Ampersand
Event-Condition-Action Rules

Software Requirement Specification

[**JG: insert:** Version 1]

Yuriy Toporovskyy, Yash Sapra, Jaeden Guo

CS 4ZP6
October 9th, 2015
Fall 2015 / Winter 2016

Table 1: Revision History

Author	**Date**	**Comment**
Yuriy Toporovskyy	26 / 09 / 2015	Initial skeleton version

Contents

List of Figures

List of Tables

[**JG: replace:**

Chapter 1

Introduction

with:

Chapter 2

Project Drivers

]

[**JG: replace:**

2.1 Project Description

with:

2.2 The Purpose of the Project

] [**JG:** *any comments I made, is a suggestion and its up to you which way you prefer to write it. thanks for getting this to us early, I'm sorry we're so unorganized*] A large part of designing software systems is requirements engineering. One of the greatest challenges of requirements engineering is translating from business requirements to a functional specification. Business requirements are informal, with the intention of being easily understood by humans; however, functional specifications are written in formal language to unambiguously capture attributes of the information system. Typically, this translation of business requirements to a formal specification is done by a requirements engineer, which can be prone to human error.

Ampersand is an approach which addresses this problem in a different way; it lets a requirements engineer translate business requirements written in natural language into a formal specification, and offers a compiler to translate this specification(the Ampersand-script) into an information system [**JG:** *Maybe we should cut out how entirely, because there's a design specification – and solely focus on what? thoughts?*

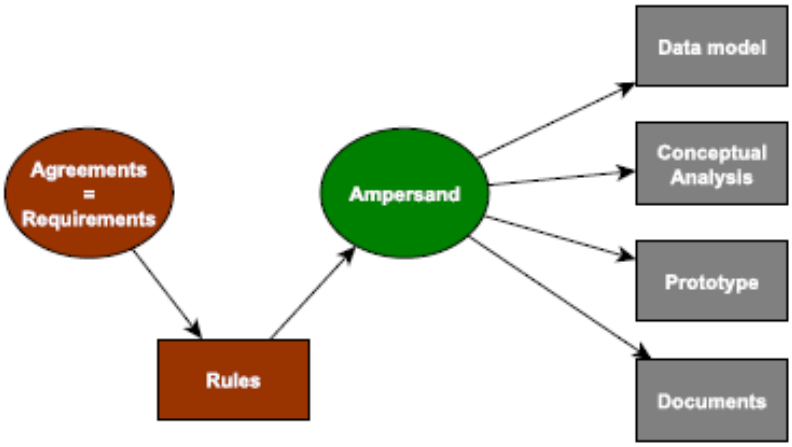

Figure 2.1: Ampersand produces a set of artifacts based on user's requirements.

] [**YT:** *The significance of Ampersand has to do with how it solves the problem, or how it works. To speak to the purpose of the project means to explain why the Ampersand solution is good and why we are spending our time making it better. The rubric has a section titled 'General System Description', so we need to say something about it*] [**SJ:** *Agree with YT. And keep it short and to the point.*] The compiler guarantees compliance between the formal specification and the information system

Ampersand also provides engineers with a variety of aids which help them to design products that fulfill all of the needs of their clients and the end-users (Figure 2.1); including data models, service catalogs and their specifications. Requirements engineering is perhaps most important in safety-critical systems; to this end, Ampersand generates modeling aids and specifications which are provably correct

Ampersand has proven reliable in practical situations, and there have been efforts to teach this approach to business analysts. A large portion of the Ampersand system is already in place; the primary focus of this project is to augment Ampersand with increased capabilities for automation.

2.3 Problem statement

[**SJ:** *The following text is perhaps meant to tell what problem you will be solving. An example is of course very illustrative, but you will also need some text of which it is an illustration. I didn't find a problem statement in this SRS, and this might be as good a place as any to put it in.*]

For example, consider a system for ordering products online. A requirements engi-

neer might want to express, for example, the following business requirement: [**JG:** *Good example, could we put something in here about restrictions of of the real world?*][**YT:** *Business requirements *are* real world requirements. Do you mean something safety-related? This example isn't the best, I know, but I deliberately chose a simple example*][**SJ:** *The simplicity is fine!*][**YS:** *The example is easy to understand. I think we should keep it.*] [**YS:** *I'm not too sure if the Haskell implementation needs to be a part of the introduction. Rubrics indicate the following critera - - - - delineate purpose, specify intended audience), system scope, definitions, acronyms, abbreviations, references, system overview, roadmap of report - - - - I believe we miss definition of E.F.A and E.C.A since it is being used a lot in the document*][**YT:** *Haskell isn't mentioned at all the intro, ECA is mentioned but is also defined (it is really quite simple, we don't define it mathematically, just in natural language, so it shouldn't be incomprehensible).*]

> *Every order must have a customer and a list of products; and the total price on the order must equal the sum of the prices of the products.*

[**YS:** *Attempt at defining ECA*][**YS:** *Should we remove some of the Haskell code and define the scope of our project in terms of conserving these data rules instead? There's a good chance Dan hasn't read our new problem statement so he doesn't necessarily have a clear idea of Ampersand. Suggestions?*] [**YT:** *There is no Haskell code!*][**YS:** *Sorry, i meant the pre-post conditions pairs, I guess we'll let that stay.*][**SJ:** *Let me give you a start:*]She formalizes this requirement as a rule in Ampersand, which is in fact a constraint on the data in the database. These constraints are called invariants, because they are meant to remain satisfied at all times during runtime. The Ampersand compiler translates each rule into, among other things, Event-Condition-Action Rules (referred to as the ECA-rules hereafter). An ECA-rule specifies the action to be taken by the system when an event takes place under a given condition. In Ampersand, the purpose of an ECA-rule is to restore invariants, i.e. to change data in the database such that the constraint becomes true again, after having been violated. Currently, an information system generated by Ampersand produces runtime-events that signal violations of the constraints. It also contains the ECA-rules that are generated by the Ampersand compiler. However, these ECA-rules are not being called to restore in this version. One reason is that there is insufficient guarantee that an ECA-rules actually restores the invariant from which it was generated. The other reason is that we need to figure out how to incorporate that functionality in the information system generated by the Ampersand compiler. [**SJ:** *End*]

The information system may contain a function for manipulating orders. (Ampersand can also generate prototype software models, including functions types like these,

from business requirements - but this is not the topic of our contribution). For example,

```
addToOrder : ( o : Ref Order, t : Product )
```

[**SJ:** *The example is ill chosen, because it uses addition. Ampersand cannot do computations (yet).*] where `Ref x` represents the type of references to values of type `x`; `Order` and `Product` the types of orders and products, respectively.

