Envisioning Group Policy in a heterogeneous Linux environment

David O Neill 0813001
Project Management in Practice
dave@feeditout.com

November 3, 2011

Abstract

Group policy a term more commonly associated with the Windows Active Directory domain is defined as a set of rules that govern an environments user and computer accounts. Group policy provides a means of centralizing the management and therein the control of the configuration; of client operating systems and their features.

Centralized management in a Linux environment is seen as a far more difficult problem. The environment given the multitude of different distributions, all subscribing to their own implementation philosophies have brought about almost infinite diversity, requiring companies to employ highly skilled technicians to manage these ever changing environments. These distributions or "flavors" seen to be highly different, however employ the same characteristics in their fundamental implementation only differing in the tools provided to control these characteristics.

Since the conception of X.500 Directory Services specification eventually leading to Microsofts dominance in the enterprise management, there has been a continual drive to integrate mixed environments into Active Directory. However due to the mixed philosophies and therein the different distributions of Open Source Software (OSS) operating systems, creating policies within active directory that can manage this diversity is seen as next to impossible or at least limited.

Unlike Microsoft clients that implement a common integrating architecture this cannot be said for Open Source Software (OSS) systems. Enterprise solutions targeting systems such as Redhat Enterprise Linux and Suse Enterprise Linux aid production administrators in provisioning and maintainability but not subscribe to non - production environments where a greater degree of flexibility is expected within the environment or do they employ user friendly common language as seen in the windows domain.

Observations within the field have led to the rapid development of Samba 4 offering directory services allowing for Windows Policies to be integrated into the internal directory structure but as of yet still no viable solution for Linux polices on the horizon.

The scope of this document is to provide insight into the issues surrounding this IT domain, to provide an analysis of the two main problems.

1) The demand for technicians to be knowledgeable in all the varying flavors of Gnus not Unix (GNU) systems and the abatement of this issue via common language, a domain specific language (DSL).

2) How these varying systems and their configurations can be represented by means of a directory services schema. Like that of its Windows Group Policy Object (GPO) counterpart, the configuration, changeability and extensibility of these policies through the use of the schema, should be applicable to all these systems.

Table Of Contents

1 Overview

1.1 Overview of the Final Year Project

Throughout this over view I will ease the reader into the domain concepts by providing the relationships with the Windows world. However as the overview progresses less comparisons will be made. As really we are mixing our apples and oranges.

Group policy in the windows world provides administrators with an easy interface to control aspects of computer policies in an easy defined manner. Any computer joined to the domain is subject to these policies. Administrators can, with the aid of visual snap-ins for Active directory, modify key-value pairs, which represent all the different aspects of a windows computer. Furthermore with the interoperability or backwards compatibility built in, each successive release of windows conforms to the standards or provides a translation pair relevant that that specific version of windows.

Before the advent of the Windows Registry configurations for programs where kept in INI files, broken up into sections and properties.

[section]
property=value

As the complexity of vendor applications and the operating system as a whole grew, so did the size of these INI files. Furthermore for interoperability and the sharing of dynamic link libraries, which depended on these INI configurations, it was quickly realized that this was an inefficient manner of storing configurations. The Windows Registry solved this issue by centralizing configurations settings into a hierarchical database containing settings for low-level operating system settings as well as settings for applications running on the platform.

Now that Windows has a central place for settings on the local machine, this provided an interface for a server (domain controller) to apply settings to groups of machines also known as Group Policy.

Given this brief overview of Windows Group Policy, lets take a look at the "Gnus not Unix" (GNU) systems. Since to conception of Linux in 1992 and the accompanied Gnus not Unix (GNU) applications, file configuration settings still remains the de facto way of configuring these systems and their application preferences. The style of these configuration files is somewhat similar to that of INI files in that there is key/value pairs; differing in how comments are written.

As Linux grow in popularity in the business sector for backend main frames due to its stability and security, a need was required for a centralized management of all these machines and a common login infrastructure to allow users and administrators to have credentials common to the network as a whole. Yellow Pages (YP) also known as network infrastructure services (NIS) offered this client server distributed system of authenticating users on a network.

Configuration data compiling of user and group information along with hosts on the domain name system (DNS) domain allow for this seamless user interaction between computers, but did not do much in allowing for administrators to manage these machines in the central location.

