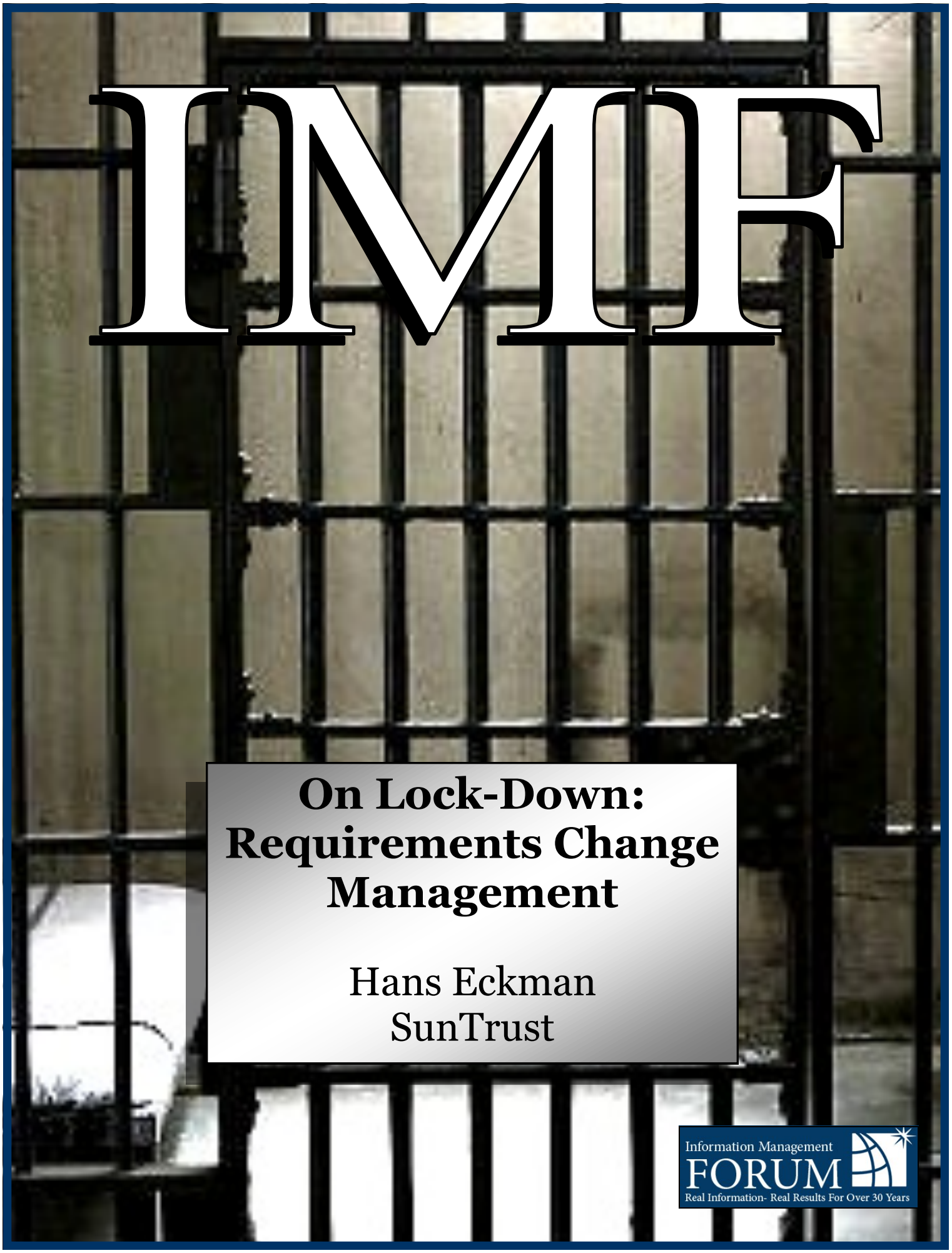


On Lock-Down: Requirements Change Management

Hans Eckman
SunTrust

Information Management FORUM
Real Information- Real Results For Over 30 Years

About Us

The Information Management Forum

MISSION STATEMENT

The IMF is the best source for unbiased, shared information in IT. We provide the highest quality personal service for a group of select members and work to be the best value in their IT budget.

The Information Management Forum was founded by five forward-thinking CIO's who envisioned peer-to-peer learning in a strictly vendor-free environment. Though vendor-sponsored networks and research companies have expanded and saturated the IT landscape over time, IMF's goal since its beginning in 1975 has been to be the best source of experiential data in IT for a select group of members while remaining strictly vendor-free. The diverse base of members utilizes IMF as an extension of their staff to achieve their goals. For over thirty years one constant remains the same: IMF's only motive is its members' success.