Ampersand can generate pre- and post-conditions for this function, based on the business requirements. This constitutes a formal specification of the information system. For example, the above function may have the following specification:

```
{ PRE: o.totalCost = t0 }
addToOrder(o,t) = ...
{ POST: o.totalCost = t0 + t.cost }
```

It is proven [**YT:** *This is 'proven' because someone told me we have a proof/presentation of the algorithm, but not its implementation. Can anyone find this? We absolutely need a reference to this.*] [**JG:** *Can you give me more context as to what is proven? I found an article on ampersand subsets, I pushed it; look for "subsets" by Joosten(both)*][**YT:** *There is no file in the repository with the word 'subset' in it?*][**JG:** *Sorry, its domain, this comment got erased with every push ¿.¡*] that for the subset of processes which Ampersand can support, there is an algorithm which will generate the necessary code to satisfy the post-conditions (ie, formal specifications) of each function. However, Ampersand does not yet implement this algorithm. Currently, a user of Ampersand must manually indicate how each violation must be corrected.

In the previous example, the implementation of the function could be as follows:

```
{ PRE: o.totalCost = t0 }
addToOrder(o,t) =
  o.orders.append(t);
  o.totalCost = o.totalCost + t.cost;
{ POST: o.totalCost = t0 + t.cost }
```

The first line includes the item in the order, and the second line fixes the violation of the post-condition which would occur without it. Currently this second line would have to be hand-written by the programmer, but the afformentioned [**YT:** *Should have a name to refer to the algorithm somehow?*] algorithm can derive it from the business rules. The main contribution of this project will be to implement the algorithm which generates the code to fix violations.

2.4 The Stakeholders

[**YT:** *These are the only real stakeholders I can think of...*] [**JG:** *you're right, I can't name names for end users so we might have to cut everything but the designers out, or find a way to use the multiple rules the professors have and play in dailylife to shovel the other subsections into the first one and strip the subsection titles. I'm not sure what to do here*]

2.4.1 Ampersand Designers & Software Engineers

Ampersand designers include Dr.ir. Stef Joosten, professor of information systems design at the Open University in the Netherlands and Dr. Sebastiaan Joosten, researcher the the Technical University of Eindhoven. An academic stakeholder is Dr. Wolfram Kahl, who supervises the development of E.F.A. Dr. Stef Joosten is the designer of the Ampersand method which have been successfully implemented in various public and private projects, among which the immigration service of the Netherlands, the Food Authority of the Netherlands, the Dutch Judiciary and a research organisation called TNO. The interest stemming from these three individuals are not only professional but also personal. Ampersand has been nurtured by a great deal of patience and dedication. The designers of E.F.A by extension are Ampersand contributors and are affected by its outcome, the piece of Ampersand we will build is a necessary component required for completion of undergraduate and is designed to test the capabilities of an undergraduate team. These individuals are the foundation of the belief system by which Ampersand is built. The Ampersand model is built on the belief that technology is highly incorporated in the use of business and affects the life of every individual. It is the belief that information system can and will be designed and built at an affordable price and perfectly compliant with the functionalities that users desire. This perspective treats information systems as a commodity of the real world that can be traded, altered, and used by any individual who is willing to learn how to use it.

2.4.2 Requirements engineers

Requirement engineers are responsible of translating business requirements into project specifications. Ampersand provides requirement engineers with the freedom to explore their creativity rather than be tried down by technical minutes. The ability compile directly from (formalized) requirements provides correctness of software, flexibility to alter specifications, and increased efficiency in designing information systems. An added benefit which engineers receive is the ability to confirm that the system they have designed matching the specifications given to them by the client. E.F.A gives (generated) information systems the ability to restore invariants automatically.

2.4.3 Architects & Functional Specification Designers

System architects, similar to functional specification designers are responsible for various designs which affect procedure and processes in a place of business. Ampersand has the ability to confirm that the design provided by the Architect matches any building regulations that must be meet. Regardless of whether the specification is for safety or aesthetics, the product must meet the client's requirements and minimal legal regulations. Ampersand is able to compose rules and regulations for data which fully incorporates both without contradiction. The addition of E.F.A allows Ampersand to automate correction on data which may contradict rules specified for a project. Architects and functional specification designers will be able to take advantage of the numerous resources Ampersand can provide such as automatic generation of databases within prototypes along with tables and relational schema's. Furthermore, there is a full range of graphics that stakeholders can choose from in order to provide visual display of their design which user may find more friendly and easy to understand.

2.4.4 Process Innovators

Process innovators covers a broad range of individuals that would not commonly be considered stakeholders. This broad category includes anyone who provides input for business processes which includes both sides of a business agreement, in addition to individuals who play a role in seeing the completion of a business process. Ampersand allows continuous input as a living system and has the ability to reorganize requirements when new constraints are added, and E.F.A has the ability to assure the correctness of system information relative to the requirements it has been given. With E.F.A's ability to verify inconsistencies and automate their correction, changes to design and restrictions due to unforeseen circumstances can be added without a system overhaul. The efficiency in which Ampersand can provide has the potential to dramatically decrease flaws that commonly litter information system.