At this point I think it's important to look at the word "domain" as it will be used in contextually and comparatively, extensively throughout this document.

"The Domain Name System (DNS) is a hierarchical, distributed database that contains mappings of DNS domain names to various types of data, such as Internet Protocol (IP) addresses. DNS allows you to use friendly names, such as www.microsoft.com, to easily locate computers and other resources on a TCP/IP-based network. DNS is an Internet Engineering Task Force (IETF) standard."

"A Windows is a collection of computers in a networked environment that share a common database, directory database, or tree. A domain is administered as a unit with common rules and procedures, which can include security policies, and each domain has a unique name."

"A NIS (Linux) domain is similar to the Windows NT domain system; although the internal implementation of the two are not at all similar,

the basic functionality can be compared."

"A domain as a field of scope or activity comprised of a specific knowledge set."

With reference to the definition of a windows domain it is important as a constitute part of this report to acknowledge the concept of a domain as a group of computers. Although this may create ambiguity and defer from the scope of the application, it is however prominent to the concept and to that of the business terminology. The term DNS domain or domain name will be used as a reference to the identification label that defines an address, more commonly associated with the web in the form of uniform resource locator (URL). And of course the term domain, by itself a reference a set of specific knowledge.

Moving on from these definitions lets take a look at the problem domain. Each individual major version of Linux distribution provides utilities specific to controlling that machine and the settings therein. These utilities in some cases modify file configurations as previously discussed, For example, software provisioning is provided via "yum" on Redhat systems and "apt" on Debian systems. Furthermore Redhat provides the tools "chkconfig" and "service" in controlling boot up configuration and instant control respectively, while Debian provides similar tools. As we start to compare the major distributions we start to see the contrasting yet similar disparate natures of the utilities provided by the vendors.

Even though the underlying well-worn technologies that provide the backend implementation are primarily the same. The tools provided to control and implement changes on these disjoint systems although comparatively different from a usage perspective provide the same functionality. This brings about the need for extremely skilled well-versed technicians and of course creates more work from an administrative point of view.

Given the success of group policy in the windows domain this seems like a logical candidate in tackling these systems as a whole. By providing a framework to manage these contrasting systems through the use of a domain specific language; theoretically, an abstraction layer of the problem domain could be modeled mitigating administrators to be learned in a plethora of distribution specific commands. Martin Fowler makes this argument from a contrasting point of view in terms of Language Oriented programming, by replacing a few general purpose programming languages with many domain specific languages, he hypothesis that the requirement to learn numerous application programmers interfaces can be more of a burden than learning a domain specific language catered for an individual task.

Given this proposition for an abstraction layer to provide domain specific commands to a heterogeneous Linux environment and the need for controlling these systems from a group policy perspective, a domain specific language seems an obvious candidate in delivering a solution. We will look further at this concept in the following sub sections of chapter one where we define the scope, the objectives and terminology associated with this motive.

Now that we have an overview of what the problem domain is and a vague idea of how it can be tackled, lets contrive the other components. Firstly the Client Server Model envisioned by the idea of a central authority, The domain specific language as the intermediate language to provide instructions to the client, the schema for the specification of the database; where polices will be created and stored and finally the administrative server front end which will be an over view of the application and how it should enable administrators to create an modify policies for the underlying components.

2 Project Scope

2.1 Scope Disclaimer

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2.2 Scope Statement

The intent of this project is to provide a Linux port of the Windows Group Policy Object (GPO) editor and a means to deploy these policies to a multitude of Gnus not Unix (GNU) systems through a common non-ambiguous domain specific language (DSL).

2.3 Objectives

2.3.1 Client server model

Centralized management being the theme of this project establishes the inception of the concept of a central authority and therein a client server model. The implementation is REQUIRED to provide a client-server model that MUST exhibit the following.

1. The implementation MUST support at least two variants of linux to prove applicability (Redhat and Suse).
2. The design MUST support the ability to be extended to support a multitude of different systems with varying revisions of supporting packages.
3. The implementation SHOULD use the Practical Extraction and Reporting Language (PERL) for highest availability possible.
4. The server is REQUIRED to scale from two test clients to hundreds of clients.
5. This SHOULD be achieved via the Master Slave Design Pattern or similar pattern.
6. A master with multiple slave servers implementation is RECOMMENDED.
7. A compiled language MUST NOT be used as updates to the client SHALL be sent via the server which will deployed on varying architectures.
8. The Server SHOULD support both Push and Pull; at least MUST implement support for pull requests.

