Triple Your Chances of Project Success
Risk and Requirements

Louis S. Wheatcraft
Requirement Experts
(281) 486-9481
louw@reqexperts.com

Abstract. Requirements have a big impact on project success. Studies show the importance of having a good set of requirements that are clear, complete, correct, and consistent as well as the consequences of not having a good set of requirements. Having poor requirements places projects at risk of significant cost overruns, schedule delays, and performance shortfalls. In fact, recent studies have shown that projects who fail to take effective actions to ensure a good set of requirements triple their chances of project failure. The emphasis of this paper is to identify common requirement development and management risks that can have an impact on a project's success, possible consequences of these risks, and strategies to mitigate the risks and avoid the consequences of those risks. Recognizing the fact that poor requirements represent a significant risk to your project and mitigating these risks can triple your chances of project success.

Risk and Requirements

Recognizing the importance of a disciplined approach to product development and the need to define a good set of requirements so projects can be successful, various organizations responsible for creating product development and system engineering processes have defined processes based on best practices in industry and government. These processes are documented in various standards (see references) all of which stress the importance of upfront processes for defining product scope and requirements. The result of following these processes is a set of requirements that describe the characteristics, capabilities, functionality, performance, and quality the system needs to have to meet stakeholder expectations. These requirements provide the foundation upon which the product design is based.

With this emphasis on requirement development and management, it is logical to assume that project managers would want to avoid the risks associated with having a poor set of requirements. Therefore it is logical to assume these project managers would follow proven best practices to ensure a set of well-written requirements is established before contracting the design and development of their product. Unfortunately, this does not happen as often as it should. Many managers continue to fail to place the proper emphasis on having good requirements and communicating the importance of good requirements to their project team. Because of this, many projects are being set up for failure from the beginning, tripling their chances of project failure.

NASA's Office of Inspector General (OIG) November 2008 report [OIG 2008] focused on five major challenges; one of which concerns acquisition and contracting processes. The OIG concluded that: "NASA must be vigilant in its *process of establishing and validating project requirements.*" Program risks increase when contractual obligations are established before developing a sound business case and *clearly defining requirements*; placing "*the project at risk of significant cost overruns, schedule delays, and performance shortfalls*. Effective risk management, safety, and mission assurance controls are key to supporting robust and reliable operations in the context of very challenging launch and mission schedules." (Emphasis added.)

The United States General Accountability Office (GAO) has also reported on systemic issues involving government organization's acquisition processes emphasizing the important role requirement development and management has on the success of a project. In a June 2003 report [GAO 2003], the GAO stated that a key to a successful project lies in the "*ability to match users' needs, or requirements, with the developer's resources* (technology and design knowledge, money, and time) when product development begins." The GAO asserts their studies show that "*doing so can prevent rework and save both time and money*." (Emphasis added.)

The GAO stated: "The start of product development represents the point at which program managers make a commitment to provide a product that will perform as required and be delivered on time and within estimated costs. Our work has shown that programs are more likely to succeed if program managers are able to achieve a match between user needs, which eventually become requirements, and resources (technology, design and production knowledge, money, and time) at the start of product development. Conversely, *if they do not match requirements with resources, cost overruns and schedule delays are likely to occur*, reducing an organization's buying power in other areas." (Emphasis added.)

Effect of Requirements Definition Investment on Program Costs

DoD has also zeroed in on the impacts of poor requirements on their weapons acquisition process. In a recent article, *Executive Gov* article[Moore 2010] Jack Moore reported "In the efforts to rein in the Defense Department's acquisition process, the practice of developing requirements for combat systems is getting a looking at, according to a *Federal Times* report. "Requirements development . . . has been identified as a weakness in the department and has led to cost and schedule overruns on many programs," said deputy DoD acquisition chief Frank Kendall in a memo last month. "*Requirements development is paramount to successful acquisition outcomes*." The solution to the cost-overruns and other problems that often result from inadequately developing requirements is better training, Kendall said." (Emphasis added.)

In the *Federal Times* report[Bennett 2010], John Bennett reported: "Top Pentagon arms buyers have heard the calls to improve specification development for weapon programs and support services, and are emphasizing better training for the acquisition work force. Countless lawmakers and analysts have sharply criticized the process the Defense Department uses to decide what features its combat systems should possess. Senior Pentagon officials have said changes to the requirements-generation process could be enacted. "*Requirements development … has been identified as a weakness in the department and has led to cost and schedule overruns on many programs*," Kendall wrote in a Nov. 19 Pentagon memo. "*Requirements development is paramount to successful acquisition outcomes*." (Emphasis added.)

According to the Standish Group [Standish 1995], in 1995, "U.S. government and businesses spent approximately $81 billion on cancelled software projects, and another $59 billion for budget overruns." Their survey claimed that in the United States, only about one-sixth of all projects were completed on time and within budget, nearly one third of all projects were cancelled outright, and well over half were considered "challenged." Of the challenged or cancelled projects, the average project was 189 percent over budget, 222 percent behind schedule, and contained only 61 percent of the originally specified features. The Standish Group [Standish 2003] stated: "*Losing sight of*

requirements is often the first step on the road to projects that come in over budget, are late, do not meet specifications or are canceled." (Emphasis added.)