MEMBERSHIP BENEFITS

USE OF IMF FORUMS FOR LEADERSHIP DEVELOPMENT for their entire IT Organization, as well as for the HR functions that support IT. Forums are kept small and interactive, and include dozens of meetings and Web Forums that enable direct peer-to-peer interaction on topics of broad interest.

IMF CONNECT forms ad hoc focus groups to answer specific questions. Our members tap the experience of their peers to get answers to challenges that they are facing within their own organizations. The IMF facilitates and project manages these discussions and produces final summaries for our busy members.

IMF REPORTS provide a rich archive of real-world experience participants can draw on as they address their issues and challenges. More than 900 reports covering the full spectrum of IT issues and challenges are available online.

IMF BENCHMARK FORUM allows unlimited, on-demand access to benchmark data to be used for comparative business analysis.

TABLE OF CONTENTS

IMF Report:
On Lock-Down: Requirements Change Management

About The Presenter

WEB FORUM

PRESENTATION BY

HANS ECKMAN, VICE PRESIDENT & ENTERPRISE BUSINESS ANALYST

ABOUT THE PRESENTER:

HANS ECKMAN, Vice President & Enterprise Business Analyst

Hans Eckman provides transitional management and consulting for growing companies, with 14 of his 20 years' experience spent implementing web-based solutions across diverse industries. He is currently working as an Enterprise Business Analyst at SunTrust Banks for the Enterprise Architecture team, and provides process improvement and mentoring support through the BA Center of Excellence and Project Management Office.

Hans' Website: www.HansEckman.com

On Lock-Down: Requirements Change Management

Starting at the End

In straying from the norm a bit, let's start with the conclusions of this report on requirements management. Then we will delve into how we reached those conclusions.

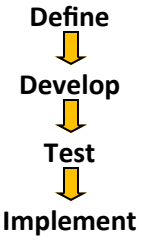
Conclusions

- Principals of release management and change control can be applied to requirements management
- Level of control must match risk and timing
- Consistency and diligence are required for success

The first is that the principals of release management and change control can be applied to requirements change management. Basically the same things you do to control change in any other aspect of your business relate to requirements too. This provides additional value and will save you having to control change in other areas by managing your requirements properly at the beginning. Next is the level of control. It needs to match not only the risk of those changes but also the timing. To be successful, consistency and due diligence are critical in the process. Looking at this, you could say the three cornerstones of requirements change management are: diligence, proper triage, and addressing issues at the proper threat level.

Ideal Project Timeline

Below is the ideal project timeline, even though a majority of projects rarely follow the timeline perfectly.



This is pretty straightforward. You figure out what you are going to do in the definition phase, develop a solution, test it, and implement it. All of your stakeholders should agree and know from the very beginning exactly what it is they want. The developers translate these requirements into a viable solution so there are no errors or omissions. Every requirement is satisfied so the solution is ready. Aside from that, during the entire project there was not a single change in business need or prioritization. What you have just read is the perfect scenario, but as you know this never seems to be the case.

High Costs of Change

One study by Barry Boehm stated that letting a defect get into production will likely cost you 100x more than had you caught it in your initial requirements and design phases. However, in a new study he conducted along with a partner, they found that on really small trivial projects the cost increase is really only 5:1. If it costs you $200 in man hours and effort to change a requirement then that means for a small project you are looking at $1000 in production, or possibly even $10,000 on a larger project. Also, current software projects spend about 40-50% of their work effort on avoidable rework. That means almost half of everything you change in a project could have been avoided. There are two major sources they found in their research of avoidable rework. One was hastily specified requirements and the other was nominal -case design and development.

Figure 1 shows the distribution and cost increase of these changes over time. There are two studies, one by Tassey and the other by Boehm, that tried to figure out how much a defect costs for every unit of cost as you move further along the lifecycle. A defect could be an error, omission, basically anything that needs to be changed. In requirements and design, that is where you point out the base unit. In this case there is one cost unit. For example, if your cost to do something is a blended rate of $65/hour then that is your multiplier as well. Once you get into coding or unit testing Tassey found the cost was 5x as much. Boehm's study showed the cost was 3x as much. During integration testing, Tassey found the cost was 10x as much, Boehm 7x. When you get into acceptance testing, Tassey found it would cost you 15x as much to fix. Tassey's study was focused more on an iterative development or prototyping, something more along the lines of agile development where you are trying to catch mistakes earlier. Boehm, in his study, found the cost was 50x more. Then when you get into production, Tassey found the cost was 30x as much and Boehm's 100x. What is the distribution of when these errors are occurring? In Tassey's study they found that about 3.5% of the defects were found in requirements and design, 10.5% during coding and testing, and 35% during integration testing. So in an iterative development you find the bulk of your mistakes in integration testing. Acceptance testing had 6% and then 15% of the errors made it all the way to production.

