

Managing Parallel Requirements

Ken Jackson, Sean Innes, Chris Hardy, Telelogic

Version 0.5

11 July 2007

Table of Contents

Acronyms and Terms	1
Executive Summary	2
Introduction	3
Portfolio Management	5
Version/Variant Evolution	6
Release Planning	9
Release Management Planning	10
Without a Release Manager	11
With a Release Manager	11
Release Manager Challenges	11
Conclusion	12
Appendix One	13
Capability Maturity Model (CMM)	13
Levels of the CMM	13
Level 1 – Initial	13
Level 2 – Repeatable	13
Level 3 – Defined	13
Level 4 – Managed	14
Level 5 – Optimizing	14
CMMI Superseding CMM	14
About Telelogic	15

Acronyms and Terms

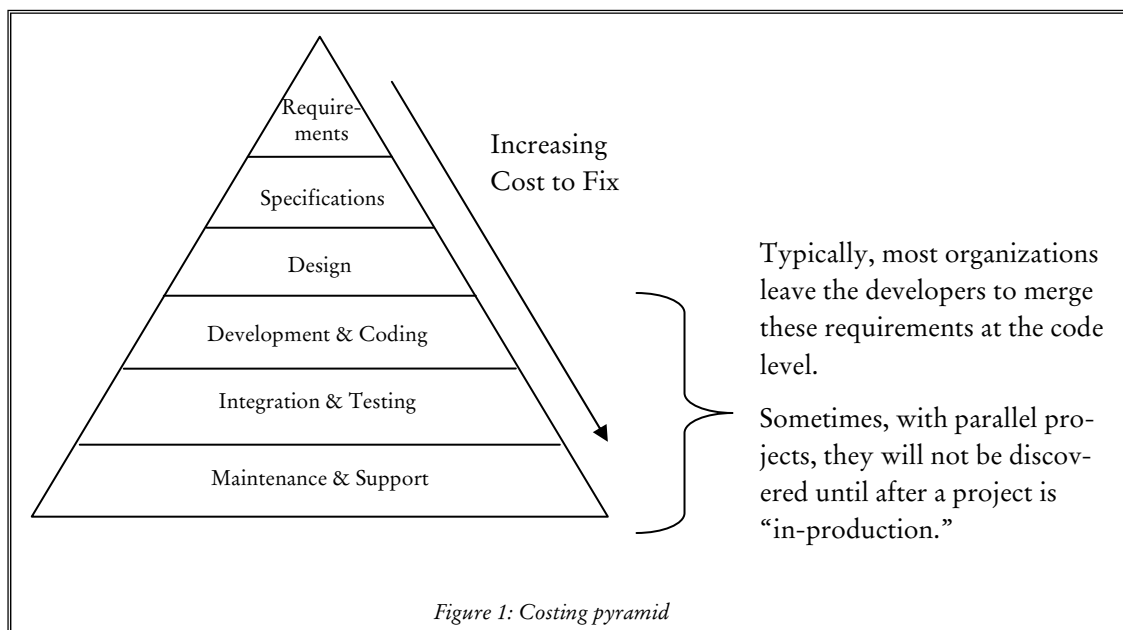
The following acronyms are used throughout this white paper and have been defined below for easy reference:

Term	Description
CCB	The Change Control Board is a committee that decides whether or not proposed changes to a project should be implemented. The CCB is constituted of project stakeholders or their representatives. The authority of the CCB may vary from project to project, but its decisions are often accepted as final and binding.
CMM	The Capability Maturity Model refers to a process improvement approach that is based on a process model. CMM is defined by levels. More information is available in Appendix One.
CMMI	The Capability Maturity Model Integration has started to supersede CMM with the goal of improving usability of maturity models for software engineering and other disciplines by integrating many different models into one framework. More information is available in Appendix One.
V-Model	The V-Model , also referred to as the VEE Model , defines a uniform procedure for product development. In systems engineering (and product development as well), the <i>VEE</i> is a process that represents the sequence of steps in a project lifecycle. The left side of the <i>VEE</i> represents the decomposition of requirements and creation of system specifications. The right side of the <i>VEE</i> represents integration of parts and their verification.
Portfolio Management	A business process by which a business unit decides on the mix of active projects, evaluation, prioritization, staffing, and dollar budget allocated to each project. Through portfolio management, the organization can explicitly assess the tradeoffs among competing investment opportunities in terms of their benefit, costs and risks.
Release Manager	A dedicated resource to coordinate and oversee the technical or day-to-day aspects of the project(s) and flow of development, testing, deployment and support of these systems.