As reported by Ivy Hooks [Hooks 2001], studies conducted by NASA revealed average cost and schedule overruns of approximately 65 percent on 29 programs. The graphic shown in Figure 1 was produced by NASA's Comptroller's Office in the early 1990s depicting NASA programs from the 1970s and 1980s. The x-axis denotes the percentage of the program cost expended "up-front" in the requirements definition and design stage. The y-axis denotes the percentage overrun of the program costs. All programs identified on this graph overran its budget. We call this the "pay now or pay later" chart. Failing to invest in upfront scope and requirements development results in significant cost overruns and schedule slips due to the resulting rework.

If a project manager reviews his or her own projections, they may find that only about 5 percent was allotted for the up-front work, placing the project's success at risk. This almost guarantees a large overrun in cost and schedule. Much of the cost overruns shown in Figure 1 can be attributed to requirements changes. Much of that change was self-inflicted because of poor up-front work. It does not take a mathematician to see that an up-front investment can pay off substantially in the development of a system.

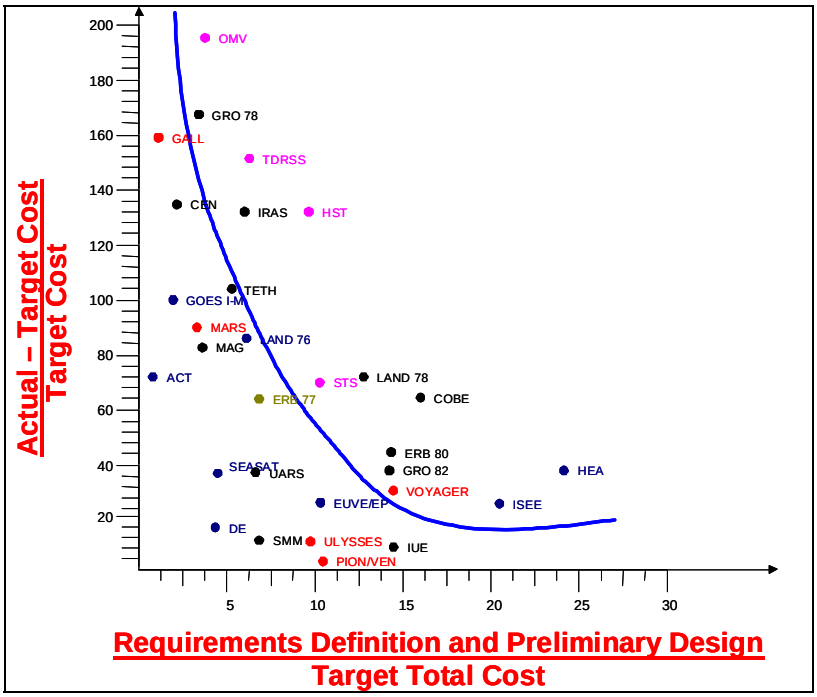

Figure 1: NASA Comptroller Cost Growth Chart

INCOSE's System Engineering Handbook [INCOSE 2007] features similar charts that tell the same story concerning both cost and schedule overruns due to not adequately investing in the needed amount of system engineering effort at the beginning of a project. The SE Handbook authors conclude: "systems engineering effort can be a positive factor in controlling cost overruns and reducing the uncertainty of project execution." Good system and software engineering processes emphasize the importance of understanding stakeholder expectations at the beginning and being

able to clearly communicate those expectations in a language (requirements) that can be unambiguously understood by developers.

Similar impacts have been reported for software-intensive projects in industry as well. The importance of using best requirement practices on project success was recently documented in a report by Keith Ellis [Ellis 2008]. In the report, Ellis studied over 100 companies with development projects in excess of $250,000.

As a result of this study, Ellis found that for 68 percent of the companies evaluated, project success is "improbable." Ellis said, "projects might succeed – but not by design. Based on the competencies present, these companies are statistically unlikely to have a successful project." While these companies indicated they recognized that requirements are important to project success, they still failed to take effective actions to ensure a good set of requirements and by doing so, they *tripled their chances of project failure.*

Ellis concludes: "Organizations understand conceptually that requirements are important, but do not internalize this understanding and change their behavior as a result. The most successful of companies do not view requirements as a document which either existed or didn't at the beginning of a project, they view it as a process of requirements discovery. *Only companies that focus on both the process and the deliverables are consistently successful at changing project success rates*." (Emphasis added.)

Based on the previous discussion, it should be clear that requirements are key to the success of a project and that when a good set of requirements is not developed, the project is at high risk and doomed to failure from the beginning. The following two quotes clearly make these points.

> Ivy Hooks, President of Compliance Automation, Inc. states: "*People who write bad requirements should not be surprised when they get bad products. But they always are*."
>
> In July 2009, NASA Administrator Charles Bolden stated: "*Putting forth the same effort, or using the same approach, then expecting different results is ... insanity*."

A Winning Product vs. Risk

The goal of all projects should be to deliver a winning product. A winning product is defined herein as: "*A product that delivers what is needed, within budget, within schedule, and with the desired quality*."

A simple and practical definition of risk is: "*Anything that can prevent you from delivering a winning product!*" Given the importance of requirements to the success of a project, poor requirements represent a major project risk.

Risks are something that could have an impact on your product or subsystem (hazard or threat). Risk has two major components: likelihood and impact/consequences. In the following discussion, the scope risk factors are things you have control over. Failing to follow these best practices means the likelihood of these risks to your project is 100%. The possible impact or consequences listed are what you are trying to avoid.