Only Change is Constant

Now we can look at that same timeline as before but with the reality that the only thing you can really count on is change. Focusing on those same phases, the question becomes when are these problems or defects going to arise. The first thing that comes to mind is scope and prioritization

Error Cost Factor	Requirements, Design	Coding, Unit Test	Integration Test	Acceptance Test	Post-product Release
Tassey[1]	1x	5x	10x	15x	30x
Boehm[2,3]	1x	3x	7x	50x	100x
When Errors are introduced[1]	3.5%	10.5%	35%	6%	15%

Figure 1: Distribution & costs

changes. These are going to start popping up part of the way through your definition phase and all the way through testing. Once you get further into testing you are really beyond the point of changing scope. Missed requirements are going to start showing in development and going to continue all the way through the end of implementation. These solutions may even go live with known missed requirements that will have to be handled elsewhere or in production. You are going to deal with ambiguities, clarifications, or invalid requirements throughout much of the lifecycle. There are also additional constraints start decisions and scope, thus causing changes in requirements. Those could be design, resource, time, or budget constraints. This is when you end up with the possibility of having to change or modify functionality to fit a new reality. There is a chance it may just be a missed implementation too. The requirements were clear, people knew, and it was simply missed or forgotten. Unfortunately those things do not show up until testing and may actually go live without that requirement implemented. The last disaster is a change in business need. This is not just the business changing what they asked for but it is also changes in the business needs/environment during the project lifecycle. You work with those changes, trying to incorporate them before the testing stage. Once you get into testing, those become a large impact you need to manage.

Managing Levels

Threat levels

In order to manage these changes, we need to be able to break them down into threat levels. In figure 2 you can see three levels of changes. The first is a non-material impact. This is a smaller change. Not one that says "I thought we were doing something and now we are doing something else." These are errors like typos, ambiguities, clarifications,

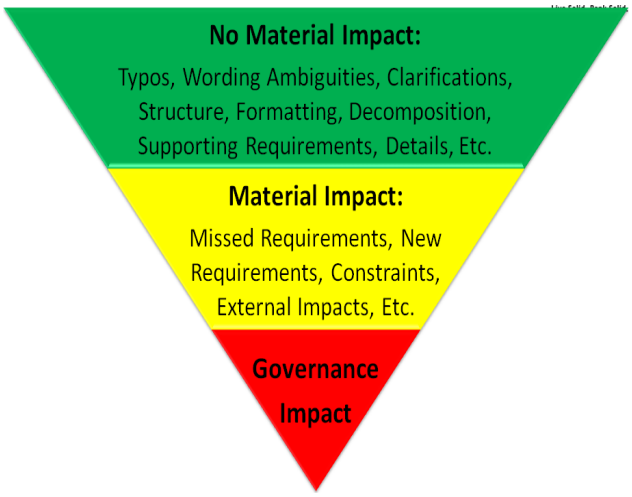


Figure 2: Threat levels

format changes, or the need for additional supporting requirements. That adds clarity so that everyone has the same picture. The next level, threat level yellow, is when you have a material impact. This is when you have missed a requirement, new requirements, impacts due to constraints that need to be overcome, or external impacts. In this scenario you really are truly changing some aspect of the solution that is going to change how it is implemented, who implements it, the timeline, and costs. The last level, threat level red, is when there is a governance impact. Now for organizations that have no governance controls this is not as big of a deal. However, when you are dealing with *SOX* controls or IT governance controls where you need official sign offs and incremental approvals then at some point these changes really step up to that next level. Now your stakeholders are going to have to sign off on the change. There is going to have to be some sort of formal change management.

Triage Levels

Figure 3 on the following page shows how you can triage these different levels of threat. The non-material impacts are the easiest change. You need