Executive Summary

In today's business environment, requirements are constantly changing. With these changing requirements, the management of parallel requirements is the key to organizational and project success. The planning and scheduling of projects with related capabilities therefore needs to be carefully considered and requires a wider knowledge of the organization's goals and strategies.

If different projects are concurrently modifying the same set of base requirements, the process of merging these differences will be more costly and complex the later in the project lifecycle they occur. Figure 1 illustrates this:



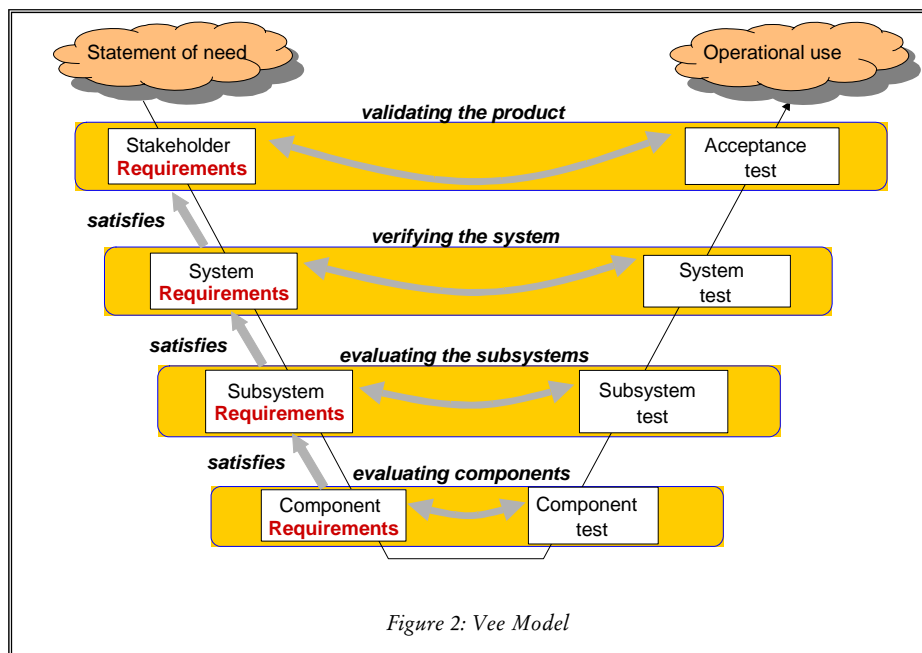
If an organization does not already have one, it should introduce a CCB to assist in the prioritization of changes to an on-going project. The CCB, along with the key sponsor(s), needs to have the power to make the required decisions and ensure the merging of these requirements is performed at the upper-most level of the pyramid that is applicable. As a result, the organization can reduce costs, avoid re-work, improve alignment with other projects and decrease time-to-market.

When the number of parallel projects increases, the organization must instigate the role of a **release manager**. It is the release manager's responsibility to manage conflicting interests that project(s) and project managers have.

Effective project and release planning fundamentally supports portfolio management and ensures that projects can take into account the wider picture, while at the same time reducing costs and delivering faster when time-to-market is a critical factor.

Introduction

Within the engineering industry, requirements management has traditionally been applied within a single project. In such a project, there is a set of high level or stakeholder requirements. These requirements are satisfied by a single set of system requirements that, in turn, are satisfied by lower level requirements or specifications and so on. This type of decomposition can be seen in the left-hand side of the well-known Vee Model, as shown in Figure 2.



This model also needs to integrate user-involved opportunity and risk management, as well as problem resolution. Additionally, the process of decomposition analysis and resolution and the process of verification analysis and resolution need to be taken into account. This approach provides a comprehensive model representing the integrated processes associated with system development and the need for system integrity.

The need to manage change complicates the situation and creates the need to assess the impact of a change whether the change comes from the top (e.g., a stakeholder changing his or her mind) or whether the change comes from the bottom (e.g., from an implementation difficulty).

The issues that arise from this situation can be addressed by an organization or project adhering to the CMM. The CMM by definition introduces concepts such as traceability and repeatability to a project or organization. (A definition of the CMM and the CMM levels has been provided in Appendix One.) However, the implementation of processes and procedures that need to be put in place to support CMM should not be underestimated because they can be quite complex, especially when taking into account the idea of traceability.