As the above discussion has shown, failing to implement an effective requirement development and management process represents a significant risk to your project's ability to deliver a winning product. Therefore, all project managers need to mitigate this risk from the beginning.

Recognizing the fact that poor requirements represent a significant risk to your project and mitigating these risks *can triple your chances of project success*.

In February 2010, NASA released *NASA Space Flight Program and Project Management Handbook* [NASA 2010]. This handbook captures program and project management best practices from experienced managers, providing the continuity of that expert knowledge base. Spread throughout the handbook are statements concerning the importance of defining and incorporating requirement development processes into your project. The handbook states:

- "All acquisitions should start with a requirement definition that clearly identifies the Agency's desired outcome for a contract."
- "Establishing a good set of program mission/operation concepts that are evolved into a useful set of program requirements is one of the most critical products for program success."
- "The most common negative finding made by independent review teams is that a project did not place sufficient effort and importance on understanding and developing project requirements."
- "One of the greatest risks that a project faces comes from ill-defined requirements."
- "Poorly written requirements, incomplete requirements, and poorly written contracts result in cost overruns and schedule slips."
- "Managers need to be able to identify risks and add the mitigation costs to the program baseline. When risks are identified and the qualitative value assigned to the risk has been verified, the PM needs to take action in the timeframe associated with that risk."

In January 2011, DOD released "*National Security Space Strategy*" report[DOD 2011]. In the report, DOD emphasized the importance of requirements and their acquisition process, stating: "In cooperation with our industrial base partners, DoD and the Intelligence Community (IC) will revalidate current measures and implement new measures, where practicable, to stabilize program acquisition more effectively and improve our space acquisition processes. *We will reduce programmatic risk through improved management of requirements. We will use proven best practices of systems engineering, mission assurance, contracting, technology maturation, cost estimating, and financial management to improve system acquisition, reduce the risk of mission failure, and increase successful launch and operation of our space systems."* (Emphasis added.)

To reduce the risk on your project, you need to integrate requirement best practices into your requirement development and management processes. The following sections walk you through these best practices, key risk factors associated with not following those best practices, and the impacts those risks can have on your project.

Scope Risk

The scope of a project constitutes the vision: the need to develop or procure a product or service; the goals and objectives of the stakeholders; information about the customers and users of the product or service; and how the product will be developed or purchased, tested, deployed and used. The scope also includes the boundaries and constraints of the product. A project with no boundaries will diverge.

Project scope definition benefits the quality of your product requirements by preventing incorrect requirements and preventing many omissions. An early scope definition keeps requirement

writers from diverging, reduces requirement inconsistencies, and keeps the big picture in view. It also shortens the time required for requirement writing and rewriting and reduces debates.

If you spend time up-front to define a clear scope, you will have more knowledge before you begin to capture requirements. You will document and confirm assumptions and resolve action items before your start writing requirements.

A major contributor to project failure is the failure to spend the time at the beginning of the project to clearly define the project scope before writing your requirements and beginning product development.

Scope Risk Factors

- *Failure to define Scope*

Failing to define your project and product scope can have major impacts and consequences on your ability to deliver a winning product. The product purpose/use will not be well understood resulting in not being able to meet stakeholder expectations. Failing to define scope also can result in your team being faced with vague and undefined desired outcomes, lacking the knowledge they need to write requirements, setting your project team up for failure.

If your team lacks clear direction due to a lack of a common vision, the volatility and divergence of individual visions will result in conflicts between stakeholders and a constant stream of issues. These will lead to constant change and scope creep, which, in turn, will have significant impacts on your budget and schedule.

- *Failure to define Need, goals, and objectives*

The Need for a project defines the "why" – why are we doing this? What are we trying to accomplish? The Need is based on an analysis of a problem that the project is supposed to solve for some stakeholder or group of stakeholders. Goals are those things that you plan to accomplish that will result in meeting the Need. Objectives are measures of performance, including key performance parameters that show that you have met the goals. Objectives show that you have "gotten there", i.e., your project accomplished what was expected of it by the stakeholders.

Failing to define the Need, goals, and objectives for your project results in your team being faced with vague and undefined desired outcomes with no clear direction. Failing to define your objectives, means you have not defined, and gotten agreement on, the criteria for success.

- *Failure to involve relevant stakeholders*

Stakeholders include key representatives from various organizations and groups that have a "stake" in your project. This can include those who buy it, sell it, use it, train others to use it, design it, develop it, test it, market it, maintain it and expect to profit from it. Each may have a very different point of view and list of priorities. Stakeholders are a major source of requirements. All stakeholders should be accounted for and considered prior to writing requirements.

Failing to involve key stakeholders can result in battles due to differing visions and different interpretations of what your project includes or excludes. Issues are not identified and resolved before investing scarce resources into the requirement writing effort. Because of this, more time is taken to write requirements and will slow down the requirement review and baseline process.

Failing to involve relevant stakeholders can result in missing requirements and subsequent change later in the product lifecycle to add in the missing requirements. Too many assumptions will have to be made, resulting in incorrect information and stakeholder expectations not met.

- *Failure to identify drivers and constraints*

Drivers and constraints are those things that are imposed on your project from outside your project that you have little control over, but have to be complaint with. These include requirements allocated to you from a higher level, standards, regulations, cost, schedule, technology, and existing systems. Drivers and constraints represent requirements.

