ISSN: 1945-3116 Vol. 3, No. 9, September 2010

Scientific Research

Special Issue for the 2nd International Workshop for Requirements Analysis (IWRA2010)

Journal of Software Engineering and Applications

ISSN: 1945-3116

www.scirp.org/journal/jsea

Announcement:

Special Issue of Journal of Software Engineering and Applications

It is our great pleasure to announce that all papers published in this issue are recommended by the Organization Committee of the 2nd International Workshop on Requirements Analysis (IWRA) 2010-held at Middlesex University, London, UK.

We hope this special issue can attract more scholars to submit their research papers to JSEA, the journal that publishes the good quality research and review articles in all important aspects of software engineering and applications.

JSEA Editorial Office

From the Special Issue Editors

Requirements Analysis: Where Theory Meets Practice

More than 70% of the IT projects in the UK fail every year and over 80% of them fail in the requirements analysis phase. Poor project management, bad organisational politics, false business priorities, lack of commitment and other issues can also cause a project to fail. Patterns of software systems failures and their study have resulted in the development of numerous models, methodologies and frameworks, yet project and system failures persist. Requirements understanding, analysis and specification have emerged as critical areas in software and systems engineering. More recent developments have indicated the shift from the technical view to the human factor embodied strongly in the agile movement. The papers were presented at the International Workshop for Requirements Analysis (IWRA2010) which focussed on concepts, ontologies, models, methods and techniques such as methods for eliciting, analysing and measuring requirements. The papers in this issue come from both academia and industry where theory is applied to case studies such as Service Provisioning for the Greek Banking Sector, home care systems and Aeronautics.

Tom Gilb distills 40 years' contribution to Software Engineering research and practice (including Software metrics, the EVO methodlogy and Planguage) in his keynote address provides an analysis of the fundamental failings of conventional thinking about software requirements, and proposed suggestions 'for getting it right'.

Professor Stamper known for his pioneering work in Organisational semiotics, and the creation of the MEASUR methodology shares his reflections on Ontologies.

Professor White provides a Review of the Impact of Requirements on Software Project Development using a Control Theoretic Model and scientists from China and Finland explore Model-Driven Derivation of Domain Functional Requirements from Use Cases. As Software Engineering is these days preoccupied with a paradigm shift to Agility Methods, agility could not be absent from such a workshop.

The papers in this issue have been contributed by academics and practitioners from several countries and domains. They explore current practices for process modelling and process improvement, examine the applicability of theoretical models and frameworks to practical problems, analyse the complexity in systems and organisations, address current issues through the use of formal, semi-formal and informal methods and above all promote the integration of theory and practice, identify trends and indicate avenues of further work.

Dedication: The IWRA210 and this special issue of JSEA are dedicated to the memory of Dr Manos Nistazakis who was a co-chair and the main driving force for organising and hosting the workshop at Middlesex University. He sadly died of natural causes suddenly aged 39 in May.

J. Software Engineering & Applications, 2010, 3, 827-900
Published Online September 2010 in SciRes (http://www.SciRP.org/journal/jsea/)

TABLE OF CONTENTS

Volume 3 Number 9 September 2010

What's Wrong with Requirements Specification? An Analysis of the Fundamental Failings of Conventional Thinking about Software Requirements, and Some Suggestions for Getting it Right
T. Gilb……827

Benefits Management Process Complements Other Project Management Methodologies
I. Karamitsos, C. Apostolopoulos, M. A. Bugami……839

Requirements Analysis and Traceability at CIM Level
M. Yamin, V. Zuna, M. A. Bugami……845

A Review of the Impact of Requirements on Software Project Development Using a Control Theoretic Model
A. White……852

IT Project Environment Factors Affecting Requirements Analysis in Service Provisioning for the Greek Banking Sector
K. Maroukian……858

Requirements Analysis: Evaluating KAOS Models
F. Almisned, J. Keppens……869

Model-Driven Derivation of Domain Functional Requirements from Use Cases
J. M. Guo, Z. Y. Zhang, Y. L. Wang……875

Towards Lightweight Requirements Documentation
Z. Y. Zhang, M. Arvela, E. Berki, M. Muhonen, J. Nummenmaa, T. Poranen……882

Investigating the Suitability of Agile Methods for Requirements Development of Home Care Systems
S. Kelly, F. Keenan……890

Introduction to a Requirements Engineering Framework for Aeronautics
R. Abo……894

Journal of Software Engineering and Applications (JSEA)

Journal Information

SUBSCRIPTIONS

The *Journal of Software Engineering and Applications* (Online at Scientific Research Publishing, www.SciRP.org) is published monthly by Scientific Research Publishing, Inc., USA.

Subscription rates:

Print: $50 per issue.
To subscribe, please contact Journals Subscriptions Department, E-mail: sub@scirp.org

SERVICES

Advertisements
Advertisement Sales Department, E-mail: service@scirp.org

Reprints (minimum quantity 100 copies)
Reprints Co-ordinator, Scientific Research Publishing, Inc., USA.
E-mail: sub@scirp.org

COPYRIGHT

PRODUCTION INFORMATION

For manuscripts that have been accepted for publication, please contact:
E-mail: jsea@scirp.org

J. Software Engineering & Applications, **2010, 3, 827-838**
doi:10.4236/jsea.2010.39096 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

What's Wrong with Requirements Specification? An Analysis of the Fundamental Failings of Conventional Thinking about Software Requirements, and Some Suggestions for Getting it Right

Tom Gilb

Result Planning Limited, Norway and UK.
Email: Tom@Gilb.com

ABSTRACT

We know many of our IT projects fail and disappoint. The poor state of requirements methods and practice is frequently stated as a factor for IT project failure. In this paper, I discuss what I believe is the fundamental cause: we think like programmers, not engineers and managers. We do not concentrate on value delivery, but instead on functions, on use-cases and on code delivery. Further, management is not taking its responsibility to make things better. In this paper, ten practical key principles are proposed, which aim to improve the quality of requirements specification.

Keywords: *Requirements, Value Delivery, Requirements Definition, Requirements Methods*

1. Introduction

We know many of our IT projects fail and disappoint. We know bad 'requirements', that is requirements that are ambiguous or are not really needed, are often a factor. However in my opinion, the real problem is one that almost no one has openly discussed or dealt with. Certainly, it fails to be addressed by many widely known and widely taught methods. So what is this problem? In a nutshell: it is that we think like programmers, and not as engineers and managers. In other words, we do not concentrate on value delivery, but instead on functions, on use cases and on code delivery. And no one is attempting to prevent this: IT project management and senior management are not taking their responsibility to make things better.

2. Ten Key Principles for Successful Requirements

In this paper, my ten key principles for improving the approach to requirements are outlined. These principles are not new, and they could be said to be simply commonsense. However, many IT projects still continue to fail to grasp their significance, and so it is worth restating them. These key principles are summarized in **Figure 1**. Let's now examine these principles in more detail and provide some examples.

Note, unless otherwise specified, further details on all aspects of Planguage can be found in [1].

2.1. Understand the Top Level Critical Objectives

I see the 'worst requirement sin of all' in almost *all* projects we look at, and this applies internationally. Time and again, the high-level requirements (the ones that funded the project), are vaguely stated, and ignored by the project team. Such requirements frequently look like the example given in **Figure 2**.

The requirements in **Figure 2** have been slightly edited to retain anonymity. They are for a real project that ran for eight years and cost over 100 million US dollars. The project failed to deliver any of these requirements. However, the main problem is that these are not top-level requirements: they fail to explain in sufficient detail what the business is trying to achieve. There are additional problems as well that I'll discuss further later in this paper (such as lack of quantification, mixing optional designs into the requirements, and insufficient background

Ten Key Principles for Successful Requirements

1. Understand the top level critical objectives
2. Look towards value delivery: systems thinking, not just software
3. Define a 'requirement' as a 'stakeholder-valued end state'
4. Think stakeholders: not just users and customers!
5. Quantify requirements as a basis for software engineering
6. Don't mix ends and means
7. Focus on the required system quality, not just its functionality
8. Ensure there is 'rich specification': requirement specifications need far more information than the requirement itself!
9. Carry out specification quality control (SQC)
10. Recognize that requirements change: use feedback and update requirements as necessary

Figure 1. Ten key principles for successful requirements.

Example of Initial Top Level Objectives

1. Central to the corporation's business strategy is to be the world's premier integrated <domain> service provider
2. Will provide a much more efficient user experience
3. Dramatically scale back the time frequently needed after the last data is acquired to time align, depth correct, splice, merge, recomputed and/or do whatever else is needed to generate the desired products
4. Make the system much easier to understand and use than has been the case with the previous system
5. A primary goal is to provide a much more productive system development environment then was previously the case
6. Will provide a richer set of functionality for supporting next generation logging tools and applications
7. Robustness is an essential system requirement
8. Major improvements in data quality over current practices

Figure 2. Example of initial top level objectives.

description).

Management at the CEO, CTO and CIO level did not take the trouble to clarify these critical objectives. In fact, the CIO told me that the CEO actively rejected the idea of clarification! So management lost control of the project at the very beginning.

Further, none of the technical 'experts' reacted to the situation. They happily spent $100 million on all the many suggested architecture solutions that were mixed in with the objectives.

It actually took less than an hour to rewrite one of these objectives so that it was clear, measurable, and quantified. So in one day's work the project could have clarified the objectives, and avoided 8 years of wasted time and effort.

1) The top ten critical requirements for any project can be put on a single page.

2) A good first draft of the top ten critical requirements for any project can be made in a day's work, assuming access to key management.

2.2. Look towards Value Delivery: Systems Thinking, not Just a Focus on Software

The whole point of a project is delivering realized value, also known as benefits, to the stakeholders: it is not the defined functionality, and not the user stories that count. Value can be defined as the benefit we think we get from something [1]. See **Figure 3**. Notice the subtle distinction between initially perceived value ('I think that would be useful'), and realized value: effective and factual value ('this was in practice more valuable than we thought it would be, because …').

The issue is that conventional requirements thinking is that it is not closely enough coupled with 'value'. IT business analysts frequently fail to gather the information supporting a more precise understanding and/or the calculation of value. Moreover, the business people when stating their requirements frequently fail to justify them

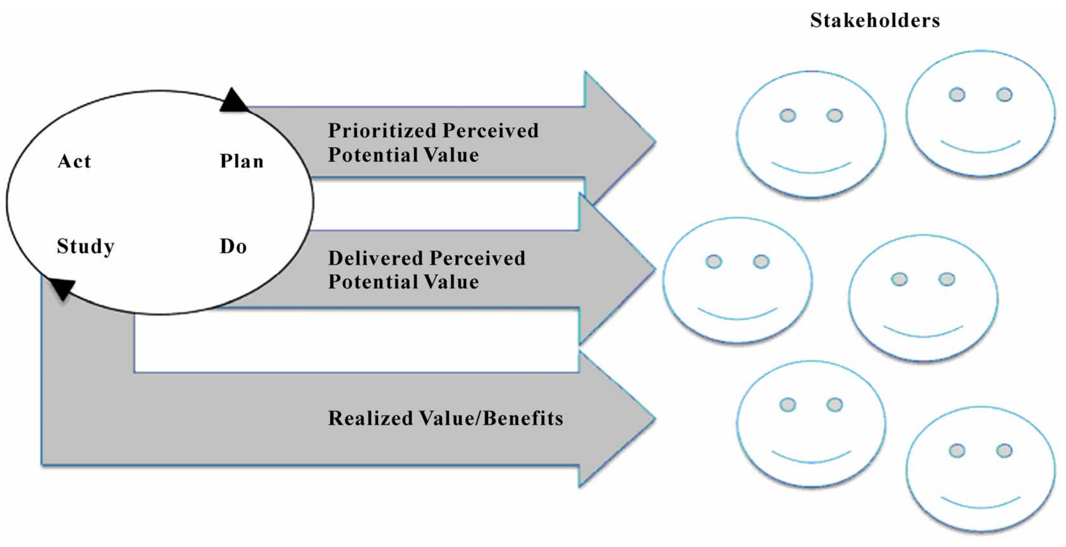

Stakeholders
Act
Plan
Study
Do
Prioritized Perceived Potential Value
Delivered Perceived Potential Value
Realized Value/Benefits

Figure 3. Value can be delivered gradually to stakeholders. Different stakeholders will perceive different value.

using value.

The danger if requirements are not closely tied to value is that:

1) We risk failure to deliver the value expected, even if 'requirements' are satisfied

2) We risk having a failure to think about all the things to do that are necessary prerequisites to actually delivering *full value* to real *stakeholders* on time: we need *systems* thinking – not just programming.

How can we articulate and document notions of value in a requirement specification? See the Planguage example for Intuitiveness, a component quality of Usability, in **Figure 4**.

For brevity, a detailed explanation is unable to be given here. Hopefully, the Planguage specification is reasonably understandable without detailed explanation. For example, the Goal statement (80%) specifies which market (USA) and users (Seniors) it is intended for, which set of tasks are valued (the 'Photo Tasks Set'), and when it would be valuable to get it delivered (2012). This 'qualifier' information in all the statements, helps document where, who, what, and when the quality level applies. The additional Value parameter specifies the perceived value of achieving 100% of the requirement. Of course, more could be said about value and its specification, this is merely a 'wake-up call' that explicit value needs to be captured within requirements. It is better than the more common specifications of the Usability requirement that we often see, such as: "2.4. The product will be more user-friendly, using Windows".

So who is going to make these value statements in requirements specifications? I don't expect developers to care much about value statements in requirements. Their job is to deliver the requirement levels that someone else has determined are valued. Deciding what sets of requirements are valuable is a Product Owner (Scrum) or Marketing Management function. Certainly only the IT-related value should be determined by the IT staff.

2.3. Define a 'Requirement' as a 'Stakeholder-Valued End State'

Do we all have a shared notion of what a 'requirement' is? I am afraid that another of our problems. Everybody has an opinion, and most of the opinions about the meaning of the concept 'requirement' are at variance with most other opinions. I believe that few of the popular definitions are correct or useful. Below I provide you with my latest 'opinion' about the best definition of 'requirement', but note it is a 'work in progress' and possibly not my final definition. Perhaps some of you can help improve this definition even further.

To emphasize 'the point' of IT systems engineering, I have decided to define a requirement as a "stakeholder-valued end state". You possibly will not accept, or use this definition yet, but this is the definition that I shall use in this paper, and I will argue the case for it. In addition, I have also identified, and defined a large number of requirement concepts [1]. A sample of these concepts is given in **Figure 5**.

Further, note that I make a distinction amongst:

1) A requirement (a stakeholder-valued end state)

2) A requirement specification

3) An implemented requirement

4) A design in partial, or full service, of implementing a requirement.

Usability. Intuitiveness:
Type: Marketing Product Requirement.
Stakeholders: {Marketing Director, Support Manager, Training Center}.
Impacts: {Product Sales, Support Costs, Training Effort, Documentation Design}.
Supports: Corporate Quality Policy 2.3.
Ambition: Any potential user, any age, can immediately discover and correctly use all functions of the product, without training, help from friends, or external documentation.
Scale: % chance that a defined [User] can successfully complete the defined [Tasks] <immediately>, with no external help.
Meter: Consumer Reports tests all tasks for all defined user types, and gives public report.
-- Analysis --
Trend [Market = Asia, User = {Teenager, Early Adopters}, Product = Main Competitor, Projection = 2013]**:** 95% ± 3% < - Market Analysis.
Past [Market = USA, User = Seniors, Product = Old Version, Task = Photo Tasks Set, When = 2010]**:** 70% ± 10% < - Our Labs Measures.
Record [Market = Finland, User = {Android Mobile Phone, Teenagers}, Task = Phone + SMS Task Set, Record Set = January 2010]**:** 98% ± 1% < - Secret Report.
-- Our Product Plans --
Goal [Market = USA, User = Seniors, Product = New Version, Task = Photo Tasks Set, When = 2012]**:** 80% ± 10% < - Draft Marketing Plan.
Value [Market =USA, User = Seniors, Product = New Version, Task = Photo Tasks Set, Time Period = 2012]**:** 2 M USD.
Tolerable [Market = Asia, User = {Teenager, Early Adopters}, Product = Our New Version, Deadline = 2013]**:** 97% ± 3% < - Marketing Director Speech.
Fail [Market = Finland, User = {Android Mobile Phone, Teenagers}, Task = Phone + SMS Task Set, Product Release 9.0]**:** Less Than 95%.
Value [Market = Finland, User = {Android Mobile Phone, Teenagers}, Task = Phone + SMS Task Set, Time Period = 2013]**:** 30K USD.

Figure 4. A practical made-up Planguage example, designed to display ways of making the value of a requirement clear.

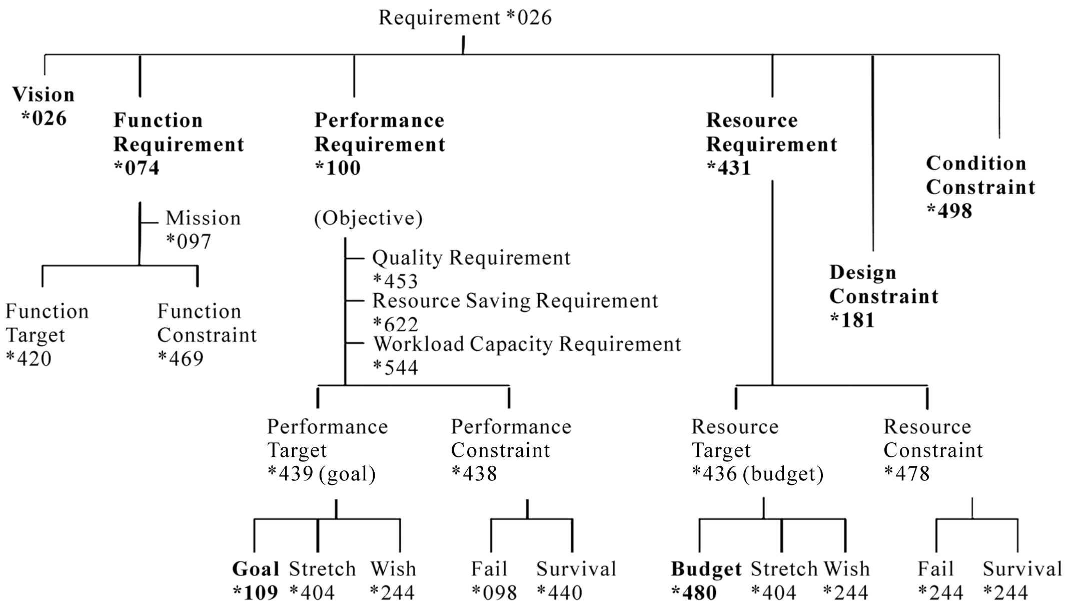


Figure 5. Example of Planguage requirements concepts.

These distinctions will be described in more detail later in this paper.

2.4. Think Stakeholders: Not Just Users and Customers!

Too many requirements specifications limit their scope to being too narrowly focused on user or customer needs. The broader area of stakeholder needs and values should be considered, where a 'stakeholder' is anyone or anything that has an interest in the system [1]. It is not just the end-users and customers that must be considered: IT development, IT maintenance, senior management, government, and other stakeholders matter as well.

2.5. Quantify Requirements as a Basis for Software Engineering

Some systems developers call themselves 'software engineers', they might even have a degree in the subject, or in 'computer science', but they do not seem to practice any real engineering as described by engineering professors, like Koen [2]. Instead these developers all too often produce requirements specifications consisting merely of words. No numbers, just nice sounding words; good enough to fool managers into spending millions for nothing (for example, "high usability").

Engineering is a practical bag of tricks. My dad was a real engineer (with over 100 patents to his name!), and I don't remember him using just words. He seemed forever to be working with slide rules and back-of-the-envelope calculations. Whatever he did, he could you tell why it was numerically superior to somebody else's product. He argued with numbers and measures.

My life changed professionally, when, in my twenties, I read the following words of Lord Kelvin: "In physical science the first essential step in the direction of learning any subject is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it. I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of *Science*, whatever the matter may be" [3]. Alternatively, more simply, also credited to Lord Kelvin: "If you can not measure it, you can not improve it".

The most frequent and critical reasons for software projects are to improve them *qualitatively* compared to their predecessors (which may or may not be automated logic). However, we seem to almost totally avoid the practice of quantifying these qualities, in order to make them clearly understood, and also to lay the basis for measuring and tracking our progress in improvement towards meeting our quality level requirements.

This art of quantification of any quality requirement should be taught as a fundamental to university students of software and management disciplines (as it is in other sciences and engineering). One problem seems to be that the teachers of software disciplines do not appreciate that quality has numeric dimensions and so cannot teach it. Note the problem is not that managers and software people cannot and do not quantify at all. They do. It is the lack of 'quantification of the qualitative'—the lack of numeric quality requirements—that is the specific problem.

Perhaps we need an agreed definition of 'quality' and 'qualitative' before we proceed, since the common interpretation is too narrow, and not well agreed. Most software developers when they say 'quality' are only thinking of bugs (logical defects) and little else. Managers speaking of the same software do not have a broader perspective. They speak and write often of qualities, but do not usually refer to the broader set of '-ilities' as qualities, unless pressed to do so. They may speak of improvements, even benefits instead.

I believe that the concept of 'quality' is simplest explained as 'how well something functions'. I prefer to specify that it is necessarily a 'scalar' attribute, since there are degrees of 'how well'. In addition to quality, there are other requirement-related concepts, such as workload capacity (how much performance), cost (how much resource), function (what we do), and design (how we might do function well, at a given cost) [1,4]. Some of these concepts are scalar and some, binary. See **Figures 6** and **7** for some examples of quality concepts and how quality can be related to the function, resources and design concepts.

My simple belief is that absolutely all qualities that we value in software (and associated systems) can be expressed quantitatively. I have yet to see an exception. Of course most of you do not know that, or believe it. One simple way to explore this is to search the internet. For example: "Intuitiveness scale measure" turns up 3 million hits, including this excellent study [5] by Yanga *et al*.

Several major corporations have top-level policy to quantify all quality requirements (sometimes suggested by me, sometimes just because they are good engineers). They include IBM, HP, Ericsson and Intel [1,4].

The key idea for quantification is to define, or reuse a definition, of a scale of measure. For example: (earlier given with more detail)

To give some explanation of the key quantification features in **Figure 8**:

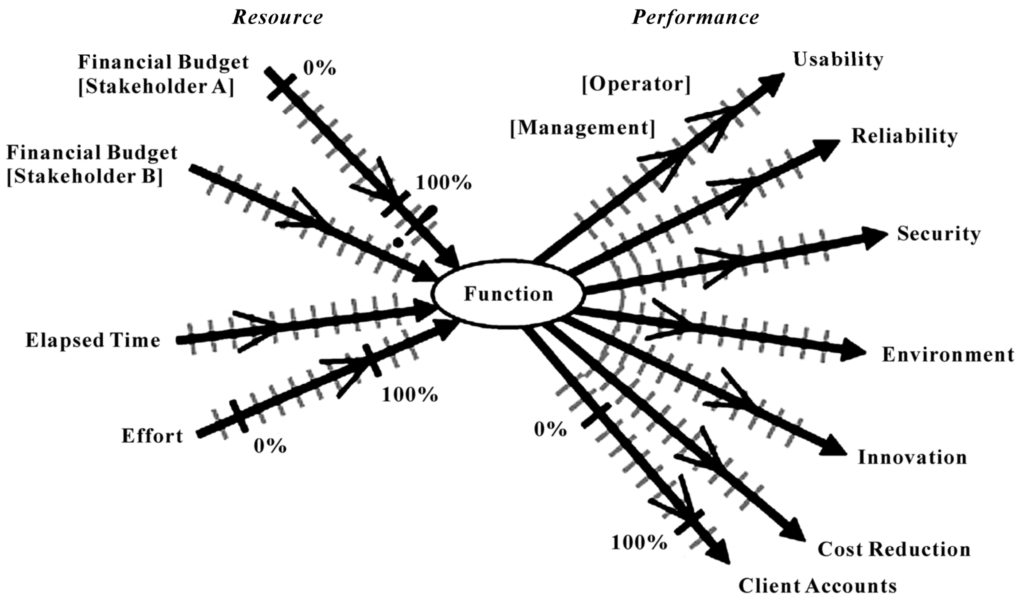


Figure 6. A way of visualizing qualities in relation to function and cost. Qualities and costs are scalar variables, so we can define scales of measure in order to discuss them numerically. The arrows on the scale arrows represent interesting points, such as the requirement levels. The requirement is not 'security' as such, but a defined, and testable degree of security [1].

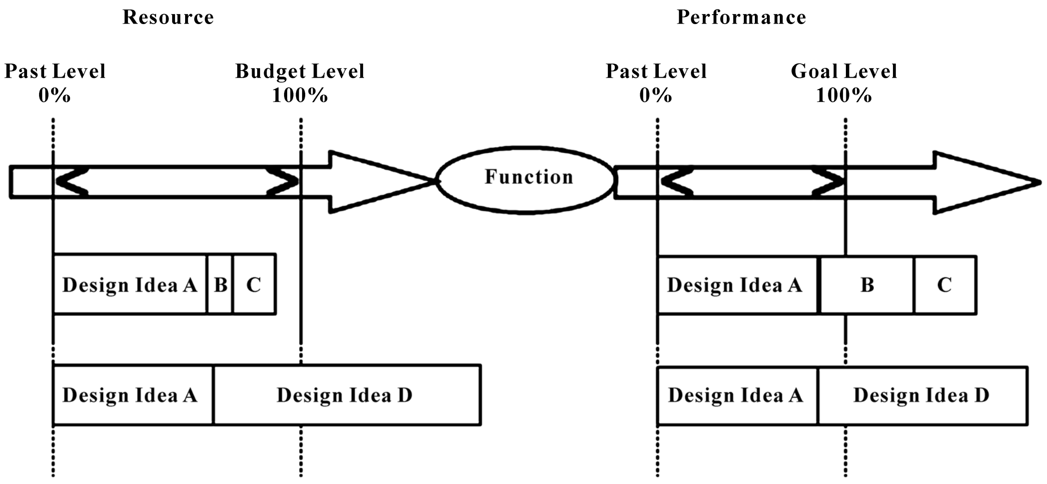


Figure 7. A graphical way of understanding performance attributes (which include all qualities) in relation to function, design and resources. Design ideas cost some resources, and design ideas deliver performance for given functions. Source [1].

1) Ambition is a high level summary of the requirement. One that is easy to agree to, and understand roughly. The Scale and Goal following it MUST correlate to this Ambition statement.

2) Scale is the formal definition of our chosen scale of measure. The parameters [User] and [Task] allow us to generalize here, while becoming more specific in detail below (see earlier example). They also encourage and permit the reuse of the Scale, as a sort of 'pattern'.

3) Meter is a defined measuring process. There can be more than one for different occasions. Notice the Kelvin quotation above, how he twice in the same sentence distinguishes carefully between numeric definition (Scale), and measurement process or instrument (Meter). Many people, I hope you are not one, think they are the same thing, for example: Km/hour is not a speedometer, and a volt is not a voltmeter.

4) Goal is one of many possible requirement levels (see earlier detail for some others; Fail, Tolerable, Stretch, Wish, are other requirement levels). We are de-

fining a stakeholder valued future state (state = 80% ± 10%).

One *stakeholder* is 'USA Seniors'. The *future* is 2012. The requirement level type, Goal is defined as a very high priority, budgeted promise of delivery. It is of higher priority than a Stretch or Wish level. Note other priorities may conflict and prevent this particular requirement from being delivered in practice.

If you know the *conventional* state of requirements methods, then you will now, from this example alone, begin to appreciate the difference that I am proposing. Especially for *quality* requirements. I know you can quantify time, costs, speed, response time, burn rate, and bug density—but there is *more*!

Here is another example of quantification. It is the initial stage of the rewrite of Robustness from the **Figure 2** example. First we determined that Robustness is complex and composed of many different attributes, such as Testability. See **Figure 9**.

And see **Figure 10**, which quantitatively defines one of the attributes of Robustness, Testability.

Note this example shows the notion of there being different levels of requirements. Principle 1 also has relevance here as it is concerned with top-level objectives (requirements). The different levels that can be identified include: corporate requirements, the top-level critical few project or product requirements, system requirements and software requirements. We need to clearly document the

Usability. Intuitiveness:
Type: Marketing Product Quality Requirement.
Ambition: Any potential user, any age, can immediately discover and correctly use all functions of the product, without training, help from friends, or external documentation.
Scale: % chance that defined [User] can successfully complete defined [Tasks] <immediately> with no external help.
Meter: Consumer reports tests all tasks for all defined user types, and gives public report.
Goal [Market = USA, User = Seniors, Product = New Version, Task = Photo Tasks Set, When = 2012]**:** 80% ± 10% < - Draft Marketing Plan.

Figure 8. A simple example of quantifying a quality requirement, 'Intuitiveness'.

Robustness:
Type: *Complex* Product Quality Requirement.
Includes: {Software Downtime, Restore Speed, Testability, Fault Prevention Capability, Fault Isolation Capability, Fault Analysis Capability, Hardware Debugging Capability}.

Figure 9. Definition of a complex quality requirement, Robustness.

Testability:
Type: Software Quality Requirement.
Version: Oct 20, 2006.
Status: Draft.
Stakeholder: {Operator, Tester}.
Ambition: Rapid duration automatic testing of <critical complex tests> with extreme operator setup and initiation.
Scale: The duration of a defined [Volume] of testing or a defined [Type of Testing] by a defined [Skill Level] of system operator under defined [Operating Conditions].
Goal [All Customer Use, Volume = 1,000,000 data items, Type of Testing = WireXXXX vs. DXX, Skill Level = First Time Novice, Operating Conditions = Field]**:** < 10 minutes.
Design: Tool simulators, reverse cracking tool, generation of simulated telemetry frames entirely in software, application specific sophistication for drilling – recorded mode simulation by playing back the dump file, application test harness console < –6.2.1 HFS.

Figure 10. Quantitative definition of testability, an attribute of Robustness.

level and the interactions amongst these requirements.

An additional notion is that of 'sets of requirements'. Any given stakeholder is likely to have a set of requirements rather than just an isolated single requirement. In fact, achieving value could depend on meeting an entire set of requirements.

2.6. Don't Mix Ends and Means

"*Perfection of means and confusion of ends seem to characterize our age*." *Albert Einstein*. 1879-1955.

The problem of confusing ends and means is clearly an old one, and deeply rooted. We specify a solution, design and/or architecture, instead of what we really value—our real requirement [6]. There are explanatory reasons for this—for example solutions are more concrete, and what we want (qualities) are more abstract for us (because we have not yet learned to make them measurable and concrete).

The problems occur when we do confuse them: if we do specify the means, and not our true ends. As the saying goes: "Be careful what you ask for, you might just get it" (unknown source). The problems include:

1) You might not get what you *really* want,

2) The solution you have specified might *cost too much* or have bad *side effects,* even if you do get what you want,

3) There may be much *better solutions* you don't know about yet.

So how to we find the 'right requirement', the 'real requirement' [6] that is being 'masked' by the solution? *Assume* that there probably is a better formulation, which is a more accurate expression of our real values and needs. Search for it by asking 'Why?' Why do I want X, it is because I really want Y, and assume I will get it through X. But, then why do I want Y? Because I really want Z and assume that is the best way to get X. Continue the process until it seems reasonable to stop. This is a slight variation on the '5 Whys' technique [7], which is normally used to identify root causes of problems (rather than high level objectives).

Assume that our stakeholders will *usually* state their values in terms of some perceived means to get what they really value. Help them to identify (The 5 Whys?) and to acknowledge what they really want, and make that the 'official' requirement. Don't insult them by telling them that they don't know what they want. But explain that you will help them more-certainly get what they more deeply want, with better and cheaper solutions, perhaps new technology, if they will go through the '5 Whys?' process with you. See **Figure 11**.

Note that this separation of designs from the requirements does not mean that you ignore the solutions/designs/architecture when software engineering. It is just that you must separate your requirements including any mandatory means, from any optional means.

2.7. Focus on the Required System Quality, Not Just its Functionality

Far too much attention is paid to what the system must do (function) and far too little attention is given to how well it should do it (qualities)—in spite of the fact that quality improvements tend to be the major drivers for new projects. See **Table 1**, which is from the Confirmit case study [8]. Here focusing on the quality requirements, rather than the functions, achieved a great deal!

2.8. Ensure there is 'Rich Specification': Requirement Specifications need Far More Information than the Requirement itself

Far too much emphasis is often placed on the requirement itself; and far too little concurrent information is gathered about its background, for example: who wants this requirement and why? What benefits do they perceive from this requirement? I think the requirement itself might be less than 10% of a complete requirement specification that includes the background information.

I believe that background specification is absolutely

Why do you require a 'password'? For Security!
What kind of security do you want? Against stolen information
What level of strength of security against stolen information are you willing to pay for? At least a 99% chance that hackers cannot break in within 1 hour of trying! Whatever that level costs up to €1 million.
So that is your real requirement? Yep.
Can we make that the official requirement, and leave the security design to both our security experts, and leave it to proof by measurement to decide what is really the right design? Of course!
The aim being that whatever technology we choose, it gets you the 99%?
Sure, thanks for helping me articulate that!

Figure 11. Example of the requirement, not the design feature, being the real requirement.

Table 1. Extract from confirmit case study [8].

Description of requirement/work task	Past	Status
Usability. Productivity: Time for the system to generate a survey	7200 sec	15 sec
Usability. Productivity: Time to set up a typical market research report	65 min	20 min
Usability. Productivity: Time to grant a set of end-users access to a report set and distribute report login information	80 min	5 min
Usability. Intuitiveness: The time in minutes it takes a medium-experienced programmer to define a complete and correct data transfer definition with Confirmit Web Services without any user documentation or any other aid	15 min	5 min
Performance. Runtime. Concurrency: Maximum number of simultaneous respondents executing a survey with a click rate of 20 sec and a response time < 500ms given a defined [Survey Complexity] and a defined [Server Configuration, Typical]	250 users	6000

mandatory: it should be a corporate standard to specify a great deal of this related information, and ensure it is intimately and immediately tied into the requirement specification itself.

Such background information is the part of a specification, which is *useful* related information, but is not *central* (*core)* to the implementation, and nor is it commentary. The central information includes: Scale, Meter, Goal, Definition and Constraint. Commentary is any detail that probably will not have any economic, quality or effort consequences if it is incorrect, for example, notes and comments.

