Module Guide for ECA Rules for Ampersand

Yuriy Toporovskyy, Yash Sapra, Jaeden Guo

February 13th, 2016

Table 1: Revision History

Author	**Date**	**Comment**
Yash Sapra	24 / 02 / 2016	Initial draft

Contents

1 Introduction

1.1 Description

The document outlines the design decision for the EFA project. EFA is responsible for generating SQL from ECA rules that will be used to fixed any data inconsistencies in the Ampersand Database.

This document follows the principle set by Parnas and Clements (D.L. Parnas, 1986) . Ampersand is currently in development where modifications are frequent, a commonly accepted practice for this situation is to decompose modules based on the principle of abstraction, where unnecessary information in hidden for the benefit of designers and maintainers(D.L. Parnas, 1984; Parnas, 1972).

Our design follows the principles layed out by (D.L. Parnas, 1984), as follows:

- Unnecessary design details are omitted for simplicity
- Each data structure is only in one module
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module. Additionally:
- Each module is broken down based on hierarchy
- Reference material are provided for external libraries but details of its use will not be provided within the module break-down

1.2 Scope

This project aims to improve upon the current Ampersand system by providing a permanent replacement for the exec-engine. EFA automatically restores system invariants according to ECA rules with no manual maintenance required.

1.2.1 Intended Audience

This document is designed for:

New project members: This document designed to be a guide to introduce new Ampersand users to EFA (ECA rules for Ampersand). It provides a basic structure that allows individuals to quickly access what they are looking for.

Maintainers & Designers: The structure of this module guide will help maintainers rationalize where changes should be made in order to accomplish their intended purpose. Furthermore, the design document will act as a guide to EFA for future designers of Ampersand.

2 Anticipated and Unlikely Changes

2.1 Anticipated Changes

It is likely that EFA will require changes to the front-end interface and an addition that integrates the front end to the back-end. Furthermore, the number of ECA rules are not permanent and if changes are made to the ECA rules, those changes will need to be incorporated into EFA.

Thus far anticipated changes include:

AC1: New front-end interface.

AC2: Addition or elimination of ECA rules.

AC3: The algorithm used for EFA.

AC4: The format of output.

AC5: The format of input parameters.

AC6: The format of initial input data and associated markers for data association.

AC7: Integration of front-end interface to back-end modules.

AC8: Implementation of SQL data structure

AC9: Testing for individual modules and internal systems

AC10: Software requirements for running Ampersand and by extension EFA

2.2 Unlikely Changes

These unlikely changes include the things that will remain unchanged in the system, and also changes that would not affect EFA.

UC1: There will always be a source of input data external to the software.

UC2: Results will always be provably correct.

UC3: The goal of EFA is to automatically correct system invariants

UC4: Output data must exist

UC5: The implementation language must be the same as that which is used for building the Ampersand system

UC6: The format of initial input data and associated markers for data association.

UC7: Type of output data will always be SQL.

3 Use Hierarchy & Dependency Graph

This section provides an overview of the module design. Modules are decomposed based on their hierarchy from top to bottom. The modules are broken down into two sections, the first section consists of the main modules used for EFA, the second section contains support module and finally the last sections contain external libraries.

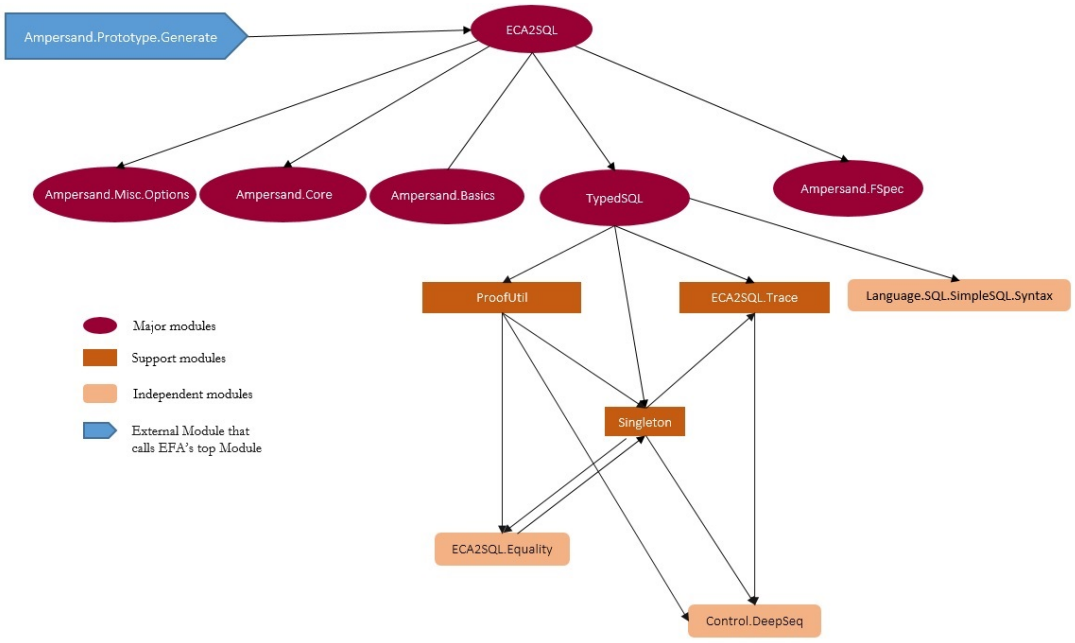

Figure 1: Dependency graph of EFA modules

4 Main Modules

5 Support Modules

6 External Libaries

7 Connection Between Requirements and Design

8 Module Decomposition

9 Behaviour-Hiding Module

10 Software Decision Module

11 Traceability Matrix

12 Use Hierarchy Between Modules

References

D.M. Weiss D.L. Parnas, P.C. Clements. The modular structure of complex systems. *Proceeding ICSE '84: Proceedings of the 7th international conference on Software engineering*, 1984.

P.C. Clements D.L. Parnas. A rational design process: How and why to fake it. *IEEE Transaction on Software Engineering*, 1986.

D.L. Parnas. On the criteria to ne used in decomposing systems into modules. *Communications of the ACM*, 1972.