Failing to identify drivers and constraints can result in missing requirements and subsequent costly change later in the product lifecycle. Failing to identify standards and regulations can result in your product not being in compliance. How can your product meet security or safety requirements if those requirements were not identified and included in your product's requirement set? If this happens, expensive rework and schedule slips result. Failing to identify the existing systems your product must interact (interface with) could result in your product failing to work with those existing systems.

- *Failure to define a feasible concept to meet the stakeholder needs*

Before writing requirements, you need to develop a set of operational concepts that address the views of all the stakeholders, all the products lifecycles, and address both nominal and off-nominal conditions. Operational concepts bridge the gap between product scope and requirements. The practice of defining and documenting operational concepts is a simple, cost-effective way to build consensus among all stakeholders and discover which questions still need to be asked and answered prior to writing product requirements.

Failing to define a feasible concept (cost, schedule, technology) to meet your stakeholder needs can result in the stakeholder expectations not being met. If you fail to address all lifecycle stages and both nominal and off-nominal cases, you will have missing requirements and your system will lack the expected robustness to handle off-nominal cases. The missing requirements and subsequent change later in the product lifecycle to add in the missing requirements can result in scope creep, expensive rework, and schedule slips.

- *Failure to define product boundaries and external interfaces*

An interface is a boundary where, or across which, two systems interact. Serious problems can arise at interfaces. Your project is particularly vulnerable to interfaces with other products over which you have no control. Because of this, you product is at most risk at the interfaces. Identifying interfaces helps you to define your system's boundaries and helps you understand the dependencies your system has with other systems and dependencies other systems have with your system. Indentifying interfaces helps you ensure compatibility between your system and other systems you need to interact. Identifying interfaces also helps to expose potential project risks.

Failing to identify an interface can have unpleasant repercussions on your project and is a common reason for products that fail to meet stakeholder expectations. Failing to identify or incorrectly define an interface is often a major cause of cost overruns and product failures. Failing to address your interfaces can result in missing requirements and subsequent change later in the product lifecycle. Failing to agree on an interface definition can result in you doing work you don't need to

do or can result in you not doing work you should have done. The end result of failing to address your interfaces is that your system may not work as expected when interacting with other systems.

- *Failure to baseline scope before writing requirements*

One of the most frequent reasons for project failure is "scope creep". Baselining your scope before writing your requirements, puts your scope under configuration control and the resulting configuration control process. Any changes proposed to your product has to first go to the configuration control board, evaluated for feasibility and impacts to cost and schedule, and formally approved.

Failing to baseline scope before writing requirements can result in scope creep and the resulting uncontrolled change. Failing to define scope also results in there not being a single clear vision for your product, rather each stakeholder has their own vision. This can result in conflicts due to an inconsistent, incorrect, and incomplete set of requirements. This then will result in significant cost and schedule impacts.

- *Mitigating Scope Risk*

The defense against all these risk factors is to clearly define your project's scope at the beginning and then validate that scope with all the key stakeholders, getting their agreement on the scope before proceeding with writing your requirements. The largest scope risk, by far, is wishful thinking (budget/cost, schedule, and technology). To avoid the scope risks discussed above, include the following in your requirement development and management process:

- Develop a clear vision – identify the Need, goals, and objectives for your project and product.
- Identify and involve relevant stakeholders
- Identify and manage drivers and constraints
- Develop operational concepts
- Identify and manage external interfaces
- Identify and manage scope risk
- Baseline Scope (Scope Review or Mission Concept Review)

Requirement Risk

Requirements describe the characteristics, capabilities, functionality, performance, and quality your system needs to have in order to meet stakeholder expectations. These requirements provide the foundation upon which the product design is based. Requirements are the technical language that communicates to the developers the stakeholder expectations as defined in the scope. A well-written set of requirements will result in a product that will meet the stakeholder expectations.

Defective requirements represent risk to your project. As shown in Figure 2[Hooks 2001], the cost to fix requirement defects increases exponentially as you move through the product lifecycle phases. This table illustrates the order of magnitude costs of finding and fixing defects as we progress in the life cycle. So, the cost of finding and fixing defects in the coding phase is 10 times more expensive than finding and fixing it during the requirements phase. Finding defects during Development Test, 15-40 times, Acceptance test, 30-70 times and so-forth.

Indentifying and removing these defects as early as possible will result in significant cost savings. Conversely, allowing these defects to go undetected until the later phases of your product development will result in significant cost and schedule risk.

