Peer review & feedback provided by Team 5.
Jason Kim, Anmol Sharma, Can Birol, Marcus Koether

November 6, 2015

Ampersand
Event-Condition-Action Rules

Test Plan

Yuriy Toporovskyy, Yash Sapra, Jaeden Guo

Could include macids as
suggested by marking scheme

CS 4ZP6
October 30th, 2015
Fall 2015 / Winter 2016

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 Static Typing . . . 4
  - 3.2.2 Formal verification . . . 4
  - 3.2.3 Random Testing . . . 4
  - 3.2.4 Unit Testing . . . 5
- 3.3 Requirements . . . 5
  - 3.3.1 Functional requirements . . . 5
  - 3.3.2 Non-Functional requirements . . . 6
- 3.4 Data recording . . . 6
- 3.5 Constraints . . . 6
- 3.6 Evaluation . . . 6

4 System Test Descriptions **7**
- T1 PA clause executing “All” subclauses . . . 7
- T2 Empty PA clause . . . 8
- T3 EFA System Compatibility . . . 8
- T4 EFA Pure Function . . . 9
- T5 EFA User Feed Back . . . 10
- T6 EFA Code Walk-through . . . 11
- T7 Degradation Test . . . 12
- T8 EFA Annotated Code . . . 12

Chapter 1

General Information

1.1 Purpose

Consider briefly explaining the product's main function/purpose, as it is unclear without reviewing previous documents.

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

Typo: preparation

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

Grammar--Fragment; consider "(...)in preparation for, and to help facilitate the actual testing 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.

The description for the document's scope is that the document intends to better describe the scope? Consider clarifying.

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.

HUnit A Haskell library for unit testing. See TODO: ref test tools TODO? Future section?

QuickCheck A Haskell library for running automated, randomized tests. See TODO: ref test tools

This section is missing initialism/acronym definitions that are used later in the document (PA, EFA, etc)

A little more info. about HUnit and QuickCheck would be good.

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 **?**. 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.

Is the ? intentional? If so why is it there? It is confusing and you should consider removing it

2.2 Test Team

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

Outlined

- Yuriy Toporovskyy
- Yash Sapra
- Jaeden Guo

2.3 Test Schedule

Test Schedule missing.

Chapter 3

Methods and constraints

3.1 Methodology

Methodology missing.

3.2 Test tools

List of test tools not listed that will be used in the testing process.

3.2.1 Static Typing

Programming languages can be classified by many criteria, one of which is their type systems. One such classification is static versus dynamic typing. Our implementation language, Haskell, has a static type system. Types will be checked at compile-time, allow us to catch errors even before the code is run, reducing the errors that need to be found and fixed using testing techniques.

Some information about dynamic typing will also be useful.

3.2.2 Formal verification

A part of our project deals with generating source code annotated with the proof of derivation of that source code, which will act as a correctness proof for the system. In particular, when we generate code to restore a database violation using ECA rules, then the generated code will have a proof associated with it, which details how that code was derived from the original specification given by the user.

Proof using Hoare Logic ?

3.2.3 Random Testing

Random testing allows us to easily run a very large number of tests without writing them by hand, and also has the advantage of not producing biased test cases, like a programmer is likely to do.

? = Uncertainty or a missing symbol?

We will be using QuickCheck **?** for random testing. The existing Ampersand code base using QuickCheck for testing, therefore, using QuickCheck has the added benefit of easier integration with the existing Ampersand code base.

QuickCheck allows the programmers to provide a specification of the program, in the form of properties. A property is essentially a boolean valued Haskell function of any number of arguments. QuickCheck can test that these properties hold in a large number of randomly generated cases. QuickCheck also takes great care to produce a large variety of test cases, and generally produces good code coverage. QuickCheck will be used for individualized module testing and well as provide a fair array of random tests for the combination of all modules **?**. Minor: either '?' or '.' could be used exclusively

3.2.4 Unit Testing

Typo: 'resultd' could be fixed

Unit testing is comprised of feeding some data to the functions being tested and compare the actual results returned to the expected resultd. We will be using HUnit for unit testing of the new source code in Ampersand. 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.

