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 properly capture the attributes of the information system unambiguously.[**JG:** *Good job! what do you think about saying "to unambiguously capture attributes of the information system" ?*] Typically, this translation of business requirements to a formal specification is done by a requirements engineer; this can be an error prone process. [**JG:** *what do you think about: "which can be prone to human error"?*]

Ampersand is a tool which aims to address this problem in a different way; by translating business requirements written in natural language into a formal specification by means of a compilation process. [**JG:** *Maybe we should cut out how entirely, because there's a design specification – and solely focus on what? thoughts?*] Even though the business requirements and formal specification are written in entirely different languages, the "compiler guarantees compliance between the two".

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; including data models, service cataogues and their specifications. Requirements engineering is perhaps most important in safety-critical systems; to this end, Ampersand generates modelling 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.

For example, consider a system for ordering products online. Ampersand takes as an input statements of business requirements like [**JG:** *Good example, could we put something in here about restrictions of of the real world?*]

> *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.*

These are translated into, among other things, formal rules concerning how the information system must react to changes in its state. 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 )
```

where `Ref x` represenets 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)*] 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.

2.3 The Stakeholders

[**YT:** *These are the only real stakeholders I can think of...*]

2.3.1 Ampersand Designers

Ampersand designers include Dr. Stef Joosten, Dr. Sebastian Joosten and Dr. Wolfram Kahl, who in addition to supervising the development of E.F.A are also Ampersand contributers. Dr. Stef Joosten is the designer of the Ampersand method which have been successfully implemented in various public sectors. The interest stemming from these three individuals are not only professional but also personal. Ampersand has been nourtoured by a greal 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 informatoin

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.

2.3.2 Requirements engineers

¡¡¡¡¡¡¡ HEAD 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 minutea. The ability to deligate time consuming tasks such as manually checking over technical specifications would increase the efficiency of the task at hand as well as decrease the number of hours required for completion. An added benefit which engineers recieve is the ability to confirm that the system they have designed matching the specifications given to them by the client. E.F.A provides the ability to ensure no contradictions are found within the design by automatically fixing inconsistency that may arise.

2.3.3 Architects & Functional Specification Designers

System architects, simliar 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 esthetics, 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 schemas. 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.

2.3.4 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 unforseen 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.

=======

2.3.5 Software engineers

¿¿¿¿¿¿¿ 2254b513e6e9df0c7cf25b177fa3ccc947f07129

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 struction 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 constrait 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 montary constraints due to the open source software opens up a broad base of opportunities for future improvements and limitless contributes 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 maintable: 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.

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 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. We may not use additional packages.

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.

3.2 Naming Conventions and Terminology

E.C.A. (Event-Condition Action): Rule structure used for data bases and commonly used in market ready business rule engines.

```
Event-Condition Action has the structure

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

E.F.A.: stands for E.C.A for Ampersand, which is the name of this project.
Graphviz: open source graph visualization software, has the ability to visually represent information in the form of charts and features. 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. This is oe of the basic components required to run Ampersand.
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
XAMPP is easy to install and can be accessed
here:

`https://www.apachefriends.org/index.html`

3.3 Relevant Facts and Assumptions

Chapter 4

Functional Requirements

4.1 The Scope of the Work

4.2 Business Data Model and Data Dictionary

4.3 The Scope of the Product

4.4 Functional Requirements

4.4.1 Input

4.4.2 Correct Translation of Input

4.4.3 Output

Chapter 5

Non-functional Requirements

This section highlights components of this project which is not required for the product to function but are highly important in the success of the product.

5.1 Look and Feel Requirements

The majority of Ampersand production features must be done through the command line, but the products that Ampersand produces has a visual component. E.F.A. will not have a direct interface as it is an internal component of Ampersand. Ideally, a template is created which an autofill feature that allows users to specify how data conflicts should be fixed. But, currently that is part of additions for the future.

5.2 Usability and Humanity Requirements

E.F.A. is engineer-friendly as it uses mathmatical logic, however it may not be as obvious for other users who are not familiar with the logical associated with mathematical relations. Ampersand is designed for any user who is willing to learn it, with that said the learning curb can be high for individuals who are not technology savvy. This product should be usable by someone with a highschool background. Ampersand was designed with multiple groups of stakeholders in mind and aims to automate the correction of data inconsistency according to the rules specified by the user. Ampersand is not geared towards individuals with learning disabilities, or those who have visual disabilities. The use of Ampersand and by extension it's extension E.F.A is geared towards individuals who have full utility of their visual and tactile senses. Currently, there is nothing in place that is especially geared towards individuals with disabilities.

However, there are various software available on the market that can help an individual with visual or tactile disability navigate a computer system. Using the same set of software, those individuals should be able to successfully use E.F.A. within Ampersand.

5.3 Performance Requirements

Ampersand is highly efficient and capable of modeling large systems in a short amount of time. The time it takes Ampersand to build a prototype depends on the complexity of the system, and the amount it takes E.F.A. to correct a model for data inconsistencies will add on the performance time of Ampersand. Though there is no minimum time for the use of E.F.A., there is a time limit in place that will halt any process once it passes the threshold of 5 minutes. Though this time may be extended and vary depending on the complexity of the prototypes and the various features the user may choose to incorporate into each model. [**JG:** *A hardline can be added later, but as of now since we are not running scenarios, it is difficult to pin point where the cut off should be. An upperbound was considered, but that can added later*]

5.4 Operational and Environmental Requirements

5.5 Maintainability and Support Requirements

5.6 Security Requirements

5.7 Cultural Requirements

5.8 Legal Requirements

Chapter 6

Project Issues

6.1 Open Issues

6.2 Off-the-Shelf Solutions

6.3 New Problems

6.4 Tasks

6.5 Migration to the New Product

6.6 Risks

6.7 Costs

All software used in the development of E.F.A are open source, there is no cost requirement for any component used.

6.8 User Documentation and Training

6.9 Waiting Room

6.10 Ideas for Solutions