Chapter 3

Project Constraints

The main constraint of this project is time; this is limited to the 8 month period in which we have to build it. It is difficult to perfect an internal construction without a large system in such a short amount of time. But time restraint is an obstacle which all projects face, because it is not limitless. Another major constraint that most project face that this project does not require is the financial constraint where projects become difficult to maintain and often halt in their progress due to monetary constraints. The lack of monetary constraints due to the open source software opens up a broad base of opportunities for future improvements and limitless contributions from the community.

3.1 Mandated Constraints

3.1.1 Project philosophy

Ampersand is an existing software project with a very sizable code base. The cost of maintaining poorly-written code can be very high and can outweigh the benefit of the contribution. In order for our code to eventually be merged into Ampersand, it must be maintainable: it must be written according to coding practices of Ampersand; it must be well documented, so it can be easily understood by other programmers.

Similarly, our code must not introduce any errors or performance regressions into Ampersand. Our code must satisfy existing tests and additional tests should be written for the new algorithm being implemented. Writing maintainable and well-documented code will help with this goal as well.

These motivations are very important to Ampersand and are considered primary goals alongside the obvious goal of producing code implementing the desired feature.

3.1.2 Implementation environment

Haskell

The Ampersand code base is written almost entirely in Haskell. Part of the prototype software is written in PHP and Javascript but we likely do not have to interact with this code base. Our code contribution must be entirely in Haskell.

Haskell software

Ampersand is designed to be used with the Glasgow Haskell Compiler (from here on, GHC) and the associated cabal build system. Ampersand also uses many open source Haskell packages, all available on the Hackage package archive. [**SJ:** *Yes, you can use additional Haskell-packages. The choice is made on the basis of maintainability. If a new package is included, this incurs a cost of maintaining an (already complicated) cabal-structure. However, it also delegates maintenance to the developers of that package. For well-developed en frequently used packages like parsec and pandoc, this is a no-brainer. For the same reason, however, immature packages that contribute little are better avoided.*]

GitHub

The Ampersand code base currently lives on GitHub. Our code contributions must also be on GitHub; this will facilitate easy integration of our code into Ampersand. This is especially useful if only parts of our code eventually become integrated into Ampersand - GitHub facilitates this especially. [**YS:** *Do we want to tell them that our code might not be integrated in Ampersand? I think we should abstract this information to make our configuration look more significant.*]

Graphviz

Graphiv is open source graph visualization software, which has the ability to visually represent information in the form of charts and graphs. It contains various designs from which the user is able to select. This is used to take descriptions of graphs in simple text and create diagrams. Ampersand generates reports about the input system using graphs and charts, so this is one of the basic components required to run Ampersand. It is not likely that Graphviz will be used in this project, unless it is deemed useful to visualize some of the proofs that are generated.

XAMPP

Ampersand generates information systems in the form of a website based on business requirements. To run, it requires a PHP-service with an SQL-database underneath. In the Ampersand project, XAMPP is used for that purpose. XAMPP stands for cross (i.e., X) platform Apache distribution containing MySQL, PHP and Perl. XAMPP is the most convenient way to set up a working database for Ampersand and access .php pages.

[**YT:** *Be liberal with what you include in this section, but if it is a 'technology' then put it in the previous section. Any acronyms, or words that the layman probably won't know, need to be here.*]

3.2 Naming Conventions and Terminology

ECA Stands for Event-Condition Action. The rule structure used for data bases and commonly used in market ready business rule engines. ECA rules are used in Ampersand to describe how a database should be modified in response to a system constraint becoming untrue. Event-Condition Action rules have the following structure

```
when <some event has happened>
and if <some condition is fulfilled>
    do <this activity>
```

[**JG:** *Please keep this in alphabetical order, or order it by chapter or something, but it needs to be in an order of some kind*]

ADL Stands for "A Description Language" From a given set of formally defined business requirements, Ampersand generates a functional specification consisting of a data model, a service catalog, a formal specification of the services, and a function point analysis. An ADL script acts as an input for Ampersand. An ADL file consists of a plain ASCII text file.

Ampersand Ampersand is the name of this project. It is used to refer to both the method of generating functional specification from formalized business requirements, and the software tool which implements this method.

Business requirements Requirements which exist due to some real world constraints; ie, financial, logistic, physical or safety constraints.

Business rules See *Business Requirements.*

EFA Stands for "ECA (see above) for Ampersand".

Functional specification A *formal* document which details the operation, capabilities, and appearance of a software system.

Natural language Language written by humans for the purpose of human communication; language intended to be interpreted and understood by humans, as opposed to machines.

Requirements engineering The process of translating business requirements into a functional specification.

[**YS:** *ADL is probably not "A data Language".*] [**YT:** *I swear to god I read "A data Language" somewhere... maybe I was delirious from lack of sleep... In any case, we should eventually find a reference for "ADL", whatever it is...*] [**SJ:** *You must have been delirious :-) . It stands for "A Description Language" or "Architecture Description Language". In fact, we don't us the acronym anymore. We just talk about "Ampersand" or "the Ampersand Language" if we need to be precise. So if you can remove the use of ADL from this SRS all together, that would be great!*] [**JG:** *this is in Joosten's article "Deriving Functional Specifications from Business Requirements with Ampersand", ADL is a tool that accompanies Ampersand (which is a method); ADL generates a functional specification consisting of a data model, service catalogue, a formal specification of the services, and a function point analysis*

It (ADL) translates business rules back into the natural language, and provides feed back to the requirments engineer for validating his work. (Also be careful with how you use the word validation vs. verification, these terms are distinguished in Rule based design and we dont want to contradict ourselves/our supervisors by using them interchangably)

ADL translates the entire set of requirements to design artifacts that are needed in subsequent software design processes (p.2)