An increasing number of commercial organizations—especially consumer products companies and service organizations such as banks—have to address these scenarios. Typically, these organizations have a number of projects under simultaneous development to create alternative versions of products or to provide enhanced facilities for the service company. Such organizations often rely on individuals and champions to bridge the gap between any process deficiencies.

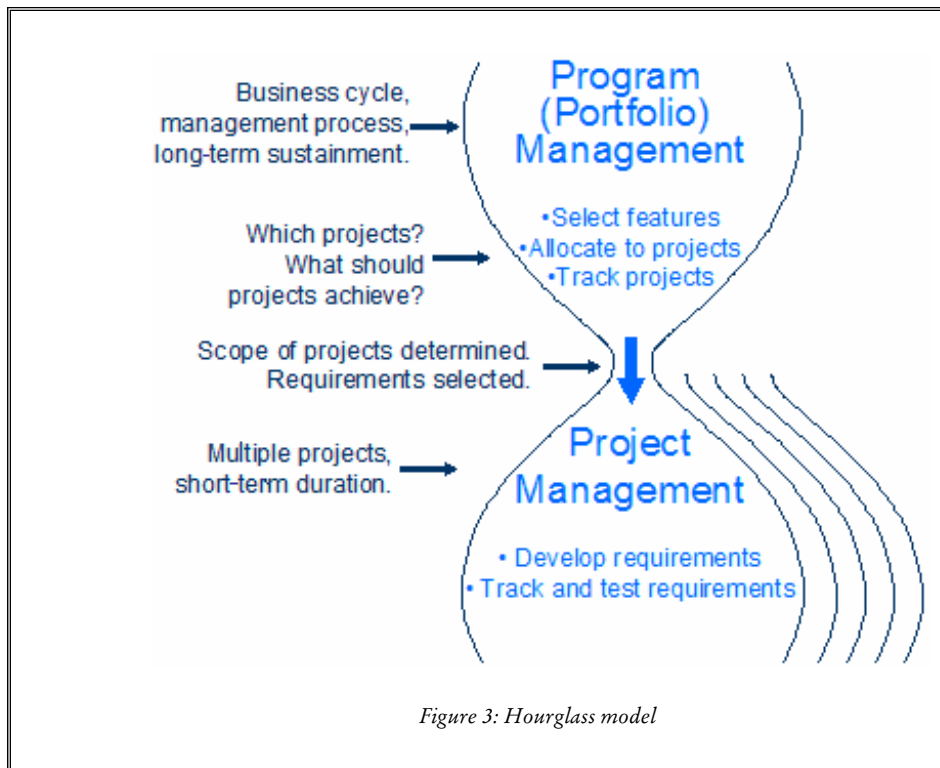
In these circumstances, the role of requirements management is much broader than a single project and must take note of the issues that are raised when concurrent projects are working on the same set of requirements or actively developing different versions of the same requirements. This wider requirements management role is referred to as portfolio or program management.

In these circumstances, managers must be aware that concurrent projects may interfere with each other because they may be attempting to simultaneously modify the same requirements. Therefore, the management of change within and across projects has to be much more sophisticated. If change management is not considered, then problems will arise much later, typically in code development, testing or maintenance, which increases the cost impact on the organization and project.

Because the number of projects simultaneously touching the same requirements is increasing, projects and organizations must be prepared to radically change or adapt their existing process and procedures.

Portfolio Management

To show this wider view of requirements management, we adopt the “hourglass” model, as shown in Figure 3, instead of the Vee Model.



The upper part of the hourglass indicates the corporate-level view of the organization and its concern with long-term planning and business sustainability. In this area, there will be many conflicting views and ideas about how the organization should evolve. It is the task of portfolio management to decide the business goals and strategies and to determine priorities among the competing views and ideas.

The neck of the hourglass represents the embodiment of the prioritization process and is usually driven by investment decisions. These priorities often represent trade-offs between benefits, costs, time and risks.

Each decision leads to the creation of one or more projects, the successful completion of which will enable the business to develop in accordance with corporate priorities. These projects will consume corporate resources (e.g., staff and capital) and create new products or services that can then be used to earn revenue for the business.

The lower part of the hourglass represents the set of projects that have been authorized by the corporate management and a project manager manages each of these projects in the normal way.