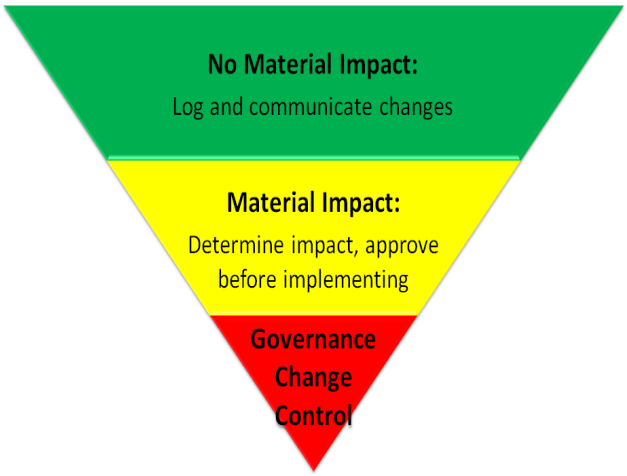


Figure 3: Triage levels

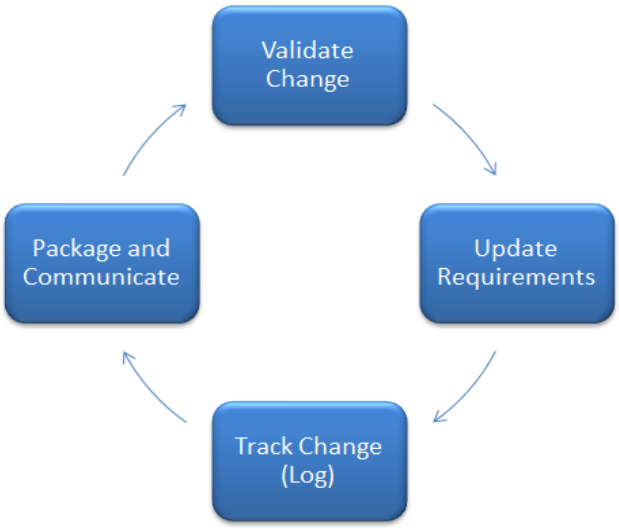


Figure 4: Non-material process

to log, track, and communicate these changes. Tell people what is changing and hopefully why it is changing. In threat level yellow really you will need to determine the impact of that material change. For example, what is it going to cost, what are the risks? These will also need to be approved by the stakeholders before they are implemented. When the sponsoring group has to get involved, the situation goes beyond your team. At the red level you are looking at a governance change control where you take that approval and run it through your standard governance change process.

Process flow

Non-material impact

Figure 4 displays a cycle for non-material impacts. Remember, those are just word changes or additional details used for clarity. The first step in this process is validating the change. Instead of making any change somebody suggests, ensure the change is real and gain a good understanding. Next you update the requirements. Then find a way to track those changes in some sort of a log system. Finally you will need to package those changes and communicate them to others. This begins after the first formal review of communications for the requirements. Any time you have a certain part where the team is conducting an official review, changes after that need to follow this update process. Anything before that is just part of your iterative or brainstorming requirements sessions. That is all part of the natural flow, as you review smaller bits of information. However, the minute you put a final draft out there you do not want your team rereading all of your documents and reviewing. You want them to focus on what has changed. It needs to update the system of record for the requirements. So wherever you are keeping your requirements, that system needs to be updated with these changes. You need to log those changes and what date those changes were made. This update process is a foundation for your change approvals. Even if you are going through an approval process with a material change it is going to follow this non-material impact update process.

Documenting changes

Documenting changes could really be something as simple as change control log in a document. It

could be the revision history if you are using a system for your requirements. There are plenty of options out there, including spreadsheets and text files. Ensure your changes are logged by document version (figure 5).

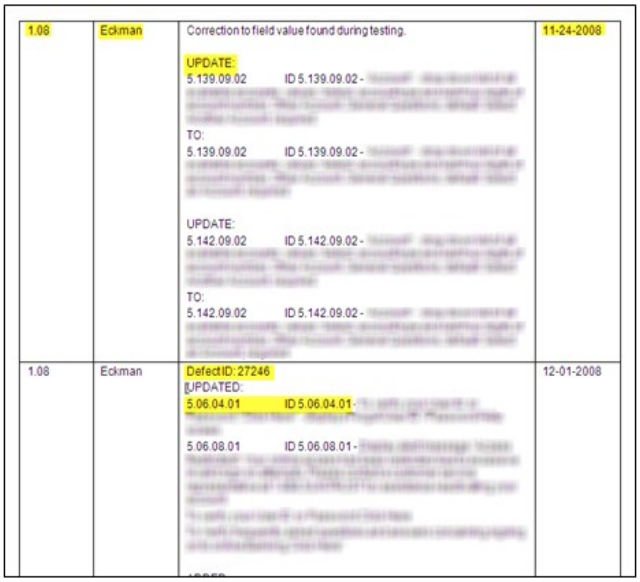


Figure 5: Documenting change

In the requirements document above, there is a column that states what version of the document it applies. Also included is the owner, what the update was, and the source. If possible, show the before and after shot. This is where, if you are documenting these changes manually you could end up doing a lot of cutting and pasting. On the other hand, if you are using a system then it might be able to give you that a little easier. It is critical to identify the owner, effective date, and whatever requirement ID. Whatever ID you are using to track these changes needs to be part of it. This means if you are using a program like *Word* and the outline format that inserting a requirement may change the numbering scheme. When possible it is a good idea to use some sort of additional or foreign key. Effective date and version date become important because as you make changes to your base document there could be a situation where your numbers have been updated and do not match.