2.3.2 Domain specific language

The domain specific language SHOULD exhibit the following characteristics.

1. The domain specific language MUST be extensible.
2. As new modules SHALL be deployed to the client, these Modules MAY provides hooks to the interpreter which it MUST accept.
3. The Domain Specific Language(DSL) SHOULD be a implemented as a hybrid language.
    - The interpreter SHALL parse the Domain Specific Language(DSL) and process it.
    - The interpreter SHALL execute embedded general purpose language instructions defined in the Client server model objective.

2.3.3 Directory services schema

The directory services schema MUST provide a means of representing and storing the following elements.

1. Files, Permissions, Lvalue and Rvalue of elements within configuration files.
2. Services and with common names, which MUST be translated by the client into the distribution specific names.
3. A representation Organizational groups in hierarchy fashion compared to active directory MUST be implemented.
4. Computer objects SHALL be recognized by a globally unique identifier (GUID) represented as a 32-character hexadecimal string.

2.3.4 Administrator front end

The Administrator Group Policy editor SHOULD exhibit the following characteristics.

1. MUST provide the ability of creating organizational units.
2. MUST provide the ability to move computers between organizational units.
3. MUST Implement a hierarchy of organizational units where by policies MAY be inherited and overrided where applicable.
4. MUST provide a means to modify policies.
5. On application or change of a policy the server MUST generate new domain specific language script.
6. The front end MUST provide a means of importing existing policies in the form of the DSL.
7. The User Interface SHOULD implement the ability to push updates to the client.

3 Work Breakdown Structure

#	Major Milestones	#	Minor Milestones	#	Constituents	Description
1	Project Initiation	1	Project conception			Formalize the idea and define rough idea of scope.
		2	Project acceptance			Describe the project t potential supervisors and get one to sign off on it.
		3	Project submission			Sign documents and submit them to the fyp office.
2	Evaluation Phase	1	Develop risk plan	1	Identify risks	A Identify the risks in the project description.
				2	Analyze risks	Use these risks to develop scope and intended work.
				3	Document risks	Document these risks for the report.
		2	Plan for quality	1	Define quality requirements	Define the areas where non functional requirements should be catered for.
				2	Define procedures for quality	Design timelines for peer review, walkthroughs and inspections.
				3	Document quality management plan	Document these meetings for the quality plan.
		3	Define Scope	1	Requirements capture	I identify the needs and conditions for project success.
				2	Develop strategy plan	I identify the miles stones and how the project should process to meet these expectations.
				3	Research previous experience	Research previous projects in the area and their outcomes.
				4	Define scope	Define scope to better realize the expectations of the product.
				5	High level work breakdown	Break down the project into constituent parts.
				6	Deliverables and acceptance criteria	Identify milestones and required fidelity prototyping.
				7	Document assumptions	Document any assumptions made during the design process.
3	Design Phase	1	Design back end	1	Client framework	Design the client framework that handles the interpretation of the DSL and the distribution specific actions.
				2	Client	Design the Client to connect the the server for updates.
				3	Server	Design the Server to handle client connections.
				4	DSL specification	Design concepts of the DSL specification and document structure.
		2	Design front end	1	Policy editor	Design the visual policy editor.
4	Development Phase	1	Back end	1	Client framework	Implement interpreter and class representations.
				2	Client	Implementation.
				3	Server	Implementation.
				4	DSL specification	Implement DSL specification.
		2	Front end	1	Policy editor	Implement the visual policy editor.
5	Testing phase	1	Assure quality	1	Participate in walk-throughs	Walk through code with supervisor.
				2	Conduct inspection and audits	Conduct finite analysis of code .
				3	Conduct project reviews	Conduct project review and timelines goals and progress.
				4	Document tests	Define test cases and results.
6	Deployment phase	1	Installation			Conduct installation test.
		3	Demo			Conduct high level presentation.