Each project must have the following:

- Allocated corporate resources, such as people and money
- Specific objectives in terms of a set of requirements that must be satisfied
- An end date by which the project should be completed

The problem of parallel requirements occurs when two or more of the projects are addressing the same requirements or even changing the same requirements. This leads to the complex process of merging requirements or specifications in order to maintain the current “as-is” view of the system.

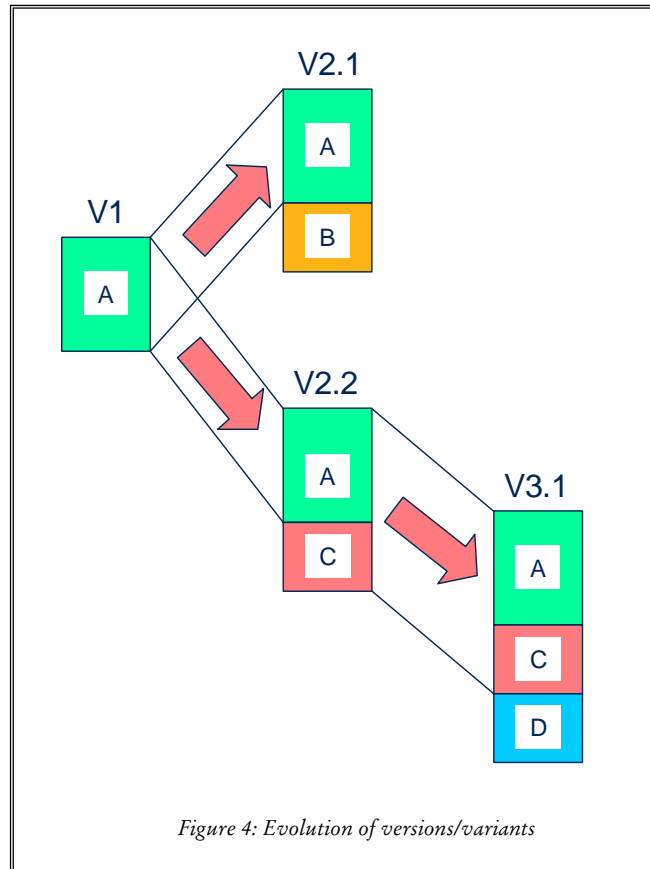
This “as-is” view is extremely important given that it is a major input into the initial analysis of system changes while also providing essential information to the organization’s enterprise architecture.

Version/Variant Evolution

This section examines the issues of parallel development in the context of versions and variants.

Figure 4 shows a set of variants that evolve in a particular way¹. The assumed starting point is the variant marked V1. This contains a set of requirements marked as A. Two variants are derived from V1 and both use A as defined in V1. Variant V2.1 adds a set of requirements labeled B to produce its end product whereas variant V2.2 adds a different set of requirements labeled C. Finally, Figure 4 indicates that a subsequent variant V3.1 is derived from V2.2. This variant takes the A developed by V1 (and used unchanged in V2.2), plus the set C developed by V2.2 and adds the requirements set D.

¹ It is not really necessary to distinguish between version and variant here. In general, one variant is different in some significant way to another variant. A set of versions tends to be sequence of successive approximations to the same end goal.



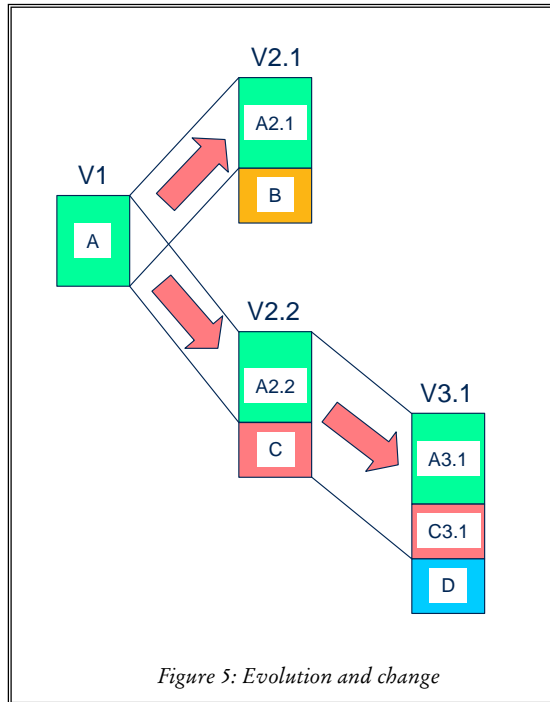
This is a simple way in which variants can evolve. However, even in this simple model it is necessary to ask the question:

- What happens if requirement set A is modified for a subsequent version of variant V1?