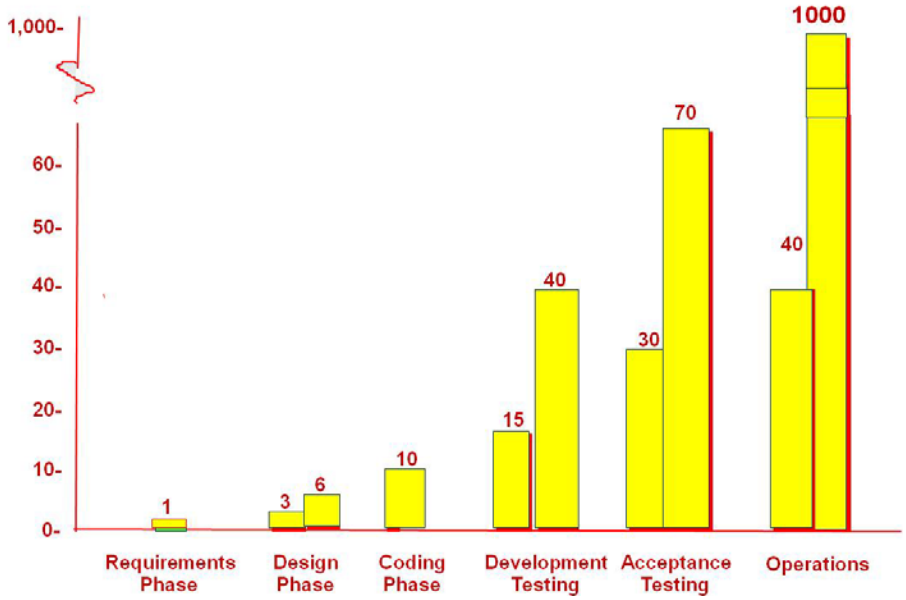

Figure 2: Cost to fix requirement defects

Requirement Risk Factors

- *Requirement not necessary*

A mandatory characteristic of a requirement is that it is needed. If a requirement is not needed, why is it in the requirement set? Requirements that are not necessary result in increased management cost of the requirements that, in turn, increases overall project costs and leaves less resources for needed requirements. All requirements need to be maintained, managed, and verified. Un-needed requirements can result in work being performed that is not necessary, taking resources away from the implementation of those requirements that are needed. In addition, implementing requirements that are not necessary can result in degraded system performance as well as introducing a potential source of failure and conflict.

- *Requirement not verifiable*

Another mandatory characteristic of a requirement is that it must be verifiable. Why ask for something when you can't prove it has been implemented as intended? A requirement that is not verifiable could result in the developers implementing the wrong thing or the people doing verification verifying the system does the wrong thing rather than what was intended. If the true intent of the requirement is not clear, stakeholder expectations may not be met.

- *Requirement not attainable*

Another mandatory characteristic of a requirement is that it must be attainable. Why state a requirement you know cannot be implemented given the existing budget, schedule, or level of technical maturity? Why state a requirement when you haven't done the assessment on whether or not it can be implemented within your existing budget, schedule, or level of technical maturity?

Stating a requirement that is not attainable will result in wasted effort, cost and schedule impacts, as well as stakeholder expectations (performance) not being met.

- *Requirement can be understood more than one way*

A requirement that can be understood more than one way is ambiguous. If a requirement is ambiguous, the result is a requirement that is not verifiable and may result in the developers implementing the wrong thing and thus stakeholders expectations are not met because the true intent of the requirement was misunderstood.

- *Requirement(s) incomplete*

A requirement is incomplete when there is information missing needed to understand the requirement or to verify the requirement. The information may be in the requirement text or in an attribute of the requirement (rationale). A classic case is when there is an interface requirement that does not point to where the interface is defined. Another case is when a value is not defined (has a To be determined (TBD) or To be supplied (TBS). An incomplete requirement can result in the requirement not being able to be implemented as intended and stakeholder expectations not being met.

As a set, incomplete requirements is also a risk. If there are missing requirements the product may not be able to do the job that was intended and again stakeholder expectations not being met. If the missing requirements deal with drivers and constraints, your system may not be in compliance with standards or regulations (law). If the missing requirements have something to do with an interface that was not identified or defined, then your product may fail to work when interacting with other systems.

- *Requirement reflects implementation*

In general, the focus on predesign requirements is on the "what" and not the "how". How is implementation. Often when an implementation is stated, the reason is that the requirement belongs at a lower level and that the real requirement is missing at the level you are at. By stating implementation, the real requirement goes unstated - the "why" is not communicated to the developers. If the real requirement is not stated, it cannot be flowed down (allocated) properly to the next level of the architecture. In addition, by stating implementation, the developer's solution space is restricted, not allowing the developer to propose the "best" solution to meet your expectations.

- *Requirement(s) subject to change*

Next to the cost of verification, rework is often the most expensive part of a project. Changing requirements often result in rework that will impact your cost and schedule. If a requirement is subject to change (or has already changed multiple times), it is a risk to your project. Because of this, requirements that are subject to change must be identified. After these are identified, you have to determine if the change matters and if it does, when is it important.

Some requirement changes are unavoidable. A standard or regulation may be updated. An interface definition may change. Technologies may not deliver what was expected. Stakeholders may change, resulting in different expectations. If you don't respond to these changes, your product may not be in compliance with the new standard or regulation or could fail to work when interacting with the changed system you have an interface with. If you haven't done the proper

assessment of a requirement change at one level of the architecture, there may be possible conflicts with requirements at higher or lower levels. If you don't change requirements to agree with the new stakeholder expectations, the wrong requirement may be implemented.

- *Requirements not allocated (flowed down)*

Allocation is the process of apportioning resources or assigning responsibility for implementation of requirements at one level to the parts at the next level of the system architecture. All requirements need to be allocated until the final level of the architecture has been defined.

Failing to allocate your requirements can result in requirements not being implemented at the next level of the architecture. Failing to allocate requirements can also result in missing lower level requirements (derived children requirements that are necessary and sufficient to meet the parent requirement.) Often when a requirement is allocated to more than one part of the architecture at the next level, an internal interface is needed. Thus, if that requirement was not allocated, you could fail to identify an internal interface. Also, allocation, when combined with traceability, allows you to do a complete change assessment of requirements in the levels above and below the requirement that is changing. Failing to address allocation can result in incomplete change assessment.

