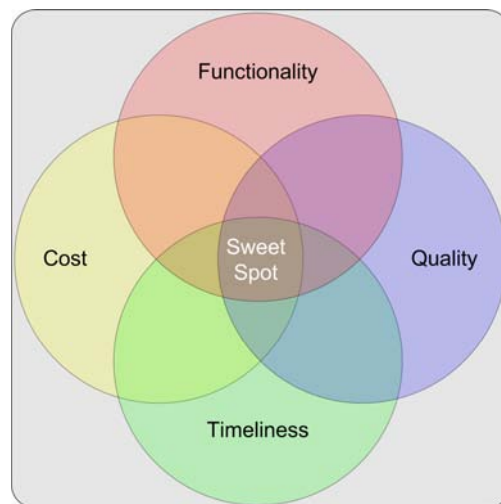


Introduction

Balancing Trade-Offs for Success

For organizations involved in software development, business stakeholders tend to have the final word on the success of a software development project. Business stakeholders generally evaluate project success based on four measures: Cost, Functionality, Quality and Timeliness. Each of these measures is dependent on the others. Cost limits functionality, quality impacts cost, timeliness (schedule) influences quality, and so on. Giving priority to any one measure unavoidably affects the other measures.

Given the interrelationships and dependencies of success measures, stakeholders must decide which measures will take precedence. Essentially, stakeholders are looking for the “sweet spot” wherein the optimal balance between measures is achieved. Finding the sweet spot does not mean that a perfect outcome is guaranteed; perfection is rarely obtainable. Finding the sweet spot means achieving the best possible outcome from balancing the trade-offs among the measures of success.



Balancing the Measures of Success

What do stakeholders need in order to fully analyze the difficult trade-off choices and make good decisions? The short answer is good information. Stakeholders will need to harvest information from many sources and one of the more critical sources is the requirements management process. If requirements management is to provide high-quality information, it must be an effective process. This presents a challenge for most organizations that must be overcome.

The Requirements Management Challenge

There is broad consensus within the software industry and academia that the success of software development projects depends heavily on effectively managing software requirements. The large volume of material that has been written on the topic is indicative of the level of importance the software industry places on this topic, and rightly so. When software projects fail, it is very often because of inadequate or ineffective requirements management.

With all that is known about requirements management, why have so few software teams mastered the discipline of requirements management? One plausible explanation is that even though organizations generally understand the concepts of requirements management, they simply don't understand how to effectively execute. That is to say, they have not mastered the skills for applying the discipline of requirements management within the unique context of their own project environment to achieve the desired results.

Effective requirements management is fundamentally difficult. It is therefore not surprising that organizations and software teams struggle with it. As with any complex undertaking, understanding what must be accomplished is an essential first step for getting the job done. To a large extent today, mature software development teams have achieved this first step. Conceptual knowledge of what to do, however, is of limited value if teams lack the knowledge and experience of how to successfully implement these concepts.

Consider the analogy of piloting an airplane. One might understand the concepts of flying a plane from point A to point B. However, if the person sitting behind the controls lacks the knowledge and experience of how to accomplish all of the required tasks of piloting the plane, would you really want to be a passenger on that flight?

To fully master the discipline of requirements management, the first step of understanding what to do can be a formidable challenge all by itself. A larger challenge still is that of understanding how to effectively apply the discipline, and to do so while contending with all of the unique circumstances that software teams experience in real world projects.

Every software development project is different. The business environment and problems to be solved are unique, and the composition of project teams is as variable as the individuals team members involved. All of these factors compound the difficulty and the challenge of implementing effective requirements management. Given these realities there clearly can be no "one size fits all" process for requirements management that produces consistently good results. This idea is rooted in common sense, but runs counter to many of the popular methodologies that have been promoted as "best practices" within the software industry heretofore.

Mastering requirements management requires that project teams use adaptable methods and processes that are tailored to specifically address the unique circumstances of each project.

An Adaptive Solution for Effective Requirements Management

Adaptive Requirements Management (ARM) is an approach to requirements management in which processes and methods are specifically and continuously tailored (adapted) to fit the unique and defining characteristics of the project and the project team.

ARM is designed to help project teams overcome the difficulty of applying the principles and concepts of requirements management within the specific context of each software development project. It achieves

this in two ways: ARM reduces the complexity of understanding the “*what*” of requirements management by defining the Key Practice Areas (KPsAs) underlying the approach. ARM reduces the complexity of the “*how*” of requirements management by providing techniques for adapting the requirements management processes and KPsAs to fit the unique circumstances of a specific project.