Background specification includes: benchmarks {Past, Record, Trend}, Owner, Version, Stakeholders, Gist (brief description), Ambition, Impacts, and Supports. The rationale for background information is as follows:

1) To help judge value of the requirement
2) To help prioritize the requirement
3) To help understand risks with the requirement
4) To help present the requirement in more or less detail for various audiences and different purposes
5) To give us help when updating a requirement
6) To synchronize the relationships between different but related levels of the requirements
7) To assist in quality control of the requirements
8) To improve the clarity of the requirement.

See **Figure 12** for an example, which illustrates the help given by background information regarding risks.

Reliability:
Type: Performance Quality.
Owner: Quality Director. **Author:** John Engineer.
Stakeholders: {Users, Shops, Repair Centers}.
Scale: Mean Time Between Failure.
Goal [Users]: 20,000 hours < - Customer Survey, 2004.
    Rationale: Anything less would be uncompetitive.
    Assumption: Our main competitor does not improve more than 10%.
    Issues: New competitors might appear.
    Risks: The technology costs to reach this level might be excessive.
    Design Suggestion: Triple redundant software and database system.
Goal [Shops]: 30,000 hours < - Quality Director.
    Rationale: Customer contract specification.
    Assumption: This is technically possible today.
    Issues: The necessary technology might cause undesired schedule delays.
    Risks: The customer might merge with a competitor chain and leave us to foot the costs for the component parts that they might no longer require.
    Design Suggestion: Simplification and reuse of known components.

Figure 12. A requirement specification can be embellished with many background specifications that will help us to understand risks associated with one or more elements of the requirement specification [9].

Let me emphasize that I do not believe that this background information is sufficient if it is scattered around in different documents and meeting notes. I believe it needs to be directly integrated into a master sole reusable requirement specification object for each requirement. Otherwise it will not be available when it is needed, and will not be updated, or shown to be inconsistent with emerging improvements in the requirement specification. See **Figure 13** for a requirement template for function specification [1], which hints at the richness possible

TEMPLATE FOR FUNCTION SPECIFICATION <with hints>

Tag: <Tag name for the function>.
Type: <{Function Specification, Function (Target) Requirement, Function Constraint}>.

================================== **Basic Information** ==================================

Version: <Date or other version number>.
Status: <{Draft, SQC Exited, Approved, Rejected}>.
Quality Level: <Maximum remaining major defects/page, sample size, date>.
Owner: <Name the role/email/person responsible for changes and updates to this specification>.
Stakeholders: <Name any stakeholders with an interest in this specification>.
Gist: <Give a 5 to 20 word summary of the nature of this function>.
Description: <Give a detailed, unambiguous description of the function, or a tag reference to someplace where it is detailed. Remember to include definitions of any local terms>.

==================================== **Relationships** ====================================

Supra-functions: <List tag of function/mission, which this function is a part of. A hierarchy of tags, such as A.B.C, is even more illuminating. Note: an alternative way of expressing supra-function is to use Is Part Of>.
Sub-functions: <List the tags of any immediate sub-functions (that is, the next level down), of this function. Note: alternative ways of expressing sub-functions are Includes and Consists Of>.
Is Impacted By: <List the tags of any design ideas or Evo steps delivering, or capable of delivering, this function. The actual function is NOT modified by the design idea, but its presence in the system is, or can be, altered in some way. This is an Impact Estimation table relationship>.
Linked To: <List names or tags of any other system specifications, which this one is related to intimately, in addition to the above specified hierarchical function relations and IE-related links. Note: an alternative way is to express such a relationship is to use Supports or Is Supported By, as appropriate>.

==================================== **Measurement** ====================================

Test: <Refer to tags of any test plan or/and test cases, which deal with this function>.

=============================== **Priority and Risk Management** ==============================

Rationale: < Justify the existence of this function. Why is this function necessary? >.
Value: <Name [Stakeholder, time, place, event>]: <Quantify, or express in words, the value claimed as a result of delivering the requirement>.
Assumptions: <Specify, or refer to tags of any assumptions in connection with this function, which could cause problems if they were not true, or later became invalid>.
Dependencies: <Using text or tags, name anything, which is dependent on this function in any significant way, or which this function itself, is dependent on in any significant way>.
Risks: <List or refer to tags of anything, which could cause malfunction, delay, or negative impacts on plans, requirements and expected results>.
Priority: <Name, using tags, any system elements, which this function can clearly be done *after* or must clearly be done *before*. Give any relevant reasons>.
Issues: <State any known issues>.

==================================== **Specific Budgets** =================================

Financial Budget: <Refer to the allocated money for planning and implementation (which includes test) of this function>.

Figure 13. A template for function specification [1].

for background information.

2.9. Carry out Specification Quality Control (SQC)

There is far too little quality control of requirements, against relevant standards for requirements. All requirements specifications ought to pass their quality control checks before they are released for use by the next processes. Initial quality control of requirements specification, where there has been no previous use of specification quality control (SQC) (also known as Inspection), using three simple quality-checking rules ('unambiguous to readers', 'testable' and 'no optional designs present'), typically identifies 80 to 200+ words per 300 words of requirement text as ambiguous or unclear to intended readers [10]!

2.10. Recognise That Requirements Change: Use Feedback and Update Requirements as Necessary

Requirements must be developed based on on-going feedback from stakeholders, as to their real value. Stakeholders can give feedback about their perception of value, based on *realities*. The whole process is a 'Plan Do Study Act' cyclical learning process involving many complex factors, including factors from outside the system, such as politics, law, international differences, economics, and technology change.

The requirements must be *evolved* based on realistic experience. Attempts to fix them in advance, of this experience flow, are probably wasted energy: for example, if they are committed to—in contracts and fixed specifications.

3. Who or What will Change Things?

Everybody talks about requirements, but few people seem to be making progress to enhance the quality of their specifications and improve support for software engineering. I am pessimistic. Yes, there are internationally competitive businesses, like HP and Intel that have long since improved their practices because of their competitive nature and necessity. But they are very different from the majority of organizations building software. The vast majority of IT systems development teams we encounter are not highly motivated to learn or practice first class requirements (or anything else!). Neither the managers nor the developers seem strongly motivated to improve. The reason is that they get by with, and get well paid for, failed projects.

The universities certainly do not train IT/computer science students well in requirements, and the business schools also certainly do not train managers about such matters [11]. The fashion now seems to be to learn over-simplified methods, and/or methods prescribed by some certification or standardization body. Interest in learning provably more-effective methods is left to the enlightened and ambitions few—as usual. So, it is the only the elite few organizations and individuals who do in fact realize the competitive edge they get with better practices [8,12]. Maybe this is simply the way the world is: first class and real masters of the art are rare. Sloppy 'muddling through' is the norm. Failure is inevitable or perhaps, denied. Perhaps insurance companies and lawmakers might demand better practices, but I fear that even *that* would be corrupted in practice, if history is any guide (think of CMMI and the various organizations at Level 5).

Excuse my pessimism! I am sitting here writing with the BP Gulf Oil Leak Disaster in mind. The BP CEO Hayward just got his reward today of £11 million in pension rights for managing the oil spill and 11 deaths. In 2007, he said his main job was "to focus 'laser like' on safety and reliability" [13]. Now how would you define, measure and track those requirements?

Welcome if you want to be exceptional! I'd be happy to help!

4. Summary

Current typical requirements specification practice is woefully inadequate for today's critical and complex systems. There seems to be wide agreement about that. I have personally seen several real projects where the executives involved allowed over \$100 million to be wasted on software projects, rather than ever changing their corporate practices. \$100 million here and there, corporate money, is not big money to these guys!

We know what to do to improve requirements specification, if we want to, and some corporations have done so, some projects have done so, some developers have done so, some professors have done so: but when is the other 99.99% of requirements stakeholders going to wake up and specify requirements to a decent standard? If there are some executives, governments, professors and/or consultancies, who want to try to improve their project requirements, then I suggest start by seeing how your current requirements specifications measure up to addressing the ten key principles in this paper.

5. Acknowledgements

Thanks to Lindsey Brodie for editing this paper.

REFERENCES

[1] T. Gilb, "Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage," Elsevier Butterworth-Heinemann, Boston, 2005.

[2] B. V. Koen, "Discussion of the Method: Conducting the Engineer's Approach to Problem Solving," Oxford University Press, Oxford, 2003.

[3] L. Kelvin, "Electrical Units of Measurement," a Lecture Given on 3 May 1883, Published in the Book "Popular Lectures and Addresses, Volume 1," 1891.

[4] T. Gilb, "Principles of Software Engineering Management," Addison-Wesley, Boston, 1988.

[5] Z. Yanga, S. Caib, Z. Zhouc and N. Zhoua, "Development and Validation of an Instrument to Measure User Perceived Service Quality of Information Presenting Web Portals," *Information & Management*, Vol. 42, No. 4, 2005, pp. 575-589.

[6] T. Gilb, "Real Requirements". http://www.gilb.com/tiki-download_file.php?fileId =28

[7] T. Ohno, "Toyota Production System: Beyond Large-Scale Production," Productivity Press, New York, 1988.

[8] T. Johansen and T. Gilb, "From Waterfall to Evolutionary Development (Evo): How we Created Faster, More User-Friendly, More Productive Software Products for a Multi-National Market," *Proceedings of INCOSE*, Rochester, 2005. http://www.gilb.com/tiki-download_file.php?fileId=32

[9] T. Gilb, "Rich Requirement Specs: The Use of Planguage to Clarify Requirements," http://www.gilb.com/tiki-download_file.php?fileId=44

[10] T. Gilb, "Agile Specification Quality Control, Testing Experience," March 2009. www.testingexperience.com/testingexperience01_08.pdf

[11] K. Hopper and W. Hopper, "The Puritan Gift," I. B. Taurus and Co. Ltd., London, 2007.

[12] "Top Level Objectives: A Slide Collection of Case Studies". http://www.gilb.com/tiki-download_file.php?fileId=180

[13] "Profile: BP's Tony Hayward, BBC Website: News US and Canada," 27 July 2010. http://www.bbc.co.uk/news/world-us-canada-10754710

J. Software Engineering & Applications, 2010, 3, 839-844
doi:10.4236/jsea.2010.39097 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

Benefits Management Process Complements Other Project Management Methodologies

Ioannis Karamitsos[1], Charalampos Apostolopoulos[2], Moteb Al Bugami[3]

[1]ON Telecoms S. A, 21 Spirou Merkouri Str, Athens, Greece; [2]4 Nicomidias Str, Nea Smirni, Athens, Greece; [3]Department of Management Information Systems, King Abdulaziz University, Jeddah, Saudi Arabia.
Email: ch_apostolopoulos@yahoo.co.uk

ABSTRACT

The benefits management approach complements most of the common project management methodologies such as critical chain project management (CCPM), and PRINCE2. The majority of these methodologies focus on how to comply with three parameters: time, cost and quality instead of identifying the positive outcomes and benefits for an organization. In this paper, a different approach for the organization is presented, which focuses on positive outcomes named as benefits. Moreover, a comparison between Benefits Management and PRINCE2 methodologies is illustrated.

Keywords: *Benefits Management, PRINCE2, Requirements Analysis*

1. Introduction

Benefits Management is the definition, planning, structuring and actual realisation of the benefits of a business change or business improvement project. The benefits management approach is necessary for the business projects and programmes so as to deliver benefits, however, they are frequently criticised for failing to achieve their objectives. Standish Group, Chaos report [1] showed that around 70% of business improvement projects fail to deliver their expected benefits, and even when they are achieved in part, often they are far from fully realised. The reasons for this are varied, but significant elements can be directly related to, for example:

- Business cases focused on target savings instead of expressing business benefits in a manner that can be understood and implemented
- Too much emphasis on deliverables or outcomes (e.g. capabilities) which on their own do not deliver specific benefits
- No mechanisms or in particular structures to manage their realisation

However, Apostolopoulos and Karamitsos [2], explained project failure reasoning in terms of behavioural perspectives. More precisely, the lacking of efficient communication may be a result of different individual and environmental approaches of a project. In other words, the client's inputs are rooted from an operational environment whereas the project manager's ideas are rooted from the past experience in a project-based environment.

In effect, because of lacking of understanding, communication barriers exist, project managers do not understand what their clients really expect (lack of user input) and projects fail. In order to overcome these barriers, it is necessary both parties to enhance their dialectic relationships, commit to a certain goal planning and if necessary change the requirements and specifications so as to reach the desired outcome, which is project success.

In general for a project to be successful, is has to be delivered on time, within budget, and conform to the client's requirements; requirements analysis is mainly related to determine the needs and conditions to be met for a successful project, taking into account the possible risks and of course, understand customer's needs and expectations.

In literature, there exist many different methodologies which are related to analyzing the requirements of a project, which in effect become a tool for effective project management.

Proper requirements analysis drive almost every task and activity, however, the identifications of when, how and what has to be done should be a bidirectional activity among all the parties involved.

In benefits management approach, the benefits (project outcomes) are analysed prior to starting managing a project, whereas, in traditional methodologies, such as Agile (cyclic software development process, encourages leadership philosophy), DSDM (Dynamic System Development; software development methodology), PRINCE2 (structured approach), focus is given more on the successful completions of different tasks, which in effect will lead to beneficial outcomes; in CCPM methodology emphasis is given on the resources (physical and human) so as to execute project tasks.

Projects are often considered to be finished when their deliverables are complete. Nonetheless, the benefits of a project are typically realised over time; this may leave no one responsible during the realisation phase and often no structure through which to manage this important element.

For benefits realisation to work, it is crucial to identify clear benefits (early in the lifecycle) that are related to unambiguous business objectives, and to assign ownership to those "responsible" for planning and managing their achievement.

A central goal of this process is to bring structure, accountability, clarity and discipline to the definition and delivery of the benefits inherent in business projects. It is therefore a key aspect of programme management and relates to other business processes, such as portfolio management and must start in the earliest stages of the change/business improvement cycle.

While investment appraisal may provide the justification for the proposition in a business case, effective realisation planning enables organisations to understand and maximise the potential benefits that can be modelled using such techniques. It must also identify and address the changes that will be required, including any resistance that may be encountered. These changes themselves may well need to be managed carefully as part of a change management programme.

The most obvious thing to say is that experience demonstrates that organisations do not find this task easy, as businesses are not abundant in skills or track record in its execution (in a formal way).

But, what are the things that typically have to improve most?

- understanding what constitutes a specific benefit (versus general outcomes or target savings for example) in any specific business and differentiating them from objectives, outcomes, and their end financial (or other) results
- the way benefits are expressed and structured in business cases and their alignment with strategic business objectives in particular, (this is fundamental to success)
- the whole planning and management of this activeity or process

2. The Benefits Management Framework

The framework must be driven by the organisation's strategic planning and portfolio management processes. To be effective, it needs to become a standard management practice throughout the business change lifecycle, especially during programme and project definition.

The first step is to establish a framework that defines how benefits should be identified, structured, planned and realised. (See **Figure 1**)

The framework should classify types of benefits of value to businesses, and reference the organisation's current strategic goals and objectives, for example:

- service/process/quality/productivity/improvements
- cost avoidance/reduction
- staff morale/motivation
- revenue generation/customer retention

The potential benefits identified must not simply exist as a list. It is important to identify dependencies to understand where the achievement of one benefit is dependent on the realisation of another.

Once they have been identified, analysed and structured, the next task is to create a realisation plan. This should also enable the organisation to identify the management actions required to support and execute that plan.

2.1. Benefits Focused Business Cases

A business case should set out the basis of an investment or change. Business cases must demonstrate the return or value that the owning organisation will achieve by the proposition in the business case. Business cases must demonstrate how the value or return will be delivered, by identifying specific benefits that will be accrued via making the investment/change. This is often very different from making summary statements about planned or targeted financial savings that will be achieved.

Many business cases in the past went no further than

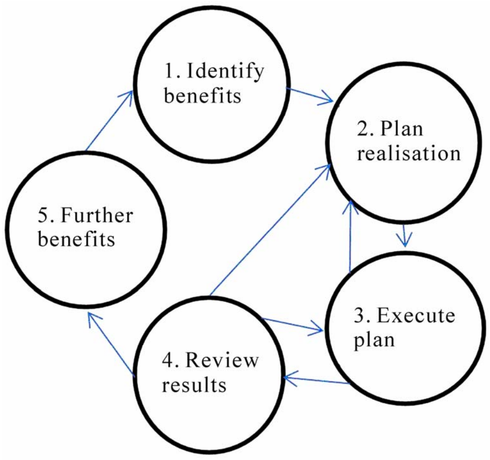


Figure 1. Benefits management framework.

identifying outcomes of potential value to stakeholders (such as capabilities), with little or no identification of planned changes. It should be of little surprise that in many of those examples, limited measurable improvement was achieved.

Any business case should not necessarily require volumes of text, but the core should be summarised succinctly against the following structure:

- goal, objectives, outcomes and planned benefits, risks, assumptions

2.2. Delivering Strategic Goals and Objectives

Most organisations have current strategic goals and objectives. These should be articulated and be very evident throughout benefits identification and planning. The business case needs to be evaluated thoroughly to ensure that it is focused on and maximises delivery or achievement of strategic goals. Following this, the realisation plans will provide a control mechanism to provide continual feedback against strategic goals.

2.3. Maintaining the Focus

During the life of a project it may be necessary to modify the objectives, change priorities or redefine the desired outcomes in the light of changing circumstances. It is important that structure and accountability continues through and beyond the life of the project and beyond, to ensure that the benefits of most value are realised at affordable cost and on schedule.

2.4. Ownership and Implementation of the Benefits Realisation Plan

Many of the anticipated benefits will not start to materialise until after the project has been delivered. It is therefore essential that the ownership of the benefits realisation plan is maintained beyond project delivery through to complete realisation. The process should also include a post implementation review, thereby allowing time for analysis and a proper evaluation against the original business case.

In practice most business managers are happy enough to accept these challenges as they recognise that Benefits Management provides them with an effective way of tackling a significant issue in their organisation.

2.5. Do’s and Don’ts of Benefits Management

According to Ward and Murray [3] there are some Do’s and Don’ts as far as Benefits Management is concerned:

- Do start Benefit Management on day one of every project
- Do involve all potential and known stakeholders early in determination of benefits
- Do make sure all dis-benefits are exposed and understood-ensure they are a price worth paying
- Do carry out a pilot or prototype if benefits are uncertain to determine what benefits are achievable and how to realize them
- Do ensure that all changes that affect the plan are interpreted in terms of the benefits and the benefit plan
- Do publicise benefits that have been achieved
- Do use Benefit Management to stop bad projects
- Do not expect to be able to predict all benefits in advance-many will only be understood after implementation
- Do not stop managing the benefits when the ‘project’ is finished

3. Benefits Management and PRINCE2 Comparison

PRINCE2 is an example of a structured project management approach that it is used widely in both the private and public sectors. It was developed by the Office of Government Commerce (OCG) and is the recommend approach by UK government projects [4].

PRINCE2, can be considered a refinement of an earlier approach PRINCE2 which is based on existing best practises in project management and other methodologies. OGC was involved in the early stages of the research that led to the development of the benefits management process described in this paper.

There seems to be a high degree of correlation between the two approaches allowing them to be used together in a way that draws on the specific strengths of each approach. Nevertheless, even though being consistent, differences also are existent.

PRINCE2 is defined in eight distinctive processes for the effective management and governance of a project. (See **Figure 2**)

Each distinct process is described briefly here and then, how benefits management and PRINCE2 can be combined to complement each other follows.

1) Starting up a project; this sends to be the first and a short process in which the project management team is appointed and the aims of the project are communicated. For this processes a Project Mandate is required, in which the reasons and the products (outcome) is the project is defined.

2) Directing a project; this is a process for the project board, in effect the senior management responsible for the project to direct its activities and resources. The process lasts for the full duration of the project and has five major strands within it:

Authorising initiation; Approval of the business case;

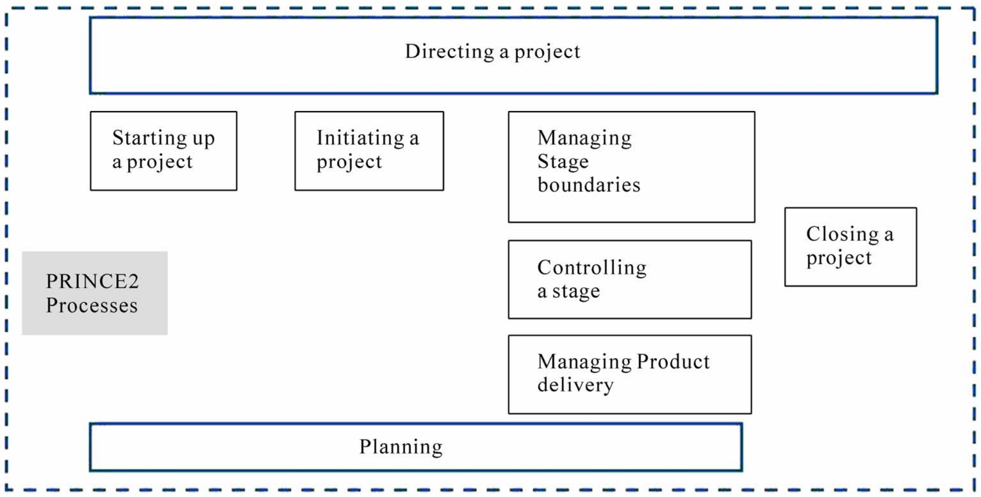


Figure 2. Effective management and governance processes Source: OGC, PRINCE2 Reference Manual (2005), p. 13.

Review of the project at stage boundaries; Ad hoc direction (progress monitoring) and ensuring the project comes to a controlled close and that lessons are shared with other projects.

3) Initiating a project; this process seeks to develop a business case for the project which, is contained in a project initiation document (PID). It includes also, the plan and the cost of the project as well as ensuring that the investment is well justified, taking into account the respective risks. It is suggested that a PID contains much information about a project including:

- Objectives
- Critical success factors and key performance indicators
- Impacts and assumptions
- Constraints and option evaluations
- Benefits analysis
- Project costs
- Cost/benefit analysis
- Risks
- Delivery plan- including stages or milestones

4) Controlling a stage; one of the key principles in PRINCE2; controlling projects is to break them into manageable, smaller stages. In this process it is described the monitoring and control activities, which are required to keep a stage on track.

5) Managing product delivery; specifies the contract between the project and suppliers. In effect the objective of this process is to ensure that planned products (outcome) are delivered as predefined. PRINCE2 calls the work agreed in the process a "work package" and seeks to ensure agreement on issues such as timing, quality and cost. For this reason, checkpoint reports are often exchanged between team and project manager.

6) Managing stage boundaries; this process is related to reporting on the performance of the previous stage, approval from senior management so as to move to the next stage, updating the project plan and detailed planning of the next stage. It actually produces the information based on which the Project Board will take the key decisions.

7) Planning; the planning process, is a repeatable process and continues throughout the whole the project. Each project plan (stage and team) must consider key planning aspects. According to PRINCE2 all activities should be logically be put in a sequence. Further to the plan, the process has a product checklist and the risk log.

8) Closing a project; the project board decides to close the project once its products are delivered and objectives are met. Moreover it is ensured that follow-up actions are undertaken and lessons shared are learned in conjunction with other projects.

According to Ward and Daniel [5], while PRINCE2 is analytical enough by providing very detailed guidelines on how project management methods and practices can be improved, on the other hand, benefits and their management seem to be described depthlessly.

To be more specific, while it is advised that the project initiation document (project initiation stage) describes a comprehensive benefit analysis; linking for example the benefits to the changes required, appointing benefit owners, settings measures for each benefit, the details on how to accomplish it, is limited.

Benefits are considered to be very important as far as the decision to manage a project is considered. This is because benefits in many cases compensate risks. If the risks in a project, in effect the possibility of failure is high then the decision might be complicated and benefits should be taken into account.

Essentially, the limited treatment of benefits is therefore a noticeable area of weakness. As a walkthrough it can be suggested that the tools and techniques related to the first two stages of the five-stage process (identify benefits, plan realisation) are used to develop a full benefits plan. For PRINCE2, the benefits plan can be described in the PID. The simplicity of the benefits management approach is also important and lies in this early stage of the project.

Ward and Daniel [5] illustrated that the strength of PRINCE2 lies in its comprehensiveness formality attention to detail and its robustness. However, the result of being inevitably complex, has as a result, that most business managers do not want or have the time to learn the methodology or even be subjected to it.

Nevertheless, in UK it is the de facto standard for project management required by the government; there are also voices which claim that we are not far from the time that it will be used as a standard by ISO quality management for quality control of project/s environment.

It is therefore suggested, that, PRINCE2 process of controlling stages, managing state boundaries and managing product delivery are used if required to undertake the third stage of our benefits management process: the execution of the benefits plan. A key part of PRINCE2 approach is the breaking of projects into phases. The benefit plan particularly the benefits dependency network, can prove a means of identifying and comparing possible phases.

According to the closing process, as described in PRINCE2 handbook, it is suggested that the success of the project is reviewed and shared among the stakeholders so as best practised to be revealed. However, these best practices do not specifically focus on benefits. Moreover, identification of further potential benefits is not accurately described or described at all. In effect it is suggested that benefits management process is followed in these activities.

Finally, for organisations that have chosen as project management methodology PRINCE2, the benefits evaluation which come up from the project evaluation, is better to be included in the project closing process.

In **Figure 3**, it is illustrated how the benefits management process and PRINCE2 are related and which approach suggested should lead at each stage.

4. Conclusions

Benefits Management should be considered the first priority of any project. This is because it describes effecttively the "steps" of how a project should be managed, and consequently what will be the outcome, "benefits". The main purpose of Benefits Management or any other similar process is to avoid project failure.

In effect, great attention is given in testing and imple-

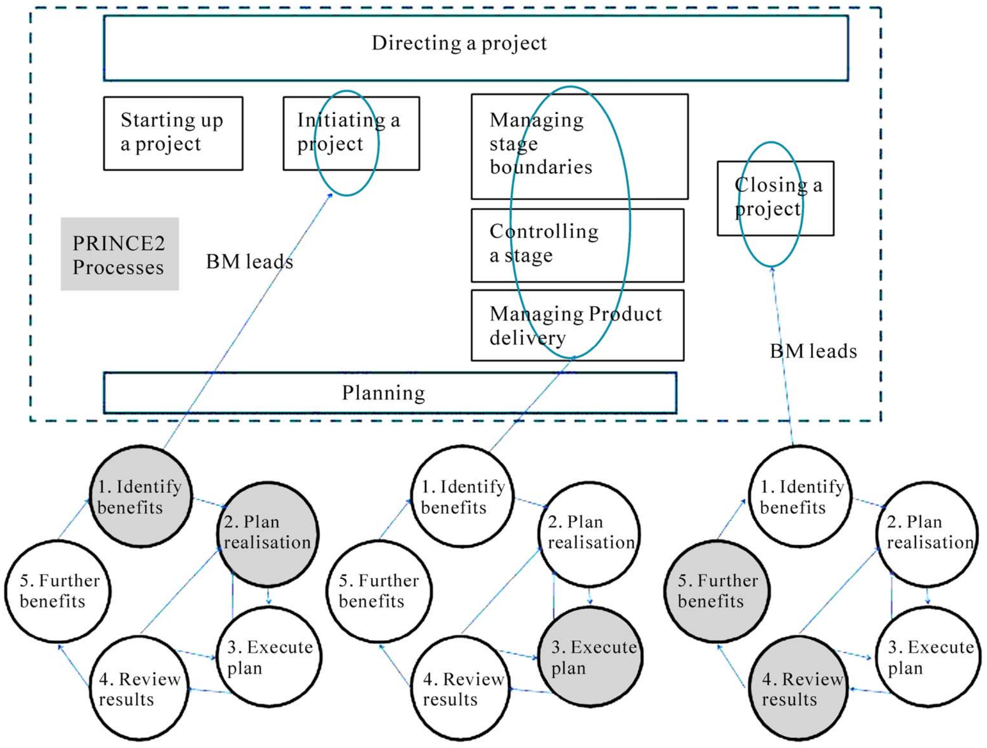


Figure 3. Benefits management and PRINCE2 relation, Source: Ward J. and Daniel E. (2006), p. 274.

menting business solutions. According to the benefits management approach, the first step is to identify the benefits, which in turn have to be structured, planned and realised.

PRINCE2, is an alternative to structured project management methodologies and approaches. Compared to Benefits Management one, benefits are treated in a limited way which suggest an obvious weakness, but it is a lot stronger in analytically defining the details of the processes.

Business management methodologies are not a panacea against project failure; nevertheless, they can be seen and used as a powerful tool in the hands of the stakeholders which can lead to project success.

Proper requirements analysis drive almost every task and activity, however, the identifications of when, how and what has to be done should be a bidirectional activity among all the parties involved. Failure in projects is a status which every project manager tries to avoid; with the aid of project methodologies and especially with benefits management one, which complements other project management methodologies the possibility of success is enhanced.

Finally, the key message for managing project expectations is effective, efficient communication and cooperation between the project manager and the client.

REFERENCES

[1] The Standish Group, “Chaos Report,” 2007. http://www.standishgroup.com/search/search.php

[2] C. Apostolopoulos and I. Karamitsos, “The Success of IT Projects Using Agile Methodology,” 1*st International Workshop on Requirements Analysis Proceedings*, Pearsons Education, Elista, September 2009, pp. 13-21.

[3] J. Ward and P. Murray, “Benefits Management Best Practice Guidelines,” Information Systems Research Centre, Cranfield School of Management, 2000.

[4] Office of Government Commerce, “Managing Successful Project with PRINCE2 Reference Manual,” TSO (The Stationery Office), London, 2005.

[5] J. Ward and E. Daniel, “Benefits Management-Delivering Value from IS & IT Investments,” John Wiley & Sons, New York, 2006.

J. Software Engineering & Applications, 2010, 3, 845-851
doi:10.4236/jsea.2010.39098 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

Requirements Analysis and Traceability at CIM Level

Mohammad Yamin[1], Venera Zuna[2], Moteb Al Bugami[1]

[1]King Abdoulaziz University, Saudi Arabia; [2]Albanian Mobile Communications, Tirana, Albania
Email: myamin@kau.edu.sa, vzuna@amc.al, maalbugami@kau.edu.sa

ABSTRACT

Poernomo suggested an approach for requirement analysis within the CIM level of the MDA framework. His approach combined MEASUR, goal and object oriented analysis, and developed a new methodology that can be integrated within the CIM level of the MDA. This paper adds requirement traceability capabilities to the method developed by Poernomo and applies the extended method on a case study based on a high profile international law firm.

Keywords: *Organizational Semiotics, MEASUR Requirements Traceability, CIM Requirements Traceability*

1. Introduction

Quite a lot of research has been conducted to identify the reasons of the failure of Information Systems. We all know that a huge amount of money is spent every year on Information Systems and in the efforts to understand their failures. A very low rate (as low as one out of eight) of successful projects is becoming a matter of great concern. As much as 35% of the projects failed as a result of poorly defined software requirements, for details see [1]. The requirements are evidently the most important deliverable of the software engineering activity. Since the requirements are the foundation of the end product, all other product steps are based on the requirements. Errors made at this stage would have a completely overwhelming effect on the rest of the project, for details see [2]. It is at the stage of user acceptance testing to realize that the incomplete requirements and specifications would produce a camel instead of a horse required by the client. According to Von Schlag [3] the majority of the defects occur during the requirements phase. In order to deliver successful projects it is essential to clearly understand what the business needs are.

A number of methods and approaches have been developed to deal with the problem of user requirements, such as MEASUR, KAOS, object oriented analysis and many more. These methods and approaches examine information systems from a different view. MEASUR approaches the systems from a semantic point of view, KAOS from an goal oriented view and object oriented from a structural point of view. All these methods have their own benefits and drawbacks. In 2000, the Object Management Group [4] developed the Model Driven Architecture (MDA) framework. This generated a new environment that requirements analysis methods should be compatible with. The key idea behind MDA is that models can be used to auto generate other models. By model we mean shapes, diagrams and code. The basic MDA engine includes four layers namely the Computational Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model and code. Transformations allow the PIM to be transformed to PSM and PSM to be transformed to code. Transformations from CIM to PIM are very primitive and a great deal of work still needs to done for requirement analysis at CIM level, see [2], the lack of which results in poor quality product.

Poernomo [5] suggested an approach for requirement analysis within the CIM level of the MDA framework in 2008. His approach combined MEASUR, goal analysis and object oriented analysis, and developed a new methodology that can be integrated within the CIM level of the MDA. This approach solved most of the issues of these methods while maintaining the benefits of individual methods. However his approach did not include a mechanism for tracing requirements. In this paper we will enhance Peornomo's method with a requirement traceability repository in order to achieve inbuilt requirements management. The method will then be used to conduct requirement analysis at the CIM level for top tier law firm of our case study.

2. The Selected Method

A recent attempt to integrate a requirement analysis model at MDA's CIM level is by Poernomo in 2008. In this proposal parts from the approaches: MEASUR, Goal driven Analysis and Object Oriented Analysis were used together [6]. The figure below shows an over view of that approach.

As it can be seen from the diagram in the **Figure 1**, the methodology focuses on conducting requirements analysis at CIM level of the MDA framework. According to the method, at the beginning, stakeholder analysis should be carried out and its findings should be captured and categorised based on the organizational onion. Organizational onion is MEASUR's equivalent for stakeholder analysis. This not only lists the stakeholders and their needs but also prioritises them, based on how critical they are for the success of the project. In parallel with organizational onion goal analysis should be conducted. This will identify all the business goals and needs of the client and ensure that they are properly documented and captured. The goal analysis will also associate the business goals dependencies in a hierarchical order. The results of both organizational onion and goal analysis will be fed to the technical requirement table.

Table 1 consists of eight columns. The first column is the actual business goal; the second column lists the business goal dependencies that must be achieved prior to achieving this goal. The third column is the development priority of this goal. This is calculated by taking an account the business priority and any functional dependencies. For example, assuming that the main goal is to move a car, a sub goal would be to move each individual wheel of the car. In order to move the car we must first move the wheels of the car. Hence, the goal moving the car is dependent on the sub goal of moving the wheels of the car. Let's assume that move the car goal has a higher business priority than move the wheels of the car. However there exists an architecture priority as it is not possible to move the car without moving the wheels of the car. As a result of this moving the wheels of the car is pushed to a high priority. The fourth column is a list of all the business owners. These are the stakeholders of this task and are extracted from the organizational onion. It's worth noting that this can also affect the priority column. For example, if this stakeholder is not close to the system (this can be found in the organizational onion) than by default this goal would have a lower priority than the stakeholder's goal that is closer to the system. The fifth column is a list of users that will be affected by the achievement of the specific goal. The start and finish time columns are used for initial planning. The last column can be either yes or no and shows if the goal has been approved or not. Only goals that have been confirmed will be pushed to the next phase.