They also specify the difference between requiement and specification: where specification is used to prescribe properties of a system to be built; and requirement describes an explicit or implicit need need of users] [**YT:** *Stef's article describes ADL as 'an accompanying tool' but also as the 'language which is input to Ampersand'. This is confusing and I feel like the former definition is not that useful so I decided to omit it. Just refer to 'Ampersand' as both the tool which is written in Haskell and this method of generating functional consraints from business requirements. To this end, I added a definiton of 'Ampersand'. But thanks for the reference to 'A description language'*] [**JG:** *I'm.. so confused lol the more we do this the more lost Ive become as to what is what and how we're going to do what we're suppose to do – I can't even tell if Ampersand is a system, model or tool.. or all three?*] [**SJ:** *Sorry for that. We use the name Amper-*

sand primarily for the approach (the method). But the Ampersand compiler is also called 'Ampersand'. And to really confuse everybody: the language is called 'Ampersand' nowadays. Only when we need to be precise (which is the case in SRS-es) we talk about the Ampersand-approach, the Ampersand-compiler, and the Ampersand-language. Why? Because everybody else is usually not capable of following the details and just calls everything 'Ampersand'.]

3.3 Relevant Facts and Assumptions

Chapter 4

Functional Requirements

4.1 The Scope of the Work

[**YT:** *The template has a bunch of stuff for this section. But I think this diagram is a good summary. Let me know what you guys think.*]

Figure **??** is a simplified view of the software design cycle, intended to highlight the role of Ampersand in this cycle. This view omits many of the uses of the design artifacts generated by Ampersand; instead it focuses mainly on the primary purpose, which is to help create a finished software system.

The contribution of this project is denoted with dashed lines. Note that it is isolated to a process completely internal to Ampersand. It should not affect the interface to Ampersand.

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description
Real World		
Requirements engineers		

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description

Ampersand system

[**SJ:** *Better to call this 'Ampersand compiler'*]Software engineers

ADL File

[**SJ:** *Better to call this 'Ampersand script' and reserve the phrase 'Ampersand file' for*]Real worl

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description
Finished product		
Business requirements		

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description
Internal representation		
Prototypes		

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description
Data models		
Design documents		
Functional specification		
Distribute product		
Reevaluate constraints		

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description
Interpret constraints		
Formalize		
Reevaluate requirements		

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description
Verify and compile		
Automatically fix violations		
Write software		

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description

Product fulfills requirements?

Software matches specification?

Table 4.1: Description of entities present in figure **??**

Entity	Type	Description

Design is feasible?

---

we do with this? It doesn't seem as relevant as the previous section....

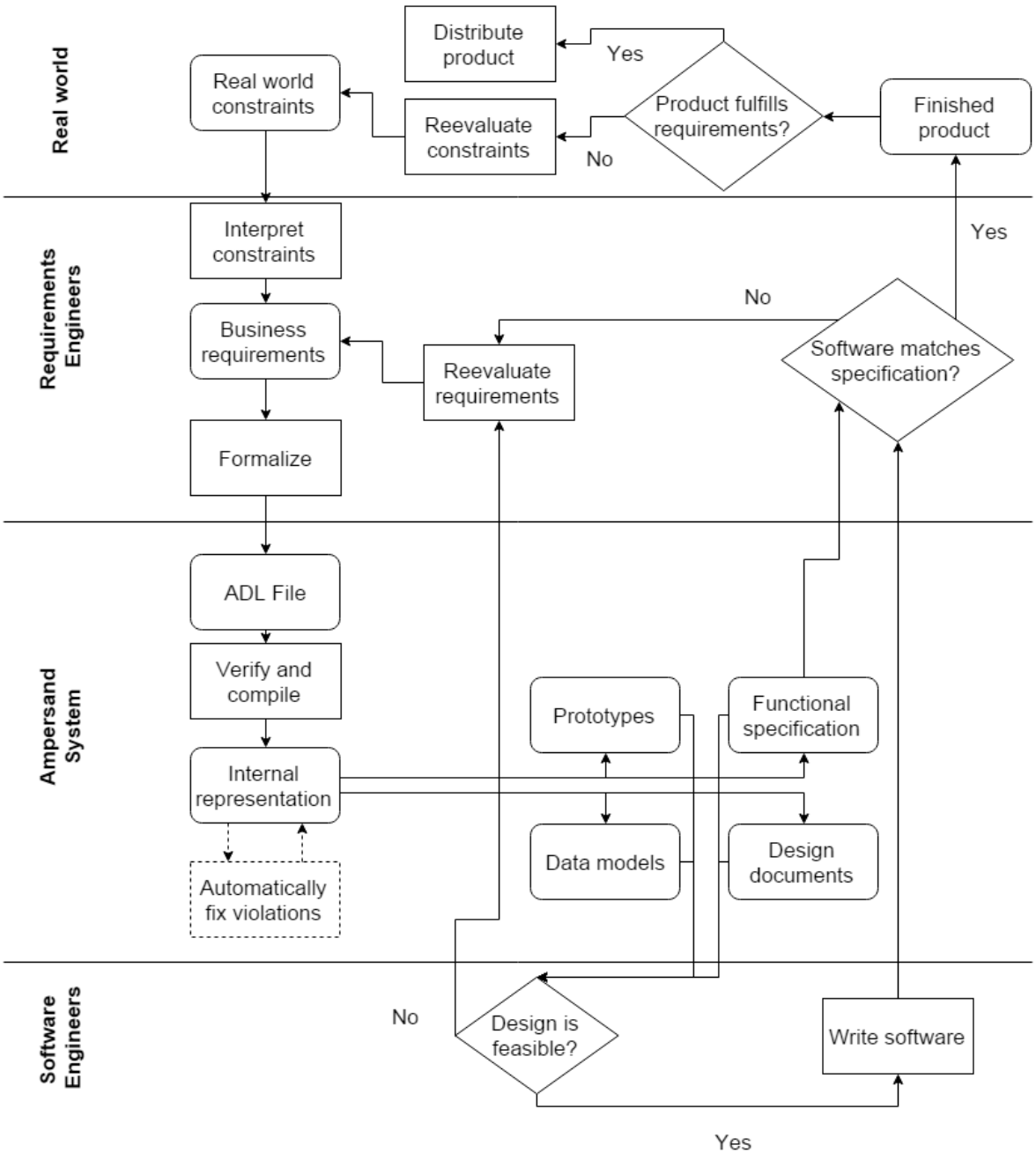

Figure 4.1: Business process diagram representing the role of Ampersand in the software design cycle