- *Requirements not traceable to a parent*

Traceability is the concept that all requirements need to be linked (traced) to their source, referred to as their parent. Being able to trace to a parent, is one way to determine if a requirement is needed. In cases where there is no trace to a parent, it could mean that the requirement is not needed and gold plating is taking place. Gold plating is adding features to as system that are not needed. Good plating is a source of requirements creep. Like requirements that are not needed, gold plating can impact the cost and schedule of your project.

As stated above, traceability, when combined with allocation, allows you to do a complete change assessment of requirements in the levels above and below the requirement that is changing. Failing to address traceability can result in incomplete change assessment.

Traceability also allows you to determine whether or not a parent requirement is being properly implemented by allowing you to access whether or not the children requirements linked (traced) to the parent requirement are necessary and sufficient to meet the intent of the parent. Without both allocation and traceability, this assessment cannot be done.

Mitigating Requirement Risk

The defense against all these requirement risk factors is to clearly define your requirements before beginning development. Do the best job you can to ensure you have a well written set of requirements. To avoid the requirement risks discussed above, include the following in your requirement development and management process:

- Define and enforce a requirement development process
- Follow the "Writing Good Requirements Checklist" (Contact Requirements Experts for a copy.)
- Include key attributes: rationale, traceability, verification method, allocation, priority, risk
- Train your requirement writers, management, developers, testers, reviewers on how to write defect free requirements

- Practice continuous requirement validation. Don't allow defective requirements you're your requirement set. Project managers should not wait until the major milestone reviews, especially the System Requirement Review (SRR), to find out they have a bad set of requirements. There is always the danger that sub-par requirements will be baselined, especially if there is a multitude of problems with the requirements at the SRR and the schedule is tight. Baselining bad requirements always leads to wasted resources needed to correct the requirements - putting the project at risk of schedule and budget overruns.
- Identify and manage requirement risk as stated above.
- Baseline Requirements (SRR) and have a firm change management process.

Requirement Management Risk

Requirement Management is the overall process that involves the development and validation of your requirements, ensuring your requirements are allocated and traced, and managing change. Requirement management does not happen by itself. Your team needs a well defined process for accomplishing these activities as well as the necessary time and resources. The following risk factors address the impacts of not following best practices involved in requirements management.

Requirement Management Risk Factors

- *No official process/process not followed*

Not having an official process for managing requirements or having a process but the process in not followed will result in wasted resources as well as all the scope and requirement risk factors discussed earlier.

- *Not enough time and resources allocated to define and baseline scope*

Failing to allocate sufficient time and resources to define and baseline your project and product scope will result in the scope risk factors discussed earlier. In addition, you will not have defined and baselined a feasible concept that, when implemented, will meet stakeholder expectations. Not having a baselined scope will also result in constant change and scope creep, impacting your cost and schedule.

- *Not enough time and resources allocated to develop and baseline requirements*

Failing to allocate sufficient time and resources to define and baseline your project and product requirements will result in the requirement risk factors discussed earlier. In addition, the direction to developers will be poorly communicated, ambiguous, incomplete, and inconsistent. Not having a baselined set of good requirements will result in constant change and requirements creep, again severely impacting your cost and schedule

- *Poor change management*

The result of not having or not enforcing a well defined change management process can result in uncontrolled change, scope creep, and requirements creep. Without some clear criteria for which changes are allowed, defective and unnecessary requirements will get into your set of requirements. You will be allowing change to control your project rather than you controlling change. Uncontrolled change results in both unnecessary rework and unneeded work, both of which will result in significant cost and schedule impacts as shown in Figure 2.

Managing Change

When you baseline your scope or requirements, you are defining a scope or requirement bucket as shown in Figure 3. Your scope (stakeholder expectations) or requirements (technical depiction of stakeholder expectations) are bound by cost, schedule, and technology. There is always some risk associated with your ability to achieve what you have put into the bucket. By baselining your scope and requirements you will have defined your scope and requirements buckets. Managing change is managing these buckets.