The next phase is the generation of problem statements and also known as stories in the agile communities. This is a piece of text with its size to vary from one paragraph to 3 pages. It provides more details of what the client expects for this goal. This text is usually full of business terminology and free of any technical details. In this phase a problem statement will be written for each confirmed goal. Parallel to the problem statement the analyst is required to produce user scenarios (use case diagrams) for each goal. The number of required use case diagrams depends on how many user functions are associated with this goal.

The next phase is the generation of the ontology chart. At this stage, the ontology and ontological dependencies

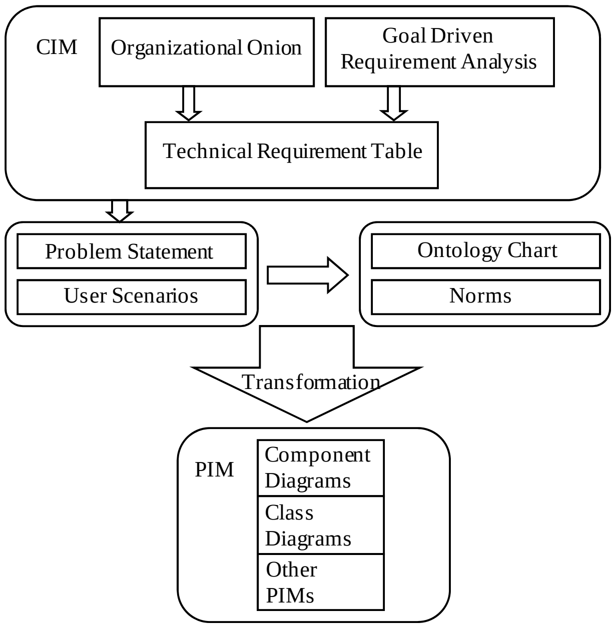


Figure 1. An overview of the selected approach.

Table 1. Technical requirements table.

Goal	Dependencies	Priority	Owner Stakeholder	Actor	Start Time	Finish Time	Confirm
Move Car	*Move car wheels*	High	Andrew	Driver	1/8/2008	1/11/2008	yes
Move car wheels	N/A	High	Andrew	Driver	1/8/2008	1/11/2008	yes
Clean the car	N/A	Low	John Block	Cleaner	1/10/2008	1/10/2008	no

will be identified from the problem statement. Once the ontology chart is complete it will be tested against the User scenarios which are stored in the form of use case diagrams. To complete the dynamic aspect of the system the analyst must specify ideally by the use of formal methods the dynamic business norms that govern the information system.

The proposal includes a MOF formal meta-model that allows ontology charts to be used within the MDA framework. Finally, the proposal also includes an automatic transformation from ontology charts and the formal norms to an object oriented diagram and suggested that transformations are also possible for components, class diagrams as well as other PIMS.

This methodology brings to light many advantages as it builds upon all the other methods mentioned above. This methodology proposed by Poermono and others is immune to business changes and analyses the requirements of complying with the business goals as to analyse the right system to add value to the system and the right way to produce this system. By proposing a meta-model for the ontology charts, it allows all the benefits of this method to be carried over automatically to the computer system by utilizing the MDA framework. If there is a change at the requirements due to a change of the business goal, the methods provides mechanisms for capturing and reviewing this objective and can automatically be applied to the computer system without any effort and without increasing complexity of the system.

The MDA framework is capable to rebuild the system with the new requirements without any effects to the rest of the users apart from the ones impacted by the change to the business goal. Another benefit of the methodology is the simplicity. Its diagrams can be used and produced by people that do not have computing background. This methodology is a step towards bridging the gap between the business analysis and software development.

3. Requirements Traceability

Requirements keep changing even during the project development. A challenge for the requirements analyst is to keep track of the changes in business requirements. Anthony Finskenstain [7] has proposed requirements traceability approach.

Requirements traceability is the ability to trace a requirement at any stage of its life cycle, revisit or even modify it. This is achieved by the use of appropriate software tools and manual processes. Such tools are document repositories able to search the documents for key words, compare documents for similarities and retrieve them for read or modification. Requirements traceability allows the software development team and the business stakeholders to locate and modify requirements at any stage of the requirements life cycle.

A recent survey on requirements management tools showed that there are more than 44 tools in market offering Capturing Requirements/Identification, Capture System Element structure, Requirements Flowdown, Traceability Analysis, Configuration Management, Documents and Other Output Media, Interfacing to Other Tools and many more [8].

4. Extending the Selected Method

4.1. An Overview

Poernomo's method is capable of delivering the benefits of MEASUR, Goal Analysis and object oriented analysis in the form of formal design, compatible with the MDA framework and capable to generating high quality code. The drawback however of that method is that, although it supports future changes on requirements, it does not have a mechanism for managing and tracing requirements. Such an addition will allow the methodology to trace, evaluate requirements, auto-generate test condition and test cases, proving information about the cost, duration and other information that can be used for planning as well as the rest of the benefits of requirements traceability. None of the current traceability tools auto-generate code from requirements hence they are just used as document management system.

The solution proposed in this paper will hold formal models that can be used to produce other models and code with the use of MDA framework. At the same time, the basic functionality of trace requirements will be allowed.

Figure 2 above shows how Poernomo's original proposal which is modified to accommodate requirements traceability. Initially the technical requirements table is stored to the traceability repository. This will be temporary and will keep track of all changes in the traceability table. There is no point in storing any information from the goal analysis or the organisational onion as the summary of these information is stored in the traceability table.

The problem statement and the use cases will also be stored in the traceability repository and be associated with the requirements from the technical requirements table. Finally the ontology chart and the business norms will be stored in the repository and associated with problem statements.

4.2. Traceability Repository Structure

The following schema in **Figure 3** shows the proposed structure of the repository.

In the object schema above, the requirements table stores information about the actualgoal in text form, it's

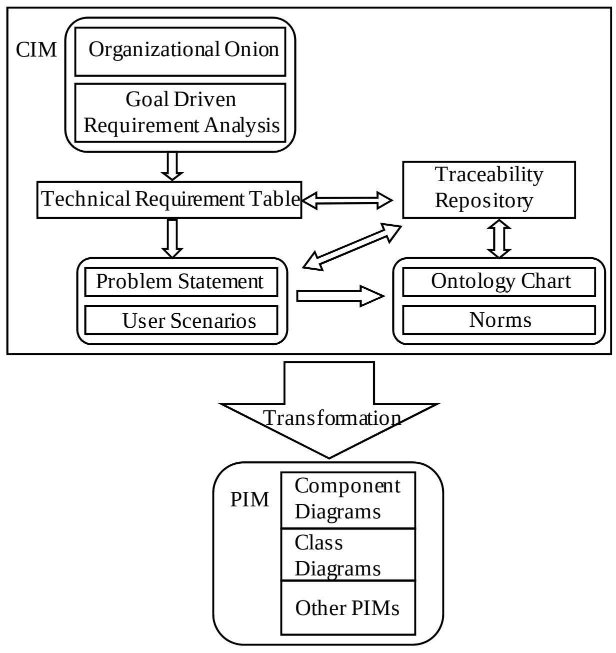


Figure 2. Extension of the selected method.

priority, stakeholder, actor, start and finish time as well as if it has been confirmed or not. The Dependences table stores all the sub goals and associate them with a parent goal. For each goal, there can be many use case diagrams.

Each use case diagram consist of one to many cases, each includes the text describing the case. Each case can be either a main case, an include case or an extend case. The attributes include_id and extend_id allow the system to store such information. Each requirement has one or more problem statements. The entity Problem statement includes the actual description of the goal in the form of text. For each problem statement there are a number of ontology charts. Each ontology chart has a title and a domain as well as zero to many OCL statements used to capture the business norms and one to many universals. These are the notes of the ontology chart. Each of them has a type, a label and can be associated with zero (if it is the root note only) or two other universals.

The above schema is capable of capturing all the information generated during the requirement analysis phase

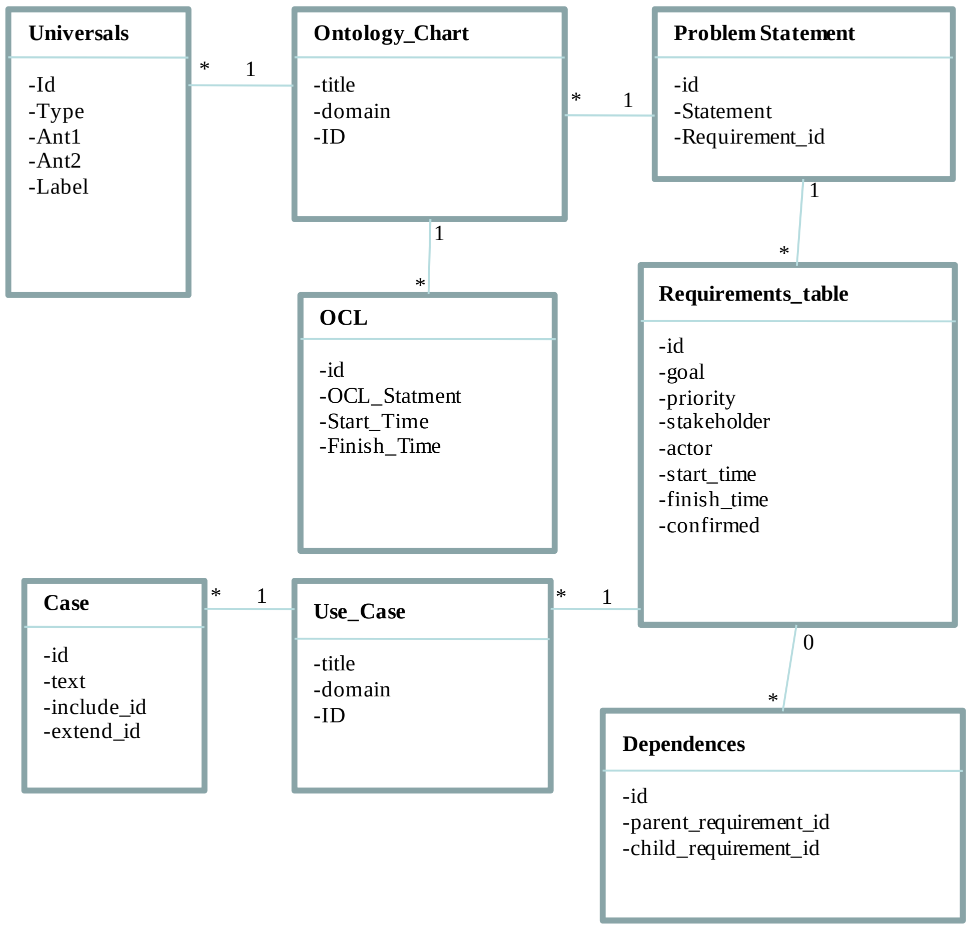


Figure 3. Traceability repository structure.

and retrieve them if required. It is also temporary as it keeps history of changes and supports non-destructive updates. It is therefore capable of enhancing Poernomo's 2008 method with requirements traceability capabilities.
It is the author's believe that such addition will improve the requirements management capabilities of the selected method and will provide a great tool for requirement analysis and management at CIM level.

5. Case Study

5.1. The Business

A top tier international law firm offers many legal services across a broad range of areas such as finance, merger and acquisitions, employment and benefits, energy and infrastructures *etc*. to a vast number of clients. The client and matter proceedings results in a big amount of paperwork. All the documents are saved in different profiles and a huge number of databases need to be utilised.

To deal with this problem in the past, the firm employed an IT solution based on profiling lotus notes. The management has decided to change the technology by upgrading to a new technology. The replacement of the ABC Profiling Lotus Notes databases has been under review for some years and different technology approaches have been discussed. The most recent technology approach was a study conducted in 2008, which culminated in a Proof of Concept to prove that the majority of ABC requirements could be encompassed into the, Beta version of Sharepoint 2007. The main disadvantage of this approach is the data in the databases has to be converted to the new system format inheriting the risk of destroying the sensitive data. Projects can now be built upon this Proof of Concept and the Sharepoint seeks to build a single replacement solution for the current Lotus Notes databases and migrates the data into a new Sharepoint 2007 application.

ABC has four Lotus Notes databases. In these databases the relevant ABC team captures extensive profiling information regarding their matters (*i.e.* legal transactions or legal deals); this could be likened to extremely detailed metadata. This profiling information is used for legal precedents and is a critical part of ABC's knowledgebase. Each profile can relate to a 'bible'. A bible is ABC's term for one or more key documents selected at the end of a matter, which form crucial reference and precedent information for legal transactions of a similar nature going forward. Sometimes it is possible to capture profile information when a bible has not yet been created, but then reference the profile to the bible at a later date. The proposed new solution for ABC Bibles Profiling will allow a certain user group (Administration or Profile User) to create and maintain profile information. The General Users will then be able to search on this profile information. All bible profile information can link into any existing bibles that reside in the Document Management System. This is an electronic repository of bibles held within the Document Management System.

5.2. Organizational Onion and Goal Analysis

The organizational onion of this system is as shown in **Figure 4**.

The system is the ABC Bibles and all the layers of the "onion" are labelled with the right entity corresponding to the relation engagement to the system. Closer to the system are the users. The users have different access rights. There are three different types of users Administrations users, profile users and general users.

After the organization onion the Goal driven analysis is conducted. Goals get extracted by the business owner of the system. One example of Goal Analysis would be General User which would search on the Profile for information. **Figure 5** shows Goal Analysis of our chosen case study of law firm.

5.3. Technical Requirements Table

The next stage is to populate the requirements of **Table 2**.

The Search Profile is dependent on Create Profile goal, being so the Search Profile goal is of high priority as the Create Profile since someone cannot search a profile unless it has been created.

5.4. Problem Statement and Use Case

After the technical requirement table is created the problem statements are created for each goal identified in the table. Below is an example of creating a new profile.

"*Administrator users create new profiles. Every new profile includes detailed information about the clients and the legal case. This information consists of Client name, Client Address, Client Litigation Party, Legal Case description, Case Number, Involved parties. The profile information needs to be linked to the document*

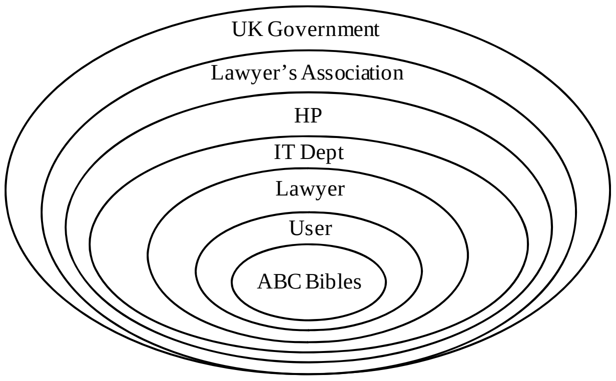


Figure 4. Organisational onion.

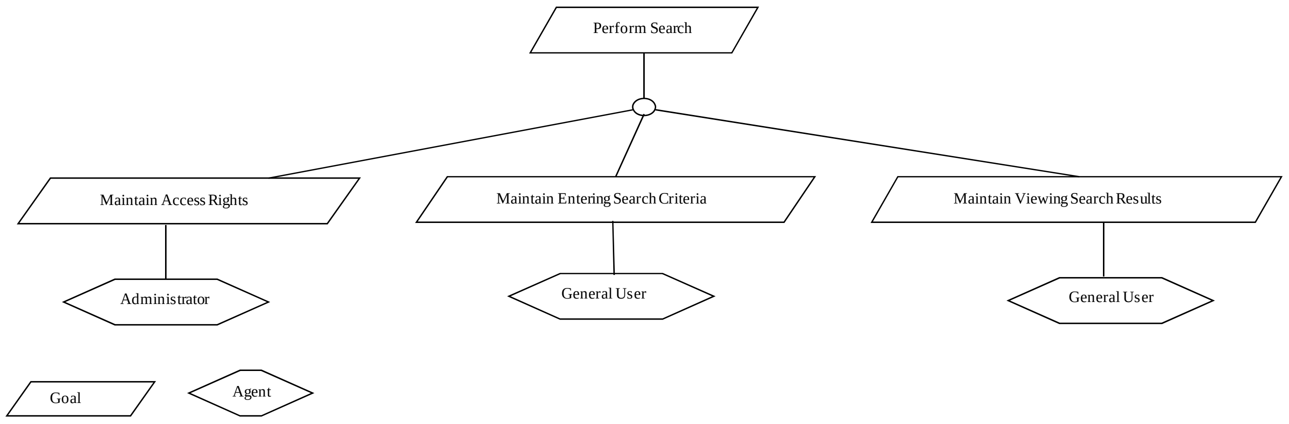


Figure 5. Goal analysis.

Table 2. Case study's technical requirements table.

Goal	Dependencies	Priority	Owner Stakeholder	Actor	Start Time	Finish Time	Confirm
Create Profile	N/A	High	Web Team	Administrator User	1/10/2008	1/11/2008	yes
Search Profile	Create Profile	High	IT Dept	General User	1/11/2008	1/12/2008	yes
Grant Access	N/A	Medium	User Admin	Profile User	1/10/2008	1/10/2008	yes

management system in the back end. The profile has to be linked with the document management system as to relate the clients paperwork with the legal case paperwork stored in the back end databases."

Following the problem statement the use case is created. The following use case diagram shows how the profiles are created by the administrator user. (see **Figure 6**)

5.5. Ontology Charting

The ontology chart is created to depict affordances and antecedents. The ontology chart could be used as the input to transformation as to produce the Platform Independent Models such as Class Diagrams, Components diagrams *etc*. (see **Figure 7**)

The Requirements table, the problems statements, the use cases and the ontology charts with the business norms where automatically stored to the traceability repository. This will now allow the analysts to trace the life of any requirement, assuming that the business analyst wants to change an existing requirement. This can be achieved by updating it the goal in the requirements table. The old goal will be kept in the repository. The user will then be required to perform the appropriate changes to the problem statement, the use case, the ontology charts and the business norms. Once this is completed the traceability repository will not destroy the old entities. It will put a finish time on them and let them be creating

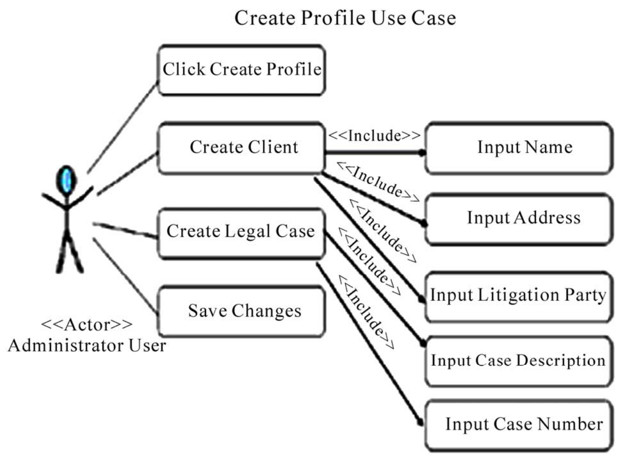


Figure 6. Create profile use case.

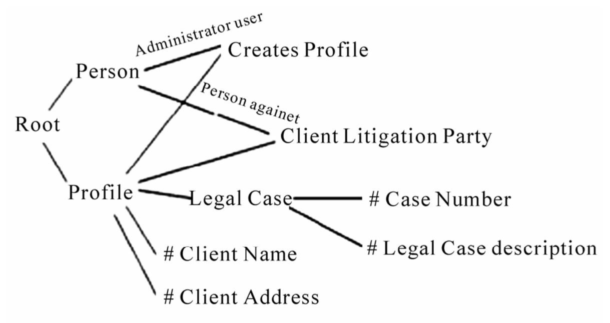


Figure 7. Create profile ontology chart.

new entities and associating the appropriate rows of data with them. After all the updates are finished the software system will be able to be regenerated with the use of the latest data, such as the latest ontology chart and norms by the use of the MDA framework.

6. Conclusions and Future Work

This project reviewed all the major methodologies for requirements analysis, MEASUR, Goal Analysis, Object Oriented Analysis as well as a methodology that combines all of them and can be integrated within the MDA framework. The last was selected and applied to the case study from a law firm. The method was also enhanced with a requirement traceability repository that allowed analyst to store, trace and modify user's requirements.

For future work the requirements traceability system can be developed and integrated within an industry standard tool such as eclipse. Additional search functionality that will allow the system to search models for similarities would also be welcomed. Last both the method and the requirements traceability mechanism need to be test on more case studies.

REFERENCES

[1] M. Roper, "Software Testing," *ACM Computing Surveys*, Vol. 23, 1994, p. 103.

[2] R. Wieringa, "A Survey of Structured and Object-Oriented Software," 1998. http://portal.acm.org/citation.cfm

[3] V. Schlag, *Patrick Certification Magazine*, Vol. 8, No. 9, September 2006, pp. 30-35.

[4] "OMG, MDA Guide Version 1.0.1," 2000. http://www.omg.org/docs/omg/03-06-01.pdf

[5] I. Poernomo, G. Tsaramirsis and V. Zuna, "A Methodology for Requirements Analysis at CIM Level," 2008. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-376/paper2.pdf

[6] V. Castro, J. M. V. Mesa, E. Herrmann and E. Marcos, "From Real Computational Independent Models to Information System Models: An MDE Approach," *Proceedings of the* 4*th International Workshop on Model-Driven Web Engineering*, Tolouse, 2008.

[7] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," *Proceedings of* 1*st International Conference on Requirements Engineering*, Vol. 11, 1994, pp. 94-101.

[8] A. Salter and L. Kecheng, "Using Semantic Analysis and Norm Analysis to Model Organizations," *Proceedings of International Conference on Enterprise Information Systems*, Vol. 3, 2002, pp. 1-7.

[9] T. Philip, G. Schwabe and E. Wende, "Early Warning Signs of Failures in Offshore Software Development Projects," *Management Services*, Vol. 51, No. 3, 2007, pp. 38-43.

J. Software Engineering & Applications, **2010, 3, 852-857**
doi:10.4236/jsea.2010.39099 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

A Review of the Impact of Requirements on Software Project Development Using a Control Theoretic Model

Anthony White

School of Engineering and Information Sciences, Middlesex University, the Burroughs, Hendon, London, UK.
Email: a.white@mdx.ac.uk

ABSTRACT

Software projects have a low success rate in terms of reliability, meeting due dates and working within assigned budgets with only 16% *of projects being considered fully successful while Capers Jones has estimated that such projects only have a success rate of* 65%. *Many of these failures can be attributed to changes in requirements as the project progresses. This paper reviews several System Dynamics models from the literature and analyses the model of Andersson and Karlsson, showing that this model is uncontrollable and unobservable. This leads to a number of issues that need to be addressed in requirements acquisition.*

Keywords: *Requirements Models*, *System Dynamics*, *Control Systems*, *Observability*, *Controllability*

1. Introduction

Software projects have a low success rate in terms of reliability, meeting due dates and working within assigned budgets [1-3] with only 16% of projects being considered fully successful while Capers Jones has estimated that such projects only have a success rate of 65%. The American "Standish Group" has been involved for 10 years with research into ICT. In their research, they aim to determine and change success and failure factors regarding such projects. Their study, which has been appropriately baptised "*Chaos*" [4,5], appears every two years. This study also shows that in 2003 only 34% were successful, 51% did not go according to plan but ultimately did lead to some result and 15% of the projects fail completely.

Despite these failures significant progress has been made in the use of System Dynamics methods to describe the development of software projects. The models of operation of the software development process were described by the successful System Dynamics (SD) models based on the work of Abdel-Hamid & Madnick [6], which set up equations relating levels such as the *number of perceived errors*, or *the number of reworked errors* and relates them to rates such as *the error detection rate* or *the rework rate, significant features of these models included the decision processes.* These models were validated against NASA project data for a medium size project and the agreement is strikingly good.

Many of these failures can be attributed to changes in requirements as the project progresses. Capers–Jones [7] states that as the project gets larger the probability of requirements creep becomes more likely, typically 1-2% per month and as high as 10% in a single month. Lorin May [8] talks about poorly established guidelines that determine when requirements should be added, removed and implemented. Deifel and Salzmann [9] describe a view of "requirements dynamics" relating to the process of changing requirements. They go on to develop a strategy to deal with the regime in which some requirements are invariant and some migrate.

Coulin *et al.* [10] state that "the elicitation of requirements for software systems is one of the most critical and complex activities within the development cycle" and that "this is preformed after project initiation and preliminary planning but before system conception and design." This would not be strictly true if evolutionary or iterative methods were used. The later the requirements in the cycle of development change, the more costly is that revision (Boehm & Pappacio [11]). It is certainly the case as Hoorn *et al.* [12] report that owing to many shifts in focus and priorities, stakeholders become inconsistent about what they actually want to accomplish with the system. If we are to improve the requirements process then proper models of a process are needed. Kotanya &

Sommerville [13] outlines the requirements engineering process as shown in **Figure 1**. Although there is feedback between requirements validation and specification and in the elicitation and specification as will be shown this is not represented in the current models. It is not clear in any of the texts on the subject whether the involvement of the use is mandated at these stages.

The whole purpose of this paper is to present simple control system models of the project development process including requirements, as in inventory analysis, and demonstrate rules for stability.

2. System Dynamics

Wolstenholme [14] describes System Dynamics as:

"A rigorous method for qualitative description, exploration and analysis of complex systems in terms of their processes, information, organizational structure and strategies; which facilitates simulation modelling and quantitive analysis for the design of system structure and control".

This definition is expanded in **Table 1** taken from Wolstenholme.

The SD model structure is highly non-linear with a number of theoretical assumptions, for example about how the errors in the coding are propagated.

These structural assumptions do not allow for System Dynamics models to enable any general rules to be developed by academics for managers to make sound judgments based on good analysis. The distinction with models of inventory processes, which are related, is the rationale for this research program. Early SD inventory models developed by Forrester [15] were also non-linear and contained a number of factors, such as employment rate, that made the problem too complex for simple rules to be developed.

The simplest expression of representation of requirements in SD models is that use by Madachy [16], shown in **Figure 2**. In this case requirements are added to by a rate of generation, usually constant. The time taken to acquire the whole requirements is dictated by the acquisition rate. Häberlein [17] proposed a different structure for the development of the whole project. In his model (**Figure 3**) the rate of generation of requirements is split into several phases depending on the comprehension of the supplier and how this is influenced. This model could show considerable promise but no equations are presented. The model of Williams [18] (**Figure 4**) could not be evaluated further at this time due to incomplete equations. The structure indicated shows dependence on quantities such as customer satisfaction that are not readily measured during the process. The model of Andersson and Karlsson [19] (**Figure 5**) is the most complete and useful model out in the literature. Not only are all the equations given, with data, but the results are of a project in industry. This model shows that the process of gaining requirements is split into a phase where the level of requirements tasks to be completed is gained via an input pulse function. The required tasks to be completed are fed from the previous state by a constant requirements completion rate. Rework is discovered in these requirements and this is fed back at a constant rate to the first level. Inadequate requirements are discarded at a rate that

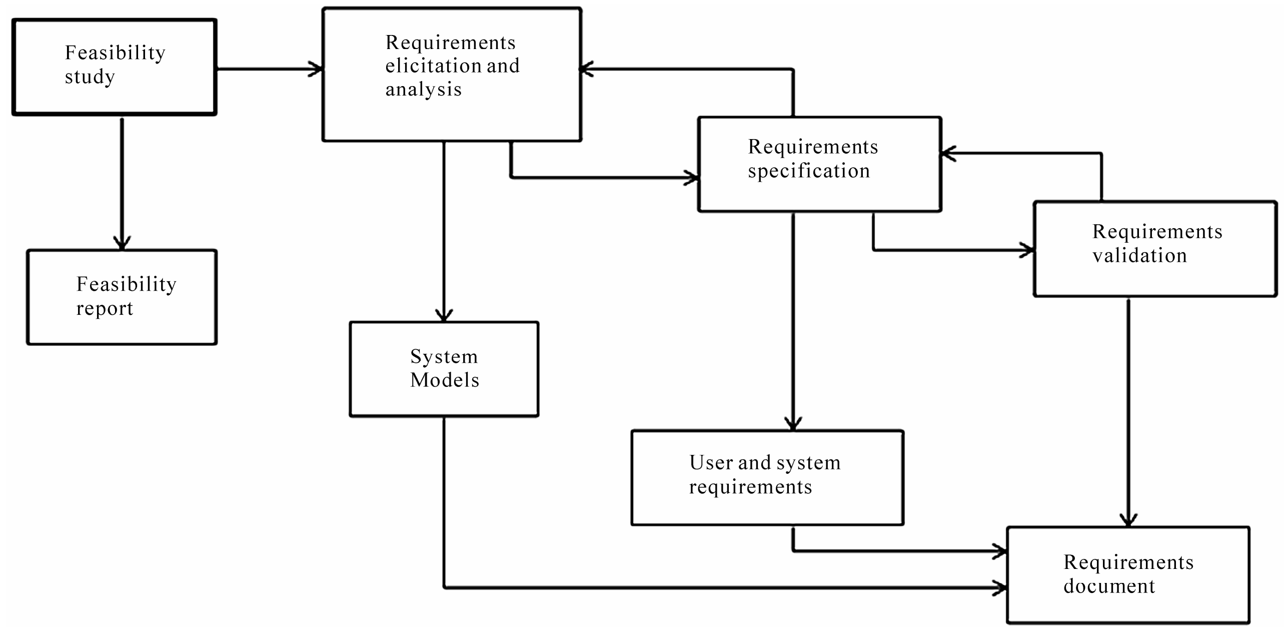


Figure 1. Requirements engineering (from Kotanya & Sommerville).

Table 1. System Dynamics a subject summary from Wolstenholme [14].

Qualitative system dynamics	Quantitative system dynamics	
(diagram construction and analysis phase)	(Simulation phase)	
	Stage 1	**Stage 2**
1. of existing/proposed systems 2. To create and examine feedback loop structure 3. To provide a qualitative assessment of the relationship between system process structure, information structure, delays organizational structure and strategy	1. To examine the behavior of all system variables over time. 2. To examine the validity and sensitivity of the model to changes in • Information structure • Strategies • Delays and uncertainties	1. To examine alternative system structures and control strategies based on • Intuitive ideas • Control theory analogies • Control theory algorithms: in terms of non-optimizing robust policy design

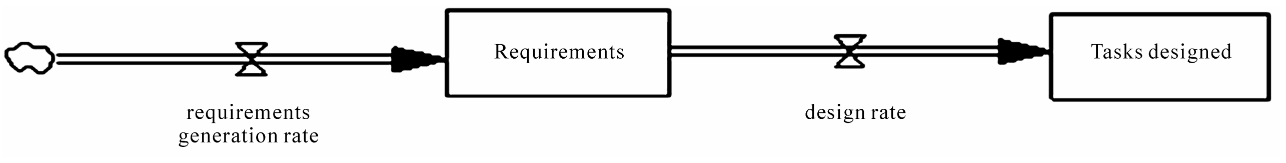


Figure 2. Raymond Madachy's model.

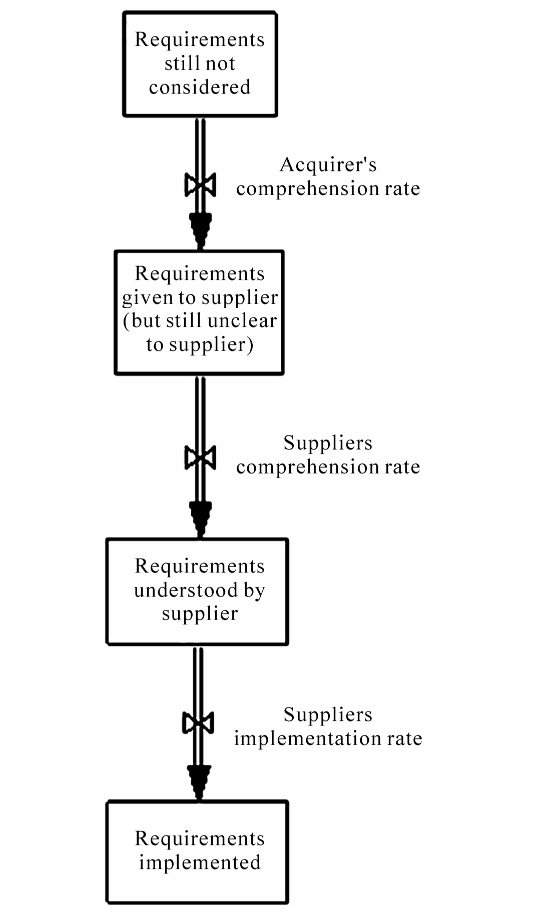

Requirements still not considered
Acquirer's comprehension rate
Requirements given to supplier (but still unclear to supplier)
Suppliers comprehension rate
Requirements understood by supplier
Suppliers implementation rate
Requirements implemented

Figure 3. Requirements as a total process in comparison to Abdel-Hamids' task based mod.

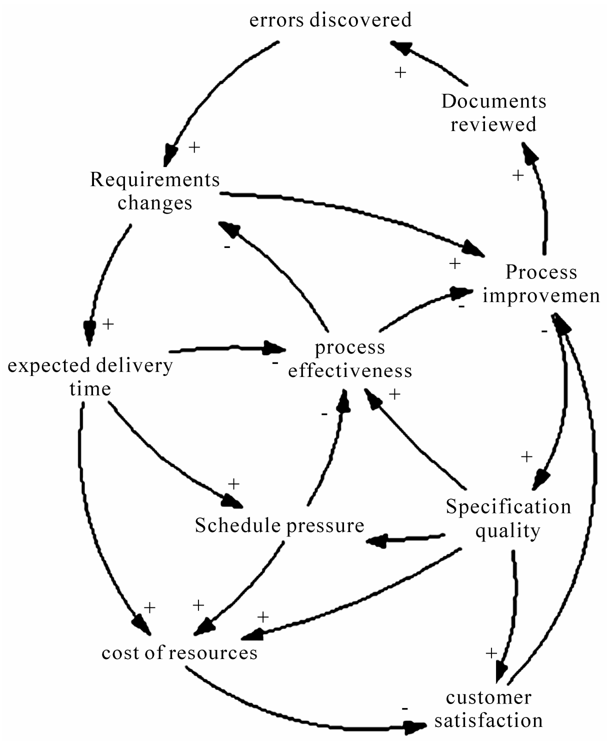


Figure 4. Requirements model of Williams [17].

is also a constant'. The final finished requirements are fed by a finished requirements rate. A number of non-linear "constants" are embedded into the system. No proper validation is made of this model or any of the models given here (this is normally very difficult).

