Proven, Practical Tactics for Agile IT Release Management
A Case Study

© By David W. Larsen

Proven, Practical Tactics for Agile IT Release Management – A Case Study

Table of Contents

Proven, Practical Tactics for Agile IT Release Management – A Case Study

© By David W. Larsen

Overview:

This study will explain how an IT organization delivered a release management process and results that exceeded its management's expectations and provided a foundation for continued success. There are 5 basic sections:

1. How did we get here – THE CONTEXT
2. First solution steps – DEFINITIONS, ROLES AND TRIAGE
3. Intake and Release Planning – THE CORE SOLUTION
4. Production Change Control – FINAL QUALITY CONTROL
5. Metrics and Insights – LESSONS LEARNED

Summary:

Many Information Technology organizations flounder when they are tasked to understand, organize and implement numerous changes to the system and application software serving their clients and end customers over a period of several years. This work explains at a detail level the very practical and common sense framework and processes that successfully conquered the problem for one corporation and its IT team. How successful was this framework? Frankly, IT metrics is a notoriously difficult and obscure element to discuss methodically. But this organization accomplished the following:

- In one year, it increased its client satisfaction ranking from 2.5 to 4.0 on a 5 point scale.
- In one year, it delivered 85% more change requests and project changes into production than in the prior 12 months.
- The organization exceeded its own stretch targets for throughput by 40% and change request cycle time by 10%.
- It accomplished these results with _no_ headcount increases and _no_ expenditures for IT "toolware".
- It did increase the IT expense budget by 1.8% to cover the extra cost of a single consultant to instantiate the framework and processes for agile release management.

What was the secret sauce to make these accomplishments possible? The answer requires that we carefully consider the context for this organization.

The Context:

The company and its IT department can be characterized as follows:

Demographics

Company

- Industry – telecommunications – one segment of a very large Regional Bell Operating Company
- Primary Products – voicemail service and ancillary features
- Consumer base – 4 million consumer accounts with 25% growth forecast
- Total company headcount - about 500 people
- Primary operation – a 24X7 call center of 300+ people selling and servicing consumers on voicemail products and features
- Financial Results – High Line-of-Business Profit Margins within very large corporate structure
- Everyone worked in the same building

IT Organization

- IT staff – about 60 – most with 2-10 years of organizational history
- Functionally aligned into – Operations, Project Management and Analysis, System Development, QA and Help Desk, Configuration Management
- Applications – 7 major home-grown subsystems serving the company's direct operations
- HR/Financial/Corporate functions were served by the corporate parent and processes, with interfaces
- Technology – fairly current languages, operating systems and technical infrastructure (hardware, network, DBMS)
- Recently installed improvements:
    - Software Configuration management tools, staff and processes
- Perceived primary problem – no effective control of changes submitted to production
- Everyone worked on the same floor

SWOT Analysis

The solution(s) would also require recognition of the strengths, weaknesses, opportunities and threats of the environment to produce an optimum outcome.

Strengths

- Strong and growing revenues
- Company Management – generally very experienced in call center management and product improvement processes
- IT Management – 80% had 4+ years within this organization and very little churn, only 2 levels of IT management
- Mature and successful IT processes included:
    - Project Management
    - Quality Assurance Testing
- Several strong IT manager advocates for improved Release Management
- Co-location of IT and its direct clients – the managers of the business functions

Weaknesses

- Company managers negotiated private deals to get their change requests and projects installed "earlier"
- No central clearinghouse for adjudicating departmental requests for IT changes
- No tracking system to account for all change requests and projects demanded and delivered
- About 325 requests/projects believed to be in play
- A haphazard intake and control/tracking process for "small" change requests
- Programmers could *independently* implement an application change to production
- No single point of contact/communications between the IT organization for each small change request
- Current status and target implementation date of any single change request difficult to obtain/pin down
- IT operations changes were totally independent of organizational change control and viewed as disruptive

Opportunities

- A new chance to consolidate and share information about everything on IT's plate in a single place
- A chance to leverage the existing knowledge and maturity of the IT staff
- A chance to reduce the start/stop nature of IT work due to competing and vociferous input from company managers
- A chance to incorporate IT infrastructure changes from Operations in a planned manner

Threats

- Software developers desired new toolware – not more management processes
- Company business managers enjoyed calling the shots directly with programming resources
- Tension between IT managers on what were the best paths for organizational improvement
- IT had failed on its first attempt the prior year at Change Control and Release Management processes
- Consultants rarely added value

Conclusion

The CIO, facing this situation, agreed to allow the Manager for Project Management and Analysis to contract for a resource to implement Release Management (Version 2). The CIO believed that she could deliver better results to her constituency by implementing change in a series of well-understood application package upgrades at regular intervals. She also wanted to take back to her peers a plan that they could understand and use to directly influence the order of implementation for their changes. The Manager of PM retained me as the Release Manager with the mandate to institute the processes and controls needed, and by engaging all IT staff and VPs in business departments as needed for success.

