input files with UTF-8 encoding

Requirement	**Interface with Type Checker**
ID	`PL-IIF-010`
Category	Internal interface requirement
Source	Current parser
Description	The NAP code shall interface with the existing Type Checker via the P-structure

Requirement	**Generate P-structure**
ID	`PL-IDR-010`
Category	Internal data requirement
Source	Current parser
Description	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 match

Requirement	**Windows environment**
ID	`PL-ENV-010`
Category	Environment requirement
Source	Current parser
Description	The NAP shall run in the Microsoft Windows 7 operational system

Requirement	**Haskell language**
ID	`PL-CSW-010`
Category	Software resource requirement
Source	Stef Joosten
Description	The NAP shall be written in the Haskell programming language

Requirement	**Haskell compiler**
ID	`PL-CSW-020`
Category	Software resource requirement
Source	Stef Joosten
Description	The NAP shall be compiled with GHC version 7.8.3

Requirement	**User-friendly errors**
ID	`PL-SQF-010`
Category	Software quality factor
Source	Project description
Description	The NAP shall give user-friendly error messages for the most common parsing errors in a consistent and clear way

Requirement	**Code annotation**
ID	`PL-SQF-020`
Category	Software quality factor
Source	Stef Joosten
Description	The NAP source code shall be documented in Haddock format

Requirement	**Ampersand code base**
ID	`PL-DIC-010`
Category	Design and implementation constraint
Source	Stef Joosten
Description	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

Requirement	**Git repository**
ID	`PL-PAK-010`
Category	Packaging requirement
Source	Stef Joosten
Description	The NAP code shall be managed in the 'AmpersandTarski/ampersand' GitHub repository, in the 'ABI_Parser' branch

6 Knowledge acquisition

6.1 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 his findings. The planning of the investigation is presented in a section per project member.

6.1.1 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 supports both monads and combinators and has quite some good features. One big advantage for this library is that the current code base can be very helpful as a source of documentation. However, being an academic project, it has more experimental features, less users and may be maintained less well than other libraries. Other libraries may also be able to provide other feature sets more suitable for the project.

The purpose of this research is to choose the library best suited to the development of the new Ampersand parser. Besides the UU.Parsing library, the Parsec library will be also analyzed. Another option, in case the grammar is documented, is to use a parser generator, e.g. the Happy Parser Generator. The result of this research could also be that the UU.Parsing library is the best choice.

There are also advantages and disadvantages on using monads and combinators, which must be well understood by the team.

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. If possible, a small proof-of-concept can be created to test the libraries.

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 [7].

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

6.1.2 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 the 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

6.2 Research context

After understanding more about the domain and the techniques involved with the project, the next step (phase 3b) 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 this project phase, the focus is on 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.

6.3 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 11.

7 Project realization

During the project phase 3c, the software solution for the customer is designed, developed and tested. This section describes the approach for the development and testing.

7.1 Coding conventions

The coding conventions are described on the Ampersand wiki [8] 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.

7.2 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 9);
- 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 12).
- 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?
  - Are there any impediments or stumbling blocks?

7.3 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 defined in communication 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 afterwards. 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 4.2) 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 predefined validation approach:

- Upon the completion of a work product, the project team will do the verification and validation of the work product;
- The verification and validation will be reported and handed over to the customer for acceptance (with a copy to the supervisor);
- After review by the customer, all remaining remarks and non-conforming topics will be solved;
- The final work product then is presented to the customer for acceptance.

7.4 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.

Automating tests for Haskell programs should be possible, since the programs should be 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 (also known as pretty-printing).

All test scripts shall be added to source control (see section 12 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 is ready.

See also section 12 for a list of tools that will be used for testing.

8 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 iteration, 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 will be merged and tested in the master branch;
2. The customer will be informed, including any other applicable information;
3. A git pull request will be issued;
4. The customer should realize any complementary tests and accept the request;

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

9 Risk management

Risk management is one of the measures taken within the project to guarantee the process quality and hence project success. This approach is loosely based on the PMI Risk Management approach [9]. This risk management approach consists out of 5 steps: Risk identification, risk qualification, determination of risk mitigation measures, monitoring & control and evaluation & fine-tuning.

9.1 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 'an 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'.

The risks will be kept together in a central repository, the risk register, in which a summary with the main characteristics is provided such as: description, risk qualification, status and identified mitigation actions.

9.2 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 the project.

The exact qualification of the risk type is based on the risk likelihood (what is the chance of the risk becoming a fact), and the risk impact (how hard would the risk hit the project if it happens). Table 2 identifies the risks qualification based on these 2 parameters:

	Likelihood		
Impact	Low	Medium	High
High	Low	High	High
Medium	Low	Medium	High
Low	Low	Low	Medium

Table 2: 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.

9.2.1 Determination of risk mitigation measures

The possible risk management strategies 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 will be documented for each distinct risk in the risk register.

9.2.2 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.

9.2.3 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.

9.3 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:

Risk:	Some requirements are unclear and/or ambiguous				
Poss. impact:	The features might be built incorrectly				
ID:	1	Domain:	Scope	Impact:	High
Likelihood:	Medium	Qualification:	High	Mitigation:	Avoidance
Approach:	For each unclear requirement, a detailed description of the need including the solution approach will be documented. This will presented towards the customer to ensure that the delivered functionality meets expectations.				

Risk:	Requirements are not ordered by priority				
Poss. impact:	Lower priority features might be built before the high priority features				
ID:	2	Domain:	Scope	Impact:	High
Likelihood:	Medium	Qualification:	High	Mitigation:	Avoidance
Approach:	Each time an additional topic can be initiated, the most urgent one will be indicated by the customer.				