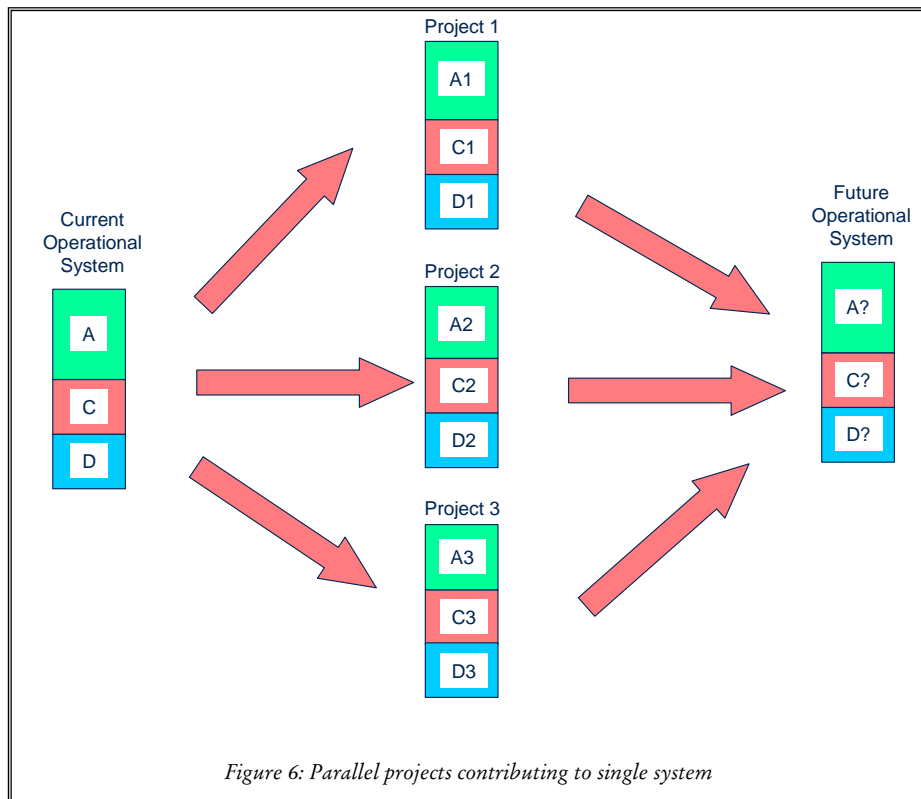
Possible answers include:

- Inheriting variants get the change immediately.
- Inheriting variants know there has been a change but cannot immediately see it.
- Inheriting variants have a choice of whether to accept the change, defer acceptance or ignore it.
- Nobody sees it because the new version is different from the point of inheritance.

The simplicity of Figure 4 is indicated by the fact that nothing that has been taken from when a previous variant has changed. A more realistic evolution is shown in Figure 5, where each variant modifies the requirements that were taken as its starting point from a previous variant. Such changes can involve altering requirements, adding new requirements, and deleting requirements.



The possible scenarios shown in Figure 4 and Figure 5 assume that the variants will never be merged together. However, in some organizations, and in particular service organizations such as banks, there is really only one product, and so the results of a set of parallel projects have to become part of the system on completion. One possible scenario for this is shown in Figure 6.