4.2 The Scope of the Product

[**JG:** *Scope = work that needs to be done to delivery a product*] E.F.A. require formal proofs and documentation that are provable in addition to its implementation; the implementation requires the final product to be able to implement an algorithm in Haskell that restores rules made by the user (i.e., invariants) and correct any data which have violated these rules. A secondary component of this project is the implementation of E.C.A. (Event-Condition Action) Rules used for reactive rule-based systems which has the ability to discover and systematically repond to triggers for events in a timely manner. Currently, implementation and architecture of ECA rules systems are intensively studied in Active Database Management Systems. There are various problems that may arise during implementation of the E.C.A. rules translation, what those problems may be are unforseeable at this stage of development. Although there may be an algorithm for this implementation, we have no been able to find it much less an example of its implementation.

4.3 Functional Requirements

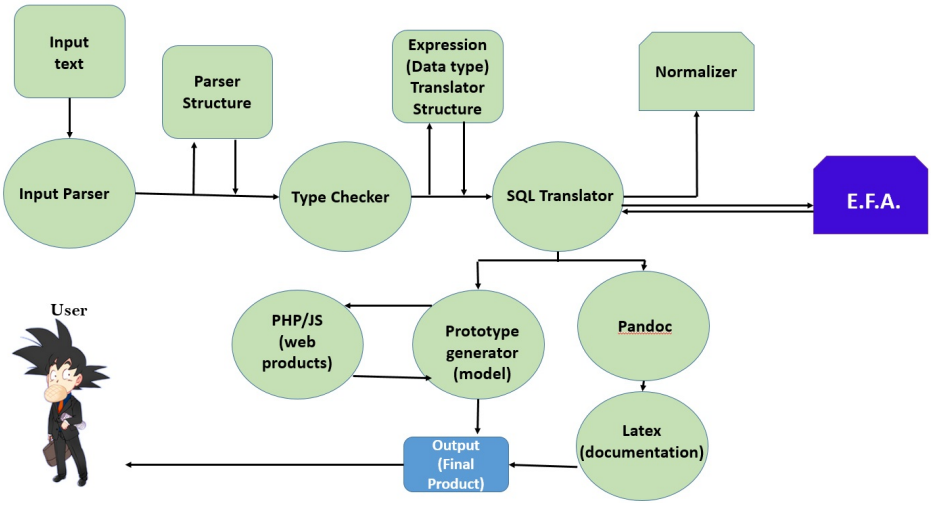

Figure 4.2: Diagram of how E.F.A. fits into Ampersand

Summary of Functional Requirements:

- Our code contribution must provably implement the desired algorithm.
- Our implementation must accept its input in the existing ADL format.
- Our implementation must produce an output compatible with the existing pipeline.

- Our implementation must be a pure function; it should not have side effects.
- Our implementation must not introduce appreciable performance degredation.
- Our implementation must provide diagnostic information about the algorithm to the user, embedded in the chapter 'Diagnostics' of the functional specification document.

[**JG:** *To YT: I'm sorry I used bullet points, but I needed more before writting this out and its 4:30 in the morning, I'm not sure you're available*]

4.3.1 User Requirement

[**YS:** *Second Approach*] [**JG:** *Are we choosing or combining?*]

Requirement # : U1 **Priority:** 4
Description: If the ECA-rule generator leaves choices open for the user, the static choices must be made by the requirements engineer and the dynamic choices must be made by the user of the prototype.

Rationale: To ensure that decisions are made in situations that are undecidable.
Fit Criterion: The Data in the database preserves these rules and maintains consistency.

Requirement # : U2 **Priority:** 1
Description: The user shall be able to write ECA-rules in the Ampersand-script.

Rationale: To allow user to interfere with the ECA-rule generation process.
Fit Criterion: The Data in the database preserves these rules and maintains consistency. **Consideration:** This is highly dangerous functionality, because invariants may be compromised on runtime.

4.3.2 Product and System Requirement

[**JG:** *maybe this note should go into glossary or something?*] *Note to reader:* Customer Satisfaction/Dissatisfaction: 1 to 5 (low to high)

Conflicts:

→ none: no existing conflict

→ minor: does not afffect functionality and can be remodeled to fit into current system

→ medium: requires immediate attention, but can be resolved

→ major: severe system conflict with Ampersand and requires alternative implementation

Priority: 1 to 5 (low to high)

→ 1 is the lowest, and it can be left out if conflicting with a higher priority item

→ 5 is the highest, and it is not something that can be compromised or replaced at any point during product development

Requirement:	S1	**Type:**	Functional
Description:	Cohesion of new product into existing system by using Haskell		
Rationale:	Necessary for E.F.A. to be added to source code		
Originator:	Client/Designers		
Fit Criterion:	Able to compile using GHC or Ampersand – Haskell function		
Customer Satisfaction:	5 - Highest		
or			
Dissatisfaction:	5 - Highest		
Priority:	5 - Highest	**Conflicts:**	none
Supporting Materials:	Ampersand Documentation (Rule Based Design)		
History:	N/A		

Requirement:	S2	**Type:**	Functional
Description:	Able to recognize and parse input		
Rationale:	Necessary for E.C.A. rules to be added, system needs input		
Originator:	Client/Designers		
Fit Criterion:	Able to take input and passes it out as correct type		
Customer Satisfaction:	5 - Highest		
or Dissatisfaction:	5 - Highest		
Priority:	5 - Highest	**Conflicts:**	none
Supporting Materials:	Ampersand Documentation (Rule Based Design)		
History:	N/A		