Definitions, Roles and Triage

Overview:

What was the secret sauce that made Release Management a success? The answer begins with core problem analysis, proposed solutions and a triage effort. These efforts were concluded within the first six week of the consulting engagement, setting the stage for new release management controls.

Core Problem Analysis:

The company and its IT department clearly wanted a solution implemented that avoided historical mistakes. As a retained consultant, I conducted a series of one-on-one interviews, examined remnants of Change Control documents from a predecessor, investigated commercial off-the-shelf packages for both processes and operational tracking, and discovered the following core problems.

Problem #1 – What Trees are in this Forest?

A substantial source of confusion and discussion involved defining the scope of what things should be controlled under the umbrella of Release Management and Change Control. There were divergent opinions about whether to include/exclude major projects, infrastructure changes from operations, bug fixes, hardware changes, Customer service changes, emergency patches, etc. As with most debates within IT, the stakeholders frequently used similar terms to mean very different things. Very specific language needed to be written, socialized and implemented that reduced this ambiguity and confusion.

Problem #2 – Who is Responsible for What?

Projects were very clearly controlled end-to-end by Project Managers, and at any given time the 4 PMs would have 2-5 projects underway. These generally covered scope for multiple applications and multiple person-months of programming effort. Beyond that good start, Client Change Requests, Bug Fixes, Operations infrastructure changes, Customer service changes (move/update/fix my workstation) and emergency patches had no one role identified for control, communications and accountability throughout their life span. The IT assembly line for such work was disjointed at best and lacked fundamental structure. The role of Release Manager itself was undefined, with various stakeholders holding unique viewpoints on the scope of the assignment.

Problem #3 – How To Introduce Order Upon Chaos?

This was a very critical concern, as the inflow of new requests from the business could not be halted, due to political and practical matters. If the organization knew tactically where it stood on Monday for every item, by Wednesday the landscape had changed, and priorities for older work were being adjusted on-the-fly, either overtly or covertly by business leadership.

Problem #4 – How Frequently Should We Release Change Packages?

A practical concern was whether IT should embark on weekly, biweekly, monthly, quarterly or other frequency of planned production software change. The frequency of change would end up driving the timing of the real-world series of gates and meetings necessary for control and adjustment.

The first solution proposal entailed purchasing a complete software application and documentation package from a market leader that promised to cover the full scope of their interests. The alternative proposal was founded upon a low level of automation, and a high degree of inter-personal collaboration to achieve similar ends.

The rest, they say, is "Scrum/Agile" history. To learn what it really takes, our story continues next with DEFINITIONS, ROLES AND TRIAGE.

Definitions

I will remark on Definitions first, because the management team needed to ground itself and communicate in a consistent fashion about the key objects and controls with Release Management processes.

Problem #1 - What Trees are in this Forest? - Scope

We took the perspective that anything that is planned to change the configuration of the production computing environment within the controlled data center and network configurations was subject to Release Management processes. As a result, <u>*we included*</u>:

- Application software changes requested by clients *or* from IT itself (re-factoring, etc)
- Application software fixes that were "not immediately damaging" clients' business
- <u>Each</u> software change package from projects (projects typically had more than one release package over several months)
- Network or server infrastructure upgrades (OS, DBMS, middleware, hardware, etc.).

<u>*We excluded*</u> from Release Management processes:

- Customer Service requests (fix/move my workstation, office moves, etc)
- Emergency production software application fixes (fix it now)

This last exclusion was troublesome, but necessary. We assigned total responsibility for managing the emergency fixes to the Software Development Manager, and set an overall target to keep these to fewer than 5% of the total changes made into production. (We tracked the numbers, but didn't stand in the way).

The practical outcome of these agreements was that each individual thing included in Release Management was a <u>Change Request</u>, to be initiated with a simple form. Each would be assigned a unique Number (and key attributes) and controlled in immediate form by an Excel spreadsheet updated and distributed by the Release Manager.

Problem #2 – Who is Responsible for What? -- Roles

Single Point of Contact (SPOC)

The organization had a good model of behavior and accountability for projects, but there was disjointed accountability for all the other Change Request types. To solve this, we defined a role called the Single Point of Contact (SPOC). The role was accountable for conveying the requirements, correct status of IT progress, and sponsoring the Change Request for its ultimate release to production. The SPOC was individually accountable for telling our clients the timing and impact of the implementation of a change, so that our clients were adequately prepped for a release. The assignments to this role were expected to last for the duration of the Change Request, as opposed to the previous pattern of shifting responsibility. As a practical matter, the SPOC assignments for 75% of the Change Requests fell into the Project Manager/Business Analysis team.