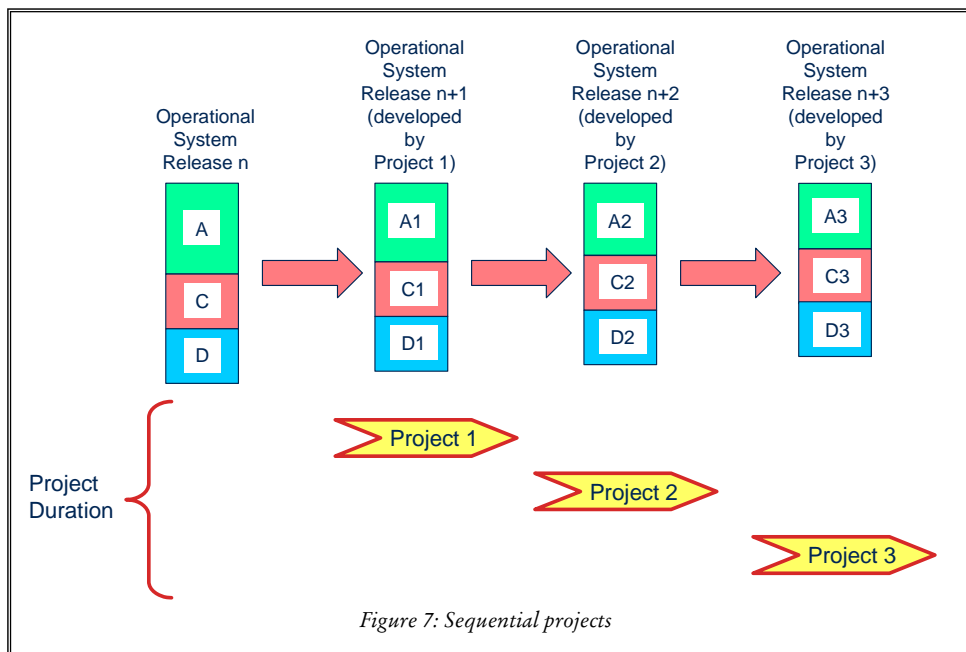


Here we see three parallel projects starting from the current operational version of a system. Each project works independently and modifies all three components. On completion of the three projects, a new operational system must be created from the results of the three projects. However, as indicated in Figure 6, it is far from clear what the final system consists of.

Clearly, such a situation is not one to be recommended! Instead, what is required is a plan for how the operational system will evolve, and this leads to the concept of release planning, which is a vital component of portfolio management.

Release Planning

In an ideal world, each project should have a single predecessor and a single successor, and each project would not start until its predecessor had finished (see Figure 7). Unfortunately this ideal world is not realistic. Therefore, a compromise has to be struck.



In general, projects have to overlap, and therefore, there is a possibility of two or more concurrent projects changing the same requirements. As indicated earlier, each project makes a contribution to the operational system. For the simplistic example above, we will assume that a project can only make a single contribution and that this can only be made when it is complete. Since projects have different durations, the only way that a plan can be made is for the date of contribution to be set (i.e., the end date of the project). At this point, a set of facilities can be made available to the operational system. (It is irrelevant whether these are new or changed facilities or even removed facilities.)

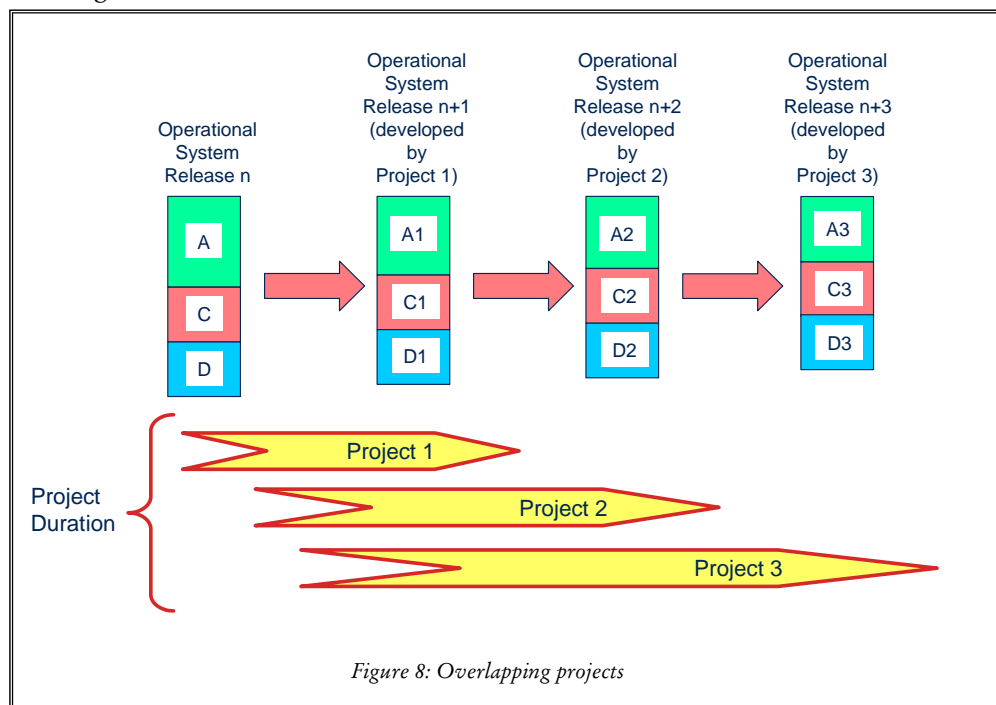
Therefore, from a planning point of view, it is necessary to determine when each set of facilities will be added to (or removed from) the operational system. The version of the operational system that is generated is called a release. In general, there will be a se-

quence of releases, each building on the previous release. Typically, this planning will be undertaken by a release manager and is known as release management planning.