Requirement:	S3	**Type:**	Functional
Description:	Able to translate input into E.C.A. rules		
Rationale:	Major portion of project and E.F.A. is useless without		
Originator:	Client/Designers		
Fit Criterion:	Outputs E.C.A. rules in proper format for F-Spec (i.e., SQL translator)		
Customer Satisfaction:	5 - Highest		
or Dissatisfaction:	5 - Highest		
Priority:	5 - Highest	**Conflicts:**	none
Supporting Materials:	Rule Based Design **E.C.A. for Web Development**		
History:	N/A		

Chapter 5

Non-functional Requirements

Non-functional requirements are fundamental to the success of any product on the market. This type of requirement is especially important to this project as it heavily relates the needs of the clients to the changes that need to be made to the final product before it is ready for the public. The human element guides the principals introduced in non-functional requirements, and for this project, it is motivated by a simple perspective. Tools should make a users job easier to do, and for that to happen the user must be able to under how to use it. [**JG:** *How do I pull the tree up to the first page of non-functional?? *facepalm* doesnt work*] [**YT:** *LaTeX floats have 'position modifiers'. Generally you should use !htb, this works in over 99 percent of case in my experience. Full details of floats and their positioning here: https://en.wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions*]

5.1 Usability and Humanity Requirements

[**SJ:** *Don't worry about the lack of user interface. There is plenty of Usability and Humanity Requirements around anyway. One is the question whether it is necessary to guide the user in picking the right ECA-rule, if there is a choice to be made. Another one is how the effect of running an ECA-rule on run time is displayed in the (current) user interface. An important one is also how the system documents this functionality, when generating a functional specification.*] *E.F.A. is built as an internal structure of Ampersand and does not have a graphical interface for users. Ampersand uses the command prompt to compile user prototypes which includes various graphical representations and webpages. It is very unlikely that users will be able to distinguish E.F.A. from Ampersand, it would be similar to someone attempting to distinguish a persons hand as a separate entity from that person. E.F.A. may not look very*

personable, or user friendly, but it will hopefully relieve some frustration and make the user feel more at ease with their design because if things do not line up, there is something that can automatically fix it for them. It would be another thing users would not have to worry about when designing a new system, this feeling of ease could extend to smooth the interactions between the designer and the client as it goes through various cycles of overhaul. The user will be notified of any changes that have been made to their design and from there Ampersand provides options of how much of the changes the user would like to keep, fix themselves, or add to a repository of changes to keep in mind for future instances. Thus E.F.A. can return corrections according to what the user specifies.

This product shall be used by any individual who is willing to dedicate the time to learning the basics of the Ampersand model and wishes to design an information system. The time it take to learn Ampersand is truly up to the user, a normal individual may take anywhere from a week to a month. While an engineer who has a strong background in mathematical logic or minor experience in programming will be able to use Ampersand efficiently within a week. The most pronounced problem concerning Ampersands usability is that its limited to two languages: English and Dutch. Those who are not fluent in neither of these languages will have great difficulty learning the Ampersand system.

*However, since Ampersand is a successor of CC, (**?**, p.10) and similarly it could be used to create consensus between groups of stakeholders with strong diverging opinions on the same matter. Once the prototype is generated, it can be presented to different individuals who may wish to conduct their own investigation concerning the best course of action. After the creation of the prototype, common users are able to manipulate various aspects of a project due to an autofill feature.*

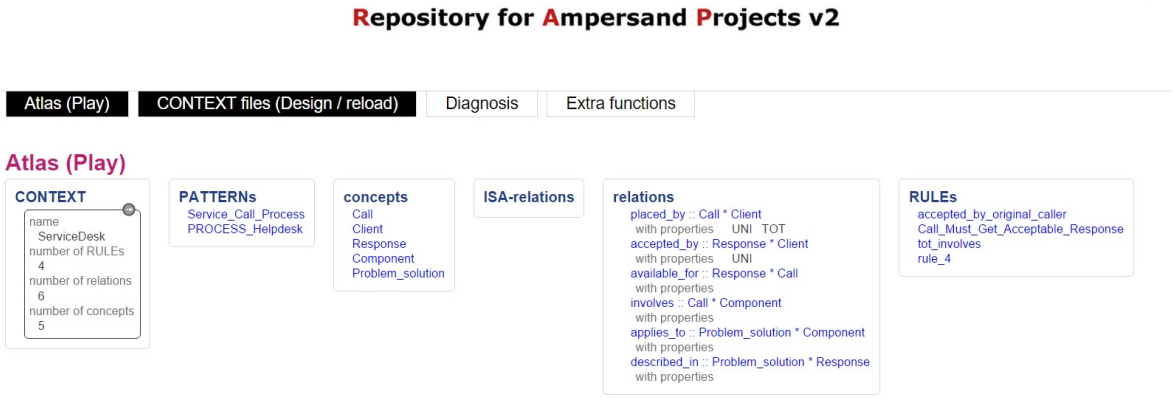

Figure 5.2: An example of what Ampersand users see when they compile a prototype, each tab and figure they have created can be modified.