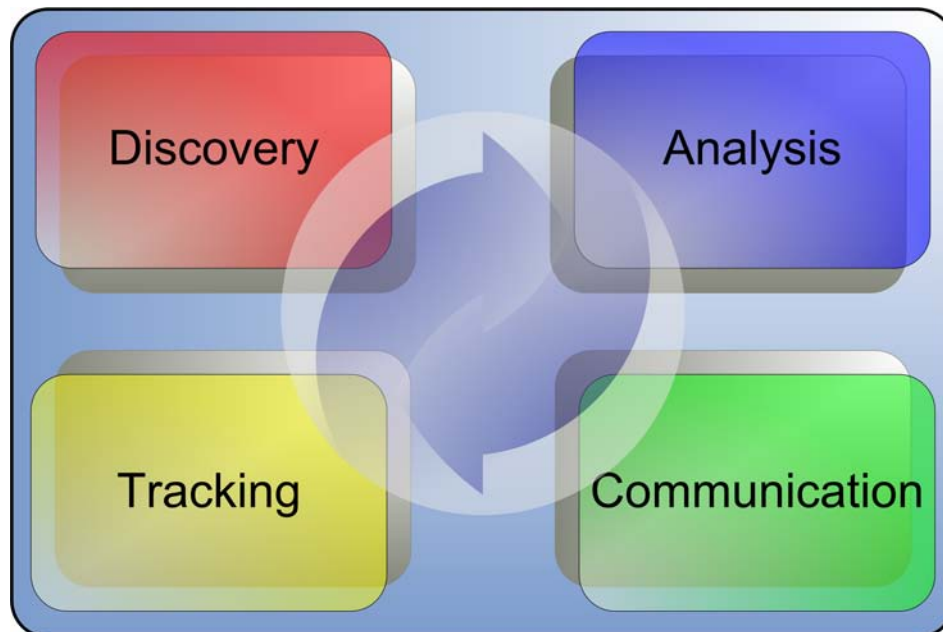
Key Principles

ARM is based on a set of fundamental principles that together form the foundation of the approach:

- 1. There is a core set of essential practices that define the discipline of requirements management, and each of these practices must be applied at the optimal level (just enough effort and no more) throughout the full lifecycle of the project.**
- 2. Every software project is unique, and therefore the requirements management approach must be adapted to fit the project and the team.**
- 3. The requirements management process must continually adapt to changes in the project environment based on well defined measures and regular evaluation of work-in-progress results.**

Key Practice Areas

The first ARM principle acknowledges the essential practices that are the core of requirements management. Within the ARM framework, these are described as Key Practice Areas. There are four distinct KPsAs which define the activities and methods for implementing the ARM approach.



ARM Key Practice Areas

Discovery

Discovery activities focus on “mining” requirements from a wide variety of potential sources. Common sources of requirements include:

- ▶ Business Stakeholders
- ▶ Domain and Subject Matter Experts
- ▶ End Users
- ▶ IT and Support Staff
- ▶ Existing System and Process Documentation
- ▶ Legacy Systems
- ▶ External Sources
- ▶ Standards and Regulatory Sources

Discovering requirements is accomplished using a variety of elicitation tools and techniques that have proven effective for working with all manner of requirements sources. Examples of tools and techniques include:

- ▶ Seminars and Workshops
- ▶ Interviews
- ▶ Research
- ▶ Systems Analysis
- ▶ Business Process Analysis
- ▶ Prototypes
- ▶ Information Feedback Loops

The information mined from requirements sources in the process of Discovery is like “raw ore” that is rich with gold. The artifacts of Discovery will typically include a variety of documents, notes and other materials that will be refined through analysis.

Analysis

The purpose of Analysis activities is to further process all of the information captured through Discovery activities. The goal is to extract the gold (useful requirements) from the raw ore. Analysis tasks include the following:

- ▶ Cleanse
- ▶ Clarify
- ▶ Organize
- ▶ Categorize
- ▶ Identify Dependencies
- ▶ Correlate Results

An important goal of Analysis tasks is to validate requirements information. This is done by focusing on the four “Cs”. Useful requirements must be Complete, Concise, Correct and Consistent. Once the four Cs have been satisfied, Analysis enables stakeholder prioritization of requirements based on quantifiable business benefit, cost, and risk.

Invariably, the process of Analysis leads to more Discovery. This is the iterative nature of the relationship of KPAs.