Release Management Planning

A release manager (or in this case it could be referred to as a requirement manager) is responsible for managing how, when and in what order these projects should be progressing.

The release manager should ensure that the release plan reflects the prioritization of the project, as each project that is contributing to the next release needs to take precedence over those that are contributing to later releases. It is this precedence relationship that enables the organization to control how and when updates to the requirements sets can take place. Figure 8 shows the same set of projects as Figure 7; however, in this scenario the project durations overlap in time. This overlap is what needs to be managed by the release manager.



This situation clearly illustrates the potential conflict between the three projects. The conflicts should be resolved by looking at them in the context of the releases they are contributing to. Project 2 contributes to Release n+2 and therefore must take the version of the requirements generated by Project 1 for Release n+1. Similarly, Project 3 must take the version of the requirements developed by Project 2 for Release n+2 because they also contribute to Release n+3.

In reality, the versions of the requirements to be used by the later projects may not be ready when those later projects need them. The project manager must therefore observe what is happening to these base requirements since they are the foundation for the preceding projects. It is also possible that in later projects, requirements may need to be changed due to requirements changing in an earlier project. This means that the full im-

part of any change required by the later project can only be determined when there has been a commitment to a set of requirements in the earlier project.

Without a Release Manager

In the event that there is a conflict between the changes required by each project and an organization does not have a release manager, the conflict must either be resolved between the project managers or escalated to a higher-level CCB. Such a CCB will operate at the level of the portfolio and release management, which are above and across all active projects.

With a Release Manager

Alternatively, if an organization actively engages in release management, a release manager should perform this activity. It is the release manager's primary job to manage the conflicting interests that projects and project managers have. This is achieved by the release manager performing the following roles:

- **Facilitator** – Serves as a liaison between varying business units to guarantee smooth and timely delivery of software products or updates
- **Gatekeeper** – “Holds the keys” to production systems/applications and takes responsibility for the inclusion of new or modified implementations
- **Architect** – Helps to identify, create and/or implement processes or products to efficiently manage the release of code
- **Coordinator** –Coordinates disparate source trees, requirements, projects, teams and components
- **Communicator** – Interfaces with sponsors where required

Release Manager Challenges

Some of the challenges facing a release manager include the management of:

- Software Defects – found either in a release or during a testing phase
- Issues – related to system resources, etc.
- Risks – derived from scheduling etc.
- Changing Requirements – related to changing stakeholder requirements
- New Development Requests – for additional features and functions
- Deployment and Packaging – the delivery and rollout of the system

Conclusion

In summary, the organization seeking to manage parallel requirements should ensure they are process mature and ready for n combinations, m variants and various deployment scenarios.

With this in mind, the following actions should be taken:

- Setup of the processes and tools to manage requirements and control the updates and change management of editing these requirements. When doing this, the following items should be addressed:
 - The need for process change
 - The need for organizational change
 - The need to create a baseline or define a common source of “truth” for the system based on the gathering of all existing information including business rules
 - The need to educate everybody in the organization so they understand their role and the roles of others—with respect to the operational system or set of products, internal processes and procedures, tools and industry best practices, and especially the need to adopt strict guidelines on the production of requirement/specification statements
 - The need to understand how projects overlap in time and how the precedence of one project with respect to others is governed by organizational priorities and objectives
- Evaluate the organization’s current maturity with respect to CMM.
- Set in place initiatives to ensure that the organization can, where necessary, increase its level of CMM maturity so that it can address projects with improved repeatability, traceability and less reliance on project heroes and key “indispensable” project personnel.
- Ensure that the role of release manager is understood and that the people who undertake the role are empowered to make the necessary project-level decisions, acting with the following responsibilities:
 - **Facilitator** to serve between various business units and time the delivery of software products and/or updates
 - **Gatekeeper** to “hold the keys” to production systems/applications and take responsibility for the inclusion of new or modified implementations
 - **Architect** to help identify, create and/or implement processes to handle parallel requirements and manage the release of projects

- **Coordinator** to cast an impartial view on the importance of delivery for varying projects
- **Communicator** to interface with sponsors where required

Appendix One

Capability Maturity Model (CMM)

The CMM refers to a process improvement approach that is based on a process model. Process improvement is a series of actions taken to identify, analyze and improve process within an organization. It specifically refers to a model developed by the Software Engineering Institute in the mid-1980s. The CMM can be broken down into five levels as described below.

