VSTTE'13

Team Name Goes Here

April 19, 2013

Chapter 1

Introduction

1.0.1 Detailed Description

TBC

Chapter 2

Process Overview

2.1 REVEAL™ Overview

Software Engineering is concerned with the controlled and correct development of Systems (Machines) that impact upon the environment in which they are deployed (World). Where the objective of the deployment of a Machine is to improve the World.

The purpose of Requirements Engineering is to take a set of Stakeholder Requirements and arrive at a complete, correct and usable set of System Specification Statements for the system under development. Within REVEAL™ the application of the process is also intended to have a wider objective of adding benefits to the project Stakeholders through learning, negotiation and the development of trust between all parties involved in the production of the system.

REVEAL™ is Altrans' systematic method for the elicitation, specification and management of System Requirements. Through its application we clearly identify three key artefacts:

- Requirements (R) which are statements about things in the World that we want the System to make true.
- Specification Statements (S) describe the Systems external behaviour.
  - Specifications include only shared phenomena in the interface between the World and the System being developed.
  - Specifications can only constrain shared phenomena that the System can control.
- Domain Statements (D) which are knowledge about the World, that is generally assumed and not normally documented, but must hold if the Specification Statements are to fully satisfy the System Requirements.

Once this information has been gathered, a Satisfaction Argument is constructed to show that all the Stakeholder Requirements have been addressed through the

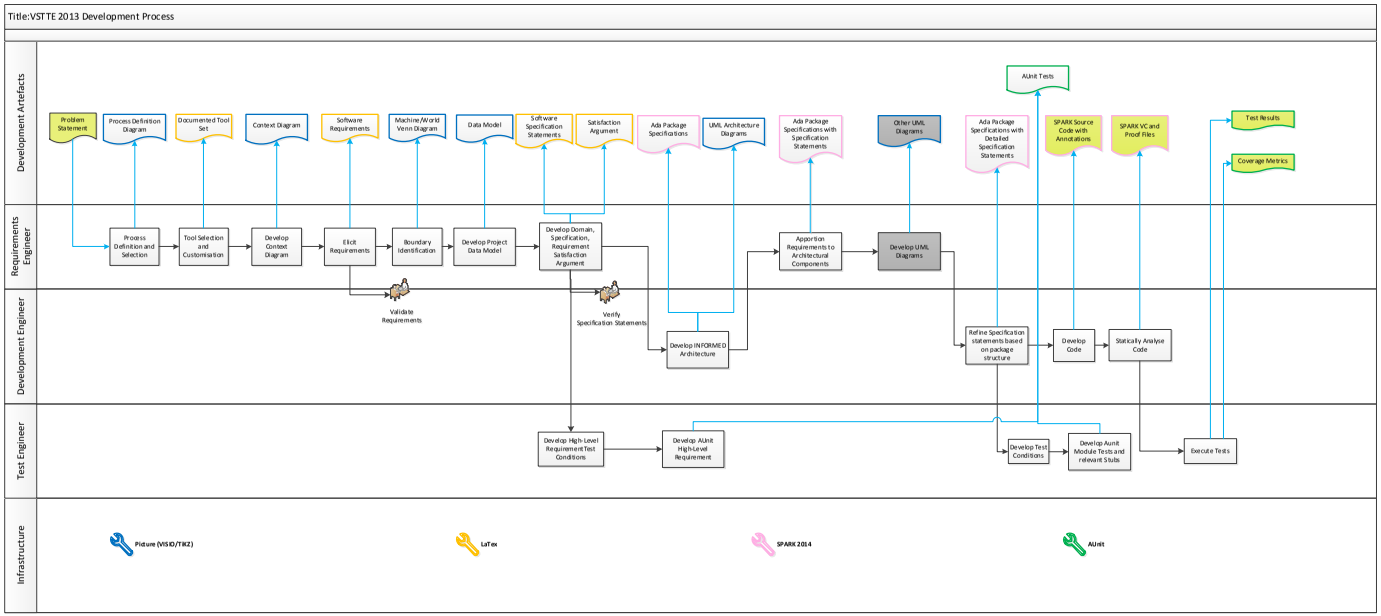

Figure 2.1: Our process

System Specification and Domain Statements. (As Verification and Validation progresses, V&V Plans and summary results can also inform the satisfaction argument, although this may not be done for the competition.)