4 Project Schedule

4.1 Activities

#	Info	Title	Given Planned Work	Expected Start	% Complete	Flag Status
0		**Linux Group Policy**		05/09/2011	23%	
1		**Project Initiation**		05/09/2011	100%	
2		Project Conception	5 days ?	05/09/2011	100%	
3		Project Acceptance	4 days ?	09/09/2011	100%	
4		Project Submission	1 day ?	15/09/2011	100%	
5		**Evaluation Phase**		19/09/2011	100%	
6		**Develop Risk Plans**		26/09/2011	100%	
7		Identify risks	1 day	26/09/2011	100%	
8		Analyze risks	1 day	27/09/2011	100%	
9		Document risks	1 day	28/09/2011	100%	
10		**Plan for Quality**		26/09/2011	100%	
11		Define quality requirements	1 day	26/09/2011	100%	
12		Set up standards and procedures for quality management	1 day	27/09/2011	100%	
13		Document quality management plan	1 day	28/09/2011	100%	
14		**Define Scope**		19/09/2011	100%	
15		Requirements capture	8.5 days ?	19/09/2011	100%	
16		Develop strategies and plans	1 day ?	19/09/2011	100%	
17		Conduct Planning Workshop	1 day ?	20/09/2011	100%	
18		Research previous experience	1 day ?	21/09/2011	100%	
19		Define scope	1 day ?	22/09/2011	100%	
20		Develop high level work breakdown	1 day ?	23/09/2011	100%	
21		Specify deliverables and acceptance criteria	1 day ?	26/09/2011	100%	
22		Document assumptions	1 day ?	27/09/2011	100%	
23		End of evaluation		28/09/2011	100%	
24		**Design Phase**		03/10/2011	46%	
25		**Design Back end**		03/10/2011	60%	
26		Client Framework	12.5 days ?	03/10/2011	100%	
27		Client Design	12.5 days ?	10/10/2011	100%	
28		Server Design	12.5 days ?	19/10/2011	50%	
29		DSL Specification	15 days ?	26/10/2011	0%	
30		**Design Front End**		02/01/2012	0%	
31		Policy Editor	15 days ?	02/01/2012	0%	
32		Specification complete		22/01/2012	0%	
33		Project Presentation		14/10/2011	50%	
34		**Development Phase**		17/10/2011	4%	
35		**Back End**		17/10/2011	7%	
36		Client Framework	20 days ?	17/10/2011	20%	
37		Client	20 days ?	31/10/2011	0%	
38		Server	20 days ?	14/11/2011	0%	
39		**Front End**		16/01/2012	0%	
40		Policy Editor	30 days ?	16/01/2012	0%	
41		End of Development		01/03/2012	0%	
42		**Testing Phase**		16/11/2011	0%	
43		**Assure Quality**		16/11/2011	0%	
44		Participate in walk-throughs and reviews	55 days	16/11/2011	0%	
45		Conduct inspections and audits	12 days	16/01/2012	0%	
46		Conduct project reviews	14 days	01/02/2012	0%	
47		Documentation	13.5 days ?	15/02/2012	0%	
48		**Deployment Phase**		27/02/2012	0%	
49		Installation	2.5 days ?	27/02/2012	0%	
50		Launch finished	3 days ?	29/02/2012	0%	
51		Demo	1 day ?	10/04/2012	0%	