5.2 Understandability

[**SJ:** *Nice example...*] *What E.F.A. does is simple to understand, it fixes logical inconsistencies with minimal if any input from the user. It looks for contradictions and violations, an overly simplistic but accurate example would be: a chair is red, a chair is blue, and there is only one chair. That chair cannot be blue and red simultaneously, to fix the problem either another chair is added or one of the colour conditions is eliminated. What is eliminated may be based on what is more important for the system and the user. In the previous example, if it is most important that there is only one chair, then one of the colours statements is erased. Otherwise, E.F.A. could add a new chair, now ever if adding a new chair decreases the efficiency of the system because it must keep track of two chairs now. It may pick a chair colour depending on another criteria that is not directly part of this problem. Such as if most individuals who may use the chair prefer it to be red rather than blue. That may be an external condition that Ampersand has been programmed to keep in mind.*

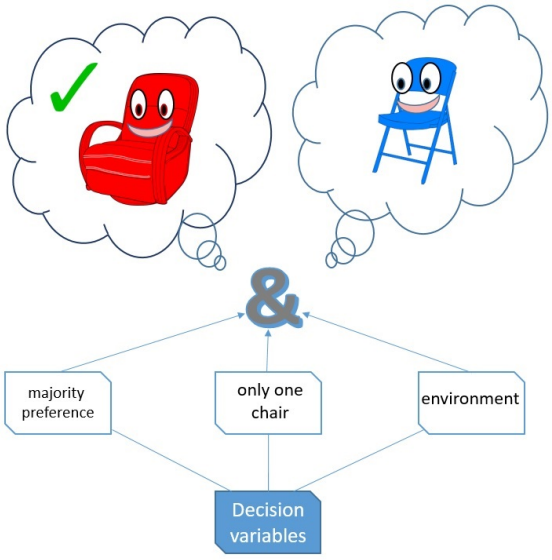

Figure 5.3: Simplistic diagram of decision making process, for full example of decision making process see. Figure 5.

5.3 Performance Requirements

[**SJ:** *If ECA-rules are assembled, there may be loops. That might cause nontermination. If such loops arise, they will be easy to signal. Yet, the runtime code must have some provision to interrupt an infinite loop.*] *Ampersand can efficiently process a large amount of information in a short amount of time, the process that E.F.A. requires will be added onto Ampersands process time. The duration of time it would take to verify data inconsistencies would depend on the complexity of the system design and the number of corrections it may take to update the system to a valid prototype according to specification. The current upper bound is expected at 120 seconds, or 2 minutes, the correction process is terminated if exceeds beyond this threshold for simple Ampersand models. The prototypes termed simple models are tasks that can be accomplished by an individual within a short amount of time, such as selecting the best actor/actress to play lead according to experience and audience appeal. Simple models are able to take into consideration not only the task at hand but the context and environment to which the task must be completed in. Performance is measured on speed, but also on validation of limitations set forth by the user; it will not matter how quickly E.F.A. can do something if it does it incorrectly and creates more work for the user. Currently, the quantification of the desired accuracy of the results produced is unknown, as E.F.A. is in its initial stages. Furthermore, the allowable time between failures is unknown and untested.*

E.F.A. can be a great tool, but in case of failure, in the unlikely event that errors are

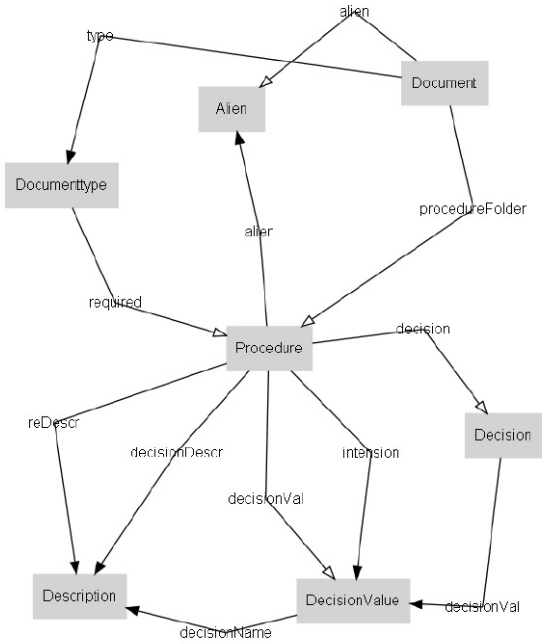

Figure 5.4: This is an example of Ampersand's complete decision process. Source : Rule Based Design

*propagated through the system and no one through any cycle of development catch it. If the designer puts full faith in E.F.A. it could be catastrophic, especially if mistakes are not caught and products proceed into the production stage. However, Ampersand possess an internal structure to detect inconsistencies and allows for manual correction. E.F.A. although important is not the last line of defense, this secondary system greatly minimizes the chances of complete failure. The likely if E.F.A. completely fails, is that the secondary system detects errors and compels users to manually correct data ((***?***, 153)).*

5.4 Operational and Environmental Requirements

It is not feasible to predict an accurate workload capacity for E.F.A. for its state, however, E.F.A. is made to handle Ampersand loads interactively and is not designed to accumulate stored data but rather return the user a modified version of the input they submitted. E.F.A. is meant to operate locally within Ampersand, there no limit to the number of users that can use this product. Extensive maintenance is not required, although minor adjustments and additions may be necessary for further improvement. As Ampersand grows in popularity, it will not be necessary for E.F.A. to grow in

proportion to Ampersand to remain useful. Due to this it is difficult to determine the longevity of this project, as it could remain until the client determines the current model is no longer useful.

Any system that is capable of running Ampersand is automatically compatible with E.F.A. [**SJ:** *These operational requirements look very artificial. On my part, you don't have to invent requirements just to fill a void in the SRS. I'd rather see useful requirements.*] *Currently those system are limited to Windows OS 7 and up. There are very few physical conditions required to maintain E.F.A. and they are the basic conditions required to keep most computer system running. This includes minimal moisture, and temperatures no lower than 35 (degree farenheight) or 1.7 (degree celcius) as computer hardware of change by flexing* [**JG:** *Is this common knowledge or does this need a citation?*] *and temperatures preferably no higher than 90 (degree Farenheight) or 32.2 (degree celcius)*

