

Ampersand Event-Condition-Action Rules

Test Plan

Yuriy Toporovskyy, Yash Sapra, Jaeden Guo

Table 1: Revision History

Author	Date	Comment
Yuriy Toporovskyy	27 / 10 / 2015	Reorganized document
Yuriy Toporovskyy	27 / 10 / 2015	Initial version - template

Contents

1	General Information	1
1.1	Purpose	1
1.2	Objectives	1
1.3	Acronyms, Abbreviations, and Symbols	2
2	Plan	3
2.1	Software Description	3
2.2	Test Team	3
2.3	Test Schedule	3
3	Methods and constraints	4
3.1	Methodology	4
3.2	Test tools	4
3.2.1	Unit Testing	4
3.3	Requirements	4
3.3.1	Functional requirements	4
3.3.2	Non-Functional requirements	5
3.4	Data recording	5
3.5	Constraints	5
3.6	Evaluation	5
4	System Test Descriptions	6

List of Figures

List of Tables

1	Revision History	i
---	----------------------------	---

Chapter 1

General Information

1.1 Purpose

This document outlines the test plan for ECA for Ampersand, including our general approach to testing, system test cases, and a specification of methodology and constraints. This test plan specifically targets our contribution to Ampersand, namely ECA – elements of Ampersand, such as design artifact generation, will not be tested.

1.2 Objectives

Preparation for testing

The primary objective of this test plan is to collect all relevant information in preparation of the actual testing process, in order to facilitate this process.

Communication

This test plan intends to clearly communicate to all developers of ECA for Ampersand their intended role in the testing process.

Motivation

The testing approach is motivated by constraints and requirements outlined in the Software Requirements Specification. This document seeks to clearly demonstrate this motivation.

Environment

This test plans outlines the resources, tools, and software required for the testing process. This includes any resources needed to perform automated testing.

Scope

This test plan intends to better describe the scope of our contribution, ECA, within Ampersand.

1.3 Acronyms, Abbreviations, and Symbols

SRS Software Requirements Specification. Document regarding requirements, constraints, and project objectives.

ECA Rule Event-Condition-Action Rule. A rule which describes how to handle a constraint violation in a database. See SRS for details.

Chapter 2

Plan

2.1 Software Description

Ampersand is a software tool which converts a formal specification of business entities and rules, and compiles it into different design artifacts, as well as a prototype web application.

This prototype implements the business logic in the original specification, in the form of a relational database with a simple web-app front-end.

A particular class of relational database violations can be automatically restored; the algorithm for computing the code to fix these violations is called AMMBR [Joo07]. This class of violations is realized within Ampersand as ECA rules – our contribution to Ampersand will add support for ECA rules, in both the Ampersand back-end and the generated prototype.

2.2 Test Team

The test team which will execute the strategy outline in this document is comprised of

- Yuriy Toporovskyy
- Yash Sapra
- Jaeden Guo

2.3 Test Schedule

Chapter 3

Methods and constraints

3.1 Methodology

3.2 Test tools

3.2.1 Unit Testing

The group will be using HUnit for automated unit testing of the new source code in Ampersand. The testing tool will be critical in analyzing the effects of our project on the existing system. In unit testing, a test case is the unit of test execution. That is, distinct test cases are executed independently and the failure of one is independent of the failure of any other test case.

HUnit is a library providing unit testing capabilities in Haskell. It is an adaption of JUnit to Haskell that allows you to easily create, name, group tests and execute them. The idea of using HUnit is similar to that of JUnit in Java; we feed some data to the functions that we're testing and compare the actual results returned to the results we're expecting.

3.3 Requirements

3.3.1 Functional requirements

The functional requirements for ECA for Ampersand are detailed in the SRS; they are also briefly summarized here. Our implementation must

F1 provably implement the desired algorithm.

- F2** accept its input in the existing ADL file format.
- F3** produce an output compatible with the existing pipeline.
- F4** be a pure function; it should not have side effects.
- F5** not introduce appreciable performance degradation.
- F6** provide diagnostic information about the algorithm to the user, if the user asks for such information.

3.3.2 Non-Functional requirements

The functional requirements for ECA for Ampersand are detailed in the SRS; they are also briefly summarized here. Our implementation must

- N1** produce output which will be easily understood by the typical user, such as a requirements engineer, and will not be misleading or confusing.
- N2** be composed of easily maintainable, well documented code.
- N3** compile and run in the environment currently used to develop Ampersand.
- N4** annotated generated code with proofs of correctness or derivations, where appropriate.
- N5** automatically fix database violations in the mock database of the prototype.

3.4 Data recording

3.5 Constraints

3.6 Evaluation

Chapter 4

System Test Descriptions

Bibliography

[Joo07] Stef Joosten. AMMBR: A Method to Maintain Business Rules. 2007.