Fig. 4.1: Activity schedule

4.2 Gannt chart

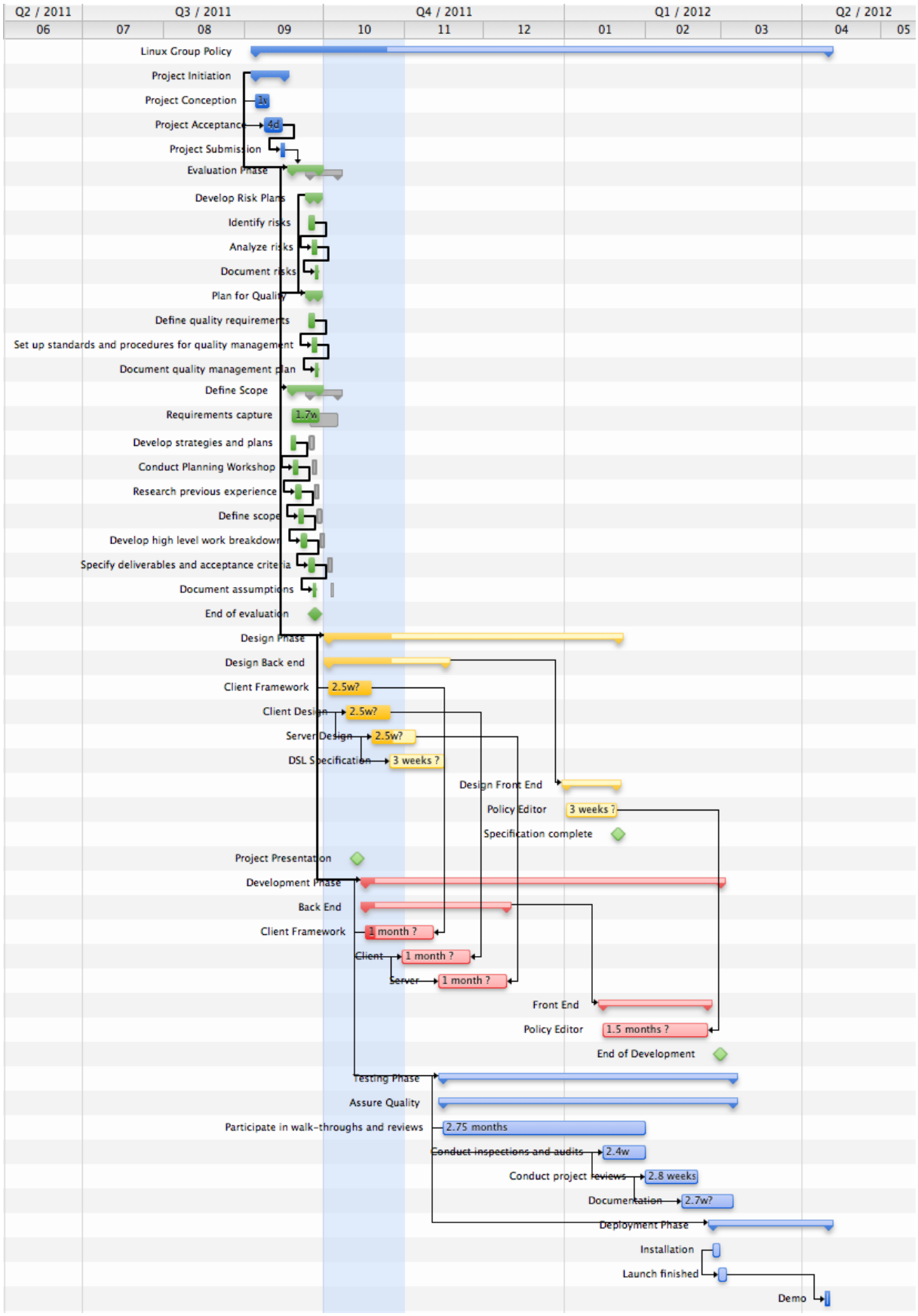

Fig. 4.2: Gannt chart

5 Communications Plan

5.1 Introduction

This communications plan presents the information pertinent to Student and the relevant parties during the process of the final year project. It outlines roughly how each party is responsible for administrating and delivering operational programmes.

5.2 Aims

The aim of this communication plan is to present an overview of the series of events or programmes as part of the final year project project management plan. This overview will highlight the roles and contributions of the parties involved and the timelines for deliveries therein.

5.3 Target audience

The targeted audience or stakeholders for the communications plan include the following

- The final year project coordinator
- Supervisor
- Second reader
- Student
- Project manager reviewer

5.4 The communications plan

5.4.1 Fyp Coordinator

Activity	Stage	Finish Date	Medium	Responsibility	Audience	Frequency
Fyp Guidelines	Understand	Week 12 Semester 1	Meeting	Coordinator	Students	weekly
Fyp Guidelines	Understand	Week 8 Semester 2	Meeting	Coordinator	Students	weekly

5.4.2 Project Supervisor Semester 1

Activity	Stage	Finish Date	Medium	Responsibility	Audience	Frequency
Meet students	Envision	Week 1	Meeting	Students	Supervisor	once
Accept students	Acceptance	Week 2	Document	Students	Supervisor	once
Supervise Students	Development	Week 12	Meeting	Students	Supervisor	weekly
Agreement Form	Submission	Week 10 Thursday	Document	Student	Supervisor	once
Marking Scheme	Acceptance	Week 10 Friday	Document	Students	Supervisor	once
Interim report	Acceptance	Week 16 Tuesday	Document	Students	Supervisor	once