Changes requiring approval

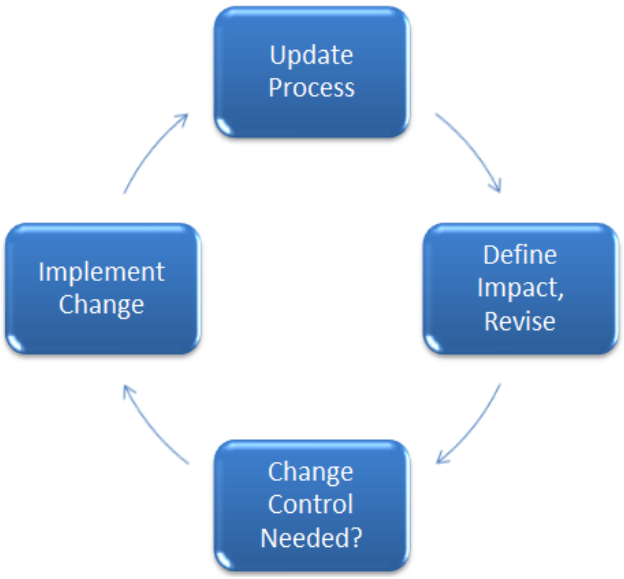


Figure 6: Approval process

Here you are doing the same tracking, changing, and validating as before in the non-material process update process. Next you must big thing define the impact (figure 6) which may result in a requirements revision. If the decision is made to change something that could have a massive impact to your scope and schedule, you may want to revise and focus on a smaller portion of the business need that is viable within the project constraints. After that you need to establish if a governance change control is necessary. Does this change violate governance rules by timing or scope to the point you need official approval? Once approval is granted, the change may be implemented. Then I would implement that change. If you sign off on a requirements document and then make material changes to it, at some point you are going to bundle those changes into a new version and approve it. When changing the functionality, or solution's capabilities, you would need

approval if it impacts other deliverables. By impact we are talking about impacts that cannot be absorbed. If you change process A for process 12 for example, it may cancel out and the impact may be zero because you are just trading hours in one bucket for hours in another bucket. That is okay if it impacts the cost or schedule. Now some groups, if the costs go down, may not need an approval. It may only be when the costs rise. That is up to your organization. Leverage the update process and make sure to define the impact and costs. If it is something that falls under your governance control, then seek stakeholder approval.

Tracking Approvals

You may track approvals manually if the tool you are using does not meet your personal requirements for traceability. It can be done a spreadsheet, as seen in figure 7. As you can see below, there is a change control ID, the date it was identified, a column for change details, and a reason (dropdown list optional).

You can track the status of those changes all the way through the process, from draft to approval. Other columns include:

⇒ Hours of impact

⇒ Primary contact of that change

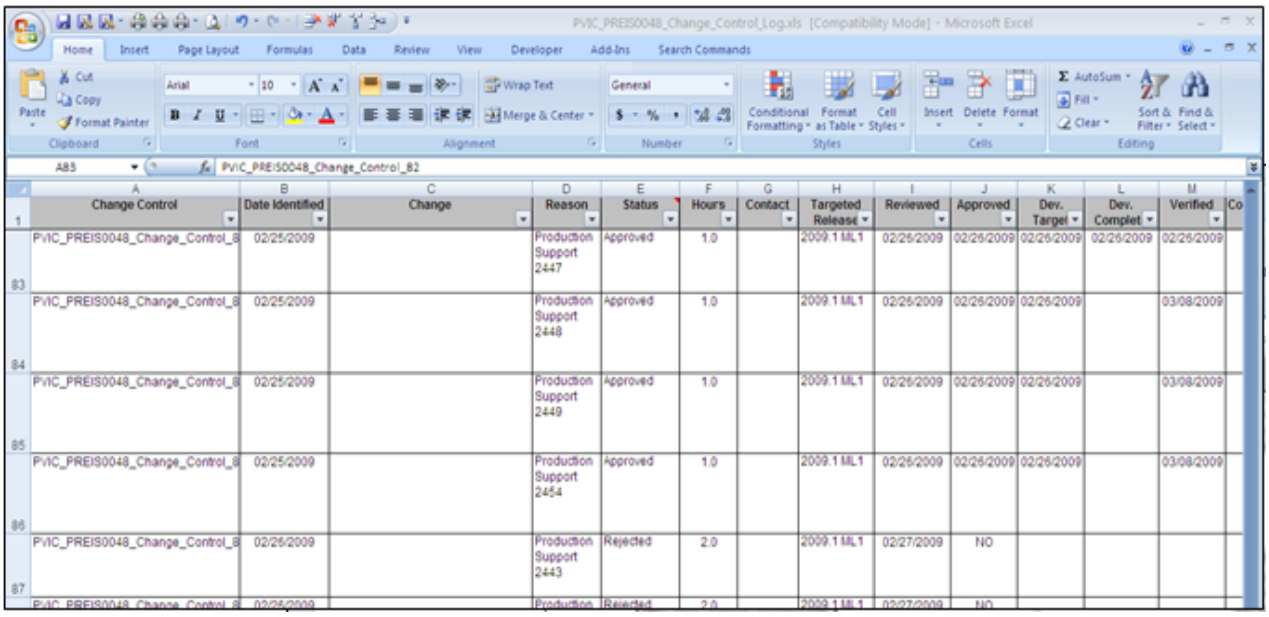
⇒ Targeted release for the change

⇒ Additional dates for historical purposes

As for the last arrow, it is good to know what date a change was reviewed, approved, the target development date, development implementation date, and the verified date that change had been made. You can also include a second linked spreadsheet that provides additional detail.

Governance change controls

Governance change controls are really for organizations that have this process. Generally you have a site or some kind of form to fill out with the required data for this process. It is basically the same as your approval workflow with a different



	Change Control	Date Identified	Change	Reason	Status	Hours	Contact	Targeted Release	Reviewed	Approved	Dev. Target	Dev. Complet	Verified
83	PVIC_PREIS0048_Change_Control_8	02/25/2009		Production Support 2447	Approved	1.0		2009.1 ML1	02/26/2009	02/26/2009	02/26/2009	02/26/2009	02/26/2009
84	PVIC_PREIS0048_Change_Control_8	02/25/2009		Production Support 2448	Approved	1.0		2009.1 ML1	02/26/2009	02/26/2009	02/26/2009		03/08/2009
85	PVIC_PREIS0048_Change_Control_8	02/25/2009		Production Support 2449	Approved	1.0		2009.1 ML1	02/26/2009	02/26/2009	02/26/2009		03/08/2009
86	PVIC_PREIS0048_Change_Control_8	02/25/2009		Production Support 2454	Approved	1.0		2009.1 ML1	02/26/2009	02/26/2009	02/26/2009		03/08/2009
87	PVIC_PREIS0048_Change_Control_8	02/26/2009		Production Support 2443	Rejected	2.0		2009.1 ML1	02/27/2009	NO			

Figure 7: Spreadsheet tracking

routing for how you complete and approve that change. This is useful when you have changes after artifact sign-off. If you have significant project impacts, like scope, functionality, or costs, you may be regrouping this scope into different releases or implementations. After all of the changes go in and the clean-up is done, it is a good idea to put a governance change control through approving the final version of that requirements document as the official approved requirements document for that project. Essentially you lump everything together at the very end and approve that document as the official version. That is then what will be passed on for future projects.

Best Practices

A good best practice to start off with is communicating the change control process and templates as part of your requirements approach. You want to eliminate surprises. Tell people upfront and let them know and fully embrace the fact that you are going to have changes. Things will not work all of the time. Instead of worrying about that and dealing with it later, deal with it upfront. Maintain consistent control and communication. Nothing will drive your project teams crazier than constantly telling them every time you correct a typo or make a change. That kills their productivity because they are reading changes over and over again, which is why tools like the change log are so important. They can browse through and decide for themselves if viewing that change is necessary. It is recommended if you are doing no or low risk changes that you make an update no more than weekly on most projects. The impact assessment is the key to risk management. You have to understand what the risks and impacts are or your projects will suffer. Every little change will eventually add up, even if it is only a 15 minute impact. Times that by 100 changes and now it becomes significant. It can be very difficult to group those changes without first understanding their effect.

Any time you can, leverage the available tools. Use a requirements management system. There are some incredible products out there, like *Borland's CaliberRM*, that do an amazing job of tracking changes, linking changes to defects, and linking changes and requirements to releases, packages, and code. If you need something to help track these changes, any type of log you use for defects or change controls works perfectly for requirements. Your bug tracker can be easily repurposed into a requirements change tracker or manager. Employees often use *Microsoft Word* and its *track changes* feature is amazing. It gives you in-line changes that people can read through and see the changes in context, knowing instantly what changed. You may want to continue making changes to a document using *track changes* for a period of time and then cut a release. Afterwards, update a minor version, accept all the changes, and start from scratch. Start with a clean version maybe once a month so it is easier for people to skim through and find the changes.

Cross-referencing every change is a best practice that involves a lot more effort but it will save you so much work in the long term. Find out what the source was, the reason for it, the date, and then any supporting documentation. What was the defect and change request? Where is the impact analysis and what is the change control? This helps because you have the signatures, approvals, and you know exactly what was done. It takes a lot of the emotional stress out of these changes.