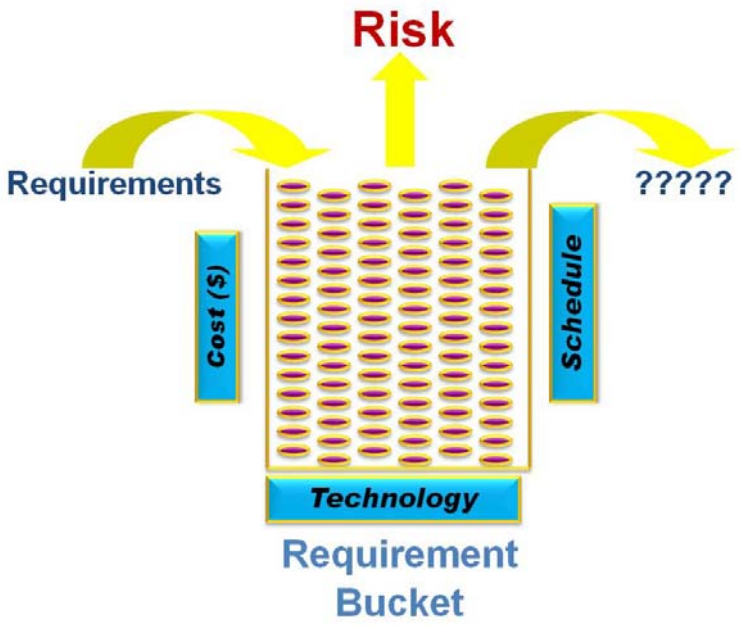

Figure 3: The Requirement Bucket

If the current set of requirements exactly fits within the scope's technology, cost, and schedule boundaries then you will have a tool to manage change. What if someone wants to add additional requirements into the bucket? What is your response? One response is to not accept additional requirements without increasing the boundaries (i.e. changing scope – asking for addition funds and time to account for the added work.) Another response could be to take out lower priority requirements from the bucket so the new requirements can be implemented within your current budget and schedule. There is also the response that no manager wants to hear – "No". Saying "No" is not always a good response when considering one's future. A more politically correct response may be "Yes, but ……" Whatever your response, adding to the bucket without making a change to the bucket will add risk and endangers the ability of your project being able to deliver a winning product that meets stakeholder expectations.

A bit of advice. People keep coming to you for more and more. If you don't say no, they will keep asking. So if you are being a nice person and saying yes to every change being asked for, you are going to get your project into a great deal of trouble. There is no reason for doing that. The person asking you to do this, wants it, but doesn't want to pay anymore and they don't want to wait any longer. The truth of the matter is you can't keep adding things without someone paying more or it taking longer than you originally planed. By saying yes all the time, you are going to get your project in trouble and you are going to disappoint the customer. You don't have to tell them "no"

all the time, but you have to be honest with them. You need to say "Yes, we could add that, but if we do add that we are not going to be able to finish this project on the date we promised – it is going to extend a week, a month, a year". It is not possible to just keep cramming it all in the bucket without risk. By saying yes, and meaning no, you are creating many problems for yourself. So don't just say no all the time, but explain yes, if you are willing to slip the schedule, if you are willing to pay more, then we can do this – otherwise we shouldn't try to do it now. People understand this.

Mitigating Requirement Management Risk

As stated for both mitigating scope risk and requirement risk, the defense against all requirement management risk factors is to follow the best practices stated. Do the best job you can, the first time. To avoid the requirement management risks discussed above, include the following in your requirement development and management planning:

- Allocate sufficient time and resources to define and baseline Scope before writing requirements (Define your scope first, baseline and control it. This is absolutely the most critical first step in the requirements process.)
- Allocate sufficient time and resources to develop and baseline requirements.
- Use requirement attributes to manage requirements.
- Develop and enforce a formal requirement development and management process.
- Train your team in your requirement development and management process.
- Manage change
  - Do not baseline a bad document. Don't say "its okay, we'll go ahead and get started and fix everything with change requests."
  - Put as much rigor in the baseline as in the changes that will follow. When someone comes in with a change after your baseline, what information do you need? The requirement, the rationale, and the impact. If you have this same rigor on the front-end as you do after baseline, you will have a much better document.
  - "Design for change". Adopt a design for change philosophy, e.g., design it so you can make a change without having to recompile, size it for more volume or concurrent users.
  - Establish criteria for change. Clearly state which changes are appropriate and will be considered and which type of changes are not appropriate and will be not be considered. Remember the old saying "Better is the worst enemy to good enough." In this case good enough is meeting stakeholder expectations, gold plating is adding features to make the product "better" when there is no requirement to do so, and by doing so, your cost and budget could be impacted negatively.

Wrap up and Parting Thoughts

Having poor requirements will place your project at risk of significant cost overruns, schedule delays, and performance shortfalls. Projects who fail to take effective actions to ensure a good set of requirements *triple their chances of project failure.* The emphasis of this paper was to identify common requirement development and management risks that can have an impact on your project's success, possible consequences of these risks, and strategies to mitigate the risks and avoid the consequences of those risks. The case was made concerning the importance of requirements from a risk mitigation standpoint and the importance of project management

ensuring a requirement development and management process is in place for their project and their project team is trained in that process.

Failing to implement an effective requirement development and management process represents a significant risk to your project's ability to deliver a winning product. Therefore, all project managers need to mitigate this risk from the beginning. Recognizing the fact that poor requirements represent a significant risk to your project and mitigating these risks can *triple your chances of project success*. What kind of a project manager would knowingly not maximize their changes of project success?

In conclusion, avoid the risks associated with poor requirement development and management practices by adopting the risk mitigation for each of the risk areas discussed in this paper and adopt the following requirement development and management best practices for your project:

- Address scope and requirement risk at the beginning of your project. Defining scope results in understanding stakeholder expectations. If you don't, you are setting your project up for failure and probably will not be able to develop a product that meets stakeholder expectations. It is worthwhile to do the best job up-front to ensure the scope of your project is clearly understood, is feasible, is agreed to, and baselined. This results in a firm foundation to develop and manage your requirements.
- Identify drivers and constraints and external interfaces as part of the scope definition process. Develop operational concepts that are thoroughly thought out in the beginning of a project to allow the writing of better and more comprehensive requirements. This will eliminate rework and multiple recertification cycles later in the product lifecycle, preventing cost overruns and schedule slips.
- Develop, implement, and enforce a formal requirement development and management process that includes continuous requirement validation. This is critical during the initial development push as well as during final requirement development and baselinging, development of the corresponding verification requirements, and managing your requirements.
- Pay particular attention to your change management process. If you don't control change, change will control you. Changes to requirements result in design changes and rework, which impact schedule and budget. Frequently, these design changes are larger and more expensive than planned.
- Train your team and enforce the requirement development and management process through project leadership. Do not only send team members to training, -- have them use the language taught in training, set up processes to match what is presented during training, and invest in either continual "refresher training" or provide a mentor to ensure a learned behavior is followed by the team; the process must be learned and it does not come naturally to most.
- Allocate the time and resources needed to do the job right – the first time. Small investments early on will provide large dividends later in saved or avoided costs and schedule slips.
- Clearly communicate to your team how important good requirements are to you and to the success of the project. Never accept defective requirements. Reward team members for doing a good job in developing and managing your requirements.