5.5 Maintainability and Support Requirements

*Maintenance is a key requirement from our clients as non-maintainable code takes far too long to decipher and upgrade. The code is required to be well documented and easy to understand so that any programmer who wishes to make additions or expand upon what is already there, can do so easily without struggle. Well documented code creates a sense of cohesion among individual parties participating on the same project. In order to meet the maintainability, E.F.A. must all requirements, such that each specification is traceable back to the motivation for its existence and the purpose it drives ((***?***, 2)). Although there are various groups, such as Dr. Joosten and his team who are available to assist individuals and businesses with any part of Ampersand, our team believes that given a specific enough error message, an individual may not require outside assistance. Moreover, E.F.A. will be accompanied with documentation that will have various examples for the user to explore. Part of the reason that non-functional requirements as essential to E.F.A. concerns the requirements for interfacing with Adjacent Systems. Thus features concerning how implementation is done which is normally based on the choice of the designers is a functional requirement when it comes to E.F.A.*

5.6 Security and Integrity Requirements

[**SJ:** *Are there any security risks for the generated information system, as a consequence of E.F.A.?*] [**JG:** *this section includes access requirements: who is au-*

thorized to access the product. Since its open source, anyone can access or change it, it just wont be part of the official source code? We're not dealing with sensitive information (there's no confidentiality breach possible..) I mean we could name system function name, and user roles but thats a rather short two sentences

There also specificiation of requried integrity of databases; this is partially applicable, it applies to Ampersand and how it requires a local database but E.F.A. doesn't.. however, it is part of Ampersand.

We dont have privacy issues.. do we? Audit requirements — specification of what the product has to do (work?) —- we might need audit requirements I mean we could say that if we fed it standard rules it should be able to recognize, translate and fix easily. like if A (subset) B and B (subset C) then A (subset) C. If a is need for b to function, and b is needed by c to function, then by ¡insert property name¿ a is required for c.

Immunity requirements: we dont.. really have security, so immunity is confusing.

[ ]

As an open source project any individual who clones the Ampersand Github has access to source code and can manipulate it as they wish. Irrelevant of what changes the user makes locally, it does not effect the Github respository, as only developers of Ampersand (i.e., our client) has permission to change source code in the respository. Access to the database and software which supports the various functions of Ampersand are run locally and subjected to the security system the user has in place on his or her work station.

5.7 Validation and Verification Requirements

[**JG:** *We can distinguish validation from verfication here, like Joosten did, and link how validation is required for verification and validation must be done by the user. As Joosten specifies that validation is only something that the user can do.. however, does E.F.A. in theory validate the state of the system by fixing inconsistent data?*

Someone help a brother out?What's below cant be all there is]

Validation and verification are important parts of non-functional requirements used to test the quality of the final product. Validation is a human activity and requires user input; the rules for the software system to follow are provided by the user, and only the user can judge if the rules are correct for the system that they are attempting to create. The validation that E.F.A. offers is a logical test to confirm that all data

sets do not violate the rules that the system follow for the model to properly mimic realistic conditions.

Verification test the accuracy of E.F.A., and how well it can replicate what the use has in mind according to an E.C.A. rule system. Verification is an automated process that confirms correctness of the project in accordance to previously validated rules. Furthermore, verification confirms the success of E.C.A. rule adoption, and is able to quantify the level of success or failure encountered by E.F.A..

5.8 Legal Requirements

The implementation must eventually be included in Ampersand, which is licensed under GPL3. To comply with this license, all of the implementation code must be either written by us so we may license it under GPL, or must already be licensed under GPL, or a compatible license, by its original author. We do not plan to use existing code, other than as a reference. [**YS:**] *Rephrase this one, I wanted to put forward the idea*] [**YT:**] *I rephrased it more concretely: we have to follow the license agreement of Ampersand, which appears to be GPL3. That means all the code we include must either be our own so we can place it under that license, or must already be under GPL. I don't see any reason we couldn't take code from somewhere if it was GPL.*]

Chapter 6

Project Issues

6.1 Open Issues

N/A [**YS:** *Since we're not concerned with any problem or issues associated with Ampersand.*]

6.2 Off-the-Shelf Solutions

No off the shelf solutions exist for this project.

6.3 New Problems

N/A [**YS:** *Since we're not concerned with any problem or issues associated with Ampersand.*]

6.4 Tasks

- *Analyze and the existing software code base and do an impact analysis of our project on the existing software.*
- *Propose a solution to the supervisor and product owner.*
- *Implement the solution and provide the annotated source code to the supervisor and the product owner for a review.*

- *Incorporate any changes suggested by them and create a pull request in the main Ampersand repository upon successful completion of the task.*

[**YS:** *Add in any significant task you feel is worth mentioning*]

6.5 Migration to the New Product

Upon final review by the client and intensive testing, if the client is satisfied by the quality of code and its maintainability, the implementation will be made part of the production stream. This process is quite simple due to the nature of the project; the core development team of Ampersand is quite small, and the project is hosted on GitHub; so our migration will consist of submitting a pull request to the Ampersand repository.

6.6 Risks

- *The new code must not introduce any errors or performance regressions into Ampersand.*
- *The code must satisfy existing tests and additional tests written for the new algorithm being implemented.*

6.7 Costs

Currently the Ampersand software system is open source and maintained by Tarski Systems. All the software subsystems used in Ampersand are also open source. There will no change in cost as a result of our implementation. The client (Tarski Systems) will be responsible for managing the cost of maintenance of the software in the future. All software used in the development of E.F.A (GHC, LaTeX, etc.) are open source as well; there is no cost requirement for any component used.

6.8 User Documentation and Training

User documentation will accompany E.F.A. which will give a brief explaination concerning how E.F.A. works and how users can use E.F.A. to maximize their efficiency. E.F.A. requires minimal input from user, but each possible input will be explained thoroughly and various examples will be provided in a step by step tutorial to help

familiarze the user with E.F.A.. Furthermore, a practice simulation will be in place for the user. The test will include a well documented default script which the user can compile to try out different priority options. The manual will include pictures as a type of visual guide concerning what the screen should look like from beginning to end. The documentation will likely be part of the Ampersand documentation, but for the purposes of this project, it will be submitted independently.

6.9 Waiting Room

6.10 Ideas for Solutions