Dealing with baseline documentation

Some groups are fortunate enough to have baseline documentation, which can dramatically reduce cost and risk. Anytime you start up a new

system, try to document all of the system functionality, provided it is not a vendor application. It does not make sense to repeat what they have already done. Your baseline, or your existing requirements, plus any changes you make for the release, become your new baseline (Figure 8).

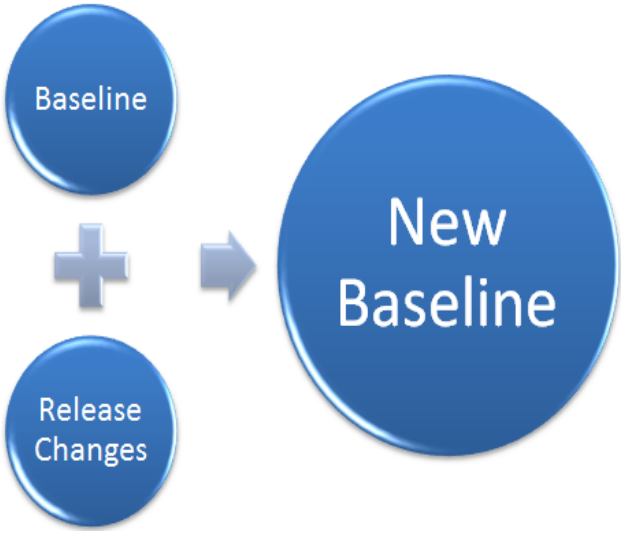


Figure 8: Baseline approach

The scope, changes, and requirements can all run through the same requirements change management process. This fits extremely well into your more iterative methodologies, like *RUP*, Agile, or a combination of both, where you are trying to pull business needs from a queue and implement them. Now you can pull them in as changes, have a method of updating and approving those changes, and then get them into a release. By starting with a baseline, since you have hopefully already documented 80-90% of current functionality, you dramatically reduce your cost and risk. This way you end up treating all of your scope as simply changes to the baseline requirements. The sum of all of those changes is your release scope. The sum of the changes in your baseline becomes the production release and your new baseline. This is going to take additional time and a lot more effort to maintain an accurate baseline but it can be an extremely valuable asset.

Recap

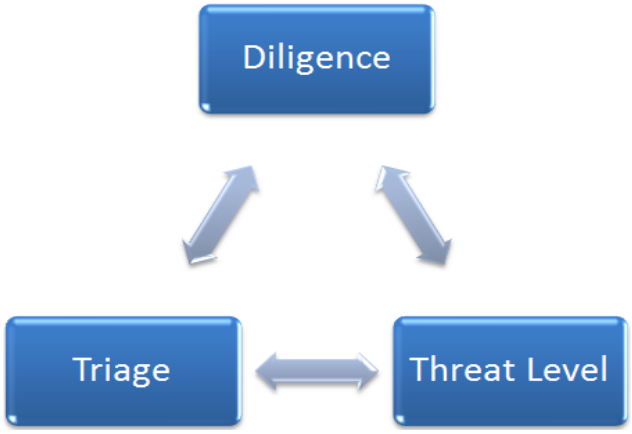


Figure 9: How to manage requirements change

Let's take a look back at the conclusions we went over in the beginning of the report. Again, all the principals of release management and change control can be applied to requirements management. This also means that best practices learned in other areas can be applied to the requirements management process as well, with similar value. Your level of control must match the risk and the timing. Approvals and governance approvals are not necessary when working on draft documentation or the first requirements solicitation. Those come into play later, once you have approved a document and people are doing work based on a certain expectation. That is when risk is escalated so your management and control level needs to escalate. Consistency and diligence are required for success. The three cornerstones are: apply diligence to manage; properly triage the changes; handle each according to the threat level (Figure 9) ■■■