References

1. Bennett, John T., *DoD urges more training for requirements writers*, December 27, 2010, Federal Times.
2. CAI *Requirements Development and Management,* Seminar Workbook, February 2010, Compliance Automation, Inc. 2010.
3. DOD, *National Security Space Strategy Unclassified Summary*, February 6, 2011
4. Ellis, Keith. *Business Analysis Benchmark – The impact of Business Requirements on the success of technology projects*, IAG Consulting, 2008.
5. Moore, Jack, *DoD Takes on Developing Weapons Requirements, Ending Creep*, Dec 28, 2010, ExecutiveGov.
6. GAO-03-598, *Matching Resources with Requirements Is Key to the Unmanned Combat Air Vehicle Program's Success*, United States General Accounting Office, June, 2003. <http://www.gao.gov/new.items/d03598.pdf>.
7. Hooks, I. F. and Farry, K. A., *Customer-Centered Products: creating successful products through smart requirements management*; AMACOM Books, NY, NY, 2001.
8. INCOSE, Systems Engineering Handbook - a guide for system life cycle processes and activities, Version 3.1, INCOSE-TP-2003-002-03.1, August 2007, ed, Cecilia Haskins.
9. ISO/IEC 15288, System Engineering-System Life Cycle Processes, October 2002.
10. NASA OIG, Inspector General, *NASA's Most Serious Management and Performance Challenges*, Report, November 2008. <http://oig.nasa.gov/NASA2008ManagementChallenges.pdf>.
11. NASA, *System Engineering Handbook*, SP-2007-6105, Rev. 1, December 2007. <http://education.ksc.nasa.gov/esmdspacegrant/Documents/NASA%20SP-2007-6105%20Rev%201%20Final%2031Dec2007.pdf>.
12. NASA, *Systems Engineering Processes and Requirements*, NPR 7123.1A, March 2007 <http://nodis3.gsfc.nasa.gov/displayDir.cfm?Internal_ID=N_PR_7120_005D>.
13. NASA, *Space Flight Program and Project Management Requirements*, NPR 7120.5D, March 2007. <http://nodis3.gsfc.nasa.gov/displayDir.cfm?Internal_ID=N_PR_7120_005D>.
14. NASA, *Space Flight Program and Project Management Handbook*, NPR 7120.5, February 2010. < http://www.nasa.gov/pdf/423715main_NPR_7120-5_HB_FINAL-02-25-10.pdf >.
15. Software Engineering Institute, *CMMI for Development – Improving processes for better products*, Version 1.2, CMMI Product Team, Carnegie Mellon, August 2006.
16. The Standish Group Report, CHAOS Chronicles, 1995 and 2003
17. Wheatcraft, L. S. and Hooks, I. F., *Scope Magic*, 2001. <http://www.complianceautomation.com>.
18. Wheatcraft, L. S. *The Importance Of Scope Definition Prior to Developing Space System Requirements*. INCOSE *INSIGHT*, Vol. 4 Issue 4, January 2002. <http://www.complianceautomation.com>.
19. Wheatcraft, L. S. *Delivering Quality Products That Meet Customer Expectations*. Published in CrossTalk, The Journal of Defense Software Engineering, January 2003, Vol. 16 No. 1. <http://www.complianceautomation.com>.
20. Wheatcraft, L. S. *Developing Requirements for Technology-Driven Products*. Presented at INCOSE 2005, July 2005. <http://www.complianceautomation.com>.

BIOGRAPHY

Lou Wheatcraft has over 40 years experience in the aerospace industry, including 22 years in the United States Air Force. Over the last 10 years, Lou has worked for Compliance Automation (dba Requirements Experts), where he has conducted over 140 seminars on requirement development and management for NASA, Department of Defense (DoD), and industry. Lou has had articles published in the International Council of Systems Engineering (INCOSE) *INSIGHT* magazine and in DoD's magazine, *CrossTalk.*

Lou has made presentations at NASA's PM Challenge, INCOSE's International Symposium, and at the local Project Management Institute (PMI) and INCOSE Chapter Meetings. Lou has a BS degree in Electrical Engineering, an MA degree in Computer Information Systems, an MS degree in Environmental Management, and has completed the course work for an MS degree in Studies of the Future.

Lou is a member of INCOSE, co-chair of the INCOSE Requirements Working Group, a member of PMI, the Software Engineering Institute, the World Futures Society, and the National Honor Society of Pi Alpha Alpha. Lou is the recipient of NASA's Silver Snoopy Award and Public Service Medal and was nominated for the Rotary Stellar Award for his significant contributions to the Nation's Space Program.