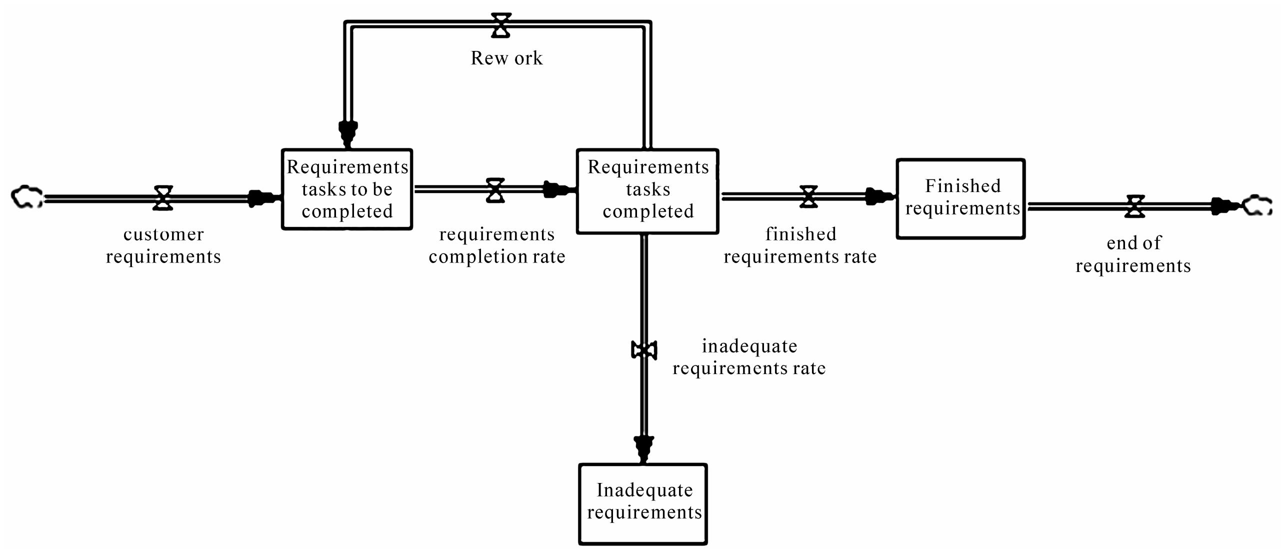

Rew ork
Requirements tasks to be completed
Requirements tasks completed
Finished requirements
customer requirements
requirements completion rate
finished requirements rate
end of requirements
inadequate requirements rate
Inadequate requirements

Figure 5. The model of Andersson and Karlsson [18].

Do any or all of these models match the published material on requirements engineering? In the broadest sense, yes, they do match what is contained in books such as Sommerville. To make further progress let us assume that the Anderson and Karlsson model is correct. This non-linear SD model has been linearised and analysed using control theory to see any general lessons can be learned.

3. Control Analysis

Part of the simplification of the Project Model is being tackled in the USA by the newer control system models of software testing (Cangussu *et al.* [20]) and the approach to control of software development by White [21].

In this case the model of Andersson and Karlsson was linearized and the following state equations obtained:

$$\frac{drttbc}{dt} = crr - rcr + rw \tag{1}$$

$$\frac{drtc}{dt} = rcr - frr - rw - irr \tag{2}$$

$$\frac{dir}{dt} = irr \tag{3}$$

$$\frac{dfr}{dt} = frr \tag{4}$$

The linearized auxiliary SD equations are:

$$crr = fi\delta(t) \tag{5}$$

(where this is a pulse of height *fi*, the initial estimate of the number of requirements).

$$rcr = rprod \tag{6}$$

$$irr = \left(\frac{rtt}{rp}\right)rtc \tag{7}$$

$$rw = rwp(rtc) \tag{8}$$

$$frr = \left(1 - rwp - \frac{rtt}{rp}\right)rtc \tag{9}$$

These equations can be represented by a state-space equation

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{A}\dot{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{B}'\boldsymbol{v} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \end{aligned} \tag{10}$$

where ***A***, ***B*** and ***B'*** are given by:

$$\boldsymbol{A} = \begin{bmatrix} 0 & rwp & 0 & 0 \\ 0 & -\left(\left(1 - rwp - \frac{rtt}{rp}\right) + rwp + \frac{rtt}{rp}\right) & 0 & 0 \\ 0 & \frac{rtt}{rp} & 0 & 0 \\ 0 & 1 - rwp - \frac{rtt}{rp} & 0 & 0 \end{bmatrix} \tag{11}$$

$$\boldsymbol{B} = \begin{bmatrix} fi \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

$$\boldsymbol{B}' = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{13}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

$$\boldsymbol{D} = 0 \quad (15)$$

$$\boldsymbol{x} = \begin{bmatrix} rttbc \\ rtc \\ ir \\ fr \end{bmatrix} \quad (16)$$

where $\boldsymbol{u}$ = pulse function and $\boldsymbol{v}$ = *rprod*. In this configuration $\boldsymbol{v}$ acts as a disturbance.

State-space theory can be used to see if this system is either controllable or observable.

We can define two matrices that will allow a measure of these properties if they are both full rank. The control stability is defined by the four eigenvalues two zero and two damped complex conjugates. The system is neutrally stable at best.

$$\boldsymbol{Cm} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B} & \boldsymbol{A}^3\boldsymbol{B} \end{bmatrix} \quad (17)$$

The rank of $\boldsymbol{Cm}$ is 1! The observability is given by:

$$\boldsymbol{Om} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \boldsymbol{CA}^3 \end{bmatrix} \quad (18)$$

The rank of this matrix is also 1. This means that the system described by the linearized state equations is uncontrollable and unobservable! The principle reason for this is that no corrective forces exist to alter the rate of production of requirements and that the rework and inadequate requirements cannot be altered independently of each other. Although a set of parameters will allow the requirements to be produced, once set in train no process exists to vary that process. No variation in workforce for example is set up in this model. No simple solutions allow this model to be put into a controllable form, although it can be made observable.

4. Conclusions

All the SD models illustrated here would appear to use a constant rate of conversion of requirement wishes from the customer to specifications, depending strictly on staff productivity. The number of staff in the cases cited appears to be fixed at the start of the process and altered only reluctantly, taking no account of project size or complexity. If this is generally true it has severe implications for the later analysis and development of the project. The most comprehensive model cited, due to Andersson and Karlsson has been analysed from a control system viewpoint. This analysis shows that such models are neutrally stable since there are no feedback mechanisms to establish when all the requirements are obtained, and they are neither controllable nor observable. The problem is that only the group of states *fr*, *ir* and *rtc* together are specified, one of them cannot be separately described or made to achieve a particular trajectory If the staff productivity is fixed and the number of staff is decided beforehand then the final outcome is proscribed. They can with some manipulation be made stabilizable.

REFERENCES

[1] J. Smith, "The 40 Root Causes of Troubled IT Projects," *Computing and Control Journals*, June 2002, pp.109-112.

[2] K. T. Yeo, "Critical Failure Factors in Information System Projects," *International Journals of Project Management*, Vol. 20, 2002, pp. 241-246.

[3] Royal Academy of Engineering, "*The Challenges of Complex IT projects*," Report of working group of RAE and BCS, 2004.

[4] The Standish Group International Inc., Standard Group CHAOS Report. 1998

[5] The Standish Group International Inc., Standard Group CHAOS Report, 25 March 2003

[6] T. Abdel-Hamid and S. E. Madnick, "Software Project Dynamics: An Integrated Approach," Englewood Cliffs, Prentice Hall, New York, 1991.

[7] C. Jones, "Large Software System Failures and Successes," *American Programmer*, April 1996, pp.3-9.

[8] L. J. May, "Major Causes of Software Project Failures," *Crosstalk*, July 1998.

[9] B. Deifel and C. Salzmann, "Requirements and Conditions for Dynamics in Evolutionary Software Systems," *Proceedings of the International Workshop on the Principles of Software Evolution, IWPSE*99, Fukuoka, 1999.

[10] C. Coulin, D. Zowghi and A. Sahraoui, "A Situational Method Engineering Approach to Requirements Elicitation Workshops in the Software Development Process," *Software Process Improvement and Practice*, Vol. 11, No. 5, 2006, pp.451-465.

[11] B. Boehm and P. N. Pappacio, "Understanding and Controlling Software Costs," *IEEE Transactions on software engineering*, Vol. 14, 1988, pp.1462-1477.

[12] J. F. Hoorn, M. E. Breuker and E. Kok, "Shifts in Foci and Priorities. Different Relevance of Requirements to Changing Goals Yields Conflicting Prioritizations and Its Viewpoint," *Software Process Improvement and Practice*, Vol. 11, No. 5, 2006, pp. 465-485.

[13] G. Kotonya and I. Sommerville, "Requirements Engineering Processes and Techniques," Wiley, 1988.

[14] E. Wolstenholme, "A Current Overview of System Dynamics," *Transactions on Institute MC*, Vol. 114, No. 4, 1989, pp. 171-179.

[15] J. Forrester, "Industrial Dynamics," MIT press, Boston, 1961.

[16] R. Madachy and B. Khoshnevis, "Dynamic Simulation Modeling of an Inspection-Based Software Lifecycle Process," *Simulation*, Vol. 69, No.1, 1997, pp.35-47.

[17] T. Häberlein, "Common Structures in System Dynamics Models of Software Acquisition Projects," *Software Process Improvement and Practice*, Vol. 9, No. 2, 2004, pp. 67-80.

[18] D. Williams, "Challenges of System Dynamics to Deliver Requirements Engineering Projects: Faster, Better, Cheaper," 21st *System Dynamics Conference*, New York, 2003.

[19] C. Andersson and L. Karlsson, "A System Dynamics Simulation Study of a Software Development Process," Lund Institute of Technology Report, March, 2001.

[20] J. W. Cangussu, R. A. DeCarlo and A. P Mathur, A Formal Model for the Software Test Process, *IEEE Transactions on Software Engineering*, vol. 28, no.8, August 2002, pp.782-796.

[21] A. S. White, "Control Engineering Analysis of Software project Management," *BCS SQM Conference*, Stafford, Section 3, April 2007.

Symbols

C*rr*	***Customer requirements rate***
fi	***Initial value of requirements assumed***
fr	***finished requirements***
frr	***finished requirements rate***
ir	***Inadequate requirements***
irr	***inadequate requirements rate***
rtc	***Requirement Tasks Completed***
rcr	***requirements completed rate***
rp	***requirement part***
rprod	***requirement productivity***
rtt	***fraction of tasks inadequate***
rttbc	***Requirement tasks to be completed***
Rw	***rework rate***
Rwp	***rework fraction of RTC***

J. Software Engineering & Applications, 2010, 3, 858-868
doi:10.4236/jsea.2010.39100 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

IT Project Environment Factors Affecting Requirements Analysis in Service Provisioning for the Greek Banking Sector

Krikor Maroukian

Printec Group of Companies, Athens, Greece.
Email: K.Maroukian@printecgroup.com

ABSTRACT

The research undertaken within a Greek IT organisation specialising in service provisioning to the Greek banking sector discusses the various aspects of a number of identified environment factors within five distinct IT projects which affect the requirements analysis phase. Project Management (*PMBOK*® *Guide* 4th *ed.*), *IT Service Management* (*ITIL*® v3) *and Business Analysis* (*BABOK*® *Guide* 2.0) *framework practices applied to the various IT projects are highlighted in regard to improved activity execution. Project issue management, stakeholder management, time management, resources management, communication management and risk management aspects are presented. These are then linked to the identified environment factors so as to indicate the adaptability of an IT support team to changing environment factors in IT project environments and how the fulfilment of these factors can significantly contribute to effective requirements analysis and enhance the requirements management cycle.*

Keywords: *Requirements Analysis*, *Project Management*, *IT Service Management*, *Business Analysis*, *Greek Banking Sector*

1. Introduction

International transaction systems have grown rapidly in the past decade, servicing more efficiently an ever growing number of debit and credit card holders throughout the world. Telecom Point-of-Sale (POS) devices offer an extended prism of services to cardholders in their daily purchase transactions. The Greek banking sector having sustained for many years the POS market, has now reached a high maturity level. This has enabled banking institutions to set a vision in terms of acquiring new technologies such as POS devices with embedded GPRS or Wi-Fi capabilities. It also means that suitable telecom network and IT infrastructure is established whereby POS management systems provide the required everyday service from banks to merchants and ultimately the cardholders. Requirements analysis is of significant importance in IT projects undertaken to support the POS management system environments that are currently installed at various banking institutions within Greece. As a consequence, it is the primary aim of this paper to investigate and identify the project environment factors that affect the requirements analysis phase in the Greek banking sector, through a series of five IT projects with distinct characteristics each, carried out by a POS management systems Support Team acting as part of Printec Group's Software R & D Division.

1.1. Background

The research paper focuses on the various issues faced in the requirements analysis during the service design, service transition and service operation execution of five large to medium-scale IT projects undertaken within the Greek banking sector. The primary deliverable of all projects was the deployment of a POS or Terminal Management System (TMS) within five Greek banks, three of them forming part of the four largest banking establishments within the Greek banking market. Services are the means of delivering value to customers by facilitating outcomes customers want to achieve, without the ownership of specific costs and risks [1]. Moreover, there is extensive reference on the various methodologies and practices employed towards client requirements identification, categorisation and dissemination of information to other corporate teams and the effect of this analysis on project deliverables. The paper also discusses the established practices which aided the overall process of requirements management. Four teams were highly involved in the IT projects undertaken. These consisted of the TMS implementation and support team, the TMS development team of Printec Group Software R&D Division and the Printec Greece e-Payments Department teams of POS application development and POS Helpdesk; a subsidiary of Printec Group. Teams from Printec Group reported to the Steering Committee and both had an individual appointed at a managerial role. The Steering Committee consisted of the Software R&D Division Director and the Group General Manager. The Printec Greece e-Payment Department POS Helpdesk and POS Development teams were headed by the Technical Supervisor and the Head of Software Engineers.

1.2. Project Management Plans

The aim of IT projects undertaken was to install or upgrade the TMS software for five banking institutions. An eighteen month period covered the entire duration of the five executed IT projects starting from May 2008 and lasting till November 2009, see **Table 1**. Notice that all project implementations refer to system upgrades except for Sigma Bank which resulted to a fresh installation of TMS at Printec Greece premises.

Printec Group's IT service provisioning to the five banking institutions can be further divided into areas such as POS management, merchant management, POS issue management, reports management and client specific business needs. In addition, the client's point of contact with Printec Group's internal software development department was the TMS support team. Therefore, the TMS implementation and support team was responsible for highlighting or escalating any issues or requirements that might arise regarding client needs. Soon it became apparent that healthy relationships between the client and the IT services supplier when maintained, through good professional practices such as regular communication updates on current issues faced, post-visit reporting on decisions reached, issue response and resolution or simply updates on a new product release, can result in higher client utility and improved practice in the requirements capture activity.

2. Project Management, Service Management and Business Analysis Frameworks for Requirements Analysis in the Greek Banking Sector

From the very beginning of the IT projects, it was realised that a defined period for requirements identification cycle would ensure that all client requirements would be recorded as part of the documentation practices already established within Printec Group. These would be taken into full consideration by all involved stakeholders for the release of the new TMS software [2]. Identification of business and user requirements was directly related to functional and non-functional requirements. Functional requirements relate to the scope of work or functionality the software must have whereas non-functional requirements refer to look and feel, usability, performance, security and maintainability and support requirements of the software [3].

Moreover, maintaining an up to date set of requirements gained significant priority and became a crucial aspect of applied IT project management practices. It was important that all stakeholders of the TMS support and software development teams had a perfectly aligned perception of what the client required so as to avoid a misconception of expressed client requirements.

2.1. Applied Requirements Analysis and Business Analysis Practices and Methodologies

The requirements analysis process refers mainly to recorded and accepted requirements that will form part of project deliverables. The acceptance stage can be con-

Table 1. Project management plans duration for five IT projects in the Greek banking sector.

Bank	Project Start Date	Project End Date	Service Delivery Duration (months)	TMS version
Beta Bank	5/5/2008	14/02/2009	9¼	Upgrade TMS6→TMS7
Gamma Bank	7/7/2008	18/12/2008	5	Upgrade TMS7→New version of TMS7
Delta Bank	7/11/2008-25/11/2008 & 20/05/2009-25/8/2009	25/8/2009	3¾	Upgrade TMS7→New version of TMS7
Sigma Bank	23/4/2009	11/05/2009	¾	Install TMS7
Omega Bank	25/5/2009	24/11/2009	6	Upgrade TMS6→TMS8

ducted under the auspices of a formal meeting with the client during which essential business requirements are discussed, recorded and timelines of service delivery provided to the client by senior management. Later on, the agreed requirements to be implemented and timelines should be communicated to all project stakeholders so that organisational efforts for the release of the new TMS software are aligned according to specifications. In effect the purpose of the requirements analysis process is to establish which requirements have been identified, the case of whether there needs to be provided clarification in regard to a requirement and finally accept the implementation of the necessary software development to release the new software product which would fully conform to client requirements. Three processes were identified throughout the requirements analysis process as the key ingredients to best practice. These are the Requirements Identification, Requirements Categorisation and Requirements Prioritisation. It was made clear that there was a set of requirements to which the client agreed upon a formal meeting with Printec's senior management and that there was flexibility in new requirements to be accepted after a new release had been scheduled and implemented in terms of establishing an independent project altogether to implement these new requests.

Firstly requirements capturing or identification procedures were established whereby client requirements were identified as follows:

1) Senior management buy-in for the project indicated strong commitment from Printec Group's side to the requirements analysis process;

2) Enquiries and incident logging through the Service Desk;

3) Frequent formal visits to the bank's site;

4) The use of a coherent vocabulary or common glossary among Printec Group and bank employees was instrumental towards an improved understanding of requirements identification;

5) A requirements identification period resulted to improved requirements capture and a better understanding of the software functionality the client was expecting to receive.

Secondly the requirements categorisation process refers to identifying the software component or module to which a requirement refers. There were two types of requirements in regard to resource allocation to tasks for their implementation. Those that could be handled by the support team and those that had to be escalated to the software development team. Resource allocation to tasks was managed through the issue management system. In addition, requirements categorisation can be carried out having in mind functional and non-functional requirements. Functional requirements describe the functionality of a system. These are sometimes known as software capabilities. On the other hand, non-functional requirements act to constrain the solution and can be referred to as constraints or quality requirements. In fact, non-functional requirements can be further classified according to whether they are performance requirements, maintainability requirements, security requirements or reliability requirements.

In addition to categorisation of requirements the established issue management system within Printec Group assisted in the prioritisation of what requirements required immediate implementation in the new software release and which ones could be implemented in a later software release. Within Printec Group the following points were considered important prioritisation criteria:

1) Importance of requirement to client satisfaction. In effect this is what the client expressed as necessary to sign off user acceptance testing and eventually project implementation;

2) Criticality of requirement in client production environment; This attributes to the severity of implications caused in the client production level if the requirement was not implemented;

3) Capability of re-tracing previously documented requirements for re-use;

4) Resource allocation to requirements tasks;

5) Cost of implementation per requirement.

Note that there are certain challenges associated with requirements prioritisation which need to be carefully considered as defined in BABOK® [4]:

- Non-negotiable demands whereby the client is unwilling to commit to any trade-offs and ranks all requirements as high priority;
- Unrealistic tradeoffs whereby the service provider's solution development team may intentionally or unintentionally try to influence the result of the prioritisation process by overestimating the requirements implementation complexity.

Subsection 2.2 refers to project management practices established for Beta Bank, Gamma Bank, Delta Bank and Omega Bank. In the Sigma Bank case, the installation of the TMS software was outsourced. Therefore TMS was deployed at the Printec Greece premises thus mitigating the execution of the requirements analysis process to the Printec Greece-e-Payments Department POS Helpdesk team which in turn collaborated with Sigma Bank - Cards Division officials. The mental mismatch model can be also considered for the Sigma Bank TMS deployment project, see Subsection 3.1.4, in a slightly different stakeholders' context but always indicating the necessity to cater for misconception of expressed client requirements between various parties and teams as shown in **Figure 1**.

2.2. Applied Project Management and Service Management Practices and Methodologies

The TMS implementation and support team performance was constantly measured through Key Performance Indicators (KPI) which indicated that most of the time all issues were resolved and whenever serious issues arose with high business impact these were escalated to the TMS development team to address the issue in software development terms. Some of the frequently monitored and reported KPIs include, issues closed at first call at the Service Desk, issues closed after second level support consultation, average time per call per analyst, issues escalated for third level support to TMS software development and POS software applications development teams. In effect, there was usually a small backlog of client issues relating to TMS installations which made it possible to counteract more effectively to newly reported issues.

Throughout the duration of the IT project the senior management style was consultative whereby decisions were taken by seeking the opinions and views of the teams prior to a decision being made [5]. However final decisions lied solely in the judgment of the Steering Committee. Furthermore, knowledge transfer on TMS usability matters from the development team to the support team was vital. This was conducted through formal and informal meetings, online material and communication, computer based training (CBT) and constant involvement to client issue resolution.

An established Service Desk operating according to ITIL specifications, was utilised to control and monitor TMS support performance levels. In particular, it was decided who would take responsibility of escalating issues to the development team, aligning client site TMS software items with Printec Group's TMS related configuration items (CI) and who would trigger and organise formal team meetings. Moreover, work delegation was essential in assigning tasks to resources by taking into consideration resource availability and the individual's expertise on the different functionality aspects of TMS. In this way incident response timeframes to the client would be minimised. Performance reporting to the Steering Committee was essential. This involves collection and distribution of performance information to project stakeholders [6].

The established flow of information and client requirements identification to escalation management required the TMS Support team members be at the heart of the process as depicted in **Figure 2**. Notice that the Steering Committee and Software Development team, at

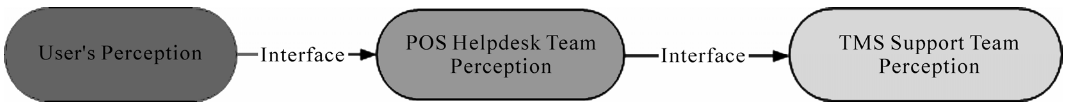


Figure 1. The mental model mismatch for the Sigma Bank-TMS deployment project.

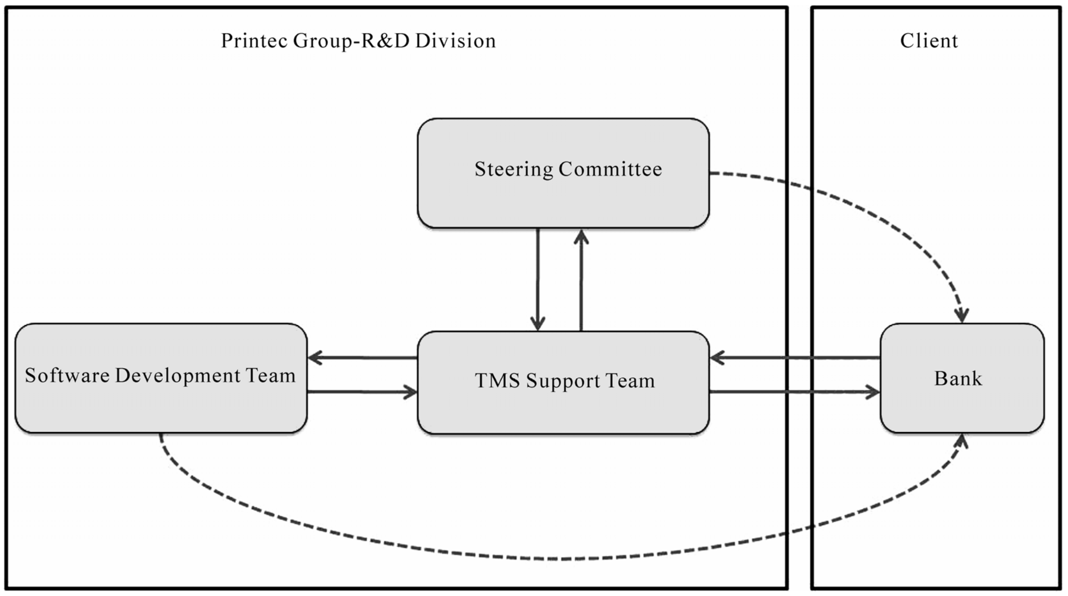


Figure 2. Printec Group-Software R & D Division communications mapping with clients.

exceptional cases, had to be involved in the IT support process with the client. This occurred due to the fact that certain requirements had to be clarified in technical terms to avoid ambiguity so as implementation and the end result was exactly what the client requested initially. In addition, this also occurred in situations when senior management had to engage in the decision making process with the client or even to schedule high profile meetings. In fact, decisions or agreements reached in collaboration with Printec Group's senior management would indicate higher commitment to the TMS project from the client side.

2.2.1. Established Service Desk

The implementation of the Service Desk for the TMS support team entailed the recording of landline incoming/outgoing telephone communications. This was achieved by monitoring and responding to all outstanding activities/requests/complaints, creating and maintaining a Known Error Database (KED) and reporting on a monthly basis to the Software R & D Division Director on TMS support productivity and efficiency.

The established Service Desk within the TMS support team served as a technique for capturing and recording client requirements

2.2.2. Issue Reporting and Statistics Provision

A suitable frequency of reporting and review was established, depending upon the importance of the review. Providing results in graphical form is useful for presenting management overviews on major areas of interest [7]. To provide a common service objective, it was important that all TMS stakeholders were aware of major issues, concerns, performance levels and achievements of the entire Software R & D Division and not just their team. **Table 2** below, shows the monthly performance statistics produced for the Director of the Software R & D Division by the owner of the TMS Support Service Desk.

3. Investigation of Environment Factors—Five Case Studies in the Greek Banking Sector

This chapter presents a discussion on each of the five IT projects the Printec Group TMS support team embarked on and unravels the factors that affected the requirements analysis phase which in turn resulted in delays or improvements to project timeframes. The case studies are presented in time sequence as they occurred throughout an eighteen month period.

The requirements analysis and business analysis (BABOK®) procedures implemented by the TMS support team have been previously described see Subsection 2.1. Project Management (PMBOK®) and IT Service Management (ITIL® v3) framework practices have been thoroughly described in Subsection 2.2.

3.1. Terminal Management System Deployment Projects for Five Banking Institutions within Greece

Each project presented in this section carries a different set of characteristics which distinguishes it from the rest in terms of size of client organisation, applied corporate

Table 2. Monthly performance statistics of the TMS support service desk.

	Total issues	% Change	Resolved issues	%
September 2008	25	0.00%	24	96.00%
October 2008	27	7.41%	24	88.89%
November 2008	30	10.00%	29	96.67%
December 2008	30	0.00%	28	93.33%
February 2009	29	–3.45%	28	96.55%
March 2009	28	–3.57%	26	92.86%
May 2009	27	–3.70%	27	100.00%
Jun-Jul 2009	13	–107.69%	13	100.00%
Aug - Sep 2009	19	31.58%	18	94.74%
Sep- Oct 2009	17	–11.76%	16	94.12%
Nov 2009	15	–13.33%	14	93.33%

IT policies, IT resources and human resources availability, project complexity, issue management, demand management, scope management, time management and financial management (project budget). The projects undertaken for the Greek banking sector are presented below in chronological order, see **Table 3**. It is clear that the project engaged with the largest banking institution, Beta Bank, faced the highest number of issues requiring resolution. In the ITIL context these can be classified as incidents or problems. In case problems arose the appropriate changes were requested and upon approval of the Change Advisory Board (CAB) a new TMS release was scheduled with a problem resolution patch. Moreover, the organisation criticality category describes the vitality and business operations impact each customer represents to Printec Group as a business customer. For example, **Table 3** shows that Beta Bank is considered to be a strategic customer of Printec Group and therefore any incidents or problems arising from any Beta Bank project require additional attention in accordance to the customer specific Service Level Agreement (SLA). Project complexity refers to the level of bureaucratic processes put in place by the customer, customer decision making time and processes on expression of customer requirements, time of response to service provider requests and in general elements that might materialise risks which will translate, as a consequence, in terms of project delays. Lastly, the number of tasks assigned per project was similar in all five case studies as shown in **Table 4**.

Note that throughout the duration of the five case studies a project manager was working in conjunction with the TMS implementation and support team so as both the service provider and the customer complied with the agreed project plan and project schedule. The assignment of tasks to individuals supported the need of task ownership. In this way individual responsibility for the completion of tasks on-time and within project scope was encouraged.

3.1.1. Beta Bank

The first project for the deployment of TMS at the Beta Bank premises, the largest banking institution within South-Eastern Europe, started in May 2008 and lasted till February 2009. With an approximate duration of nine months and thirty-six reported issues, see **Figure 3**, it was the largest and most complex project of all. High

Table 3. TMS deployment project characteristics for five banking institutions within Greece.

Bank	Completion Duration (months)	Project Issues	Organisational Criticality	Project Complexity	% of Tasks Completed
Beta Bank	9	36	High	High	100%
Gamma Bank	5	16	Medium	Medium	100%
Delta Bank	3¾	11	Medium	Medium	100%
Sigma Bank	¾	6	Low	Low	100%
Omega Bank	6	11	Medium	High	100%

Table 4. A standard TMS deployment project schedule.

Task	Completion Duration (days)	Owner
Project kick-off meeting	0	Software Support Engineers, Project Manager
Requirements Analysis	5	Software Support Engineers, Project Manager
Data migration from old to new system	15	Software Support Engineer 1
Setup of User Acceptance Testing (UAT) environment at client site	15	Software Support Engineer 2
Testing at client site	5	Software Support Engineer 2
UAT sign-off and software activation in production	2	Software Support Engineer 1, Project Manager

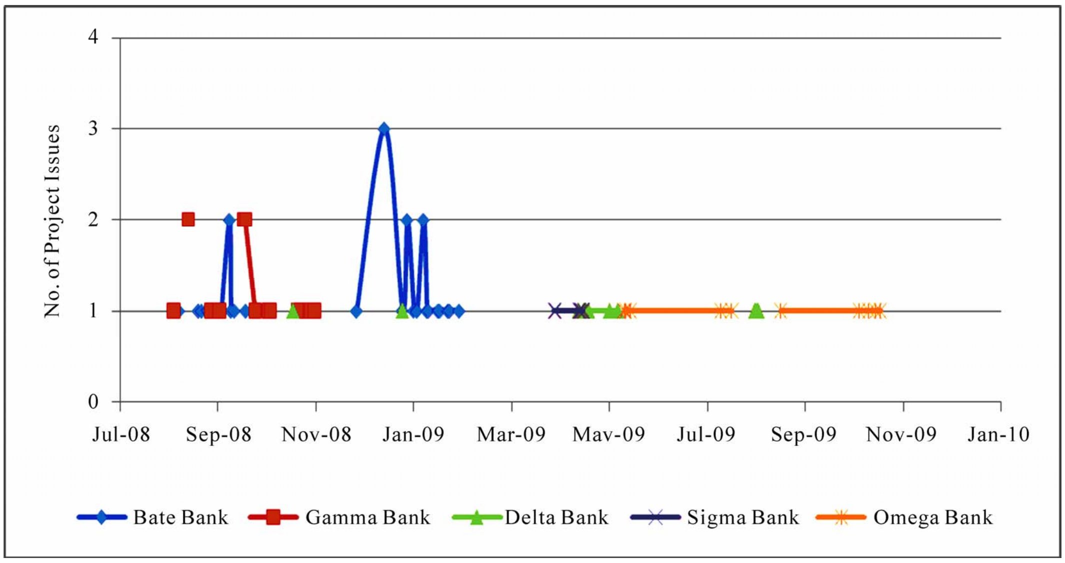


Figure 3. Recorded issue rate per day throughout the duration of all five projects.

project complexity resulting to project timeframe delays existed due to several reasons as described below:

1) High issue reporting rates, see **Table 3**, meant that TMS support and TMS Development team members had to first resolve the issues so as to progress through the project management plan;

2) Numerous change requests to project requirements and thus deliverables during the User Acceptance Testing stage. In general, software development at this stage entails high costs;

3) Numerous confirmation requests for undocumented TMS workflow which meant that Printec Group's Intellectual Property had to be preserved while satisfying client requests;

4) TMS administrators were constantly seeking reassurance on system management matters;

5) Constant requests for additional training provision;

6) The large size of corporate POS Helpdesk and POS Faults Departments resulted in scheduling requests for additional training sessions;

7) Beta Bank being a high-profile client meant that no trade-offs could be made during the requirements analysis phase. Without any ground to negotiate identified requirements implementation there is additional risk on the service supplier side during service delivery.

In addition, the prolonged requirements analysis period, at the beginning of the project, was beneficial in the sense that certain requirements were well defined and a good understanding of their implementation was acquired. However, the prolonged period of service design, service transition and service delivery in general, had a serious impact on project constraints e.g. extended project duration and contributed to higher project complexity as well.

3.1.2. Gamma Bank

The second project for the deployment of TMS at Gamma Bank premises, the third largest Greek banking institution within Greece, started in July 2008 and lasted till December 2008. With an approximate duration of five months and sixteen reported issues, see **Figure 3**, it was the third largest and complex project of all. Medium project complexity resulting to project timeframe delays existed due to the following reasons:

1) Established IT security corporate policies meant that several authorisations were required to conduct simple activities such as the installation of new software on UAT and production environments or the retrieval of a database backup file. Usually this resulted in task delays;

2) Whenever task delays occurred throughout the project lifecycle, the rate of visits increased so as to ensure that TMS deployment work was carried out as planned.

Gamma Bank having sustained a highly sophisticated and secure TMS IT environment on its premises, assisted in keeping the project complexity at a medium level compared to the rest of the projects even though high IT security levels resulted at some cases to project timeframe delays. As observed, in the case of Beta Bank, when nearing TMS upgrade dates the recorded issues to be resolved experienced an increased rate. As a consequence the Printec Group TMS Support team had to be highly responsive, during these periods, regarding requests communicated by the banking institution's management. In fact, this team behaviour had to be consistent within all client project environments.