Q. How do you incorporate "why" into the "what" normally documented into requirements?

A. *One of the challenges when doing system or functional and non-functional requirements is those are stating exactly what you are going to do. Often, if a development team or an implementation team is only looking at that level of requirements they're losing the context, they're losing the reason. You're also really losing the opportunity to leverage their intelligence and insight into possibly a more valuable solution because they do not understand why they're doing it or they may code it a certain way that ends up severely limiting you in the future. So when you talk about a "why," you're actually not talking about system requirements you're talking about business requirements. A properly executed requirements package would trace your business features, which are your high level goals or areas of the system, to your business requirements which are at the end of this "why would something be done." The "what is it that from a business value and why am I doing something" standpoint is completed at the end. Then I would want to trace those to my system requirements. So a business requirement, using SunTrust as an example, is that customers need the ability to deposit their checks in a mobile manner. In mobile checks deposit, one of the business requirements is the use of a camera to capture the image of a check. That is really saying "what is it we are trying to do." What is it the user is supposed to do? I can then trace that to the system requirements telling the development team what the flow is, what the use-case is, what the scenario is, what the constraints are, etc. I really think you get the biggest bang by working on your traceability, rather than trying to write the "why" into your system requirements. Now there's an exception to that train of thought. If you are using a prototyping or a visual method of developing your requirements, communicating them, or doing something that is very user-story or use-case based then you can include in those user-stories additions that describe "why." There's nothing wrong with using plain English and saying "we're hoping the user will be able to do this at the end" or "here's why we are trying to do it." So adding that additional information into your supporting information is a great idea. You just want to be careful and make sure that there's a distinction and the team knows the difference between what they're expected to implement versus their interpretation of the "why," which could be very vague.*

Q. Could you talk a bit more in detail about the state of requirement tools such as *CaliberRM*, *Rational RequisitePro*, and any other good ones that you prefer to use today? Also could you share any published comparison of these tools?

A. *I haven't Googled tool comparisons but I have been involved with companies that have done some comparisons in the past. Based on the progression of the tools there's none that I feel that I could really use as a basis. There are a number of tools out there. When you're talking about really a requirements tool for text based requirements, there is CaliberRM by Borland. That is definitely an industry leader. IBM has their Rational RequisitePro as part of their Rational Suite. They have a new tool that's just been released that is part of their new project planning and management platform. It is basically an enhanced version of RequisitePro that contains a lot more support for visual requirements, use-case scenarios, data mapping,*

actually as part of the tool. SunTrust does use a RequisitePro. I would say it's an adequate tool. We've been able to use it but its support for files and file attachments, images, and any ability to really track changes within the tool are very slim. From the work I've done, and I haven't reviewed a lot of them, I lean strongly towards the Borland tool set. I think they've really put together some outstanding packages with their traceability and ability to package, tag, cross-reference, and then build views. Also the thing I like about CaliberRM that isn't supported in most of the tools like RequisitePro is the ability to create a Word or Excel template and generate requirements documents on the fly. In RequisitePro I actually create a view, export requirements, paste them into Word, and then do a quick reformatting. So it only takes about 10-30 minutes to build a document once I've got it set up but CaliberRM does that on the fly. There are also other tools that are much more on the visible side, like iRise. It is basically a visual prototyping tool for people with decent computer skills but no programming. You actually start with screen captures from a system and it lets you create screen flows, scenario flows, comment boxes, and hot spots where you can create mock functionality. You actually end up integrating your requirements into the screen flow and the screen requirements. Those visual tools, while hard for traditional BA's to accept and work with, actually have the best results when you start looking at the business understanding what they're getting and your development team understanding what they're going to do. So I would say, of the ones I've looked at, iRise is the leader in that area ■■■

Recently Published Reports

Connect: Batch Job Management

Rick Schwarz
Communications Manager
The IMF

Connect: Corporate Email on Personally Owned Devices

Rick Schwarz
Communications Manager
The IMF

Connect: Public Cloud Computing

Rick Schwarz
Communications Manager
The IMF

How Much Money are You Wasting on Documentation

Cass Van Gelder
Sr. Technical Writer
Caesars

Connect: Data Center Strategy

Rick Schwarz
Communications Manager
The IMF

Connect: Capacity & Performance Management

Rick Schwarz
Communications Manager
The IMF

Information Management FORUM
Real Information - Real Results for Over 30 Years

***For a complete list of all published reports please visit:**
www.theIMF.com/Published-Reports.htm

Upcoming IMF Forums

Information Management FORUM
Real Information- Real Results For Over 30 Years

Fall Senior Executive Roundtble

October 3 - 4, 2011

The Intercontinental Harbor Court Baltimore, MD

*For more information on upcoming IMF Forums, please visit our website:
www.theIMF.com/IMF-forums.htm

MAXIMIZE YOUR BENEFITS WITH...

IMF ONLINE

TWITTER

IMF Twitter updates provide the most real-time information about blog posts, published reports, and other updates related to the IMF communi-

@ITInfoForum

LINKEDIN

The IMF LinkedIn group is a great way to network with IMF members at other organizations, share information, and post questions.

GROUP ID: 1873190

BLOG

The new IMF blog provides the latest happenings at and around IMF. Check back often, or subscribe to our RSS feed!

HTTP://BLOG.THEIMF.COM/

Information Management FORUM
Real Information - Real Results for Over 30 Years

For more information about our services please contact us at: **770.455.0070** or **information@theimf.com.**