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

Overall formatting for this section could be improved. Indenting or otherwise indicating that it was a list might make it better, also semicolon after the 'must' to

F1 provably implement the desired algorithm.

F2 accept its input in the existing PAClause format.

F3 produce an output compatible with the existing pipeline.

F4 annotated generated code with proofs of correctness or derivations, where appropriate.

Formatting here could be improved.

F5 automatically fix database violations in the mock database of the prototype.

F6 not introduce appreciable performance degradation.

F7 provide diagnostic information about the algorithm to the user, if the user asks for such information.

3.3.2 Non-Functional requirements

Non-Functional Req.

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

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 be a pure function; it should not have side effects.

3.4 Data recording

Consider adding a N/A or explanation for why these headings are blank.

3.5 Constraints

3.6 Evaluation

It is unclear to me whether you have any unit tests planned, consider including them or a section on what kind of unit tests you will do in the future. This could be difficult to determine at the moment due to not having any units which need testing, but a section including future plans would be good.

Chapter 4

System Test Descriptions

T1 PA clause executing "All" subclauses

Test type	Black box/Unit Test
Schedule	January 2016
Requirements	Functional

Input

The input is an ECA rule of the form:

On Ins(Δ, r_0) **Do All**(**Ins**$(e_0, \mathbb{I}_{C_0})$, **Ins**$(e_1, \mathbb{I}_{C_1})$)
where $C_0, C_1 ::=$ Concept
$r_0 : [C_0 \times C_1]$
$e_0, e_1 :$ Expression

Output

The output is an abstract SQL statement of the form:

```
INSERT INTO Concept_0_population VALUES e_0_query;
INSERT INTO Concept_1_population VALUES e_1_query;
```

Description

ECA rules of the input format are generated using QuickCheck, converted to abstract SQL, then compared against the expected output format using HUnit.

T2 Empty PA clause

What is a PA clause? It does not seem to be defined.

Test type Black box/Unit test
Schedule January 2016
Requirements Functional

Input

The input is an ECA rule of the form:

On $<$ **Ins/Del** $>$ Δr_0 **Do Nop**
where $C_0, C_1 ::=$ Concept
$r_0 : [C_0 \times C_1]$

Output

The output is the empty abstract SQL statement; that is, the SQL statement which represents "do nothing".

Description

ECA rules of the input format are generated using QuickCheck, converted to abstract SQL, then compared against the expected output format using HUnit.

What is EFA? Too many undefined initialisms/acronyms. If they were defined in a previous document, add them to this one as well.

T3 EFA System Compatibility

Minor: Unnecessary '/'

Test type Functional/Black box/
Schedule Dec 2015
Requirements F3

Any reason for using these 'coded' references instead of just typing the text? They make it hard to follow the test cases. Consider hyperlinking to referenced section.

Input

ADL File Input 1:
Based on the Rule: Only members who have relevant experience may apply for this job
Using Sets: JOBS-AVAIL, APPLICANTS, EMPLOYEES-WITH-RELEVANT-EXPERIENCE
With ECA rules:
ADL Files Input 2:
Based on the Rule: Only members who have relevant experience may apply for this job
Using Sets: JOBS-AVAIL, APPLICANTS, EMPLOYEES-WITH-RELEVANT-EXPERIENCE
With ECA rules: APPLICANTS must be a member of both EMPLOYEES-WITH-RELEVANT-EXPERIENCE AND have a relation to (i.e. applied for) JOBS-AVAIL

Output

EFA User Output for Input 1:
Reading <file>.adl..
Generating..
Rules Done..
Sets Done..
No Errors
No Violations
EFA User Output for Input 2:
Reading <file>.adl..
Generating..
Rules Done..
Sets Done..
ECA Rules Done..
No Errors
No Violations

Description

Two different version of the same script is given as input, the first is without ECA rules the second is with ECA rules that this project adds. Both of these scripts should pass through the Ampersand generator without causing errors or violations. If the second script which contains ECA rules successfully passes through each part of Ampersand then the new additions generated by EFA is compatible with the old Ampersand system.