3.1.3. Delta Bank

The third project for the deployment of TMS at Delta Bank premises, subsidiary to the largest worldwide financial institution based in the USA, started in November 2008 and lasted till August 2009 with a halting period of six months due to events caused by the recent global economic recession. With an approximate duration of three months, three weeks and eleven reported issues, see **Figure 3**, it was the fourth in scale project of all with medium complexity. Medium project complexity resulting to project timeframe delays existed due to certain reasons as described below:

1) Established IT security corporate policies meant that several authorisations were required to conduct simple activities such as the installation of new software on UAT and production environments or the retrieval of a database backup file. Usually this resulted in task delays;

2) Whenever task delays occurred throughout the project lifecycle, the rate of visits increased so as to ensure that TMS deployment work was carried out as planned;

3) The TMS deployment project timeframe of delivery was largely affected by the recent economic turmoil. Even though the active project duration was recorded as three months and three weeks; adding the inactive period to the project duration equals to nine and a half months.

The Delta Bank TMS environment IT setup shared a lot of similarities to that of Gamma Bank since both were seeking upgrades of older TMS7 systems to the most recent TMS7 software releases. Therefore, the major factors which affected requirements analysis and project completion are stated along the same lines.

3.1.4. Sigma Bank

The fourth project for the deployment of TMS on behalf of Sigma Bank at Printec Greece premises, a small banking institution based in Greece, started at the end of April 2009 and lasted till mid-May 2009. With an approximate duration of three weeks and only six reported issues, see **Figure 3**, in terms of scope, time and budget, as stated in PMBOK®, this was a successful project. Low project complexity resulted to the effective application of project management and service management practices and an improved project timeframe for reasons described below:

1) High control of outsourced TMS deployment service owned by Printec Greece e-Payments Department POS Helpdesk team;

2) A quick issue resolution process in collaboration with Printec Greece e-Payments Department;

3) IT security policies were set internally by Printec Group TMS support and Printec Greece e-Payments Department POS Helpdesk teams.

Sigma Bank on its own involved a fresh installation of TMS at the premises of Printec Greece; the Greek subsidiary of Printec Group. Requirements definition and management was conducted in an organised and concise manner, from the beginning of the project. The materialisation of a TMS6 system failure meant that reactive tasks had to be put in place for urgent issue resolution purposes. All tasks falling within the data migration and installation barriers were executed in a timely and highly responsive manner. Furthermore, good communications among project stakeholders throughout the duration of the project was executed. This involved communication of the TMS support team with counterparts in the Printec Greece e-Payments Department. The Technical Supervisor of the Printec Greece e-Payments Department POS Helpdesk team was appointed the TMS administrator. Regarding risk management techniques risk owners were appointed for the data migration, installation, maintenance and administration tasks. As a result emerging issues were resolved within reasonable timeframes. A request was made for risk control and monitoring purposes so that tasks were put in place to avoid future risk materialisation e.g. establish daily TMS system database backup plan.

3.1.5. Omega Bank

The fifth project for the deployment of TMS at Omega Bank premises, the Greek subsidiary of a French financial services Group, started in May 2009 and lasted till November 2009. With an approximate duration of six months and eleven reported issues, see **Figure 3**, it was the second largest and most complex project of all. High project complexity resulting to project timeframe delays existed due to several reasons as described below:

1) Low availability of IT resources;

2) Insufficient IT infrastructure capabilities;

3) Inexistent corporate IT security policy;

4) Lack of appointment of TMS administrator(s);

5) As a result additional TMS performance issues were recorded due to compliance failure of the client to minimum system specifications;

6) An increased rate of visits so as to ensure that TMS deployment work was carried out as planned meant that the project timeframe suffered additional delays;

7) Numerous training sessions had to be organised due to an apparent indifference of Omega Bank POS Helpdesk staff to acquire the knowledge necessary to proceed to service delivery;

8) The project was part of the first live deployment of new corporate TMS software product during which, the TMS support team underwent a knowledge transfer and knowledge acquisition period as part of the service tran-

sition process.

During the project initiation stage Omega Bank seemed to have many similarities to the Beta Bank TMS project environment. However, a known risk materialised over the course of the project which regarded low IT resources support capability from the client side. This meant that most of IT resources management had to be conducted by the Printec Group TMS Support team which increased the responsibility and accountability factors on the side of the service supplier. Although the initial project plan presented to Omega Bank did not include this kind of additional tasks the overall project duration was largely affected. Moreover, the inexistence of an appointed TMS administrator added to the already high complexity level of the project. However, due to the fact that the newly released TMS8 software was to upgrade the old TMS6 system project completion was of high priority and significance and therefore the efforts focused in enabling a smooth transition for the day-to-day Omega Bank POS management activities.

3.2. Identified Environment Factors Affecting Requirements Analysis

In Subsections 3.1.1 through 3.1.5, a series of IT projects has been presented each with its own distinct characteristics. The five project environments highlighted in terms of project management, service management and business analysis distinguish themselves though certain identified influential factors. These factors relate to three domains. Firstly, factors relating to *human behaviour and competencies* which entails individual task ownership, responsibility, accountability, competence and skills. Secondly, factors related to *policy and standards compliance* and lastly factors influenced by *IT systems architecture and IT resources availability*.

The total project duration recorded at the end of each project lifecycle indicated a correlation between the organisation criticality, see **Table 3**, and total project issues per project, see **Figure 4**, to form a similar pattern as shown in **Figure 5**.

Throughout the lifecycle of a project and from information recorded during project closure in the lessons learned documentation, certain environment factors were identified which carry a major influence on service delivery time and budget management as well as requirements analysis and implementation acceptance by the client. These factors are as follows:

1) IT security policies and standards applied, e.g. ISO/IEC 27000;

2) Corporate software deployment policy procedures on the client side;

3) Software documentation provision e.g. user manual,

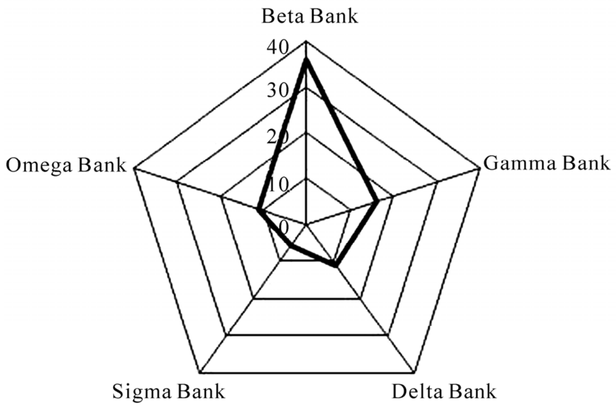


Figure 4. Total project issues per project depicted on a radar chart.

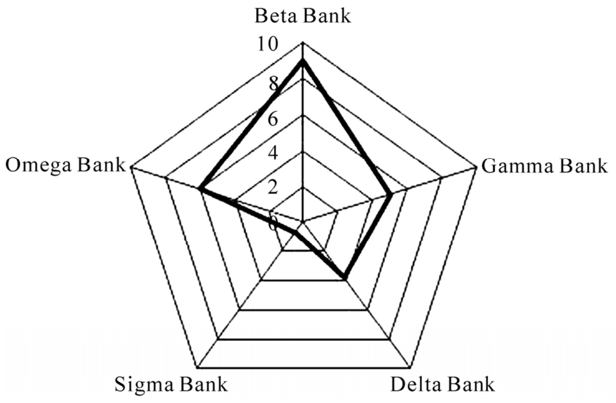


Figure 5. Total project duration (months) per project depicted on a radar chart.

operator manual, etc.;

4) Rate of software training sessions for system administrators and system users;

5) Rate of formal walkthroughs, meetings and visits at the client site;

6) Appropriate IT support team skills on the service provider side;

7) System administrator(s) attitude(s) in terms of new requirements requests made, commitment to software administration, comprehension skills of new software through training sessions and user acceptance testing sign-off willingness;

8) IT resources availability *i.e.* hardware, servers, etc.;

9) Senior management commitment from both, the service supplier and the client;

10) Selection of appropriate deployment period. For example, festive periods which result in high transaction rates, summer time when seasonal shops operate e.g. touristic shops, restaurants, hotels, should be carefully considered when making request for change for TMS;

11) Definition of project stakeholders' task ownership at the beginning of the project, for both the service supplier and the client, using techniques such as a RACI (Responsible-Accountable-Consulted-Informed) chart.

The aforementioned factors should not be assigned to a weighting system whereby each factor is categorised according to impact on project deliverables. The reason behind this being that throughout the duration of the five case studies presented wherever one or more of these factors was not satisfied there were bound to be user acceptance issues and system deployment delays. The question at hand is how a service supplier can satisfy as many of the factors stated previously, in order to fulfil client requirements and attain high client satisfaction.

Sigma Bank forms a distinct exception to the above identified factors due to the fact that it signifies an outsourcing IT project *i.e.* the TMS software was installed internally at Printec Greece premises. In this case, the control over the IT competency of POS management system was given exclusively to Printec Greece–POS Helpdesk team. Therefore, certain environment factors were considered unnecessary. For example, as mentioned earlier the Sigma Bank TMS7 installation was a result of an unexpected system failure and urgency emerged for the issue to be resolved as soon as possible. Therefore the factor regarding suitable yearly period deployment cannot be applied in this case since the system recovery and deployment processes commenced as soon as the system failure occurred.

4. Findings

A number of noteworthy findings are evident regarding the service provision of a Terminal Management System, developed from Printec Group–R&D Division, to five banking establishments operating within Greece. Moreover, a full cycle of requirements analysis has been described whereby requirements identification, categorisation and prioritisation processes combined with best practice in Project Management according to PMBOK®, IT Service Management according to ITIL® and Business Analysis according to BABOK® can be instrumental towards successful requirements analysis.

The identification of eleven factors that have potentially influenced the five project environments of Beta Bank, Gamma Bank, Delta Bank, Sigma Bank and Omega Bank indicates the necessity to cater for each one in order to fulfil successfully client requirements in the Greek banking sector. In other words, to increase project implementation success rates these factors should be considered thoroughly. If success is to be measured in terms of Scope, Time and Budget as stated in PMBOK®, then that means that projects should be governed by generally accepted best practice frameworks. For the five case studies presented in Section 3, the project completion rate and projects within scope rate was 100%. In general, these were caused by the project environment factors identified in Subsection 3.2. There was no case of forceful project closure even though on one occasion there was a halting period which lasted up to six months as a consequence of the effects of the current economic recession on Delta Bank. It is also notable, that the Sigma Bank project signified an outsourced IT competency to Printec Greece. The TMS deployment project was a success in project management, service management and requirements analysis terms but this does not necessarily mean that outsourcing is an option for any client of any size. Sigma Bank holds a relatively small corporate structure, the lowest number of POS devices in its TMS database and therefore the lowest number of requirements.

It has been shown that constantly employed requirements analysis tools, techniques and methodologies as well as issue management, time management, resources management, risk management, stakeholder management and service supplier Service Desk Operations reporting, assist greatly in the process of capturing client requirements in a timely fashion. The need of senior management buy-in in the process of requirements identification has been also stressed. This increases client confidence and maintains a healthy client-supplier relationship. Requirements categorisation has revealed that when prioritising requirements, influencing factors should be mainly customer centric.

Finally a thorough account of PMBOK®, ITIL®v3 and BABOK® methodologies has been given whereby scope management, time management, resources management, issue management, Service Desk, availability management, stakeholder management, risk management, business analysis and release management processes have been blended with requirements analysis tasks in order to maximise business benefits. As PMBOK®, ITIL®v3 and BABOK® methodologies are used to a certain extent, though limited within Greece, this research has indicated the significant opportunities presented for improved requirements analysis when the PMBOK®, ITIL®v3 and BABOK® frameworks are used in conjunction in IT project environments for the Greek banking sector.

5. Conclusions

This paper has presented the various environment factors with either negative or positive impact on the requirements analysis phase while undertaking five IT projects with differing environmental setups within the Greek banking sector. The essence of keeping healthy client-supplier relationships, appropriate IT and human resource provisioning by the customer side, appropriate visit frequency rates at the customer site, provision of high-quality software documentation from the service supplier side, training provision of system administrators and users from the service supplier, compliancy to corporate IT security and software deployment standards at the client site, setting service transition milestones in accordance to specific yearly periods, service supplier and client senior management commitment to project deliverables as well as the use of PMBOK®, ITIL® and BABOK® methodologies has been highlighted. Vital issue management tools and techniques have also been presented such as the importance of keeping an up-to-date set of recorded issues as part of Service Desk operations and issue resolution statistics reporting to the Director of Printec Group–Software R & D Division. Moreover, the effects of proper guidance in the requirements analysis process which entails the requirements identification, categorisation, prioritisation, implementation and release stages has been clearly indicated. The specific PMBOK®, ITIL® and BABOK® functions applied within Printec Group–Software R & D Division TMS support team have also been presented. Finally, the importance of defining services provision in an 'IT-enabled business' context has been discussed whereby the development and deployment of IT software products support corporate strategic envisioning, business vitality and viability through the achievement of specific business goals.

6. Acknowledgements

Special thanks to Printec Group of Companies and the Software R & D Division which contributed to a significant degree to the research presented. Special thanks to Dr. Fotis Moulatsiotis without whom the research would not have been realised. His impeccable judgment and proper guidance filled me with more determination to complete a thorough research. Many thanks to Alexander Poutos, Christos Michas and George Dimitropoulos for their contribution on IT service support matters and thanks to the POS Applications Development and POS Helpdesk teams of Printec Greece e-Payments Department. Many thanks also to the unknown referees. All contributions filled me with greater motivation and enthusiasm to conduct the research.

REFERENCES

[1] Office of Government Commerce, *ITIL® v3–IT Service Management Core Books*, TSO, 2007.

[2] Project Management Institute, "A Guide to the Project Management Body of Knowledge," 4th Edition, Project Management Institute, 2008.

[3] B. Hughes, R. Ireland, B. West, N. Smith and D. I. Shepherd, "Project Management for IT Related Projects," British Computer Society in Association with Biddles Ltd., London, 2004.

[4] IIBA, "A Guide to the Business Analysis Body of Knowledge (BABOK Guide)," International Institute of Business Analysis, 2009.

[5] P. Wheatcroft, "World Class IT Service Delivery," British Computer Society in Association with CAPDM, London, 2007.

[6] J. Robertson and S. Robertson, "Mastering the Requirements Process," 2nd Edition, Addison Wesley, Reading, 2006.

[7] E. Harrin, "Project Management in the Real World," British Computer Society in association with CAPDM 2007.

J. Software Engineering & Applications, 2010, 3, 869-874
doi:10.4236/jsea.2010.39101 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

Requirements Analysis: Evaluating KAOS Models

Faisal Almisned, Jeroen Keppens

King's College, London, UK.
Email: faisal.almisned@kcl.ac.uk, jeroen.keppens@kcl.ac.uk

ABSTRACT

Wigmore's charts and Bayesian networks are used to represent graphically the construction of arguments and to evaluate them. KAOS is a goal oriented requirements analysis method that enables the analysts to capture requirements through the realization of the business goals. However, KAOS does not have inbuilt mechanism for evaluating these goals and the inferring process. This paper proposes a method for evaluating KAOS models through the extension of Wigmore's model with features of Bayesian networks.

Keywords: *KAOS, Requirements Evaluation, Wigmore's Chart, Bayesian Networks*

1. Introduction

The alignment of requirements analysis to business goals and objectives is essential for the return of investment of any project. KAOS is a goal driven requirements analysis method that defines a goal tree with parent and sub goals. KAOS assumes that achieving all sub goals of a parent goal will guide to the achievement of the parent goal. The inferring process in KAOS is informal, due to the nature of deduction in KAOS, which is based on the assumption that the completion of sub goals leads decisively to the parent goal. However, there is no guarantee that the previous assumption is always valid. The lack of precise assessment for KAOS goals requires further consideration. Usually, in realty some sub goals does not lead to the parent goal due to some contextual knowledge that was not measured completely in KAOS representation. Another cause of the uncertainty of goals originates from the possibility of assigning multiple values to one goal rather than only two possible values (true or false), which is the only option taking into account in the current features of KAOS. For instance, if one of the sub goals was completed partially, there is no feature to measure the impact of this sub goal to the parent goal.

This paper takes into account the possibility of failures in achieving the ultimate goals in KAOS models. This paper will propose a new graphical representation model, which can absorb KAOS models to be represented through it. The new model enables analysts to provide measurable ultimate goals accompanied with probability to give analysts statistical results. These results will facilitate the evaluation process of the whole KAOS model. The new Model will formalize the inferring process to be mathematically valid.

2. KAOS

KAOS is a goal oriented requirements analysis method, developed by University of Oregon and university of Louvain. KAOS stands for Knowledge Acquisition in automated Specification [1]. The main advantage of KAOS over other requirements analysis methods, which are not part of the goal analysis family, is its ability to align requirements to business goals and objectives. This alignment increases the chances that the new development will add value to business.

KAOS focus on realizing and indicating the business goals, then specifying the requirements that infer to the business goals. "Each goal (except the leaves, the bottom goals) is refined as a collection of sub goals describing how the refined goal can be reached" [2]. The structure of the various connected requirements and goals is represented hierarchically in graphical notation in an upwards direction. The top goals are strategic objectives for the business. As low as the diagram level reaches as closer to the low level requirements. The root of the diagram is the ultimate business goals. Then, the analysts must identify the penultimate goals followed by the lower goals and so on. The previous step is recurring until the analysts reach the basic goals. The lower goals are linked with the parent goals through union. The union indicates that the completion of the lower goal successfully will definitely cause the completion of their parent goal. **Figure 1** shows an example of a simplified KAOS model. KAOS main focus is on the business requirements, disregarding if this requirement is part of the

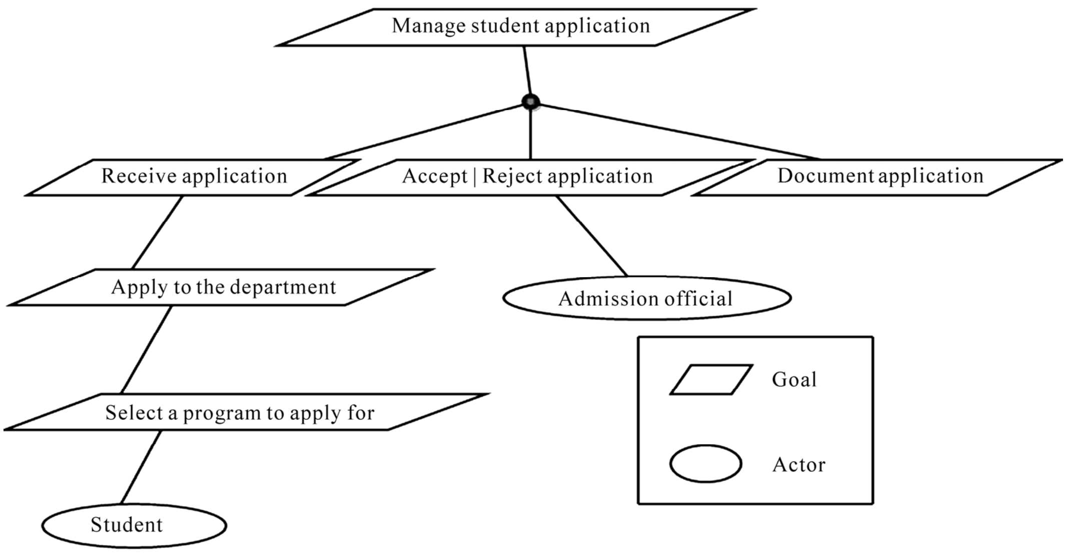


Figure 1. An example of KAOS.

computer system requirements or not. Each goal is accompanied with obstacles and the stakeholders involving in this goal. A limitation of KAOS is the lack of any inference evaluation capabilities. The achievement of sub goals does not imply the achievement of their parent goals in all cases. The next section presents a review of two candidate approaches to solve this issue.

3. Related Work

In this section, two graphical representation models will be studied as possible methods to evaluate KOAS models. The features of these approaches will be examined to check the suitability of them to enclose KAOS models.

3.1. Bayesian Networks

Bayesian Network (BNs) is a general statistical tool that can be applied to various applications. BNs are helpful to assess the weight or the influences of premises, to determine the strong inference links. [3] Bayesian Network is a graphical representation tool using symbols, numbers and arrows to enable analysts to reason logically far from doubt. It is an appropriate tool to gather and analyze evidences, in order to produce strong arguments. There are two components to construct BNs. First, nodes are representing the noticed evidential facts, propositions and variables. Second, arrow that connects between various nodes in the diagram. These arrows indicate the dependency probabilities. The value or the weight of each node is affected by the value of the nodes influencing this node and linked with it. The final conclusion of the network is affected by the probabilities of each proposition and inference. (See **Figure 2**)

Bayesian Network is a method to reason logically and rationally using probabilities. The simplest way to understand the goals of BNs is to think of a circumstance you need to "model a situation in which causality plays a role but where our understanding of what is actually going on is incomplete, so we need to describe things probabilistically" [4]. There, BNs allow analysts to compute the overall probability of the final conclusion. By, computing the probability of propositions connected directly, then the higher connections, then the higher and so on. The benefits from BNs are obvious in the prediction of outcomes in doubtful cases. Also, the benefits are apparent in the detection of the causes of certain results. The influencing relations are not decisive but probabilistic; the precise probability is assigned for each node and relation. BNs are a Directed Acyclic Graph. BNs are constructed from nodes and directed links. Arrows that connect various propositions are accompanied with the probabilistic information required to define the probability distribution all over the network. To achieve that, initial probability value should be assigned to the nodes with no earlier nodes. Then, calculate the provisional probability for the

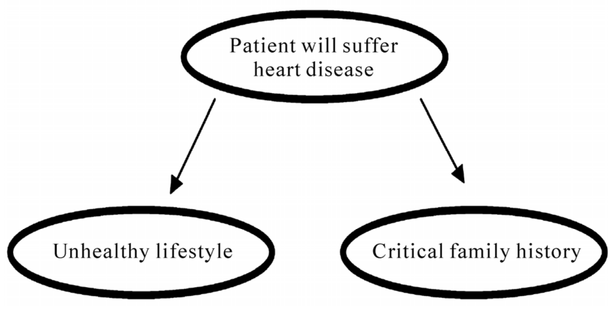


Figure 2. Simple bayesian network.

rest of the nodes and for all possible combinations of nodes and their antecedents. BNs permit the computation of the provisional probabilities of every node, bearing in mind that the value of some of the nodes has been specified before that computation took place. The diagrams' direction of Bayesian Network is downwards. In brief, the strength of the final argument is affected by the probability calculations of the supporting evidences and facts. The connections in the network represent the direct inference probabilities. The structure of the network illustrates the probabilistic dependency between various variables in a case. Each node is accompanied with a conditional probabilistic table of that node. The mixture of values for the nodes' ancestors will be provided. [5].The main incompatibility between Bayesian Networks and KAOS modeling is the fact that the direction of BNs is downwards which contradicts with the deduction process of KAOS. However, the probabilities feature is an important aspect to be added to the evaluation process.

3.2. Wigmore's Chart

Wigmore's chart (WC) was created by John Henry Wigmore (1913) to help lawyers. [6] Wigmore's chart acts as a legal reasoning diagramming method. Wigmore's chart considered as an argument diagramming techniques to demonstrate the structure of reasoning and inferring for an argument in a legal case. The diagram as a whole identifies the logic, structure and grounds behind the reasoning of arguments in legal cases. WC is a tool which enables the creation of arguments followed by the examination of those arguments, then the recreation of those arguments. WC is valuable in cases surrounded with doubt and uncertainty. In order to create WC, analysts of legal cases must identify the connections in all steps of the arguments. Then, the analysts should break-down the argument into propositions and facts. After that, the analysts should connect these facts and propositions together towards inferring the final conclusion of that argument. The chart method of Wigmore has a number of symbols to represent the different types of propositions and evidences. These symbols are connected with arrows to specify the direction, influence and weight of the inference. The final conclusions of the chart illustrate the logical deduction of the propositions and facts that assemble the inference. One of the main characteristics of WC is the production of key lists. The key list contains a list of all propositions, facts, evidences and assumptions, which are used to build the final conclusion of the arguments presented. In addition, inference maps show the gathering and linking process of evidences, this validates the argument construction procedure. The chart direction is upwards from facts to assumptions. The chart contains symbols, numbers and arrows only, but, will be accompanied with a key list clarifying the statement of each proposition or evidence (see **Figure 1**). There are five main symbols required for the construction of the Chart Method of Wigmore according to Schum [7] (See **Figure 3**).

Wigmore's chat properties can be used to evaluate the deduction process of KAOS models. But, the lack of measurable results affects the reliability of the evaluation process of KAOS models.

3.3. Comparison

Bayesian networks and Wigmore's chart have valuable features, which can aid the needed evaluation of KAOS models. However, their weakness does not provide a sufficient method for evaluation. The following table compare the two models.

Bayesian Networks	Wigmore Chart
Based on statistics, using probabilities calculation for premises and relations	Based on the natural logic of rea-soning. In addition to the skills and knowledge of the chart creators
The network direction is downwards	The chart direction is upwards
Not extendable notations, BN is a Directed Acyclic Graph	Extendable notations, richer semantics and it has some understanding of what it represents
Applicable to wide range of domains, used in various applications	Designed for law domain, but can be applied to other domains only if it can be extended
Produce supportive probabilistic arguments for the final conclusion	Enable the production of argument in favour and disfavour of the desired outcome
More Complex generation	Less complex generation
Top down approach	Enable Top down and Bottom up approaches
Measurable results, not decisive	Either for or against the intended outcome
The perspective of the creators does not play any role in the outcome of the network	The Chart Method of Wigmore allows the occurrence of multiple evaluations and considerations of same evidences in legal cases, from various perspectives.
The information flow from the basic fact or variable to the final outcome or goal	The information flow from the final outcome or goal to the basic fact or variable

As shown in the earlier comparison, the need to combine selected features from these two approaches could prove to be beneficial in terms of producing valuable method to evaluate KAOS models, as explained in the next section.

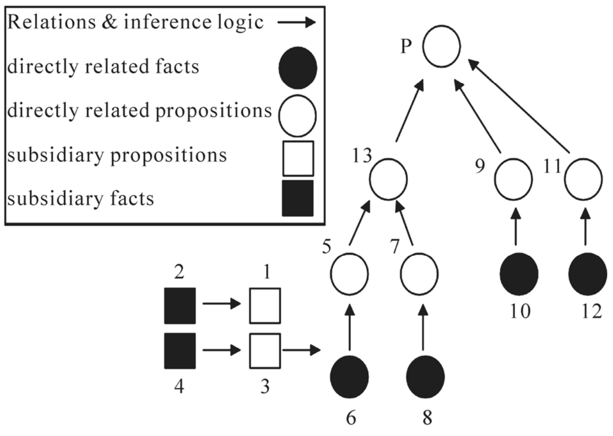


Figure 3. An example of wigmore chart.

4. Extending Wigmore's Chart

Bayesian networks and Wigmore's chart have number of practical and valuable features. The integration of some of these features will offer a model with superior capabilities and usage. The offered model will encapsulate several characteristics from both earlier models that do not contrast with each other. The extended model has to capture the properties, which are compatible with each other. This will allow the production of useful model, which facilitates the graphical representation of various tasks.

The suggested model extracts most of its properties from the chart method of Wigmore with the inclusion of one property of Bayesian networks and other external aspects. The model has to include additional aspects in order to address the gaps, which are not fulfilled completely by BNs and WC. The new model has several features. First, it enables both Top down and Bottom up Approaches, in order to facilitate the generation of models starting from the basic premises or starting from the desired conclusion. Second, the new model allows the production of measurable results to provide more accurate and reliable representation, through the introduction of probabilistic calculations. The third feature states that the new model should be extendable to be applicable to various domains. This is related to the notations of the models and the observation of contextual knowledge. Fourth, the new model eases the creation of representation supporting the desired goal, and against the desired goals. Finally, the new model will eliminate the complexity and ambiguity raised from representing the multilayered nature of cases or similar repetitive patterns. This multilayered nature could cause high complexity as stated by Hepler "If all these features are represented in one diagram, the result can be messy and hard to interpret" [8]. Another cause of complexity is the reappearance of a similar pattern of evidences and relations between facts and propositions, within the same case or in similar cases. And it would be "wasteful to model these all individually" [8]. It allows any network to contain an instance of another network without showing the detailed structure until requested. Moreover, it authorizes the creation of general networks that contains repeated patterns of evidences and relations, which can be reused after few amendments to customize the structure to the current case. This feature can be represented in the diagram as a special symbol. This model aims to simplify the creation of probabilistic graphical models and to convert the presentation into more efficient and understandable form.

There are six main symbols required for the construction of the new model. The five foremost symbols are derived from Wigmore's chart in a generalized manner to make them applicable to various domains. The last symbol is to represent the new feature of representing a repeated pattern or inclusive diagram. First, white circles are representing the directly related propositions or goals. Second, the black circle is a symbol of the directly related facts. Third, white squares stand for the subsidiary propositions. Fourth, the Black squares, which corresponds to the subsidiary facts. Fifth, arrows are showing the flow of relations between propositions and facts. Arrows are used to clarify the inference logic or flow in the arguments. Finally, the black rectangle illustrates the presence of repetitive pattern or another inclusive diagram. The number inside the rectangle will refer to the final conclusion of the repeated pattern or inclusive diagram. The diagram direction is upwards. Every symbol will be accompanied with the probability calculation function, which will calculate the provisional probability of each node based on the probabilities of the precedent directly connected nodes. The black circles and squares will be assigned with initial probability values.

There are a series of sequential steps to construct the new model representation. First, the analysts should start by identifying the ultimate goal from the analysis. Second, the analysis team should realize and assign the final conclusion of the model usage, the penultimate propositions which support the final conclusion and the middle propositions that support the higher propositions. The previous step could be repeated recursively. Third, the analysis team have to define the provisional facts and evidences that support all of the propositions in the chart. This can happen by indicating the scenario behind the construction for or against the goal of the analysis team in this case. Fourth, the analysts have to list all key premises and inference links to simplify the construction process. Fifth, analysts should commit to the appropriate construction of the model, by using the accurate symbols and right features. Numbers will be assigned to each symbol indicating the correct proposition or fact from the key list. Finally, after the existence of real arguments, the evaluation process should start. The analysis team should assess the arguments and evidences behind them. The analysts have to assign the initial probability values then calculate the provisional probabilities for the whole diagram. Afterwards, the joint probability for the whole diagram must be calculated, according to the probability computations rules. By this, the evaluation process could be emphasized. This will help to generate measurable outcomes to solve various issues. **Figure 4** presents the usage of the new models symbols. The next section will provide a glimpse about the significant of using our new model.

5. Evaluation KAOS

This section will show how the new extended model could be used to evaluate KAOS. The extended model will enclose KAOS goals and provide a measurable evaluation of the possibility of achieving the final outcome. **Figure 5** shows a basic KAOS model with three goals.

The constructed KAOS model could be evaluated by transferring the current goals and requirements in this KAOS model to the new models' graphical representation. This step is quite simple. The new model is similarly upwards. Each goal will be in the same position in the diagram as it was. The direct goals and requirements are represented as white and black circles. All nodes in KAOS tree will be represented as circles, the basic requirements with no earlier nodes are black and the goals are white. The accessory goals and requirements correspond to white and black squares. In the new model, analysts will be allowed to represent partially related goals and their requirements as squares, the basic related requirements are black and the related goals are white. Unions inside the KAOS model can be represented as arrows in the model. The constraints within KAOS model can be represented as a rectangle, which can symbolize the contextual knowledge or any compound model that involves repeated pattern or another model structure.

Figure 6 shows how to enclose the previous KAOS model into the new model. Then, the initial probability values have to be assigned to the nodes in the diagram. After that, analysts have to calculate and assign the provisional probability to all goals. These provisional probabilities will be produced by computation functions assigned with the inference process, which will calculate the provisional probability of each node based on the probabilities of the precedent directly connected nodes. This function should be following the acknowledged probability rules.

6. Conclusions

This paper proposed an extension of Wigmore's chart model, intended for evaluating the inference process among goals in KAOS models. Additionally, it provided a mechanism to measure the possibility of achieving a parent goal if its sub goals are achieved.

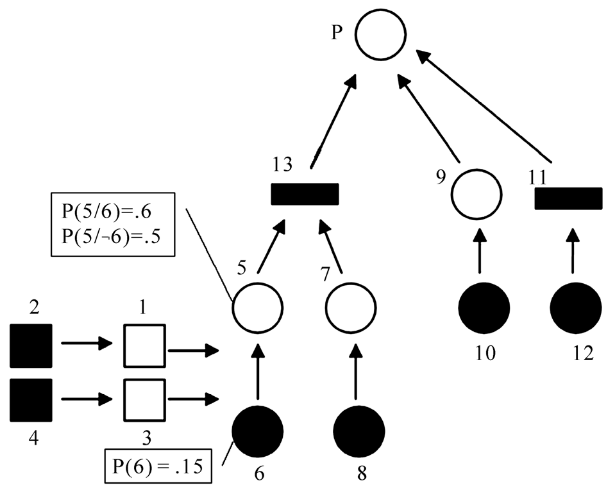


Figure 4. An example of the new model with sample probabilities.

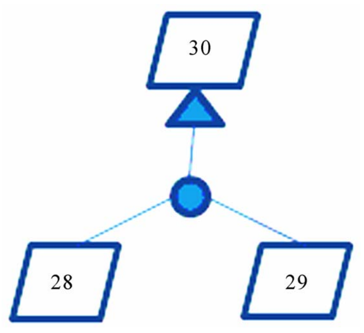


Figure 5. A basic KAOS model.

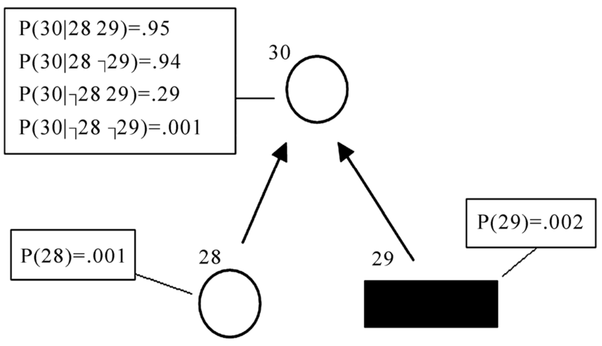


Figure 6. An example of the new model with the accompanied provisional probabilities.

Both Wigmore's chart and Bayesian networks were reviewed before an extended Wigmore's chart could be proposed. The new model provides a mathematical evaluation of KAOS, increasing the chances of constructing the right model. The new model presents a method for producing measurable results of the overall goals.

The main obstacle of the proposed evaluation approach is that it is not always feasible to know and assign the possibility of the inference from the leaves to the parent node. The proposed model suggested the use of a new separate model rather than extending KAOS model. This is to avoid adding complexity to KAOS models, in addition to the standardization grounds.