The REVEAL^{TM} process is far more extensive than that described above including detailed processes for:

- Stakeholder Identification,
- Knowledge Elicitation,
- Requirement Verification and Validation,
- Conflict Management and Resolution,
- Requirements Maintenance and Management.

More information on the REVEAL^{TM} method can be found on:

http://intelligent-systems.altran.com/technologies/systems-engineering/revealtm.html

Since by its nature the VSTTE 2013 competition has a very short timescale, and the Requirements are provided by the competition organisers it is not expected that there would be time to apply a full REVEAL^{TM} process, however the benefits of Requirements capture, expression, satisfaction arguments and traceability can be shown by their application in this context.

2.2 INFORMED Overview

INFORMED is the Altran method for the construction of high-quality software at a low-cost, it uses elements of both object oriented (OOD) and functional design. OOD is used to establish the architecture of the system and the elements of system state it contains. The result is an annotated framework of SPARK packages, constructing an architecture that is modifiable and maintainable, which is capable of being analysed at an early stage using the Examiner.

Altrans' analysis of many software designs has identified that there are 3 key building blocks that are used repeatedly to build an application. These building blocks are:

- A Main Program, that is the top level, entry point controlling the behaviour a system or sub-system.
- Variable Packages, which are SPARK packages containing persistent "state", and are equivalent to what is commonly called an object.
- Type Packages, which define the name of a type, and the operations taht it supports.

In addition to these packages, INFORMED also includes two other types of packages:

- Utility Packages, which provide shared functionality. The need for utility packages arises when an operation is required which affects or uses more than one variable of type package.
- Boundary Variable Packages, which are particular kinds of variable package which provides interfaces between the software functionality described by the INFORMED design and elements outside it with which it must communicate. Unlike other variable packages the variable within the package is a place holder representing the stream of data arriving from, or being sent to, the outside world rather than simply an abstract name for the internal state of the package.

The INFORMED method consists of 6 steps, and these are:

1. Identification of the System boundary, inputs and outputs.
2. Identification of the SPARK boundary.
3. Identification and localization of the system state.
4. Handling initalization of state.
5. Handling secondary requirements.
6. Implementing the internal behaviour of components.

The output of the INFORMED process is represented a collection of design diagrams and a collection of well defined SPARK specifications. A summary of the INFORMED graphical elements and example representations are documented in Annex A of this document to aid the reader in the understanding of the architectural design.

As the VSTTE 2013 competition has a very short timescale, it is not expected that there is enough time to fully apply the INFORMED process, however the benefits of the benifits of the modelling approach and the translation in SPARK Package Specifications can be shown in this project.

2.3 SPARK2012 Overview

TBC

Chapter 3

Team Members

Alan Newton is a Senior Software Engineer working for Altran in Bath. He has a PhD in Software Engineering, and has experience throughout the lifecycle. Not only has he experience in developing high-integrity software using Formal Methods and SPARK, he has also developed Novel tools and processes for the effective deployment of software engineering activities.

In this project Alan has been involved in developing the processes, applying the REVEAL^{TM} and INFORMED processes and developing documentation to support the delivery of the project.

Yannick Moy is a senior engineer at AdaCore, working on static analysis and formal verification tools for Ada and SPARK programs. He previously worked on similar tools for C/C++ programs at PolySpace, INRIA research labs, and Microsoft Research. Moy received a PhD in formal program verification from Universit Paris-Sud.

In this project, Yannick has been involved in developing formal specifications, implementing code, and verifying (formally and by testing) that the code matches the formal specifications.

Chapter 4

Acknowledgements

As can be seen from the Team section, the members of this team have been drawn from AdaCore, Altran and Mitsubishi, and we would like to thank our employers for there support in our taking part in this competition.

We would especially like to thank them for allowing us to explore and use their tools and techniques to build a well-defined process for application to this problem.