5.4.3 Project Supervisor Semester 2

Activity	Stage	Finish Date	Medium	Responsibility	Audience	Frequency
Supervise students	Envision	Week 10	Meeting	Supervisor	Students	once
Draft Report	Acceptance	Week 7 Thursday	Document	Students	Supervisor	once
Product Acceptance	Acceptance	Easter break Monday	Email	Students	Supervisor	once
Report Submission	Acceptance	Week 11 Thursday	Document	Students	Supervisor	once
Cut-off Submission	Acceptance	Week 12 Thursday	Email / Document	Students	Supervisor	once

5.4.4 Student Semester 1

Activity	Stage	Finish Date	Medium	Responsibility	Audience	Frequency
Project conception	Envision	Week 1	Meeting	Student	Potential Supervisor	once
Project acceptance	Confirm	Week 2	Meeting	Student	Supervisor	once
Project submission	Submission	Week 2 Thursday	Document	Student	Fyp office	once
Develop risk plan	Define	Week 3	Meeting	Student	Supervisor	once
Plan for quality	Evaluate	Week 4	Meeting	Student	Supervisor	once
Design analysis 1	Design	Week 5	Meetings	Student	Supervisor	weekly
Presentation	Presentation	Week 6 Friday	Meeting	Student	Supervisor Panel	once
Agreement Form	Submission	Week 10 Thursday	Document	Student	Supervisor	once
Marking Scheme	Submission	Week 10 Friday	Document	Student	Supervisor	once
Development analysis 1	Develop	Week 12	Meeting	Student	Supervisor	weekly
Interim report	Submission	Week 16 Tuesday	Document	Student	Supervisor	once

5.4.5 Student Semester 2

Activity	Stage	Finish Date	Medium	Responsibility	Audience	Frequency
Design analysis 2	Design	Week 2	Meetings	Student	Supervisor	weekly
Development analysis 2	Develop	Week 6	Meeting	Student	Supervisor	weekly
Draft Report	Submission	Week 7 Thursday	Document	Student	Supervisor	once
Demo	Presentation	Week 10 Wednesday	Presentation	Student	Public presentation	once
Product Submission	Submission	Easter break Monday	Email	Student	Supervisor	once
Report Submission	Submission	Week 11 Thursday	Document	Student	Supervisor	once
Cut-off Submission	Submission	Week 12 Thursday	Email / Document	Student	Supervisor	once

5.5 Evaluation and change

As the deadline approaches this communications plan is subject to change. Communications between the supervisor and the student may be subject to change in order to ensure a quality product delivery on time.

6 Risk Management Plan

6.1 Introduction

As the product being delivered is highly conceptual, a new and unique product in the problem domain, chance is a significant element leading to risk of uncertainty of delivery. The purpose of this risk management plan is to identify risks and develop strategies to help mitigate these risks.

6.2 Risk Monitoring

During the ongoing meetings with the supervisor the main risks prevalent should be readdressed, discussed and the status of each risk readjusted. It is the responsibility of the risk manager(s) to provide these risk status updates, the trigger conditions and risk response.

6.3 Risk mitigation and avoidance

As more risk become apparent during the development these should be addressed immediately and discussed with the supervisor. These risks should be qualified by the student and avoidance and mitigation strategies should be developed with the supervisor.

6.4 Qualitative and quantitive

Risk Likelihood Rating

Low	**Medium**	**High**	**Extreme**	**Not Assessed**
L	M	H	E	NA

Risk Impact Rating

Grade	**Mitigation action**	**Description**
N	No Action	No action required unless grading increases over time
A	Alternative action	Have alternative action plan
P	Produce Minimum	Mitigation plan to provide minimum accepted
I	Implemented Execution	Identified and avoided during execution
C	Catastrophic	Identified and avoided on commencement as a priority

Combined impact / Likelihood

	Impact				
Likelihood		**Low**	**Medium**	**High**	**Extreme**
	Low	N	N	A	C
	Medium	A	P	I	C
	High	I	I	C	C
	Extreme	C	C	C	C