Communication

The purpose of Communication is to disseminate requirements information throughout the project environment. The ultimate goal is consistent understanding among all of the consumers of requirements. Consumers include potentially anyone within the project environment who must work with requirements in any capacity. Software designers and developers are obvious consumers. Stakeholders are primary consumers, as are Quality Assurance team members. Trainers, documentation writers, system operators, administrators and end users are all requirements consumers.

Given the diverse roles and responsibilities of these consumer groups, requirements information must be made available in a variety of formats and venues. While it is certainly true that requirements knowledge is communicated orally, and this is an essential element of communication, requirements must be adequately documented. The process of Communication not only facilitates deeper understanding, it provides a persistent source of information that does not fade as the memory of humans is prone to do.

One key to effective Communication of requirements is audience-specific focus. This requires that requirements are expressed in a variety of forms. Common forms of requirements documentation include:

- ▶ Conceptual View
- ▶ Vision
- ▶ Use Cases / User Stories
- ▶ Supplemental Detailed Requirements

Useful requirements artifacts are not limited to textual documents. Other forms of documentation include items such as:

- ▶ Requirements databases (Repositories) which store requirements data (and meta-data) and can provide many different consumer-specific views of requirements
- ▶ Graphics (e.g., story boards and pictures) and other visuals
- ▶ Models (e.g., architecture models, use case models, deployment models)
- ▶ Spreadsheets
- ▶ Functional prototypes

Communication is essential to the function and effectiveness of each of the other KPAs. All KPAs are intended to be carried out in an iterative, cyclical process. Communication provides the essential information feedback loops which drive the entire process forward.

Tracking

The purpose of the Tracking KPA is to facilitate the requirements management process. A fundamental part of this is managing the lifecycle of individual granular requirements. Requirements change over time both in terms of what they describe, and also in terms of over-all relevance to the project. Additionally, the status of requirements change throughout the process and this must be tracked and monitored. All of this activity is important to preserve as historical data.

Tracking enables project managers to plan implementation efforts and also to measure progress. Traceability and coverage are focal points for Tracking. Basic questions about which requirements have been implemented, which requirements have been verified, impact analysis of proposed changes, etc., are important for managing the project. Answering these kinds of questions is a function of Tracking.

To be effective, Tracking depends on having the mechanisms in place for managing requirements information. The job of Tracking requirements is nearly impossible if requirements are not documented in some way. The better equipped the project environment is for managing requirements information, the easier the job of Tracking.

The difficulty of Tracking is related to the volume of requirements information generated, and to the efficacy of the tracking tools and techniques used. The ideal solution for tracking requirements is a Requirements Repository. This tool can either be a commercial off the shelf product or a “home grown” solution.

Applying ARM

To be successful with ARM, organizations must not lose sight of the core principles behind the framework. The first principle states that each of the KPAs must be applied at the optimal level for throughout the duration of the project. The optimal levels will vary during the lifecycle of the project, but the objective is always to do just enough of each of the KPA activities, and no more.

The second principle states that there is no one size fits all process that can be equally effective for all projects and teams. This necessitates adaptation based on project need. The third principle extends the second by recognizing the need for continuous adaptation.

The first perplexing question is how to go about configuring the ARM process for a specific project. What factors or considerations drive the configuration decisions? At the highest level, there are two primary considerations for configuring the ARM process: Project Factors and Team Factors.

Project Factors

Project factors are all of the elements and details which give the project a unique profile. These factors define the project environment and must be fully acknowledged when planning how the project will be carried out. The important project factors to consider when configuring the ARM process include the following:

- ▶ Business Goals and Objectives
 - ▶ What are the priorities?
 - ▶ What are the measures of success?
 - ▶ What are the cost/benefit considerations?
- ▶ Constraints
 - ▶ What is the budget?
 - ▶ What is the required delivery date?
 - ▶ What resources are available?

- ▶ Scope and Complexity
 - ▶ What is the size and duration?
 - ▶ How large is the project team?
 - ▶ What are the cross-functional dependencies?
- ▶ Risks and Opportunities
 - ▶ What is the risk tolerance of the organization?
 - ▶ What are the consequences of failure?
 - ▶ What are the potential rewards that can justify the risks?
- ▶ Problem Domain
 - ▶ How well is the problem understood?
 - ▶ Has a similar problem been solved before?
 - ▶ Can the problem be broken down into simpler problems and solved incrementally?
- ▶ Solution Domain
 - ▶ What software development process will be used?
 - ▶ What is the technology platform?
 - ▶ What are the critical technical or logistical challenges that must be met?