This work can be extended by building on the mathematical properties of the extended Wigmore's chart and by identifying advanced means for assigning the initial probability values.

REFERENCES

[1] A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal-Directed Requirements Acquisition," *Science of Computer Programming*, Vol. 20, No. 1-2, April 1993, pp. 3-50.

[2] Respect IT, A KAOS Tutorial, Objectiver, 2007.

[3] F. Taroni, C. Aitken, P. Garbolino and A. Biedermann, "Bayesian Networks and Probabilistic Inference in Forensic Science," John Wiley and Sons, Chichester, 2006.

[4] E. Charniak, "Bayesian Networks without Tears," *AI Magazine*, Vol. 12, No. 4, 1991, pp. 50-63.

[5] R. Neapolitan, "Learning Bayesian Networks," Prentice Hall, New Jersey, 2004.

[6] C. Reed, D. Walton and F. Macagno, "Argument Diagramming in Logic, Law and Artificial Intelligence," *The Knowledge Engineering Review*, Vol. 22, No. 1, 2007, pp. 87-109.

[7] D. A. Schum, "A Wigmorean Interpretation of the Evaluation of a Complicated Pattern of Evidence," Technical Report, 2005. http://tinyurl.com/2a3elq

[8] A. Hepler, P. Dawid and V. Leucari, "Object Oriented Graphical Representations of Complex Patterns of Evidence," *Law, Probability and Risk*, Vol. 6, No. 1-4, 2007, pp. 275-293.

J. Software Engineering & Applications, **2010, 3, 875-881**
doi:10.4236/jsea.2010.39102 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

Model-Driven Derivation of Domain Functional Requirements from Use Cases

Jianmei Guo[1], Zheying Zhang[2], Yinglin Wang[1]

[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China; [2]Department of Computer Sciences, University of Tampere, Kanslerinrinne, Finland.
Email: {guojianmei, ylwang}@sjtu.edu.cn, Zheying.Zhang@cs.uta.fi

ABSTRACT

Domain analysis is essential to core assets development in software product line engineering. Most existing approaches, however, depend on domain experts' experience to analyze the commonality and variability of systems in a domain, which remains a manual and intensive process. This paper addresses the issue by proposing a model-driven approach to automating the domain requirements derivation process. The paper focuses on the match between the use cases of existing individual products and the domain functional requirements of a product line. By introducing a set of linguistic description dimensions to differentiate the sub-variations in a use case, the use case template is extended to model variability. To this end, a transformation process is formulated to sustain and deduce the information in use cases, and to match it to domain functional requirements. This paper also presents a prototype which implements the derivation as a model transformation described in a graphical model transformation language MOLA. This approach complements existing domain analysis techniques with less manual operation cost and more efficiency by automating the domain functional requirements development.

Keywords: *Software Product Lines, Domain Analysis, Model Transformation, Use Cases, Functional Requirements*

1. Introduction

Software product line engineering (SPLE) has emerged as one of the most promising software development paradigms for production of a set of closely related products. It reduces development costs, shortens time to market, improves product quality, and helps to achieve greater market agility [1,2]. Some organizations make a transition from conventional single-system engineering to SPLE in order to enable mass customization and maintain their market presence. They systematically reuse legacy systems and existing products which embody their domain expertise to develop the core assets base of product lines [3,4].

Domain analysis is a process by which information used in developing software systems is identified, captured, and organized with the purpose of making it reusable when creating new systems [5]. Its essential activeties include analysis of commonalities and variabilities from the similar existing products and elicitation of domain requirements. Many domain analysis techniques [6-10] have been used to identify and document the commonalities and variabilities of systems constituting the product line. These techniques, however, mostly depend on domain experts' experience [2,11,12], and lack automated support [2,4,11]. When existing products have a significant amount of commonalities and also consistent differences among them, it is possible to extract the product line from them. If domain requirements, together with other systems artifacts, can be automatically reasoned and extracted from requirements of existing products, the amount of effort will be reduced significantly.

In this paper, we focus on functional requirements, and propose a model-driven approach to deriving domain functional requirements (DFRs) from use cases of existing products. Use cases are a widely used technique for requirements specification, but there is no generally accepted formalism to explicitly represent the variations of scenarios in a use case [13]. Using the metamodeling [14] and the model transformation techniques [15,16], the use case template [17] is adapted for variability modeling in order to derive DFRs for product lines. By introducing a set of linguistic description dimensions into use cases to differentiate the sub-variations, the use case template is extended to model the variability. Further, we analyze the correlation between the tailored use cases and the DFRs, and present a model-driven framework to derive DFRs from use cases. We also presents a prototype which implements the derivation as a model transformation described in a graphical model transformation language MOLA [18]. Our approach reduces the manual operation cost and complements existing domain analysis techniques by introducing an automated process of commonality and variability analysis.

The remainder of this paper is organized as follows. Section 2 defines the metamodels of use cases and DFRs and analyzes their correlation. Section 3 proposes a model-driven framework for DFRs derivation and presents a model transformation definition in MOLA. Section 4 discusses related work. Section 5 concludes this paper with future work.

2. The Underlying Metamodels

2.1. Use Cases

Use cases have been a means to understand, specify, and analyze user requirements that is rather often used [19]. They are widely adopted to capture functional requirements from an outside or users point of view. A use case describes the actions of an actor when following a certain task while interacting with the system to be described. It also describes a system's behavior as it responds to a request that originates from outside of the system. Use cases are often recorded by following a template. Although there are no standard use case templates, most of them describe more or less the same issues, e.g., the system to be described, the use cases within the system, the actors outside the system, and the relationships between actors and use cases or in between use cases [20].

One of the most commonly used use case templates is suggested by Cockburn [21]. According to Cockburn's template, a use case is described with its name, goal in context, scope, level, trigger, pre- and postconditions, main success scenario, extensions, sub-variations, and other characteristics [17]. The main success scenario describes what the use case actually does. It is the main part in a use case description, and often described as a sequence of steps or several alternatives to steps, such as extensions and sub-variations. Extensions specify changes in the course of execution of the main success scenario, and sub-variations give the further details of a step's manner or mode that will cause eventually bifurcation in the scenario. The main success scenario, together with extensions and sub-variations, describes a use case behavior, and also implies a set of fine-grained functional requirements. As is shown in **Figure 1**, the step is a basic object to capture a use case behavior. It has attributes such as *step #*, *name*, *description*, *use case name*, *actor*, and *trigger*. The attributes *step #* and *name* can be used to identify a step. A step could have *sub-variations*, and can be extended to another (set of) step(s) based on a given condition. Instances of *step* form the main success scenario.

To derive DFRs from use cases, common and variable requirements have to be identified and analyzes. The above use case description includes the specifications of sub-variations and extensions, but still lacks the formalism to model variability and to support domain analysis. In order to add the formalism to support variability modeling for the product line, we extend the existing use case metamodel by adding the attribute *ProductSite* and a set of *dimensions* for the steps described in the main success scenario, as shown in **Figure 1**. The attribute "ProductSite" records the owner of the step, which clarifies the existing product to which the step belongs, and help to analyze the commonality and variability in the course of defining the product line scope. The dimensions structure the specification of the sub-variations for the steps. They, are deduced according to the linguistic characteristics of functional requirements [4] and present the different perspectives of sub-variations. The dimensions include *agentive*, *attributive*, *locational*, *temporal*, *process*, and *purpose*.

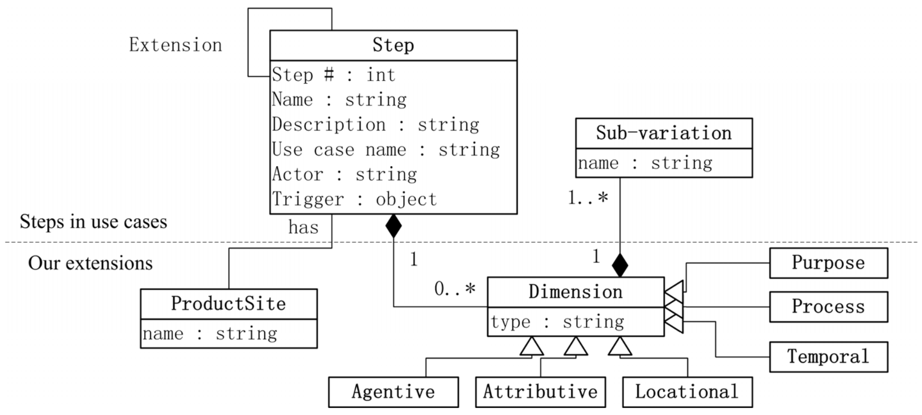


Figure 1. The extended use case metamodel.

Agentive defines the agent(s) whose activities will bring about the state of affairs implied by the step, e.g., "{author}$_{\text{Agentive}}$ submits an article". Attributive defines the attributes of the agentive or of the object associated with the action implied by the step, e.g., "submit a {research, review}$_{\text{Attributive}}$ article". Locational defines the spatial location(s) where the activity implied by the step is supposed to take place, e.g., "submit an article {at office, at home}$_{\text{Locational}}$". Temporal defines the duration or frequency of the activity implied by the step, e.g., "submit an article {every week, every month}$_{\text{Temporal}}$". Process defines the instrument, the means and the manner by which the activity is performed, e.g., "submit an article by {email, submission system}$_{\text{Process}}$". Purpose defines the purpose that the agentive carries out the activity or that the objective is affected by the activity implied by the step, e.g., "submit an article for {propagating the knowledge}$_{\text{Purpose}}$".

The extended use case metamodel mainly involves the steps, the sub-variations and the extensions because these elements from Cockburn's template have the direct relationship with functional requirements. It also models the variability through introducing a set of linguistic description dimensions to differentiate the sub-variations. The product site of each step is added for further analyzing the commonality and variability of DFRs.

2.2. Domain Functional Requirements

DFRs specify the common and variable requirements for all foreseeable applications of the product line. According to the Orthogonal Variability Model (OVM) [2], we define the variability model of DFRs in **Figure 2**.

Besides the essential attributes of a DFR, *i.e. name* and *description*, a DFR shall document its variability. Variability comprises the *variability subject* and the *variability object*. A variability subject is a variable item or a variable property of such an item [2]. If a DFR or its property has the tendency to change, it is a variability subject. The property with tendency to change can be further defined as a *variation point*. A DFR could have one or more variation points. In general, a variation point of a DFR expresses a variable semantic concern of the DFR. To present different semantic concerns of a DFR, we also define a set of types for the variation points of DFRs according to the linguistic characteristics of functional requirements [4], *i.e. agentive*, *attributive*, *locational*, *temporal*, *process*, and *purpose*.

A variability object is a particular instance of a variability subject [2]. It is represented as a variant. A variant represents a single option of the semantic concern that the variation point expresses, and a variation point may have one or more variants.

A DFR could come from one or more product sites in a product line. Such product sites can be used to calculate the *CV ratio* (Commonality/Variability ratio). A CV ratio records the frequency a DFR appears in a domain. Engineers can use the CV ratios to decide whether a DFR is common or optional. Generally, a DFR is common if it has a CV ratio "100%".

A DFR constraint documents a relationship between two DFRs, between a variant and a DFR, or between two variants. There are two types of relationship, *i.e. requires* and *excludes*. Each DFR constraint has a *domain* and a *range*. A domain of a constraint is the constraint's subject that possesses the constraint relation; a range of a constraint is the constraint's object that is affected by the constraint relation. For example, a constraint "A requires B" expresses the constraint relation "requires" between its domain "A" and its range "B". A domain or a range also has its "name" and its "type".

2.3. Correlation between Use Cases and Domain Functional Requirements

Steps presented in our prior metamodel describe the

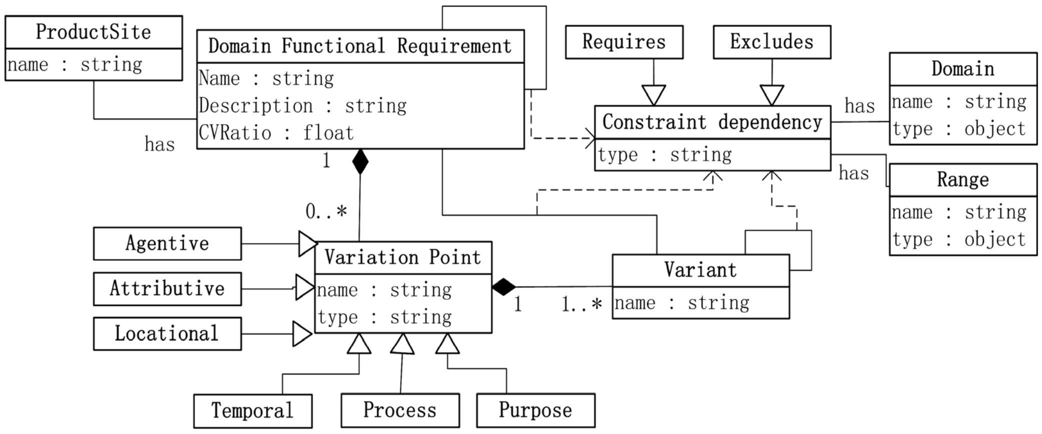


Figure 2. The domain functional requirement metamodel.

fine-gained functional requirements of a use case. The same requirements can be specified as a DFR for a product line. Since different products could have the same, the similar, or the different functional requirements, the DFRs can be obtained by merging the same and the similar functional requirements and distinguishing the variable ones. We conclude the correlation between use cases and DFRs as follows, which grounds the model transformation for DFR derivation.

2.3.1. Identifying Requirements

Steps with different ProductSite value is mapped into a DFR if their identity has the same value, *i.e.*, the same attribute *name*. Then, the sub-variations of a step are mapped into the variants of the DFR according to their same names. The dimensions of the sub-variations are also mapped into the variation points of the DFR according to their same types.

2.3.2. Analyzing Commonality and Variability

The commonality and variability can be analyzed by calculating the CV ratio. The productSite of the same step, together with the total number of products of the product line, forms the input of the calculation.

2.3.3. Identifying the Constraints

The relationships between and within the use cases shall be mapped correctly into the constraints of DFRs in terms of the same domain and the same range. Some constraints are straightforward and can be identified easily. For example, the extensions and the triggers of the steps can be mapped into the "requires" constraints of DFRs, *i.e.* "a trigger of a step" can be represented as "the step requires the trigger". The "precondition" of a use case can be mapped into the "requires" constrains of the DRF derived from the step numbered Step 1 in the use case. However, some more complex relationships in use cases are not identified in this paper. For example, according to Cockburn's template, a step also represents another use case (subordinate use case). Thus, a complete use case could have different "levels". These levels form the hierarchical relationship between the steps. In addition, the conditional relationship and the sequential relationship of the steps are also simplified. These complex relationships will be explored in our future work.

3. The Derivation Process

3.1. Framework

Based on the proposed metamodels and their correlation, it is feasible to derive the DFRs from the use cases of a set of closely related products in the same domain. We propose a model-driven framework for DFRs derivation from use cases in **Figure 3**.

The figure shows the scenario of model transformation from multiple source models (use cases) into a target model (DFR model). Both the source model and the target model are instantiated from their respective metamodels, *i.e.*, the use case metamodel and the DFR metamodel. A model transformation definition is defined based on these metamodel specifications. The transformation definition is executed on concrete models by a transformation engine.

3.2. Model Transformation Process

We present a prototype that implements the derivation of DFRs as a model transformation described in a graphical model transformation language MOLA [18]. MOLA provides an easy readable graphical transformation language by combining traditional structured programming in a graphical form with pattern-based rules. It clearly distinguishes what is from source models, what is from target models, and what is the mapping association between them. It is suitable for representing loops, which makes the time complexity analysis clearer.

A *program* in MOLA is a sequence of statements. A *statement* is a graphical area delimited by a rounded rectangle. The *statement sequence* is shown by dashed arrows. Thus, a MOLA program actually is a sort of a "structured flowchart". The simplest kind of statement is a *rule* that performs an elementary transformation of instances. A rule contains a pattern that is a set of elements representing class and association instances (links) and is built in accordance with the source metamodel. A rule has also the *action* that specifies new class instances to be built, instances to be deleted, association instances to be built or deleted, and the modified attribute values. The

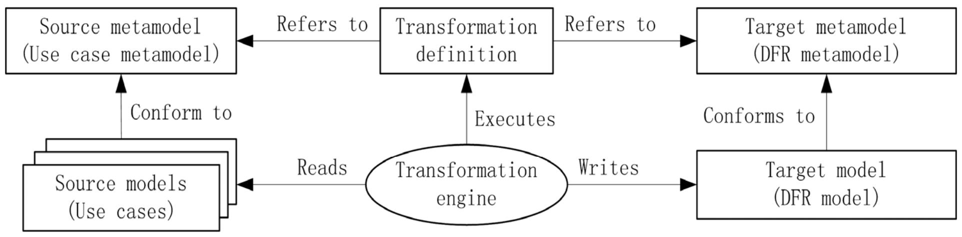

Source metamodel (Use case metamodel)
Refers to
Transformation definition
Refers to
Target metamodel (DFR metamodel)
Conform to
Executes
Conforms to
Source models (Use cases)
Reads
Transformation engine
Writes
Target model (DFR model)

Figure 3. Model-driven framework for DFRs derivation.

semantics of a rule is standard, *i.e.*, locating a pattern instance in the source model and apply the actions.

The new elements, instances, and links, are shown with *dotted* lines. The *mapping associations* prefix the association name by the "#" character. The mapping associations link instances corresponding to different metamodels; they typically set the context for next subordinate transformations and trace instances between source models and target models in the model transformation.

The most important statement type in MOLA is the *loop*. Graphically a loop is a rectangular frame, containing a sequence of statements. This sequence starts with a special *loop head* statement. The loop head is also a patter but with the *loop variable* highlighted (by a *bold* frame). The *reference* notation prefixes an instance name by the "@" character to show that the same instance selected by the loop head is used. The semantics of a loop is natural, *i.e.*, performing the loop for any loop variable instance which satisfies the conditions specified by the pattern.

As is shown in **Figure 4**, the model transformation

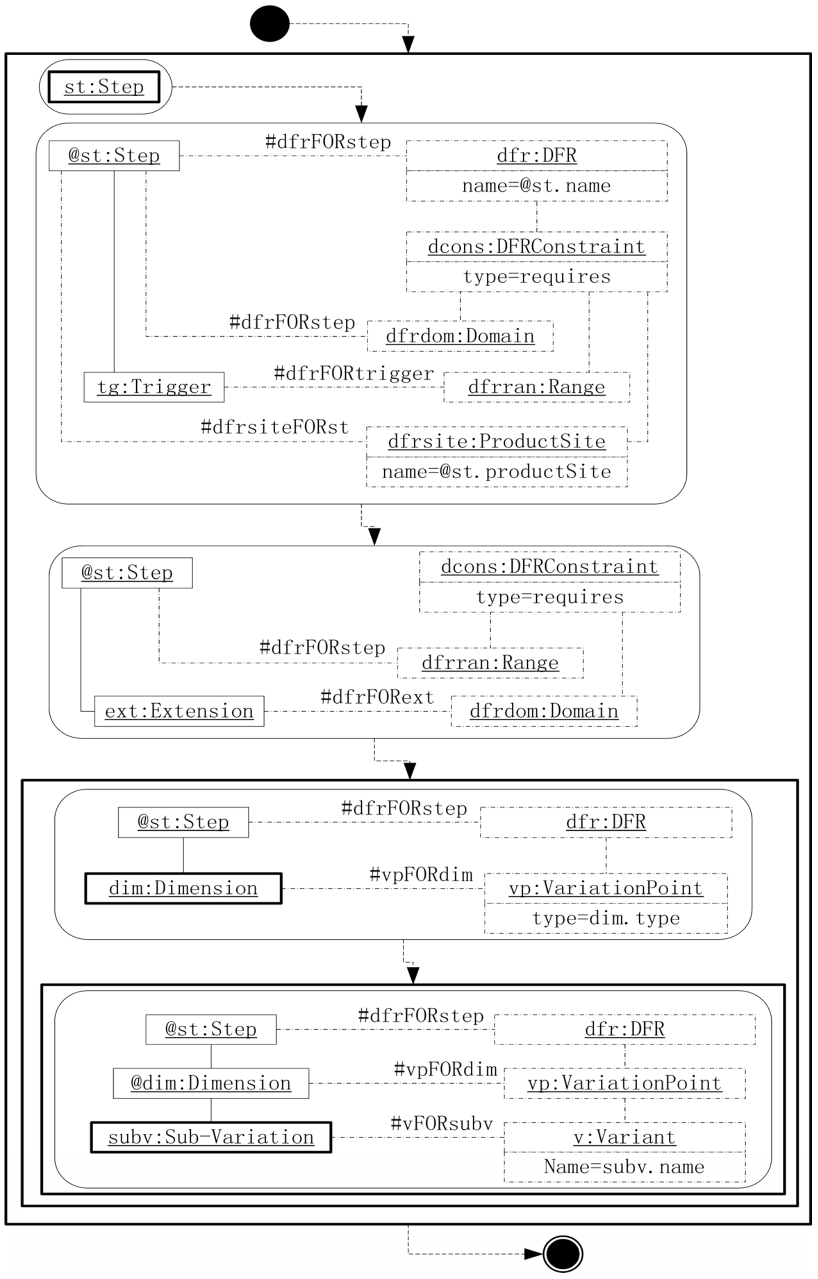


Figure 4. Model transformation definition in MOLA.

from use cases to DFRs contains three nested loops. The outer loop is executed for each step instance. The next statement is a rule building a DFR from a set of steps according to their same names. The mapping association "#dfrFORst" records which DFR from which step actually has been generated and can be reused in the following statements. Thus, all input steps with the same name will be merged as one DFR. Meanwhile, the rule also identifies these steps' product sites to build the DFR's product sites. Next, the extensions and the trigger of the step are transformed into the "requires" DFR constraints.

The middle-level loop is executed for each dimension of the steps. Its rule builds the variation points from the dimensions according to their same types. Its pattern references the "#dfrFORst" mapping association built by the previous statement. The loop head "dim:Dimension" is combined with building actions. Thus, all the dimensions, having the same type and belonging to a set of steps with the same names, will be merged as one variation point of the DFR that is generated by the set of steps.

The inner loop is executed for each sub-variation of some dimension of some step. Its rule builds a variant through merging a set of sub-variations with the same name.

Note that the CV ratios of DFRs will be calculated according to the product sites of DFRs after the model transformation process. For those steps without sub-variations, only the outer loop will be executed for analyzing their commonality and variability.

4. Related Work and Discussion

Some researchers have studied how to extend use cases with variability. They exploit and extend the use cases for product lines in different perspectives and by different means. For example, Jacobson *et al.* [8] introduce variation points into use case diagrams and use them to describe different ways of performing actions within a use case. Gomaa [22] and John and Muthig [13] introduce stereotypes, such as <<variant>>, <<kernel>> and <<optional>> for use cases for modeling families of systems. Most research remains the documentation of variabilities, but ignores the principle of SPLE, *i.e.* proactive reuse. Topics such as how to systematically reuse existing requirements to derive domain requirements lack enough research. In this paper, we extend use cases with a multi-dimensional structure for modeling the variability, and record the product site of each step in use cases. These extensions support systematic reuse of domain knowledge by deriving domain requirements from existing use cases, and analyzing the commonality and variability.

Many domain analysis methods, such as FODA (Feature-Oriented Domain Analysis) [6], FORM (Feature-Oriented Reuse Method) [7], SCV (Scope, Commonality, and Variability) [10], RSEB [8] and FeatuRSEB [9], identify and document the commonalities and variabilities of related systems in a domain. Nevertheless, these typical domain analysis techniques mostly depend on domain experts' knowledge and experience to manually acquire commonalities and variabilities. In addition, Braganca and Machado [23] and Wang *et al.* [24] propose automated approaches to transformation between feature models and use cases. Our approach complements the existing domain analysis techniques by providing automated support for developing DFRs from use cases.

Manual effort is yet indispensable in our approach. First, analysts must scope domain requirements and formulate domain terminology. Second, each use case must be specified with the predefined use case metamodel and unified domain terminology, which helps uncover the incompleteness and inconsistency of existing requirements to a certain degree. Although NLP techniques could be incorporated into DFRs development to minimize the manual operation cost [4], their accuracy can not yet be guaranteed sufficiently. Third, a comprehendsive analysis of variability dependencies must be done by analysts in terms of domain context. In summary, domain experts still play an essential role in the DFRs development.

5. Conclusions

In this paper, we propose an automated approach to DFRs derivation using the metamodeling and model transformation techniques. The main contribution is to present a model-driven approach to domain requirements analysis. The approach complements the existing domain analysis techniques by reducing the manual operation cost and improving the efficiency in DFRs development, which further enhances the transition process from single-system engineering to SPLE. In addition, the model transformation definition documents the traceability information between DFRs and use cases, which helps manage domain requirements and trace their rationale for decision making in SPLE.

Our future work is twofold. First, we will further improve our current approach, especially exploring how to handle the complex constraint dependency. Second, we will verify our approach through an in-depth experimenttal study.

6. Acknowledgements

This work is supported by the National Natural Science Foundation of China (NSFC No. 60773088), the National

High-tech R & D Program of China (863 Program No. 2009AA04Z106), and the Key Program of Basic Research of Shanghai Municipal S & T Commission (No. 08JC1411700).

REFERENCES

[1] P. Clements and L. Northrop, "Software Product Lines: Practices and Patterns," Addison Wesley, Boston, 2001.

[2] K. Pohl, G. Bockle and F. van der Linden, "Software Product Line Engineering: Foundations, Principles and Techniques," Springer Verlag, Heidelberg, 2005.

[3] C. W. Krueger, "Easing the Transition to Software Mass Customization," *Proceedings International Workshop on Product Family Engineering*, Bilbao, 2001, pp. 282-293.

[4] N. Niu and S. Easterbrook, "Extracting and Modeling Product Line Functional Requirements," *Proceedings RE*'08, Barcelona, 2008, pp. 155-164.

[5] R. Prieto-Diaz, "Domain Analysis for Reusability," *Proceedings* 11*th Annual International Computer Software and Applications Conference*, Tokyo, 1987, pp. 23-29.

[6] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Technical Report CMU/SEI-90-TR-021, Software Engineering Institute, Canadian Mennonite University, 1990.

[7] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin and M. Huh, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," *Annals of Software Engineering*, Vol. 5, No. 1, 1998, pp. 143-168.

[8] I. Jacobson, M. Griss and P. Jonsson, "Software Reuse: Architecture Process and Organization for Business Success," Association for Computing Machinery Press, 1997.

[9] M. L. Griss, J. Favaro and M. D. Alessandro, "Integrating Feature Modeling with the RSEB," *Proceedings of International Conference on Steel Rolling*'98, Victoria, 1998, pp. 76-85.

[10] J. Coplien, D. Hoffman and D. Weiss, "Commonality and Variability in Software Engineering," *IEEE Software*, Vol. 15, No. 6, 1998, pp. 37-45.

[11] M. Moon, H. S. Chae and K. Yeom, "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line," *IEEE Transactions on Software Engineering*, Vol. 31, No. 7, 2005, pp. 551-569.

[12] E. de Almeida, J. Mascena, A. Cavalcanti, A. Alvaro, V. Garcia, S. de Lemos Meira and D. Lucrédio, "The Domain Analysis Concept Revisited: A Practical Approach," *Proceedings of International Conference on Steel Rolling*'06, Turin, 2006, pp. 43-57.

[13] I. John and D. Muthig, "Tailoring Use Cases for Product Line Modeling," *Proceedings of Requirements Engineering for Product Lines*'02, Essen, 2002, pp. 26-32.

[14] Z. Zhang, "Model Component Reuse-Conceptual Foundations and Application in the Metamodel-Based Systems Analysis and Design Environment," Ph.D. dissertation, Jyvaskyla Studies in Computing 39, University of Jyvaskyla, 2004.

[15] A. G. Kleppe, J. B. Warmer and W. Bast, "MDA Explained: The Model Driven Architecture: Practice and Promise," Addison Wesley, Longman Publishing Co., Inc., Boston, 2003.

[16] S. Sendall and W. Kozaczynski, "Model Transformation: The Heart and Soul of Model-Driven Software Development," *IEEE Software*, Vol. 20, No. 5, 2003, pp. 42-45.

[17] A. Cockburn, "Basic Use Case Template," 2010. http://alistair.cockburn.us/Basic+use+case+template.

[18] A. Kalnins and J. B. E. Celms, "Model Transformation Language MOLA," *Proceedings of Working Conference on Model Driven Architecture*: *Foundations and Applications* (*MDAFA*'04), Linköping, 2004, pp. 62-76.

[19] I. Jacobson, "Object-Oriented Software Engineering: A Use Case Driven Approach," Addison Wesley, Wokingham, England, 1992.

[20] Object Management Group, "Unified Modeling Language (UML), version 2.2," 2010. http://www.omg.org/technology/documents/formal/uml.htm.

[21] A. Cockburn, "Writing Effective Use Cases," Addison Wesley, Boston, 2001.

[22] H. Gomaa, "Object Oriented Analysis and Modeling for Families of Systems with UML," *Proceedings of International Conference on Steel Rolling*'00, Vienna, 2000, pp. 89-99.

[23] A. Braganca and R. J. Machado, "Automating Mappings between Use Case Diagrams and Feature Models for Software Product Lines," *Proceedings of Southern Poverty Law Center*'07, Kyoto, 2007, pp. 3-12.

[24] B. Wang, W. Zhang, H. Zhao, Z. Jin and H. Mei, "A Use Case Based Approach to Feature Models' Construction," *Proceedings RE*'09, Atlanta, Georgia, 2009, pp. 121-130.

J. Software Engineering & Applications, 2010, 3, 882-889
doi:10.4236/jsea.2010.39103 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

Towards Lightweight Requirements Documentation

Zheying Zhang[1], Mike Arvela[2], Eleni Berki[1], Matias Muhonen[3], Jyrki Nummenmaa[1], Timo Poranen[1]

[1]Department of Computer Sciences, University of Tampere, Tampere, Finland; [2]Futurice GmbH, Taubenstraße, Berlin, Germany; [3]Nomovok Ltd., Keilasatama, Espoo, Finland
Email: {zheying.zhang, eleni.berki, jyrki.nummenmaa, timo.poranen}@cs.uta.fi, mike@arvela.net, matias.muhonen@nomovok.com

ABSTRACT

Most requirements management processes and associated tools are designed for document-driven software development and are unlikely to be adopted for the needs of an agile software development team. We discuss how and what can make the traditional requirements documentation a lightweight process, and suitable for user requirements elicitation and analysis. We propose a reference model for requirements analysis and documentation and suggest what kind of requirements management tools are needed to support an agile software process. The approach and the reference model are demonstrated in Vixtory, a tool for requirements lightweight documentation in agile web application development.

Keywords: *Lightweight, Requirements Documentation, Requirements Management Tool, Agile Software Development*

1. Introduction

Many agile methods emphasize working code [1,2] and working documentation (e.g. in the case of SCRUM), too. Agile methods concentrate on adding value to business and agile software development is an intensive communication process with users. The requirements analysis and documentation activities, however, are often carried out intuitively. Instead of a full requirements document, the most common form of user requirements includes a user story or a use case, which tells a story on how the user completes a meaningful task in interaction with the system to be built. Working on the increments based on user stories involves interaction with the end-user, where more information comes in the form of feedback from the user. This feedback contains changes, additions and refinements on requirements.

Well-defined requirements have traditionally been seen as a critical factor behind software project success [3-6]. Agile methods also emphasize the user requirements, but they are less documentation-centric. It is, however, important to consider how and to whom the requirements are presented and used. We want to develop a user-centric reference model to capture the requirements analysis and documentation environment; this can improve user participation and requirements representation, while supporting agile ways-of-working and values. The paper presents a lightweight requirements documentation environment by proceeding as follows. Initially we present the rationale of having a requirements documentation phase and comment on the three dimensions of requirements engineering (RE). The latter grounds the discussion of how and what might make the requirements documentation a lightweight process and, thus, more agile. Based on this discussion, we propose a reference model for requirements analysis and documentation, and further discuss what the requirements management tools should be like for agile projects. We finally propose Vixtory [7], a prototype tool for agile web application development, as an example to demonstrate the lightweight requirements documentation environment supported by our reference model.

2. The Three Dimensions of Requirements Engineering

Pohl [8] describes the RE activities and their goals along three orthogonal dimensions: specification, representation, and agreement. The framework assumes that there are three major facets of the RE process, namely documenting the requirements in a more complete manner, with more formality, and more consensus among stakeholders [8,9]. The *specification dimension* deals with the degree of requirements understanding at a given time [8]. Completeness and understandability of knowledge form the main concerns in this dimension. The *representation* *dimension* copes with the degree of formality of knowledge expression [8]. In general, requirements can be documented in three types of representation: informal representation (e.g. natural language), semi-formal representation (e.g. ER diagram, state diagram, etc.), and formal representation (e.g. mathematics expression). From the informal presentation to the formal one, requirements documents are shifting from the user-oriented ones to the system-oriented ones. The *agreement dimension* deals with the degree of agreement reached on a specification. It focuses on a variety of views of the same system from heterogeneous groups of stakeholders, and emphasizes the common agreement.

3. The Meaning of 'Lightweight' Requirements Documentation

Requirements documentation provides a means of communicating between diverse stakeholders and achieving common agreement on the future software artifacts description [3]. Requirements elicitation is perceived as an essence for a software development project, but the requirements analysis, documentation, validation and maintenance are very tedious processes. Many researchers claim that most requirements specifications found in industry nowadays still include many poor-quality requirements [6,7], even though there are so many different techniques to ensure the requirements' quality. Poor requirements form an important reason that causes the failure of many IT projects [5,10].

An ideal requirements document shall be correct, complete, consistent, precise, testable, and traceable [4]. In practice, however, it is hard to address all requirements up front, and to maintain a correct and consistent document throughout the project in an ever-changing environment. In agile software projects, in particular, the traditional requirements process is replaced with iterations and increments, and the documentation is replaced with user stories, working software, and changing requests. We shall ensure that stakeholders express, understand, document, use, and maintain requirements in a correct and easy way. The requirements gathering and agreement process should shift from documentation to communication efforts. A more narrative and context-specific approach should be adapted to improve the requirements analysis process, and at the same time, keep it lightweight. The latter indicates the environment which deploys the available resources to effectively support the communication and the consequent analysis and documentation. The meaning of lightweight is next discussed along three important dimensions.