Risk:	The problems of the current parser are still unknown				
Poss. impact:	A parser with the same problems may be delivered				
ID:	3	Domain:	Scope	Impact:	High
Likelihood:	High	Qualification:	High	Mitigation:	Avoidance
Approach:	A meeting with the customer will be organized to gather knowledge of the existing system, focussing on the current issues, pain points and pitfalls.				

Risk:	The code, grammar description, code comments and examples aren't consistent.				
Poss. impact:	The delivered code may not be according to the expected coding conventions				
ID:	4	Domain:	Scope	Impact:	High
Likelihood:	High	Qualification:	High	Mitigation:	Avoidance
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.				

Risk:	'Useful feedback' is a subjective term				
Poss. impact:	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 specifications but of which the quality is judged as insufficient.				
ID:	5	Domain:	Quality	Impact:	High
Likelihood:	High	Qualification:	High	Mitigation:	Avoidance
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.				

Risk:	The full list of wishes cannot be implemented within the time budget				
Poss. impact:	Unsatisfied customer, non acceptance of the project.				
ID:	6	Domain:	Budget	Impact:	Medium
Likelihood:	High	Qualification:	High	Mitigation:	Reduction
Approach:	The customer was informed that the full list cannot be implemented. 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 within budget.				

10 Issue management

During the project's execution, software and documentation issues will be found. Whenever issues are encountered, they should be logged and tracked. In section 12 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 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, the issues and corresponding change will be logged in a change request list in which they will be evaluated based on project impact, cost (in hours), added value and corresponding risk. Change requests will be handled in full transparency with the customer and supervisor to avoid scope creep and the risk that the focus drifts away from the initial project goal.

See also section 9 for risk management procedures.

11 Documentation

11.1 Documentation plan

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

Each iteration delivery, the available set of documentation will be delivered informally towards the project supervisor and, where useful, towards the customer.

11.2 Documentation deliverables

The following documentation artifacts will be delivered as part of this project:

1. Scientific article (individual report per project team member)
2. Detailed report of the 'research context'
3. Analysis & design
4. Test report
5. Concluding project documentation including all sub documentation artifacts

12 Tools, methodologies and accelerators

12.1 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 issues with the customer and other developers;

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

12.2 Documentation

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

Haddock For annotating documentation on the code;

TeXworks For writing, editing 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.

12.3 Design

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

yEd For creating diagrams and graphs;

12.4 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.

12.5 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);

Neil Mitchell's HLint for checks on the code readability and maintainability;

Glossary

A-structure The ADL code generated by the Ampersand type checker, used as input for the calculator component. 5

ADL Ampersand Design Language. 5

ADL-structure See A-structure. 5

Agile Group of software development methods in which requirements evolve through collaboration during the project's execution. 19

branching Branching is the duplication of an object under revision control so that modifications can happen in parallel along both branches. 20

Cabal Library for managing Haskell builds and packages. 25

Dropbox File hosting service that offers cloud storage, file synchronization, personal cloud, and client software. 24

EBNF Extended Backus-Naur Form. 24

Extended Backus-Naur Form Notation technique for documenting context-free grammars. 24

F-structure The functional structure generated by the Ampersand calculator, used as input for the different output modules. 5

GHC Glasgow Haskell Compilation system. 25

Git A distributed revision control and source code management. 24

GitHub Git repository web-based hosting service which offers all of the distributed revision control and source code management (SCM) functionality of Git. 24

Haddock A software documentation generator for the Haskell programming language. 24

HLint Library that reads Haskell programs and suggests changes that hopefully make them easier to read. 25

Hpc Library for checking, recording and displaying code coverage. 25

IDE Integrated Development Environment. 25

Iteration An iteration is the basic unit of development. In the end of each iteration, working software should be delivered. 19

LaTeX Document preparation system and document markup language for the TeX typeset. 24

merging The reintegration of a branch into the original code. 20

Microsoft Office Office suite of desktop applications developed by Microsoft. 24

NAP New Ampersand Parser (the software deliverable of this project). 14

OU Open Universiteit Nederland. 13

P-structure The parse-tree generated by the Ampersand parser, used as input for the type checker. 5

Phase 2 Planning phase of the project. 4

Phase 3a Phase of the project when domains and techniques are investigated. 16

Phase 3b Phase of the project when the research context is investigated. 18

Phase 3c Phase of the project when the software is actually designed, developed and tested. See section 7. 18

PoC Proof-of-concept. 9

QuickCheck Library for testing Haskell code. 25

Railroad Diagram Generator Tool for generating syntax diagrams based on EBNF notation. 24

RDG Railroad Diagram Generator. 24

Sentinel Test server for the Ampersand project. 25

TeXworks Graphical user interface for editing and compiling LaTeX documents. 24

Wiki Website that allows users to create web pages collaboratively. 18

yEd Software for editing graphs. 24

References

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

[2] Ampersand: foutvrije specificaties voor B&I-vraagstukken
Informatie magazine, August 2007
Stef Joosten, Rieks Joosten and Sebastiaan Joosten
`http://informatie.nl/Artikelen/2007/augustus/Default.aspx`

[3] Ampersand Software Architecture
`http://wiki.tarski.nl/index.php/Software_Architecture`

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

[5] Software Engineering – Product quality
International Organization for Standardization
ISO/IEC 9126:2001
`http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749`.

[6] J-STD-016 Standard for Software Development and Documentation
Annex F.2.4: Contents of the Software Requirements Specification (SRS)
Department of Defense, United States of America
September 1995
ISBN 0-7381-0427-2, SS94377.

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

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

[9] Project Management Institute
`http://www.pmi.org`