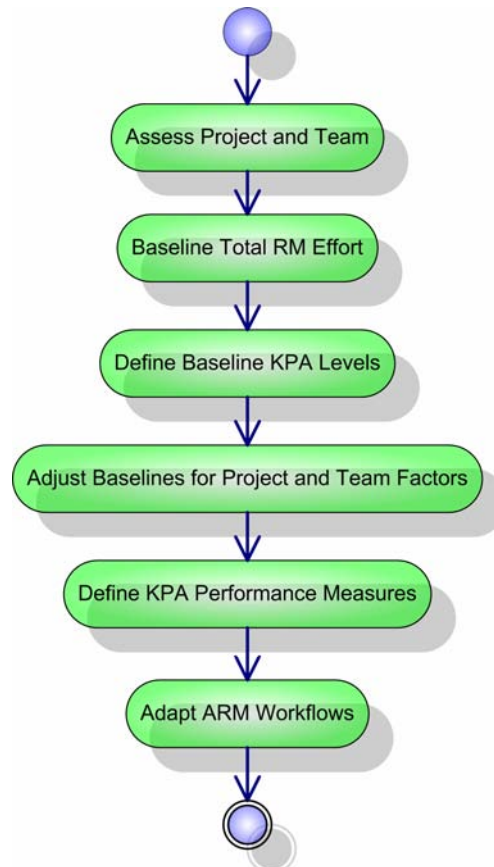
Team Factors

Team factors reflect all of the complex human elements of software development, and the unpredictability innate in human nature. Primary team factors include:

- ▶ Team size
- ▶ Maturity
- ▶ Individual member capabilities
- ▶ Track record (both team and Individual members)
- ▶ Culture
- ▶ Leadership
- ▶ Personalities
- ▶ Roles and responsibilities
- ▶ Structure (static elements)
- ▶ Process (dynamic elements)
- ▶ Communication

ARM Configuration Workflow

The high-level workflow for configuring the ARM includes six steps as illustrated below.



Project and Team Assessment

In order to configure the ARM process to meet the specific needs of the project, one must first accurately assess the Project and Team factors. This requires assessment instruments that can be used to objectively characterize the nature of the project and the team.¹ No matter what approach is used for assessment, one fact remains: assessment is not an exact science. Even with the aid of good assessment instruments, if the right questions are not asked, the results of assessment will be lacking.

Regardless of how assessments are carried out, it is important to remember that there is no absolute way to quantify the capabilities of the project team or its abilities to fulfill the needs of the project. However, with a plan for continuous adaptation of the process, all we are concerned with at this stage is defining a starting point from which to evolve.

Baseline Total Requirements Management Effort

Project and team resources are finite. It is therefore incumbent on the project management to define the total requirements management effort which will be needed for success of the project. Requirements

¹ CSG has developed project and team assessment tools for use with the ARM framework.

management effort can be defined as a percentage of the overall software development effort. There is, of course, the proverbial “chicken and egg” problem here. How does one gauge the overall software development effort when the requirements are not yet defined?

This is indeed a challenge, and often requires a “best guess” based on the information available at the time. The initial estimate of total requirements management effort will typically be strongly influenced by what resources are available, and by balancing the anticipated requirements resource needs against other critical project needs. To some extent, the priorities placed on the project success measures of Functionality, Cost, Quality and Timeliness will affect the level of requirements management total effort.

Define Baseline KPA Levels

After defining the total allocation of resources for requirements management, the next step is to define the initial allocations for each of the KPAs. Here again, the priorities of the project will provide some guidance for deciding the level of activities required for each of the KPAs.²

KPA baseline effort allocations are defined as a percentage of total requirements management effort. These levels will naturally ebb and flow throughout the life of the project, partially in response to the types of activities that are going in at a particular stage, but also on observation or recognition that, based on work-in-progress results, KPA levels must be rebalanced.

It is not useful to approach this exercise with the idea that the perfect solution will be obvious or achieved. In fact, it is better to simply consider that an initial baseline is needed, and changes and adjustments will be made along the way in the process of continuous optimization of the process.

Adjust Baselines for Project and Team Factors

Depending on how you arrived at levels for the initial effort baselines, you may have fully considered the project and team factors in the decisions. As with any decision that stems from working with incomplete or unknowable information, it is a good idea to do a gut check. Based on what you know about the project and the team, do the initial baseline levels seem reasonable? If not, change them to reflect what does seem reasonable.

Define KPA Performance Measures

Because the ARM approach is designed to continuously evolve, it is necessary to define the measures that will provide useful evidence of progress (or lack thereof). There is no theoretical limit to the quantity or types of measures one can define, but beware of over-kill. Also beware of metrics for the sake of metrics.