From the perspective of specification, requirements documentation is an ongoing process, and the details can be elaborated just in time. Requirements need not be fully specified up front, at a very early stage of the project, when many aspects are unknown and needs cannot yet be expressed, consistently and correctly, to say the least. We hereby agree with other researchers [3,12,13] supporting that requirements development is an ongoing process throughout a software development project. Meanwhile, there is no point in specifying highly detailed requirements before software or at least prototype development even starts. Software requirements can be elaborated at the right time when they are selected for implementation. This is natural as the application domains of the real world, to which the software targets, is subject to change. Furthermore, users change their understanding towards their requirements and needs as the development proceeds in new software releases that need feedback.

From the perspective of representation, prototypes or working software improve the requirements understandability by providing a context realism representation. To get some grip of the concept lightweight in the representation dimension, we make the following division of ways of working to document requirements. In the traditional waterfall model, the requirements have been documented before the actual software is being built. A typical way to express requirements is textual [7,14]. Different from the traditional waterfall model, prototyping [3,14] means building software to mimic the behavior and functionality of the target software to be built. The prototypes are used to validate requirements, but, they could also be seen as specification techniques, where requirements can be elicited upon and attached to what can be called prototyping software. Similarly, it is also possible to attach requirements to working software. This has become more of an option in incremental software development, where software is built on top of existing increments, as is typically done in agile software development. The working software can provide users a real and actual representation of the requirements. It can be regarded as a starting point to elicit and refine requirements rather than an end of a development cycle. Such a context enriched representation makes a smooth transformation from the high level requirements description to the detailed implementation, and enhances the clarity and understandability of requirements. At the same time, the requirements are not rigid with a specific form of representation, which forms a flexible and lightweight process to represent requirements.

From the perspective of agreement, timely feedback on small releases of working software supports the evolution from the individual views to a common agreement. Incremental development that utilizes prototypes or other prototype-like software artifacts, e.g. working software, gives us a possibility to attach a part of the requirements to existing increments. This allows stakeholders not only to perceive the target applications, but also to present their individual views and wishes based on the existing version. With the frequent releases, stakeholders can review the working software, adjust their understanding of the target application, and provide timely feedback as the requirements for the follow-up iterations. It flexibly supports the evolution from the personal views to a common agreement on the target system, and avoids the aforementioned problems of prototyping methods: The software does not give a promise of a functionality which may be incorrect and also the design is "as-is".

4. A Reference Model for Requirements Documentation Environments

The most essential aspects in a requirements analysis and documentation environment can be captured in a reference model. The reference model has ER like notations. The rectangles represent entities which facilitate requirements documentation. They are integrated together through a number of traceability links, represented as arrows. As shown in **Figure 1**, the model consists of three basic entities: requirements, external documents, and software artifacts. These are the most essential artifacts found in every software project, and documented in the supporting tools. Entities are interlinked with each other through a number of traceability links. Each entity and links can be specialized and instantiated to create organization or project specific requirements documentation environments. Since lightweight documentation is our main concern and target in this study, the instances and attributes, which reflect the nature of agile software development approaches, are marked in bold. In the following section, we will discuss the reference model along different dimensions of lightweight requirements documentation.

4.1. Requirements

Requirements comprise the specifications of the services that the system should provide, the constraints on the system and the background information that is necessary to develop the system [16]. They are typically represented in a natural language, supplemented by diagrams such as the use case diagram, the data flow diagram, etc. Requirements are documented with attributes such as creation date, version number, author, description, priority level, status, etc.

Instead of limiting a requirements specification to a single and rigid representation, the informal representations of users' conception of their system such as user stories [1,2], use cases, and scenarios can be elaborated

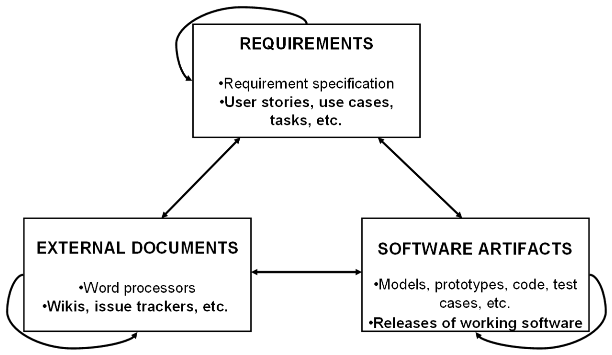

REQUIREMENTS
•Requirement specification
•User stories, use cases, tasks, etc.
EXTERNAL DOCUMENTS
•Word processors
•Wikis, issue trackers, etc.
SOFTWARE ARTIFACTS
•Models, prototypes, code, test cases, etc.
•Releases of working software

Figure 1. A reference model of a requirements analysis and documentation environment.

and attached to the working software at a right time, which can make the requirements documentation process intuitive and encourage customers' participation. User stories include a set of small sentences that express the user needs, in her/his own words [2]. The description tells a story on how the user completes a meaningful task in interaction with the system to be built. When a user story is selected for implementation, it can be further elaborated by the developers in their preferable forms.

Requirements can be documented in different levels of detail. The high abstraction level requirements, such as user stories, are documented by interacting with customers and by means of experimental use of prototypes or working software. They are customer's actual needs. When the user stories are selected for implementation, they are refined and adjusted into detailed tasks, and implemented and evaluated within the same iteration. Such divide-and-conquer tactics isolate the customers from complex technical implementation, while enable them to provide timely evaluation and feedback of accumulated implementations.

4.2. External Documents

External documents represent the documents which are not stored in any requirements management tools (RMTs). In traditional software development process, they typically describe and contain the requirements specified in general-purpose document tools, modeling tools, etc. These documents are structured ones and easy to create, but static. It is hard for different stakeholders to work in collaboration on the same document, and to document and exchange ideas in a lightweight process.

An agile principle is close collaboration between developers and customers. A lightweight documentation needs a platform that can support effective and efficient collaboration among an often large number of diverse needs and requirements stakeholders. The model we propose enhances collaboration by adding the instance of generic collaborative platforms, such as wikis [17,18] and issue trackers. These can provide a flexible way of open review, elaboration, refutation, or refinement with a discussion thread. Further the discussion and communication comments give rise to the development of narrative descriptions of the features and requirements of the software under development. This can reduce the details needed in the requirements documentation, as the detailed contextual information can be linked to the discussion on the collaborative platform. The process can be adapted to support active stakeholders participation in requirements elicitation and documentation [18].

4.3. Software Artifacts

Software artifacts represent the final or interim byproducts during the development of software. Examples include specifications, prototypes, code, testing cases, as well as working software released at the end of each iterative process. They are connected with requirements through a variety of traceability links, which provide contextual information within the development team to support software development activities, such as change impact analysis, testing, maintenance, project tracking, etc.

As one of the agile principles is ‘frequent releases’, the development team can deliver working software to the customers for their experimental use and for feedback [1]. The experimental use process expands the contextual information within the development team to that between the customers and the developers. It facilitates customers to explore the real working product, and to provide individual feedback before the final delivery, which reduces the uncertainty between the development team and the customers. Being different from a prototype, which may represent some compromise from the final production design, the working software provides customers with an actual production design, and, thus, eliminates the risks of misunderstanding and misleading. The context specific information of working software is much more real than that of a prototype. Furthermore, compared with the throw-away prototype approach [3,14], the practice of short release of working software saves time and other resources in a software project. Therefore, an easy use of the traceability link among requirements and the working software is necessary to facilitate the communication of ideas and prompt feedback.

4.4. Traceability Links

Traceability links connect the instances of every component to provide contextual information on the target system [13]. They present the relationship of entities instantiated from the same element, such as the elaboration relationship between a high level user story and a list of low level tasks, the validation relationship between a test case and a segment of the software, or the hyperlinks available in wikis or any other collaborative platforms. Furthermore, the traceability links between different elements allow developers to trace code back to the conversation from which the artifact came, back to the user story, and finally to its initial requirements [19]. They also facilitate the customers to be involved in the development process by tracing between the user stories to the working software. Consequently, there are two categories of traceability links in the requirements documentation environment: links within tools and links between tools. It is undoubted that a set of tools are deployed within a project to support the development activities from different aspects. In general, each tool can provide some sort of traceability information within the use of tool, such as the aforementioned elaboration or validation links. Besides, it is necessary for an agile project to have an integrated environment by using hyperlinks to connect the requirements and other information scattered in different tools. Examples include the traceability link between the prototyping software and the RMTs, between the collaborative platform and the RMTs or a CASE tool. These hyperlinks ease the flexible documentation and use of requirements and the related information [20], which reflects the goal of the reference model.

Access to documented traceability provides different levels of context realism. It is indeed very valuable. However, the manual burden directly contradicts with the agile principles. Developers are often reluctant to participate in the effort of documenting traceability information [13,21,22]. On the basis of the existing tools, a solution to connect the scattered information manually or automatically into the iteration is very important. The next section elaborates further on this aspect.

5. Tools Supporting Lightweight Requirements Documentation

There is a number of tools supporting requirements analysis and documentation. A tool survey conducted by INCOSE [23] compares the features of over forty different RMTs from 2004 to 2009. These tools support the requirements documentation process and, clearly, influence the quality of the documentation. Before discussing the need for lightweight requirements documentation tools, we will have an overview of the features of tools that support the traditional requirements documentation phase.

5.1. Classification of Requirements Documentation and Management Tools

We attempt to broadly classify existing RM tools into four groups: general-purpose document tools, collaborative tools, RMTs, and prototyping tools.

The *general-purpose document tools* mainly include office suites such as MS Office, Open Office, Lotus SmartSuite, etc. These are not too specialized, and many users are acquainted with since they are easy to adapt to the needs of different development environments. Surveys report that these general-purpose document tools, though not sophisticated, in practice are helpful with requirements documentation [5,6,24-26]. Though the non-specialized features can be considered as a great merit, it is difficult to support specific RE activities and ensure the quality of the derived documents. .

The *collaborative tools* offer a flexible platform that can involve a number of diverse users in common tasks to achieve their goals and for collaborative editing of contents. Wikipedia is an example for creating an encyclopedia openly and collaboratively by volunteers from all around. According to the level of collaboration, there are different categories of these tools ranging from a simple information sharing application (e.g. online chat, wikis, etc.) to sophisticated systems that facilitate group activities (e.g. knowledge management and project management systems). Instead of facilitating documentation, these tools provide a lightweight solution to creating, editing, sharing, and discussing information; the latter obviously improve the communication and collaboration for requirements analysis.

RMTs are dedicated to manage the large amount of data collected during the RE process and the volatility of the requirements [3]. There are many commercial RMTs such as DOORS, Requisite pro, CaliberRM, etc. Typically, these tools collect the requirements in a database and provide a range of facilities to access the information on the requirements. These facilities include requirements browsing, requirements converting, report generating, requirements tracing, change control, etc. The RMTs that support formal requirements representation can also facilitate requirements consistency checking and semantics verification [27]. Such tools aim at technical users, and provide a comprehensive environment to support the different dimensions of RE process. Empirical studies [25] support that RMTs provide better coverage of the RE process and the quality of requirements documentation. On the other hand, many surveys [5,6, 24-26] report that the mainstream practice relies on office and modeling tools rather than RMTs. Survey reports contradict on the industrial use and the rationale of RMTs.

The *prototyping tools* are specific tools, which rapidly represent, build, and realize important aspects of a software-based system. The prototype serves as an experimental system to demonstrate requirements and collect stakeholders feedback. Prototyping tools range from simple ones that develop a mock-up system to specialized ones that create interactive wireframes for websites and desktop software, and design user interfaces with high functionality. Examples include Axure RP, ProtoShare, etc., which generate web-based prototypes. Besides, some general-purpose CASE tools provide good support for prototyping for user interfaces and web design, such as the graphic design tools (e.g. Illustrator or Adobe Photoshop), the diagramming tools (e.g. Visio or SmartDraw), and the visual and text based HTML tools (e.g. FrontPage, Dreamweaver, etc.). Instead of specifying and managing requirements, the prototyping tools focus more on providing stakeholders with a real experimental system, which increases requirements understandability and avoids requirements creep and rework.

In addition to these four categories of tools for requirements documentation, there are also agile project management tools, such as Rally, Scrumworks Pro, which facilitate backlog (requirements) editing and report generating. All these tools provide support for requirements documentation in some aspects of the reference model depicted in **Figure 1**. In general, RMTs, as well as all other requirements documentation tools, provide support for requirements specification in different levels of formality. Besides, the collaborative tools provide more flexible support for the external documents, while the prototyping tools can provide links between the software artifacts and the requirements. Obviously, none of them can cover the components specified in the reference model of **Figure 1**.

The purpose of requirements documentation is communication among a number of stakeholders. The general-purpose document tools have widespread availability. They, however, lack adequate support for communication and collaboration in the RE process. The collaborative tools compensate for the deficiency of collaboration in the general-purpose documentation tools, but lack enough support in context enriched representation and just-on-time requirements documentation. The RMTs tools over-emphasize the specification and representation dimension of the RE process, i.e. the bureaucratic and rigid support for the RE process, but do not facilitate a close and smooth interaction between developers and customers [21,26]. The communication factor is lost. The prototyping tools, on the other hand, offer users the actual prototype for experimental use and feedback, but, most of them, lack necessary features that facilitate just-on-time specification. In a summary, **Figure 2** illustrates the tools previously discussed and their support of the goals set within the three dimensions of lightweight requirements documentation, as discussed in the beginning of this work.

Consequently, in order to better support requirements documentation, a tool should capture the three important dimensions of the RE process, as outlined in the context

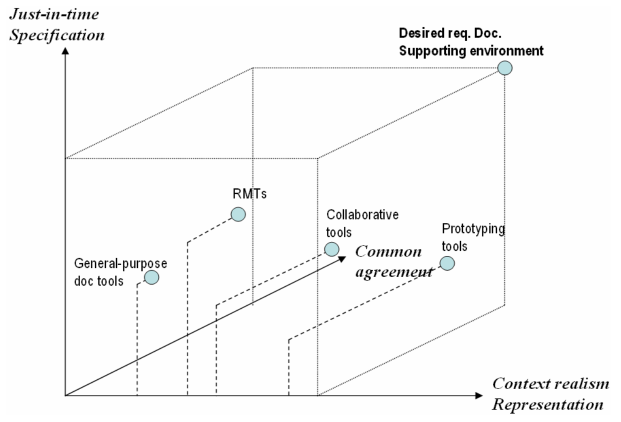

Just-in-time Specification
Desired req. Doc. Supporting environment
RMTs
Collaborative tools
Prototyping tools
General-purpose doc tools
Common agreement
Context realism Representation

Figure 2. Tools within the three dimensions of lightweight requirements documentation.

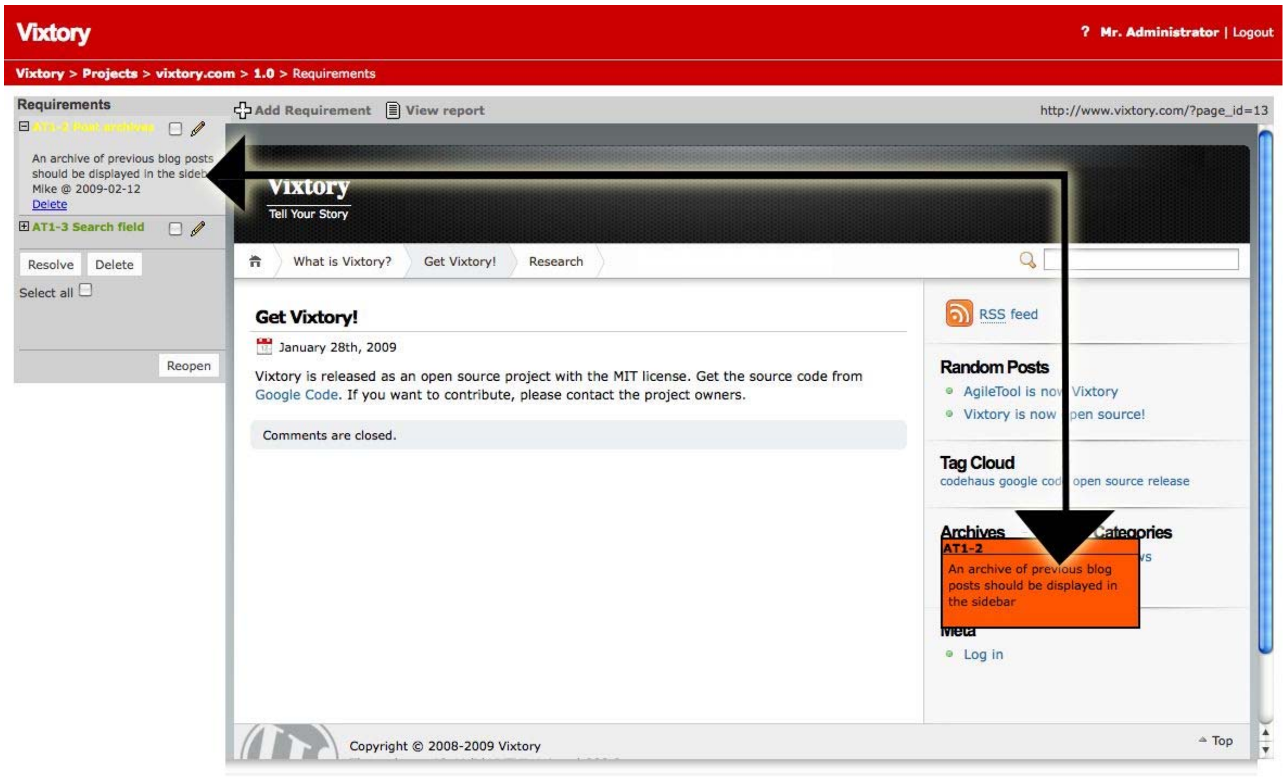
of this paper. A single tool cannot provide all the desired features for a lightweight requirements documentation process. The latter should be facilitated by a set of simple, intuitive, and widespread availability tools [20,28], which could easily and flexibly be integrated into the development environment.

5.2. Vixtory–A Target Application Based Requirements Documentation Tool

As shown in **Figure 2**, prototyping tools are the ones most close to the desired lightweight requirements documentation. The target is, thus, to improve the specification dimension of such tools and provide users with an actual experimental use of the target application. Motivated from these needs and the tools features discussion, we developed a requirements management tool for agile web application development, namely Vixtory [7]. It provides a lightweight and less burdensome documentation approach by annotating requirements directly to the target application. The stakeholders are allowed to participate in the development process and review the target application even during development.

Vixtory [7] was implemented with Groovy and the Grails framework [29] using Asynchronous Javascript and XML (AJAX) to store requirements in a relational database. Vixtory models requirements in an intuitive way: the requirements are part of the application being developed. There is no need to maintain a separate requirements document. The stakeholders can add a new version of the web application being developed to Vixtory's project database. Each version is identified by an URL address. Stakeholders can freely navigate in the Vixtory web application with a standard web browser.

As can be seen from **Figure 3**, the web page under development is on the right side of the screenshot, and the requirements pane showing a list of the requirements



An archive of previous blog posts should be displayed in the sideb
Mike @ 2009-02-12

Get Vixtory!

January 28th, 2009

Vixtory is released as an open source project with the MIT license. Get the source code from Google Code. If you want to contribute, please contact the project owners.

Comments are closed.

AT1-2
An archive of previous blog posts should be displayed in the sidebar

Copyright © 2008-2009 Vixtory

Figure 3. The layout of Vixtory.

is on the left. The stakeholders can browse the web application in question freely page-in-page and identify the requirements for individual views of the web application. The identified requirements are attached with the requirements annotation tool to the corresponding view of the web application. An annotation in Vixtory is a prioritized requirement containing a textual description, listed on the requirements pane.

The annotated requirement is visually linked to an element on the web application. Any elements from links to complicated forms can be annotated with a requirement. The annotations provide a clear traceability link between requirements and the implementation without adding a burden of a specification document, which also facilitates the communication and collaboration between stakeholders. Vixtory provides developers and on-site customers with a straightforward view of the web application being developed, which forms the actual target application context. It also supports and manages change by allowing effortless updating and replacing of requirements.

Vixtory was created with user experience and ease of use as top priorities [30]. Given that Vixtory is in its first commercial release iteration, much work still remains to be done. The requirements specification and representation will further be improved in order to provide end-users with more flexibility in the documentation process. The hypertext links, for instance, between Vixtory and the existing project management tools or collaborative platforms are missing and this is something that will further be considered. How easily Vixtory can be integrated and used with other development platforms and organizational cultures are open questions, worth considering for our ongoing research.

6. Conclusions

We discussed the need for lightweight requirements documentation and presented a reference model for addressing this need. We provided an existing RM facilitated tools taxonomy and drew conclusions on how these tools support requirements analysis and documentation in agile software development. Upon the comparison and contrast of these tools, we identified further needs for requirements documentation that have not been adequately addressed. Therefore, we proposed the adoption of the Vixtory tool and illustrated how it can be used to flexibly document requirements for agile development.

As Vixtory is a prototype tool, we do not yet have enough feedback from the Vixtory tool production use. The feedback upon the initial experimental use of Vixtory has been positive. The project managers, in particular, like the tool. An obvious reason is that the tool makes end user participation easier and it offers less vague and ambiguous requirements due to the actual target system context. In the future, we need to empirically evaluate the acceptability of the tool, asking more stakeholders on their experiences. We currently expect to gain experience from industrial and student software projects. We are particularly interested in the users' communities feedback for improvement.

REFERENCES

[1] K. Beck, “Extreme Programming Explained: Embrace Change,” Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[2] A. Cockburn, “Agile Software Development: The Cooperative Game,” 2nd edition, Addison-Wesley Professional, 2002.

[3] G. Kotonya and I. Somerville, “Requirements Engineering: Processes and Techniques,” John Wiley & Sons, Chichester, 1998.

[4] IEEE Recommended practice for software requirements specification. IEEE Standard 830-1998, 1998.

[5] H. F. Hofmann and F. Lehner, “Requirements Engineering as a Success Factor in Software Projects,” *IEEE Software*, Vol.18, No.4, 2001, pp. 58-66.

[6] U. Nikula, J. Sajaniemi and H. Kälviäinen, “A State-of-the-practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises,” Research Report, Telecom Business Research Center, Lappeentanta, 2000.

[7] Vixtory, “Tell your story”. http://www.vixtory.com/

[8] K. Pohl. “The Three Dimensions of Requirements Engineering: A Framework and its Applications,” *Information Systems*, Vol. 19, No. 3, 1994, pp.243-258.

[9] C. Rolland and N. Prakash, “From Conceptual Modelling to Requirements Engineering,” *Annals of Software Engineering*, Vol. 10, No. 1-4, 2000, pp.151-176.

[10] The Standish Group, “Chaos Chronicles Version 3.0.”, 2003. http://www.standishgroup.com/chaos/

[11] Agile manifesto, 2001. http://agilemanifesto.org/

[12] E. Berki, “Formal Metamodelling and Agile Method Engineering in MetaCASE and CAME Tool Environments,” *The* 1st *South-East European Workshop on Formal Methods*, South-Eastern European Research Center (SEERC): Thessaloniki, 2004, pp. 170-188.

[13] B. Ramesh and M. Jarke, “Toward Reference Models for Requirements Traceability,” *IEEE Transactions on Software Engineering*, Vol. 27, No.1, 2001, pp.58-93.

[14] C. Ghezzi, M. Jazayeri and D. Mandrioli, “Foundamentals of Software Engineering,” 2nd Edition, Prentice Hall, 2003.

[15] E. Georgiadou, K. Siakas and E. Berki, “Agile Methodologies and Software Process Improvement,” *Proceedings of the Virtual Multi Conference on Computer Science and Information Systems*, Virtual Online Conference, 2005,

pp. 412-417.

[16] P. Zave, "Classification of Research Efforts in Requirements Engineering," *ACM Computing Surveys*, Vol. 29, No. 4, 1997, pp. 315-321.

[17] P. Louridas, "Using Wikis in Software Development," *IEEE Software*, Vol. 23, No. 2, 2006, pp. 88-91.

[18] B. Decker, E. Ras, J. Rech, P. Jaubert and M. Rieth, "Wiki-Based Stakeholder Participation in Requirements Engineering," *IEEE Software*, Vol. 24, No. 2, 2007, pp. 28-35.

[19] C. Lee and L. Guadagno, "FLUID: Echo–Agile Requirements Authoring and Traceability," *Proceedings of the 2003 Midwest Software Engineering Conference*, Chicago, June 2003, pp. 50-61.

[20] B. Boehm, "Requirements that Handle IKIWISI, COTS, and Rapid Change," *Computer*, Vol. 33, No. 7, 2000, pp. 99-102.

[21] Z. Zhang and J. Kaipala, "A Conceptual Framework for Component Context Specification and Representation in a MetaCASE Environment," *Software Quality Journal*, Vol. 17, No. 2, 2009, pp.151-175.

[22] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," *Proceedings of the* 1*st International Conference on Requirements Engineering* (*ICRE '94*), Colorado, 18-22 April 1994, pp. 94-101.

[23] INCOSE requirements Management Tool Survey. http://www.paper-review.com/tools/rms/read.php

[24] A. Forward and T. C. Lethbridge, "The Relevance of Software Documentation, Tools and Technologies: A Survey," *Proceedings of the* 2002 *ACM symposium on Document engineering*, McLean, Virginia, USA, ACM Press, pp. 26-33.

[25] A. Persson and J. Stirna, "Advanced Information Systems Engineering," 16*th International Conference*, *CAiSE*, Riga, Latvia, June 7-11, 2004, Proceedings.

[26] A. Manninen and E. Berki, "An Evaluation Framework for the Utilisation of Requirements Management Tools-Maximising the Quality of Organisational Communication and Collaboration," *Proceedings of BCS Software Quality Management* 2004 *Conference*, British Computer Society: Swindon, 2004, pp. 139-160.

[27] C. Heitmeyer, J. Kirby and B. Labaw, "Tools for Formal Specification, Verification, and Validation of Requirements," *Proceedings* of *the* 12th *Annual Conference* (*COMPASS*'97).

[28] B. Kernighan, "Sometimes the Old Ways are Best," *IEEE Software*, Vol. 25, No. 6, 2008, pp. 18-19.

[29] G2One Inc., Grails Web Application Framework, http://grails.org/

[30] M. Arvela, M. Muhonen, M. Piipari, T. Poranen and Z. Zhang, "Agile Tool-Managing Requirements in Agile WWW Projects," *Proceedings of BIR* 2008, Gdansk, 2008, pp. 211-215.

J. Software Engineering & Applications, 2010, 3, 890-893
doi:10.4236/jsea.2010.39104 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

Investigating the Suitability of Agile Methods for Requirements Development of Home Care Systems

Sandra Kelly, Frank Keenan

Dundalk Institute of Technology, Dundalk, Ireland.
Email: {sandra.kelly, frank.keenan}@dkit.ie

ABSTRACT

The ageing population in developed countries brings many benefits but also many challenges, particularly in terms of the development of appropriate technology to support their ability to remain in their own home environment. One particular challenge reported for such Home Care Systems (*HCS*) *is the identification of an appropriate requirements development technique for dealing with the typical diverse stakeholders involved. Agile Methods* (*AMs*) *recognize this challenge and propose techniques that could be useful. This paper examines the desirable characteristics identified for requirements development in HCS and investigates the extent to which agile practices conform to these. It also sets out future work to improve the situation for the non compliant points found.*

Keywords: *Home Care Systems* (*HCS*), *Agile Methods, Requirements Development*

1. Introduction

According to the World Health Organisation (WHO), the population of over 60s in developing countries is expected to triple from "400 million to 1.7 billion" over the next forty years [1]. Rising healthcare costs and lack of resources in terms of hospitals and the availability of skilled doctors and nurses mean that by 2050, this situation will become unsustainable. In Ireland 11% of the population is currently over 65 and this is expected to double over the next 25 years, in particular a threefold increase in those aged 80 and above is expected [2]. Accommodating the ageing population is and will continue to be an increasing challenge. For example, Technology Research for Independent Living (TRIL) report that 40% of older people are prematurely institutionalised due to injuries sustained from falls in the home [3]. In general it is beneficial to develop alternative approaches to prevent or delay the institutionalisation of older people, enabling the elderly to remain at home. This has subsequently required an increase in the need to develop HCS.

HCS is defined as "a potentially *linked set of services* including social care and health care that provide or support the provision of care in the home" [4]. Along with the cared person, many other stakeholders are involved including family members, social care, home help, health nurses and GPs. Given such diversity of background, perspective and experience, and perhaps geographic distribution, identification, access to and engagement of the appropriate stakeholders to identify requirements is difficult. To complicate things further, modes of interaction are expected to be subject to continuous change over time as the condition of the home dweller may change [5]. Identification of a more suitable approach for requirements development is necessary.

Despite the numerous techniques that exist for requirements development [6] and, indeed, efforts made to classify them [7,8] it is claimed that current techniques are inadequate, while McGee-Lennon suggests that a novel approach is required. Despite this, it is acknowledged that potential does exist if techniques could be *modified* to deal with a "combination of multiple distributed and possibly conflicting stakeholder needs" along with "long term configuration and evolution of these needs" [5]. In general, for HCS, any approach should be "lightweight enough to be useable yet rigorous so as to be justifiable" [4].

A potential solution may however be offered through the principles and practices suggested by agile development. The Agile Manifesto promotes values of close and continuous *interaction* between all interested stakeholders, including the *customer* and development team, very *short iterations* of fully working, and tested, software, which provides the ability to easily *accommodate changing requirements* [9]. At an initial inspection it appears that the characteristics of agile development closely match those identified for HCS.

This paper compares the desirable characteristics of requirements development in HCS identified by [5] with the general approach recommended for handling requirements in agile development. Section 2 looks at how requirements are developed in an agile setting, Section 3 compares the characteristics of HCS with AMs, while Section 4 concludes the paper and outlines future work.

2. Agile Requirements Development

AMs recommend a highly participative and iterative approach to requirements development [10]. Initially, requirements are briefly documented, typically through *user stories*. Usually for user stories a high level user request is written on an index card or post-it note. Along with specific functionality required, other relevant information such as story title, release date, order of priority, author's initials and time estimates to complete can also be included. During development, each story is used to provoke an in-depth discussion between developers and other stakeholders to examine each requirement in detail [11].

Initially, collectively, user stories identify a high level plan for each release of the project. The customer prioritizes each story according to business value. Attention is then focused on the first release with a number of prioritized stories used to identify the first short iteration. Typically this is a week or two in length [10]. Developers further divide the stories into tasks and estimate a timeframe to complete each task. It is expected that each story is considered to be a placeholder, indicating that an in-depth analysis will be conducted during the iteration.

User stories and their associated tasks are placed together on a publicly-visible story board. The board frequently referred to as "the wall" by [12], is the main focal point of the room. Developers choose a story to complete from the board and commit to this by signing their initials on the story card and taking it from the board for development. When the user story is complete, the developer ticks the card and returns it to a new position on the story board. Alternatively, if the story is not fully complete, the card is considered to be still pending and is returned to the same position on the board. Complete stories become features of the system where they can be further developed in upcoming iterations.

The story board provides a clear indication of what work has been done and what has yet to be completed. Stories placed with their tasks also allow developers to envisage dependencies, essentially providing a visual representation of the work plan to the team. The colour of a card can also be used to convey specific meaning and can indicate warning signs. Some examples from [12] showed:

- Green cards signified stories, white for tasks;
- Blue cards related to features for staff;
- Orange flags indicated incomplete acceptance tests;
- Pink cards described bugs.

The positioning of the cards on the story board can also communicate specific meaning. The top three rows for instance in [13] contained recently completed stories. To the left of the board were scheduled stories and to the right were unscheduled stories.

2.1. The Customer Role

A key role for successful requirements development within AMs is that of the customer, which differs from that expected in a traditional development project, for example with the waterfall approach the customer is involved at the beginning of a project and the relationship between the customer and the development team throughhout the project is contractual, whereas AMs prefer the customer to be continuously involved for the duration of the project. However, a misconception with early AMs is that customer involvement was often reduced to a single on-site customer with little guidance provided on how to implement this role.

In distinguishing between traditional and agile Requirements Engineering (RE) practice in industry, Cao & Ramesh found that the "inability to gain access to the customer and obtaining consensus among stakeholder groups" were the most common challenges experienced [14]. Martin *et al*. report eight practices that were used in successful projects to enable real customer involvement [15]. The authors illustrate the complexity of customer representation, identifying ten roles on a customer team, these were informally created with little prior guidance to support the customer role. Each person on the customer team negotiates with and represents a widely diverse group of stakeholders.

Many authors have expressed the importance of having the appropriate and relevant stakeholders on board. In examining critical success factors in software development, Boehm and Turner found that the customer role should comprise of individuals who are Collaborative, Representative, Authorized, Committed and Knowledgeable (CRACK), deeming these crucial attributes stakeholder representatives should possess in implementing the customer role [16]. Hence, performing the customer role in agile projects is a challenging task, frequently taken for granted.

3. Agile Methods for HCS?

McBryan *et al.* identify ten desirable characteristics an RE method should possess for HCS [5]. **Table 1** summarizes these characteristics in column 1 with an indication of how AMs match these in column 2. Two ticks indicate that AMs comply fully with the characteristic, one tick indicates partial compliance and 'x' indicates no compli-

Table 1. HCS characteristics and agile methods compliancy mapping.

Characteristics	AMs
Iterative development	✓✓
Prioritization	✓✓
Correlation with other processes	✓✓
Appropriate stakeholders	✓
Participatory elicitation	✓
Identification of conflict	✓
Resolution of conflict	✓
Retention & traceability	✓
Annotation	✓
Distributed elicitation	✓

ance with the characteristic. The following points discuss the varying degrees of compliance between these.

Iterative development: with AMs, short iterations of fully working software provide opportunities for continuous stakeholder input. Perceived needs can be clarified and new ones emerge. As circumstances change requirements will evolve based on stakeholder input. This is a characteristic AMs are fully compliant with.

Prioritisation of requirements: different stakeholders may need to be given different priorities depending on a variety of circumstances. Specifically in [5], the authors point out that in relation to particular features, if for instance, usability is an issue then the needs of the person in care may be the highest priority whereas care professionals may have higher priority requirements if it is the case that the person concerned is a "risk to themselves or others" in the home environment. AMs are fully compliant with this characteristic as features can be prioritized at the beginning of each iteration.

Correlation with other processes and work practices calls for immediate benefit from solutions required. This is entirely consistent with the agile approach which promotes the early delivery of high quality 'working software' to satisfy the customer's business objective.

Identification and Engagement of appropriate stakeholders is partially compliant as AMs recognize the need for this. Although this is promoted within AMs and numerous successes have been reported, difficulties have emerged in certain situations particularly when multiple stakeholders are involved. However, no generalized method as such can be applied here since realizing this depends on constraints and circumstances often unique to each situation.