Grade change assessment

Identifier	**Description**
new	New Risk
—	No change
↓	Decreased risk
↑	Increased risk

6.5 Register

id	Description	Project Impact	Likelihood	Impact	Combined	Grade change	Review Date	Mitigation actions	Responsibility	Timeline
1	Supervisor unavailable	Poor planning and delivery	L	H	A	↓	Weekly	Identify next 2 milestones and required work	Student	Weekly
2	Work schedule interruptions by other academic deadlines	Delayed product delivery grade penalized	M	H	C	↑	Weekly	Plan in advance Don't schedule work during weeks 7-12 of semester 1 & 2	Student	Weekly
3	Failure to define requirements	Poor scope definition unattainable goals	L	M	N	↓	21/1/2012	No action required at this time	Supervisor Student	3/1/2012
4	Failure to define good scope Unattainable goals and expectations	Product that does not meet the requirements, quality expected	M	H	I	↓	21/1/2012	Hammer out scope with supervisor and plan for second reader expectations	Supervisor Student	1/2/2012
5	Poor design	Software that does not exhibit quality attributes	M	H	A	—	Ongoing	Plan for change in grading scheme rebalance report / code awarding marks	Student Supervisor	2/2/2012
6	Poor implementation	Software that does not realize the design	L	L	N	New	Ongoing	Refactor design as part of the iterative development	Student	1/3/2012
7	Technology cacophony Too many technologies to learn	Danger of spending too much time on one area	M	M	P	New	Ongoing	Produce the minimum requirements for demo day focus on delivery of written report expectations quantify attainable scope	Student Supervisor	1/3/2012
8	Presentation Poor explanation of project	Failure to show merit of project to supervising panel, Loss of 10% of overall marks	L	M	N	↓	Week 6	Properly define scope, applicability and intent	Student	completed
9	Agreement form submission	Failure to submit form on time could result in project failure	L	E	C	↑	Week 10	Make reminder and schedule agreement form signage with supervisor	Student	Week 10
10	Marking scheme	Marking agreement to be submitted	L	L	N	↑	Week 10	Make reminder and schedule agreement form signage with supervisor	Student	Week 10
11	Interim report	Failure to submit interim report on time will result in 10% loss	L	H	C	↑	Week 13	Make reminder and have pre interim report review meeting	Student	Week 14
12	Draft report	Failure to submit interim report on time will result in 10% loss	L	H	C	new	Week 6(2)	Make reminder and have pew draft report review meeting	Student	Week 7(2)
13	Demo day Product Launch Presentation	Result of loss of marks, if not properly presented	L	L	N	new	Week 9(2)	Have meeting with supervisor to showcase product	Student	Week 10(2)
14	Product submission	Penalty for late submissions	L	L	N	new	Week 8(2)		Student	Week 9(2)
15	Report Submission	Penalty for late submissions	L	L	N	new	Week 10(2)		Student	Week 11(2)

7 Quality Plan

7.1 Introduction

The project must comply with all of the requirements described in project requirements specification. The projects conformity to software requirements specification will be checked throughout the lifecycle as defined in the rational unified process. Upon the supervisors verification that all of the tests have been satisfactorily completed / passed, the project is considered to be of satisfactory quality. Meaning that the project complies with all requirements and is accepted by the supervisor.