Levels of the CMM

There are five levels of the CMM. A brief summary is provided below.

Level 1 – Initial

Processes are usually ad-hoc, and the environment is very unstable and chaotic. A Level 1 organization tends to frequently abandon processes, especially in times of crisis when they deem that they need a quick fix.

A Level 1 organization also relies on knowledge in the organization's collective mind, and this knowledge is usually not documented. Working in a CMM Level 1 environment is a very haphazard approach, and if the people you need are on holiday or left the organization, it can be a very difficult time for the organization.

Level 2 – Repeatable

At maturity Level 2, success is starting to become repeatable, and the projects usually use basic project management to track cost, schedule and the work involved.

Project status and delivery are also visible to management in this level, and it is common for milestones and completion of tasks to be managed at this level.

This level still has a significant risk of exceeding cost and time estimates and still relies on specific individuals, sometimes known as “heroes” to handle any problems.

Level 3 – Defined

The organization has a set of standard processes, which are the basis for Level 3. These are established over time, constantly improved and standardized across the organization.

Management has a set process and ensures objectives are addressed. The big difference from Level 2 is that the scope of standards, processes and procedures is often tailored for an individual project from the company default process.

Level 4 – Managed

At maturity Level 4, process is controlled using statistical and other quantitative techniques and is quantitatively predictable. At maturity Level 3, processes are only qualitatively predictable.

Using precise measurements, management can effectively control the software development effort.

Level 5 – Optimizing

At Level 5, process improvements are addressed to find common causes of variations, and this is used to measurably improve the organization's processes. This level is always focused on process improvement by way of incremental and innovative technological improvements.

Level 5 is concerned with addressing common causes of process variation and adapting the process to improve process performance while still maintaining the statistics and predictability defined from Level 4.

CMMI Superseding CMM

The CMM is becoming superseded by CMMI (Capability Maturity Model Integration). The goal of CMMI is to improve usability of maturity models for software engineering and other disciplines by integrating many different models into one framework.

CMMI comes with two different approaches, staged and continuous:

- The **staged model**, which groups process areas into five maturity levels, is the representation used to achieve a “CMMI Level Rating” from a SCAMPI appraisal.
- The **continuous representation**, which was used in CMM, defines capability levels within each profile. The differences in the representations are solely organizational; the content is equivalent.

About Telelogic

Telelogic® is a leading global provider of solutions for automating and supporting best practices across the enterprise—from the powerful modeling of business processes and enterprise architectures to the requirements-driven development of advanced systems and software. Telelogic's solutions enable organizations to align products, systems, and software development lifecycles with business objectives and customer needs to dramatically improve quality and predictability, while significantly reducing time-to-market and overall costs.

To better enable our customers' drive towards an automated lifecycle process, Telelogic supports an open architecture and the use of standardized languages. As an industry leader and technology visionary, Telelogic is actively involved in shaping the future of enterprise architecture, application lifecycle management, and customer needs management by participating in industry organizations such as INCOSE, OMG, The Open Group, Eclipse, ETSI, ITU-T, the TeleManagement Forum and AUTOSAR.

Headquartered in Malmö, Sweden, with U.S. headquarters in Irvine, California, Telelogic has operations in 20 countries worldwide. Customers include Airbus, Alcatel, BAE SYSTEMS, BMW, Boeing, DaimlerChrysler, Deutsche Bank, Ericsson, General Electric, General Motors, Lockheed Martin, Motorola, NEC, Philips, Samsung, Siemens, Sprint, Thales and Vodafone.

Your contact : Knowllence - Tél +33 (0) 381 382 950 - www.knowllence.com

Global Headquarters
P.O. Box 4128, SE-203 12
Malmö, Sweden

Americas Headquarters
9401 Jeronimo Road
Irvine, CA 92618 USA

Offices across Europe, America, Asia and Australia.
Distributors worldwide.