Formatting could be modified to fit this test on the page

T4 EFA Pure Function

Consider pushing this to the next page for readability.

Test type Dec 2015
Schedule F4
Requirements Two conditions must hold for a function to be considered a pure function 1. The fu

Input

The input is an ECA rule of the form:

ECA = {Condition that triggers action: Insertion of <new field into table>,
Change that initiated trigger: Insertion of <e2> into current data scheme,
Action to be done : ∀
{(take the difference of the previous result of the expressions
(take result of intersection of the returned result
(composition of the result
(Simple declaration of the result of the conversion
(convert expression e2 using the identity relatior
of e1
)
)
with e1)
with e1)
with e1 where e1 is another expression)}
}

Output

asdfasdf

Is the EPA Pure Function test output a TODO item? This would explain the placeholder.

T5 EFA User Feed Back

Following from above, fix the formatting so that all of your headings fall on one page if possible. With the above being pushed down this may be less of an issue.

Minor: Change "Feed Back" to "Feedback"

Test type Functional
Schedule January 2016
Requirements F6,N1

Brief Explanation Concerning Context

Only those who are qualified can be cast into roles, the actor must have relevant experience.

Input

The input shall be ECA rules specifying invariants that must be maintained throughout the program.

Example concerning how roles are cast for a theater performance:

User Input:

RULE "who's cast in roles" : cast; instantiates —- qualifies;comprises MEANING "an Actor may appear in a Performance of the Play only if the Actor is skilled for a Role that the Play comprises "

EFA INPUT:

ECA = {
ECA Trigger: if cast member does not have relevant experience or enough experience
ECA Violation: lack of experience
ECA Action: Remove actors without enough relevant experience
}

Output

User feed back if file has no errors:
Reading file theatreCasting.adl..
Done.
Done.

User feed back if file has errors:
Reading file theater.adl
Error(s) found:
Type error, cannot match:
the concept "Role" (Tgt of qualifies)
and concept "Performance" (Src of instantiates)
if you think there is no type error, add an order between concepts "Role" and "Performance".
Error at symbol () in file theater.adl at line 26 : 44
==========
No declarations match the relation: actor
Error at symbol () in file theater.adl at line 26 : 62
==========
ECA Rule Violation:
Error at Rule declaration and structure in file theater.adl at line 33 : 41
Error: Structure and Meaning do not match

T6 EFA Code Walk-through

Test type	Non-functional
Schedule	January 2016
Requirements	N2

Scheduling this for January seems slightly optimistic. Do you plan on having most of your code done by then such that the walk-through will be useful? Also again push this to the next page.

Brief Explanation

Input and output are not available for this test, as it requires each member of the design team to walk through the code line by line to check if it is easy to understand by another programmer and well documented. If it is easy to read and understand but not only the individual who wrote it but those around them, then it should be easy to maintain.

T7 Degradation Test

Test type Non-functional
Schedule First Week of February 2016
Requirements F6

Brief Explanation

Degradation shall be measured through a comparison of Ampersand before EFA and Ampersand after EFA. The amount of time Ampersand takes to compile a prototype will measure performance degradation; if Ampersand takes substantially longer to compile after the addition of EFA then it is an appreciable difference. A Linux distribution will be used to time multiple trials of Ampersand with and without EFA. The test materials used will be taken off of the Ampersand-models github.

T8 EFA Annotated Code

The test cases need to specify whether it is an automated or manual test

Test type Non-functional
Schedule January 2016
Requirements N4

Description

There is no input, however there will be annotations available for output, this is used for debugging purposes and the user will never see this.

Overall, good test plan. More info. about tests would make it better.

Are any of these tests to be used for the proof of concept testing? All the schedules seem to be for later dates.

Peer review & feedback provided by Team 5.
Jason Kim, Anmol Sharma, Can Birol, Marcus Koether

November 6, 2015