Participatory elicitation and negotiation requires those involved in a care network including the cared person to negotiate the suitability of a potential device or care service proposed. Essentially, an opportunity to demonstrate a candidate device or interaction mode to stakeholders in advance of decisions to be made is necessary. This characteristic is partially compliant as Active Stakeholder Participation (ASP) is encouraged in AMs but achieving this in practice remains problematic.

Identification of conflict partially complies since AMs encourage conflict to be aired as soon as possible but this is often challenging if relevant stakeholders are not actively involved or, as is frequently reported, only identified during the latter stages of development.

Resolution of conflict partially complies, although the need to air and resolve conflict early is recognized, success here is again dependent on stakeholders' active participation in the project.

Retention and traceability of requirements is partially compliant since AMs only retain artefacts such as user stories for as long as they are deemed useful. Agile practitioners recommend questioning traceability in terms of time to complete and the re-examination of what value it brings to stakeholders. This is to ensure that while traceability may be needed, it must be applied in a timely manner. An option here is to employ an appropriate project management tool that does not detract from the overall effort required in developing and maintaining the software solution. Examples of available tools include Envision VIP 9 and PACE 3.

Annotations to facilitate negotiation and traceability, is also partially compliant as story cards are often annotated. However, it is likely that agile practitioners would integrate the annotation as a main part of the user story. In particular, this characteristic intends to add further context to a requirement.

Distributed elicitation is a characteristic AMs are partially compliant with since although AMs prefer face to face communication, other means such as email and video conferencing are most often used for dispersed stakeholders. In addition to tools mentioned previously, other tools, particularly XPlanner and TargetProcess enable distributed teams to communicate with geographically dispersed stakeholders.

Here, AMs are fully compliant with three characteristics of a desirable approach to requirements development in HCS. However, AMs show only partial compliance with the remaining seven characteristics indicating future work needed to improve on these.

4. Conclusions and Further Work

In summary, due to the expected increase in older populations, the need to develop HCS is evident. Success in software systems heavily depends on accurately obtaining requirements from stakeholders. It is also difficult to identify access, engage and support continuous negotiation of requirements amongst relevant stakeholders. Despite the numerous elicitation techniques available, accommodating diversity amongst multiple stakeholder groups remains a key challenge. Suggestions made to improve on this include a new or adaptive approach to requirements development, However it is not entirely clear how this could be achieved. AMs may provide a solution but particularly, an important concern here is in effective implementation of the customer role. This paper compared the desirable characteristics for RE in HCS with agile methods and indicates that there is a close match. However, challenges still exist as AMs are only partially compliant with many of the remaining characteristics. Future work will investigate how this position can be further improved.

REFERENCES

[1] World Health Organization (WHO), "Public Health Implications of Global Ageing," 2010. http://www.who int/features/qa/42/en/index.html

[2] S. Roberts, T. Basi, A. Drazin and J. Wherton, "Connections: Mobility and Quality of Life for Older People in Rural Ireland" Intel Corporation, America, 2007.

[3] Technology Research for Independent Living (TRIL) "Falls Prevention," 2010. http://www.trilcentre.org/falls_prevention/falls_prevention.474.html

[4] M. R. McGee-Lennon, "Requirements Engineering for Home Care Technology," *Proceeding of the* 26*th Annual SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, 2008, pp. 1439-1442.

[5] T. McBryan, M. R. McGee-Lennon and P. Gray, "An Integrated Approach to Supporting Interaction Evolution in Home Care Systems," *Proceedings of the* 1*st international Conference on Pervasive Technologies Related To Assistive Environments*, Athens, July 2008, pp. 1-8.

[6] A. Davis, O. Dieste, A. Hickey, N. Juristo and A. M. Moreno, "Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review," *Proceedings of* 14*th IEEE International Conference of Requirements Engineering*, Minneapolis, September 2006, pp. 179-188.

[7] H. Van Vliet, Ed., "Software Engineering Principles and Practice," John Wiley & Sons Ltd., Chichester, 2000.

[8] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," *Proceedings of International Conference on Software Engineering*, ACM Press, Limerick Ireland, 2000.

[9] K. Beck, *et al.* "The Agile Manifesto," 2001. http://www.agilealliance.com

[10] S. W. Ambler, "Agile Modeling: Effective Practices for Extreme Programming and the Unified Process," John Wiley & Sons Inc., Chichester, 2002.

[11] D. Astels, G. Miller and N. Miroslav, "A Practical Guide to Extreme Programming," Prentice Hall, New Jersey, 2002.

[12] H. Sharp, H. Robinson, J. Segal and D. Furniss, "The Role of Story Cards and the Wall in XP teams: A Distributed Cognition Perspective," *Proceedings of Agile*, IEEE Computer, Society Press, Washington, DC, 2006, pp. 65-75.

[13] W. Pietri, "An XP Team Room," 2004. http://www.scissor .com/resources/teamroom

[14] L. Cao and B. Ramesh, "Agile Requirements Engineering Practices: An Empirical Study," *Software, IEEE*, Vol. 25, No. 1, 2008, pp. 60-67.

[15] A. Martin, R. Biddle and J. Noble. "XP Customer Practices: A Grounded Theory," *Proceedings of Agile Conference*, *agile*, Chicago, August 2009, pp. 33-40.

[16] B. W. Boehm and R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods," *Computer*, Vol. 36, No. 6, 2003, pp. 57-66.

J. Software Engineering & Applications, 2010, 3, 894-900
doi:10.4236/jsea.2010.39105 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

Introduction to a Requirements Engineering Framework for Aeronautics

Robert Abo

Cedric Laboratory, Conservatoire National des Arts et Métiers, Paris, France.
Email: robert.abo@cnam.fr

ABSTRACT

*This paper introduces a framework to produce and to manage quality requirements of embedded aeronautical systems, called the 'Requirements Engineering Framework' (REF). It aims at making the management of the requirement lifecycle easier, from the specification of the purchaser's needs, to their implementation in the final products, and also their verification, while controlling costs. REF is based on the main standards of aeronautics, in particular RTCA DO-*254*, and RTCA DO-*178*B standards. An implementation of REF, using the IBM Rational DOORS and IBM Rational Change tools, is also presented in this paper.*

Keywords: *Aeronautics, Requirements Engineering, RTCA DO-*254 *and RTCA DO-*178*B Standards, V-Model*

1. Introduction

To ensure the safety and the reliability of the aircraft's embedded systems, airworthiness authorities (e.g. *US Federal Aviation Administration* [1], *European Aviation Safety Agency* [2], *UK Civil Aviation Authority* [3], etc.) require that they are built under control of processes based on international standards. Among these standards, the main two used in the civilian domain are the well-known RTCA DO-254 '*Design Assurance Guidance for Airborne Electronic Hardware'* standard (aka EUROCAE ED-80) [4] for hardware components and the RTCA DO-178 ed. B '*Software Considerations in Airborne Systems and Equipment Certification*' standard (aka EUROCAE ED-12) [5] for software components. They are referred to as the 'DO standards' throughout this paper.

In this article, we introduce the '*Requirements Engineering Framework*' (REF for short), which aims at producing and managing quality requirements, in order to produce safe and secure embedded aeronautical systems, that must adhere to the rigorous constraints of international standards, while controlling costs. This is achieved by using formalized and mature processes as presented in the following sections. The REF described in this article, does not refer to the practices of a particular supplier or a particular firm in aeronautics.

The rest of this paper is organized as follows. In Section 2, we present the basic notions of requirements management, which form REF foundations. Section 3 presents an implementation of REF, which uses the IBM Rational DOORS tool [6] to manage requirements and to carry out requirement traceability, and IBM Rational Change tool [7] to manage changes between work teams. Section 4 is dedicated to the safety activities, while Section 5 concludes this paper.

2. Requirements Management

2.1. System Lifecycle Model

DO-254 does not prescribe a preferred lifecycle model, nor imply a structure for the performing organization. In the same manner, DO-178B does not designate a preferred software lifecycle, but describes the separate processes that comprise most lifecycles and the interaction between them. The lifecycle for each project should be based on selection, and arrangement of processes and activities determined by the attributes of the project.

Several system lifecycle models exist in system engineering, with different approaches on the manner of leading a project to develop a system: waterfall, V-model, iterative, spiral, agile, and so on. Each one has its pros and cons, and it is up to the chief technical officer and project leaders to determine the most suitable model to lead the projects of their company.

REF is based on *V-model* [8] (aka "Vee model"), which is a variation of the waterfall model. This choice is explained by its advantages. First, it is simple, well organized, and easy to use and to implement. In particular, it highlights the correspondences between the development phases (*i.e.* the descending stages, from the early specification to the implementation) and the verification phases (*i.e.* the ascending stages, from the implementation to the product delivery). Thus, it facilitates not only requirement traceability, but also the production of the certification documents required by DO standards, as we will explain in the following sections. Another great advantage of V-model is it can be tailored into a specific project-oriented V-model, because it is independent from any organization and any project. It also provides assistance on the way to implement an activity, and it supports a wide range of development methodologies, in particular *formal methods* [9-11] often use to develop critical parts of systems.

Among disadvantages of V-model, it is project-oriented instead of addressing the development of systems within a whole organization. Another V-model disadvantage is it fails at covering the maintenance of systems. But, these disadvantages do not impact REF.

2.2. Basics

The concept of *requirement* is in the middle of systems engineering, as the abundant literature on the subject attests it [12-15]. We define a 'requirement' as a customer's elementary need that is to be implemented in the product or service that he receives[1]. In systems engineering, we can refine this rough definition by distinguishing the characteristics of the system to be built, known as the *functional requirements*, from the ways the system achieves its functions, known as the *non-functional requirements* (e.g. performance, quality, interface requirements, etc.). We can also differentiate the customer's needs, from which the supplier's distributed requirements are issued, among three hierarchical levels, which are the system, the high-level and the low-level requirements sets. From now on, by "customer", we mean not only the purchaser of the building system, but also the supplier's teams who require services from other ones along an enterprise workflow dedicated to requirements management. Thus, we distinguish four main requirement levels according to their refinement level, plus a requirement implementation level as shown in **Figure 1**:

1) The 'purchaser's level' corresponds to the purchaser's specifications seen as a set of rough needs developed in the 'Purchaser Specification' (PuS) document.
2) The 'system level': the purchaser's needs are refined and reformulated, by using technical terms understandable for the development teams. The system requirements are collected in the 'System Specification' (SyS) document. It is possible to refine this level, by considering a sub-level dedicated to the embedded equipment.
3) The 'high-level requirements (HLR) level'. The notion of sub-system appears, and hardware requirements are distinguished from software ones at this level. High-level requirements are developed from the analysis and refinement of system requirements, system architecture, safety-related needs and derived requirements. The latter correspond to requirements that are the result of the sub-system development process, and may not be directly traceable to high-level requirements. The HLR are collected in the 'Hardware Requirement Specification' (HRS) and the 'Software Requirement Specification' (SRS) documents.
4) The 'low-level requirements (LLR) level'. Low-level requirements are developed from the high-level requirements, sub-system architecture, and design constraints, by refinement and reformulation. The hardware and software subsystems are directly developed from the LLR. The LLR are collected in the 'Hardware Design Document' (HDD) and the 'Software Design Document' (SDD).
5) The 'implementation level' is the last level and marks the end of the descending phase of the V-model. It corresponds to the hardware components and the source code. The implementation of a requirement consists in giving this requirement an existence from its specification as it appears in the HDD (for hardware components) or in the SDD (for software components).

Requirements are fundamental. Firstly, the supplier's requirements formalize the customer's needs. The supplier ensures the comprehension of the customer's needs, that he has translated this into a form he can use without any misunderstanding. Secondly, requirements allow the identification of the characteristics of the customer's needs. Finally, requirements simplify the taking into account of customer's needs along V-model by formalizing them. They show the customer that the final product answers the needs he has expressed.

2.3. Requirements Specification

It consists of specifying the requirements. In particular, engineers have to define the bi-directional and vertical traceability between the upper and lower requirements. The main objective of the requirement traceability is to show that the purchaser's needs are satisfied by system requirements, high-level requirements, and low-level requirements; and then implemented into the hardware

[1]DO-254 defines a *requirement* as "an identifiable element of a specification that is verifiable" [4]. DO-178B defines a *software requirement* as "a description of what is to be produced by the software given the inputs and constraints" [5].

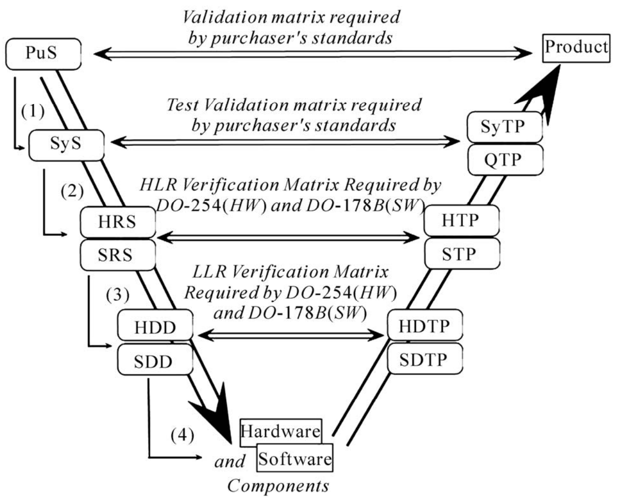

Validation matrix required by purchaser's standards
PuS
Product
(1)
SyS
Test Validation matrix required by purchaser's standards
SyTP
QTP
(2)
HLR Verification Matrix Required by DO-254(HW) and DO-178B(SW)
HRS
SRS
HTP
STP
(3)
LLR Verification Matrix Required by DO-254(HW) and DO-178B(SW)
HDD
SDD
HDTP
SDTP
(4)
Hardware
and Software
Components

Figure 1. The documents of a project, issued at the different stages of V-model with (1): System level requirement validation matrix; (2): Specification Analysis Matrix (DO); (3): Design Analysis Matrix (DO); (4): Hardware/Code Analysis Matrix (DO). The requirements are produced by successive refinements along the descending phases of V-model. In the figure, 'TP' stands for *Test Plans*, and 'Q' for *Quality*.

components or the source code.

2.4. Requirements Justification

All the supplier's requirements at any level have to be justified. A justification records the reasons for a requirement's existence, and its compliance with a customer's need. It also records the reasons for the implementation choices; and it keeps the analysis for the future designs and the modification assessments. Finally, it justifies the activities link to requirements, in particular the safety ones. Requirement justifications make the requirement analysis phase easier.

2.5. Requirements Review and Analysis

This phase is also referred to as "requirements validation". Its purpose is to ensure that all the customer's needs are specified (*i.e.* there is no under-specification of the customer's needs) and nothing more than these needs is specified (*i.e.* there is no over-specification of the customer's needs). Moreover, this analysis consists in ensuring that the requirements at each level are good and well-specified requirements, *i.e.* they are sufficiently correct, complete, unambiguous, consistent, self-contained, achievable, verifiable, etc., so the delivered product will meet all the customer's needs and airworthiness authorities' constraints including DO requirements.

We must notice that whether the writers and the reviewers are the same engineers, they cannot perform the validation of the requirements they specified, in particular for the requirements of the most critical software referred to as Level A or Level B by the DO-178B standard[2]. Project managers and team leaders must organize the work of the engineers taking this into account. A specific team performs the safety activities as described in Section 4.

2.6. Requirements Verification

This activity deals with the rise of V-model. It consists in evaluating the implementation of the supplier's requirements to determine, whether or not, they have been met. There are several means of verification: tests, code analysis, model checking, simulation, etc. For aeronautics real-time embedded software, the low-level requirements are often implemented by using the Esterel Technologies' SCADE Suite [16]. This tool complies with DO-178B, and allows for generation of a certified source code from low-level requirements without any unit tests.

3. Implementing REF

The REF processes are implemented through two main tools namely: IBM Rational DOORS [6] for the management of requirements, and IBM Rational Change [7] for the management of changes impacting requirements. This choice and the use of these tools are not mandatory, and other ones with similar functionalities can be used, according to the final customer's choices. Reviewing all of them is out of the scope of this paper, but we can quote Geensoft's Reqtify [17] or IBM Rational RequisitePro [18] as other examples of requirements management tools. IBM Rational ClearQuest [19] and Serena TeamTrack [20] are other examples of change management tool.

3.1. Requirements Management

DOORS is a requirements management tool that provides an easily collaborative environment, to make the achievement of processes linked to the specification, the analysis, the verification and the traceability of requirements easier.

3.1.1. Data Organization

Data is stored in DOORS databases, each of which are organized as folders, projects and modules. Projects are specific folders that contain data related to a particular project. They can contain folders and sub-folders, both contain modules. We define a module as a collection of objects with attributes, each of which relate to a particular topic. Each module has its own attributes as name, type, description, date of creation and so on. Different

[2]Software level is based upon the contribution of software to potential failure conditions as determined by the system safety assessment process. Their effects on the aircraft, the crew and the passengers categorize the failure conditions. They spread out from 'A' (catastrophic effects), to 'E' (no effects) [5].

kinds of modules can be defined.

Each project should contain at least:

1) Modules for customer specification;
2) Modules for system, high-level and low-level requirements;
3) Modules for applicable standards, documents, and plans;
4) Modules for requirement verification (test cases, test procedures, results, and analysis);
5) Modules for requirement justification;
6) Modules for requirement validation.

Within a module, objects can be organized in a hierarchical manner. Information is displayed through views that can filter attributes according to user choice. Objects can be linked together, in particular hierarchical objects, which is very important to define objects traceability. It is possible to define several kinds of objects:

1) Requirements collected in the specification modules;
2) Validation objects collected in the validation modules;
3) Justification objects collected in the justification modules;
4) Verification objects collected in the verification modules;
5) Other objects in particular texts, that can contains titles, notes, remarks or any other textual explanations that are not requirements but are useful to understand the specifications. Indeed, we must keep in mind that these modules can be published as official documents for the purchaser and the end users.

DOORS administrators can regularly create module baselines, which are frozen modules that cannot be modified. They record the history of the module since its last baseline, including information about objects, their attributes, and also module sessions.

3.1.2. Documents Issues

DOORS allows exporting a module into several formats, that can be Microsoft Office, HTML, FrameMaker, etc. This functionality is particularly interesting to deliver definitive documents to purchasers. It is possible to choose the attributes to be printed on documents extracted from DOORS modules. In that case, the text of the requirement is automatically put between the identification of the requirement and the ‘End of Requirement’ tag. The attributes to be printed should be, at least:

1) The requirement identifier;
2) The requirement text;
3) The upper requirement(s) covered by this requirement;
4) The delivery version of the product where this requirement appears.

3.2. Change Management

3.2.1. Basics

The configuration management process is interfaced with IBM Rational Change [7]. Specifications, test cases, test procedures and any documents are managed with DOORS. Change is a web-based tool for change management solutions, allowing teams involved in the system development to get together. Across the enterprise, it tracks change requirement requests.

3.2.2. Process Description

Updates of requirements, justification, and validation objects are decided by a committee. They are only authorized through a change management process described in the following text. Each modification or evolution need is recorded through a *Specification Change Request* (SCR) that details the origin of the evolution, the standard of applications and the evolution need. This SCR can lead to several *Requirement Change Requests* (RCR), each of them impacting one or several requirements of a specific module. The Change tool traces the links between an SCR and its RCRs. Each RCR is realized in DOORS. Thus, each requirement modification must be traced with the relevant RCR. Once the SCR is approved in commission, the requirement or procedure is then proposed for the validation process. An SCR or an RCR can be reworked, if conflicts are detected. The SCR manager can close an open SCR after having checked it:

1) All impacted requirements have been validated;
2) All modifications are well traced in DOORS;
3) All verification modules have been updated;
4) All impacts on lower and upper requirements have been taken into account;
5) All justifications have been updated;
6) All impacts on previous standard specification have been taken into account;
7) The standard of applicability has been clearly identified.

Figure 2 shows the SCR and its associated RCRs life-cycles, with the corresponding processes enabling to pass from a stage to another.

3.3. Requirements Documentation

Some attributes are generic and DOORS automatically manages them. These usually are the object identifier, its date of creation, its date of last modification, the name or the user identification, etc. The object identifier is unique, and must contain the identification of the module that the requirement belongs to, and a number. The module identifiers should be, at least: SYS for ‘System’, HW for ‘Hardware’, SW for ‘Software’, SAF for ‘Safety’, VAL

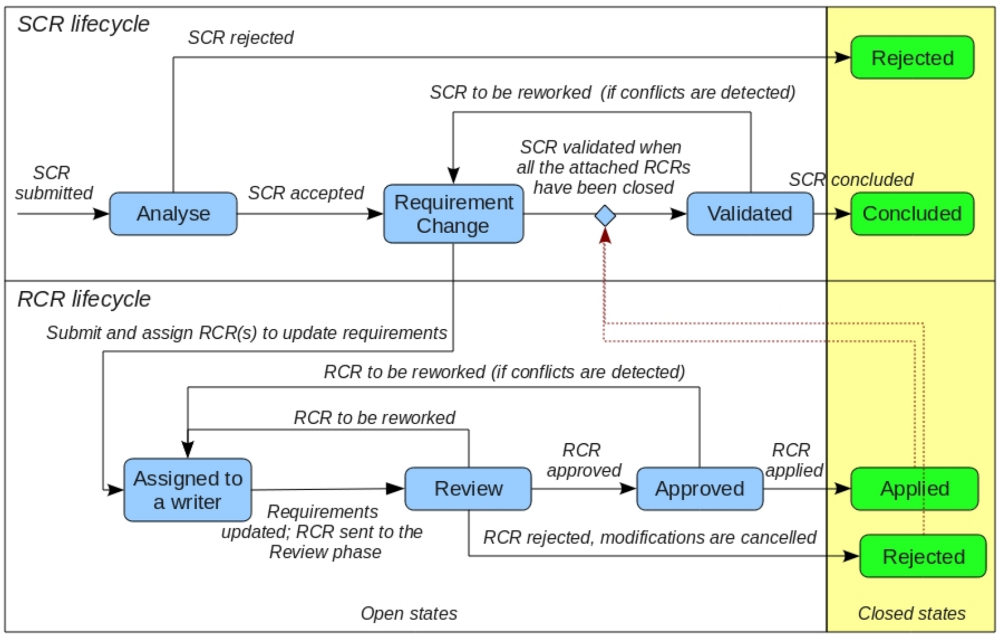

SCR lifecycle
SCR rejected
Rejected
SCR to be reworked (if conflicts are detected)
SCR validated when all the attached RCRs have been closed
SCR submitted
Analyse
SCR accepted
Requirement Change
Validated
SCR concluded
Concluded
RCR lifecycle
Submit and assign RCR(s) to update requirements
RCR to be reworked (if conflicts are detected)
RCR to be reworked
RCR approved
RCR applied
Assigned to a writer
Review
Approved
Applied
Requirements updated; RCR sent to the Review phase
RCR rejected, modifications are cancelled
Rejected
Open states
Closed states

Figure 2. the SCR and RCRs lifecycles for REF. In general, several RCRs are associated to one SCR. RCRs permit to trace requirements updates.

for 'Validation', JUS for 'Justification', and others necessary identifiers as, for example, QLY for 'Quality', PRG for 'Programs', etc. For requirements, there must be the following major attributes. They have an impact on the validation status.

1) A main description to describe the requirement. It may contain drawings, tables, figures or mathematical formulas.
2) An assumption or a set of assumptions for the requirement, if any. Assumptions must be identified, justified, and validated.
3) The domain of activity, for example, SYS for 'system' level, HW for 'hardware', or SW for 'software' level.
4) The type of requirement: 'derived' requirements, which are the results of the sub-system development process and may not be directly traceable to high-level requirements. A 'terminal' requirement cannot be traced to lower levels. A 'normal' requirement is neither derived nor terminal.
5) The delivery version of the system in which the requirement appears (for example V0, V1.0, V1.1, etc.). It is possible to qualify a version as 'partial' to indicate requirements are partially implemented in it.
6) Links to requirements not under the DOORS control.

Even if it is obsolete, a requirement must never be deleted. This basic rule is necessary to avoid losing traceability and to keep a trace of its existence. Besides, this deletion must be justified in the justification object linked to the deleted requirement.

Low-level requirements have specific attributes as the identification of the function that calls it, the description of its input and output parameters, etc., plus a data dictionary in which all data, types, variables, constants, and definitions of applications are defined.

3.4. Requirements Justification

The DOORS justification module embeds three categories of justification objects expected for certification issues:

1) Justification of all the requirements (normal, derived, and terminal).
2) Justification of the validation of requirements.
3) Justification of safety assessment of derived requirements.

As far as possible, the requirement justification process must be complete before entering the requirement validation phase as the latter contains a checklist of criteria to ensure completeness and correctness of this activity.

3.5. Requirements Review and Analysis

We perform two kinds of requirement analysis: the transversal and the unitary analysis.

3.5.1. Unitary Analysis

It is requirement-oriented. The requirement conformity with the DO standard criteria applicable to requirements is checked using DOORS. All requirements are analyzed one by one: the system requirements; the hardware high-level and low-level requirements (DO-254 Subsections 6.1.2.2, 6.1.2.4 and 6.1.2.5) and also the software high-level and low-level requirements (DO-178B, Subsections 5.3.2 and 6.3.1). We check the quality of each requirement *i.e.*:

1) Its *adaptability* for its level of specification, e.g. no detailed requirement at system or high level, or no rough and non refined requirement at low level (requires by DO-178B Subsection 5.1.2 g for SW);
2) Its *completeness* with no missing information, in particular, concerning the acceptance criteria (requires by DO-254 Subsection 6.1.2.4 for HW and DO-178B Subsections 6.3.1a, b, d and 5.1.2 f for SW);
3) Its *correctness* by expressing a need and not a solution for that need; if possible, the contrary must be rigorously justified (requires by DO-254 Subsection 6.1.2.5 for HW and DO-178B Subsection 5.1.2 g for SW);
4) Its *consistency* by not being contradictory with other requirements of the same level (requires by DO-178B Subsection 6.3.1 b);
5) Its *feasibility* by checking it can be implemented on the target architecture (requires by DO-254 Subsection 6.1.2.5 for HW, and DO-178B Subsection 6.3.1 b, c, d for SW);
6) Its *unambiguity* and *precision* by checking that nobody can interpret it (requires by DO-254 Subsection 6.1.2.5 for HW, and DO-178B Subsection 6.3.1 b and d);
7) Its *verifiability* by checking that its verification is possible (requires by DO-254 Subsection 6.1.2.5, and DO-178B Subsection 6.3.1 b, d for SW);
8) Its *traceability* by checking links with upper and lower requirements (requires by DO-254 Subsection 6.1.2.4 and DO-178B Subsection 6.3.1 a);
9) Its *conformance to standards* (requires by DO-178B Subsection 6.3.1 e);
10) Its *algorithms* (if any) must be *accurate* and *correct* (requires by DO-178B Subsection 6.3.1 g);
11) Its *topicality* by checking it does not refer to an obsolete part of the system.

NB. Software scripts can be used to check general rules automatically, that major attribute fields are not empty, editing requirement rules are complied with, etc. For this, each attribute must be correctly filled in.

3.5.2. Transversal Analysis

It is document-oriented. The DO standard criteria applicable to a document are used to validate the whole document from a quality point of view. It consists in checking several points among which:

1) Its availability and its consistency;
2) Its compliance with the purchaser and airworthiness standards;
3) The completeness of its references;
4) Its readability;
5) Its compliance and traceability with upper documents if any;
6) Its correctness, completeness and accuracy;
7) Its compliance with development standards;
8) Its maintainability.

3.6. Requirements Verification

Each requirement is associated to one or more test cases, with each of them specifying the test objective with a description. If the test case defines a test of the product (laboratory, vehicle, flight, environment, etc.) then a script or detailed procedure and the associated test results shall be written. If the test case is defined by analysis, a detailed procedure is used to reach the test result. Test cases shall only specify the objective of the analysis. Test results shall contain the full analysis and the result status for each standard. Then three levels of verification modules are provided:

1) The test case level aiming at containing test case(s) covering requirements. A tests case describes test sequences, objectives, input/output conditions, required environment and accepted criteria from a general point of view: no implementation details linked to test benching or particular tools need to be described, unless there are particular constraints.
2) A detailed test procedure or script level that is the implementation of test cases with regards to test bench facilities, software capacities, specific tools to be used, or other precise implementation details required to ease test runs and avoid mistakes in test procedure execution. Test scripts are dedicated to automated procedures and detailed procedures to manual tests. Both can be used for tests requiring manual sequences. For test cases by analysis, the detailed procedure is used to reach the test result.
3) A test result level containing all the verification results.

4. Safety Analysis

The safety activities are exclusively related to the needs impacting the safety of the system to be built. They affect the documentation, the justification, and the validation of safety-related requirements. An independent team of engineers, referred to as the “safety team”, performs the safety activities, that are based on the analysis of all the safety-related requirements (normal, terminal, and derived) that contribute to reach the customer’s safety needs. A set of safety-oriented attributes is defined for each requirement.

4.1. Safety Activities in Specification Modules

A special attribute should be used to mark any safety-related requirement. It must adhere to the lower requirements in order to identify requirement trees that need a safety analysis precisely. If a requirement is not safety-related, its attribute shall be set to ‘NO’. Safety teams shall be specially warned of every evolution of this attribute for each requirement. All updates of this attribute for any requirements must imply a new safety validation phase. When it is set to ‘YES’, this attribute must be visible in the published version of specifications.

4.2. Safety Activities in Justification Modules

Different attributes should be used to justify the safety aspect of a requirement. The first attribute should state whether a requirement has an impact on the safety analysis and must require special attention. The second should detail the reasons why the previous attribute was filled as ‘YES’. Another one should detail the analysis performed by the safety team in order to comply with the safety objectives. Some other justification attributes should be added.

4.3. Safety Activities in Validation Modules

Only the safety team fills out the attributes of these objects. They should record at least, the accepting of the requirement in accordance with the safety criteria, the reasons of the acceptance or the rejection, the name of the engineer who performed the validation, and the date of the validation in order to ensure it is still current.

5. Conclusion

This paper presents a general framework, which we have called the “Requirements Engineering Framework” or REF for short, dedicated to the management of requirements of aeronautical systems, during their whole lifecycle. It aims at producing quality, secure and safe systems in accordance with the rigorous DO constraints, while controlling manufacturing costs. This framework can be implemented in several ways according to the specific needs of suppliers. In this paper, we have outlined the interests of using DOORS [6] and Change [7] tools to implement REF.

In a future paper, we envisage to describe the possible implementations of REF in greater detail.

6. Acknowledgements

I would like to acknowledge the discussions and suggestions from different persons including my work colleagues and my friends. I would like to acknowledge especially Prof. Kamel Barkaoui of the *Conservatoire National des Arts et Métiers* in Paris, France, who encouraged me to write this paper.

REFERENCES

[1] “US Federal Aviation Administration”. http://www.faa.gov/

[2] “European Aviation Safety Agency”. http://www.easa.eu.int

[3] “UK Civil Aviation Authority”. http://www.caa.co.uk

[4] RTCA DO-254 (EUROCAE ED-80), “Design Assurance Guidance for Airborne Electronic Hardware,” 2000.

[5] RTCA DO-178B (EUROCAE ED-12B), “Software Considerations in Airborne Systems and Equipment Certification,” 2nd Edition, 1992.

[6] “IBM Rational DOORS”. http://www-01.ibm.com/software/awdtools/doors/

[7] “IBM Rational Change”. http://www-01.ibm.com/software/awdtools/change/

[8] “Das V-Modell”. http://v-modell.iabg.de/ (some pages about the fundamentals of V-model are in English).

[9] C. M. Holloway, “Why Engineers Should Consider Formal Methods,” *Proceedings of the* 16*th Digital Avionics Systems Conference*, *Irvine*, California, October 1997.

[10] N. A. S. A. Langley, “Formal Methods web site”. http://shemesh.larc.nasa.gov/fm/

[11] “Formal Methods in System Design Journal,” Springer. http://www.springer.com

[12] R. R. Young, “Effective Requirements Practices,” Addison Wesley, Boston, 2001.

[13] K. E. Wiegers, “Software Requirements,” 2nd Edition, Microsoft Press, 2003.

[14] K. E. Wiegers, “More About Software Requirements,” Thorny Issues and Practical Advice, Microsoft Press, 2006.

[15] V. I. Fort Belvoir, “Systems Engineering Fundamentals,” Defense Acquisition University Press, USA, 2001.

[16] “Esterel Technologies SCADE Suite”. http://www.esterel-technologies.com/products/scade-suite

[17] “Geensoft Reqtify”. http://www.reqtify.com

[18] “IBM Rational RequisitePro”. http://www-01.ibm.com/software/awdtools/reqpro/

[19] “IBM Rational ClearQuest”. http://www-01.ibm.com/software/awdtools/clearquest/

[20] “Serena TeamTrack”. http://www.serena.com/products/teamtrack/change-request-management.html

J. Software Engineering and Applications, **2010, 3, 827-900.**
Published Online September 2010 in SciRes (http://www.SciRP.org/journal/jsea).

TABLE OF CONTENTS

Volume 3 Number 9 September 2010

What's Wrong with Requirements Specification? An Analysis of the Fundamental Failings of Conventional Thinking about Software Requirements, and Some Suggestions for Getting it Right
T. Gilb.......... 827

Benefits Management Process Complements Other Project Management Methodologies
I. Karamitsos, C. Apostolopoulos, M. A. Bugami.......... 839

Requirements Analysis and Traceability at CIM Level
M. Yamin, V. Zuna, M. A. Bugami.......... 845

A Review of the Impact of Requirements on Software Project Development Using a Control Theoretic Model
A. White.......... 852

IT Project Environment Factors Affecting Requirements Analysis in Service Provisioning for the Greek Banking Sector
K. Maroukian.......... 858

Requirements Analysis: Evaluating KAOS Models
F. Almisned, J. Keppens.......... 869

Model-Driven Derivation of Domain Functional Requirements from Use Cases
J. M. Guo, Z. Y. Zhang, Y. L. Wang.......... 875

Towards Lightweight Requirements Documentation
Z. Y. Zhang, M. Arvela, E. Berki, M. Muhonen, J. Nummenmaa, T. Poranen.......... 882

Investigating the Suitability of Agile Methods for Requirements Development of Home Care Systems
S. Kelly, F. Keenan.......... 890

Introduction to a Requirements Engineering Framework for Aeronautics
R. Abo.......... 894