The primary goal is to define the minimal set of indicators that will provide the essential process feedback loops necessary for process improvement. It is generally useful to define KPA specific measures that provide insights into each of the KPAs. Some examples of useful measures include the following:

² CSG’s assessment tools address KPA leveling, based on twelve standard project profile possibilities.

Discovery Measures

- ▶ Effort expended
- ▶ Quantity of new requirements over time
- ▶ Distribution of requirement priorities
- ▶ Developer thrash (insufficient)

Analysis Measures

- ▶ Effort expended
- ▶ Quantity of change requests & defects over time (delta) and as % of initial requirements
- ▶ Distribution of requirement priorities
- ▶ Developer thrash (conflict)
- ▶ Quantity of unit tests developed
- ▶ Code coverage
- ▶ Lack of stakeholder acceptance

Communication Measures

- ▶ Effort expended
- ▶ Lack of stakeholder acceptance
- ▶ Approval cycle time
- ▶ Mean-time-to-update of documents

Tracking Measures

- ▶ Effort expended
- ▶ Developer thrash (too much change)

With regard to process monitoring, the effort required to produce the metrics must be justified by the value of the information. If it takes more effort than it's worth, either find a way to make the effort more efficient, or look for other useful measures that are less expensive.

Adapting the ARM Workflows

Defining baselines and measures is useful and necessary, but what matters most is actually doing the work of requirements management. There is no magic that can provide an alternative to rolling up one's proverbial sleeves and getting the job done. It is also unfortunate but true that there is no cook book formula for how to martial your requirements team to meet the challenges that lie ahead.

The success of the requirements management process is made more achievable by having a well defined approach that adapts to the project, but it does not eliminate the need to have individuals involved who have the necessary talents, skills and experience required to do the job.

It is a common fact of life in the world of software development that some team members tasked with requirements management are not fully qualified for the job. Fortunately, this does not mean that success is unachievable. It does mean, however, that skilled management and effective process are in place to compensate for the shortfalls that may exist within the team.

For the Requirements Manager, there is no perfect world. One never has full control over all of the factors and potential barriers to success. To the extent that you have inexperienced or under-qualified team members, the pragmatic response is to compensate in various ways, and work to improve the situation over time. For some projects, the constraints may be too overwhelming and the project is doomed from the start. Under these circumstances, the best course of action is not to begin the project at all.

To the extent that you have some flexibility over which team member are assigned to KPA activities, it makes sense to allocate team members based on the “highest and best use” principle. One of the interesting things about KPAs is how the nature of KPA tasks is most suitable to particular personality types of human beings. Discovery, for example is ideal work for the outgoing and engaging personality. Analysis is best suited for personalities who enjoy the more analytical, detail-oriented tasks. Communication requires, among other things, well developed communication skills and an enthusiasm for sharing information. Tracking is all about managing the process and managing information. These activities are ideal for individuals who enjoy accounting work.

The real challenge for the requirements manager is to work effectively with what you’ve got. Remember that perfection is the enemy of the good. The success of the project does not depend on perfect requirements management. It does however, require sufficient effort and results necessary to meet the challenges of the project overall.

Continuous Adaptation

Once the requirements manager understands all of the important aspects of ARM, orchestrating the process to achieve optimal results is the key to success.

ARM can be integrated with virtually any software development process, whether it is formal or informal, CMMI or Agile. It is necessary in all instances, however, to have regular review intervals, and process measures that provide essential feedback from which further process adaptations can be implemented. Without these essentials, the odds of achieving consistent high levels of success are very low.

Measure what is important to measure, and respond appropriately to the feedback received by adjusting KPA levels, and emphasis on KPA tasks.

Summary

ARM is an excellent approach for mastering the extremely difficult discipline of requirements management. It is based on common sense principles, and is designed to work in the imperfect world of commercial software development. Key focal points for succeeding with ARM can be summarized as:

- ▶ Focus on each of the KPAs
- ▶ Understand the true nature of your project by assessing your project and project team
- ▶ Continuously work to improve KPA proficiency within your team
- ▶ Monitor performance measures and thresholds for KPA results (process feedback loops)
- ▶ Evolve RM process in response to performance measure feedback by adjusting KPA activity levels

About CSG

CSG is Portland-based consulting firm specializing in custom software development for business and government. For over twelve years, CSG consultants have been highly successful delivering high-quality software solutions to our clients.

In striving to increase the value we provide to clients, CSG has worked to create innovative techniques for increasing the effectiveness of software teams. The Adaptive Requirements Management approach is a product this effort, and encapsulates the many years of experience we have spent mastering the art of requirements management.