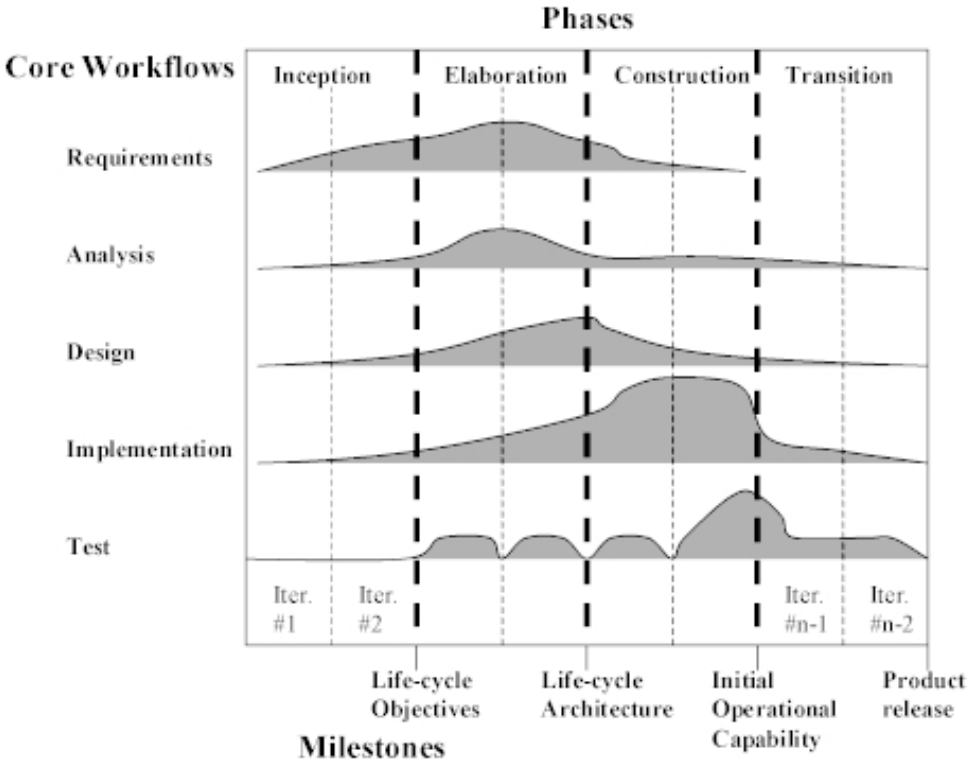

Fig. 7.1: Rational Unified Process

7.2 Review process

The communications plan outlines the meeting times when software reviews are scheduled to take place. The software reviews are in place to detect any defects in the current fidelity prototypes during its construction. The determination of new enhancements, or regressions should be documented and the risk register updated accordingly. This should be completed for each of the 4 individual components as outlines in the objectives.

The supervisor in conjunction with the student will attend each review, and critique software prototypes to ensure that the maximum number of possible defects are accounted for, moreover identify risks in the current direction.

This review process is subject to the work completed, accounting for disruptions in the work scheduled due to other academic commitments. The risk and schedule log should to addressed and adjusted accordingly based upon work interruptions. Each meeting will consist of no more than a one hours. The current prototype should be provided in advance of this meeting. Any enhancement found to be too difficult or unnecessary for product completion will be noted by the student.

7.3 Quality Compliancy plan

Deliverable	Quality Event	Quality Materials	Quality Metrics	Purpose	Presenters	Validators	Resolution
Preliminary Project Conception	High level Analysis	Project template Conformance with fyp research expectations	Area of research shows academic value	Compliance with academic expectations prior to submission.	Student	Supervisor	Re-scope project
Project Description Finalization	Accept project	Academic value	Timelines and expectations are realistic	Project submittal	Student	Supervisor	
Requirements	Business modeling Inception	Support for non functional requirements	Each requirement tracked back to project goal	Support in Definition scope	Student	Supervisor	Redefine requirements
Scope	Scope definition Inception	Properly defined scope	Clearly state whats in / out	identify clear objectives	Student	Supervisor	Redefine scope requirements
Objectives	Objectives inception	Broken down objectives that realize the scope	Objective components realize scope	Define work breakdown structure	Student	Supervisor	Redefine Objective scope
Work breakdown structure	Process inception	Each objective broken down	Clear milestones and finite process	Define Milestones and schedule	Student	Supervisor	Redefine Objective scope
Schedule	Schedule delivery inception	Timelines or other chart (gannt)	Milestones WBS Dates	Project progress analysis	Student	Supervisor	Redefine structure milestones
Design	Analysis and design Elaboration	UMl Diagrams	Class / Sequence diagrams	Realize requirements	Student	Supervisor	
Test Cases	Testing Construction	Test harnesses to test business rules	Code passes tests	Meet business rules	Student	Supervisor	Redefine design
Change Request	Business rule Conformance	Change request	Clear requirement change	Conformance with expectations	Supervisor	Student	Redefine scope requirements design
Implementation	Milestones Construction	Code	Code passes test cases	Meet design requirements	Student	Student	Fix bugs
Demo	Deployment	Deployment test harness	Product realizes requirements	Show proof of concept	Student	Supervisor	

Bibliography

[1] Project management docs.

List of Figures