Architecture Review Board (ARB)

The IT organization had a defined group called the Architecture Review Board (ARB) which convened to assess the technical and organizational impact and risks of major changes to the environment. This group consisted of the 5 IT Managers, the Applications Architect and the Operations Architect, and occasionally the CIO. As part of our definitional work it was determined that this group would exercise an expanded role – to quickly and routinely review each incoming change request. The Release Manager was added to the Board. This board was the "neck of the funnel" for all new items and through discussion, they determined very rough size, priority, and impacts. The ARB also made the specific assignment of a SPOC to each new Change Request. More on the role of the ARB is covered in the Intake and Release Planning section.

Release Planning Group (RPG)

The primary organizational element that needed to be set in place was a new group titled Release Planning. This team, facilitated by the Release Manager, met with great discipline and regularity to organize, re-organize, and commit to a comprehensive, concrete order of implementation for all Change Requests. While this sounds straightforward on paper, remember that the context for this role was to organize an average of 125 Change Requests into a series of planned releases – and do this repeatedly as new things got added each week. This was a puzzle with ever-moving parts. The Release Planning Group consisted of the 5 IT Managers and the Release Manager. The Release Manager published the current Release Schedule as an outcome of each of the group's meetings.

Change Control Board (CCB)

This group was chaired by the Configuration Management leader, and had the responsibility to review and approve or defer the completed Change Requests for implementation in production. The Operations Manager and QA Manager played strong roles within this forum. The SPOCs for each Change Request were questioned for preparedness items, including the advance notification of the client communities. The CCB made a consensus decision on each Change Request and the outcome of these decisions allowed the Release Manager to lead the Configuration Management Team to prepare the scripts and code packages for production upgrades.

Problem #3 – How To Introduce Order Upon Chaos?—Business Cycle

This fundamental problem afflicts all business organizations. Customer requirements constantly change in nature, new ones are added, and old ones wither yet refuse to die. Progress on the production line in IT is swift, stuttering, under-resourced, or overwhelmed. Managers independently made decisions from their own perspective of the best interests of the company. Frequency of change was sporadic.

To introduce order, the Release Manager defined a disciplined business cycle for Release Management Processes. The business cycle was a repetitive set of scheduled meetings of the Architectural Review Board, Release Planning Group, and Change Control Board that would be executed with defined agendas and deliverables, without failure, and with full participation of the people in their assigned roles. This business cycle would commence as soon as a triage of the Change Request queue of work could be completed. Customer feedback loops were defined for each stage of the process. Triage was a critical first step and is discussed further below.

The plan for this business cycle was presented to the CIO. She approved the plan for this business cycle, committed her management team to its principles, and successfully sold the plan to her peers in the company.

Problem #4 – How Frequently Should We Release Change Packages?—Solution

The debate on this was not difficult – once we had made the earlier decisions to include operations change requests and exclude the emergency software fixes. We settled on a 2-week release change cycle. Our internal customers were already seeing changes made (or attempted) weekly with mid-week exceptions and surprises, so this could have been viewed as a step back by IT. However, the IT managers saw many shortcomings with more rapid attempts at change and were far more confident that the company would be well-served on a 2-week cycle. Specifically, change requests would be bundled together into a Release Change Package for implementation on alternate Thursday evenings. Our fallback position, if the Release did not succeed on Thursday night, would be to switch to a Friday evening implementation.

Triage

At this stage, the CIO evaluated how quickly all this good foundation work could be put into operation. As Release Manager, I advocated for a process solution supported by an industrial-strength commercial application that could be leveraged toward portfolio management, with many people updating their component parts, and project-specific support and tracking. However, finding, funding, purchasing and implementing such a baseline tool would require an estimated 3-4 months under ideal conditions. The CIO opted to proceed with the alternative “low-tech” approach for her organization. The mandate was to “find a way” to implement the essential processes by using the lowest-budget approach.

The mandate was daunting. The prior “Change Coordinator” person had worked from a Change Control log in Excel that had fallen into disuse. The root causes for that condition appeared to be that the information could never be kept current, plus it only covered some of the Change Requests. No one had previously been assigned the responsibility to actual know and communicate the status of “small” change requests – of which there were several hundred. The log also attempted to store a lot of interim dates on small changes, and it duplicated interim dates that Project Managers maintained in MSProject on regular projects. The SPOC role had not been defined. As far as we could tell, 125 Change Requests were “open” – meaning submitted by clients and not yet completed. That number fluctuated each week as new ones got added and some got finished, but no one was certain of the status of each item.

It seems appropriate to define the term triage (courtesy of dictionary.com)

–noun

1. the process of sorting victims, as of a battle or disaster, to determine medical priority in order to increase the number of survivors.
2. the determination of priorities for action in an emergency.

The IT managers knew we couldn’t cope much longer with incorrect information about all the victims (Change Requests) that were littering our battlefield. As Release Manager, I asked them to devote the resources necessary to obtain a current, accurate view of the following for each Change Request:

1. Change Request Number
2. Customer Name
3. Change Request Label (very short description)
4. First Requested Date
5. Status (Open, Completed, Being Worked, Cancelled, or Duplicate) (if completed, wanted a completion date)
6. Target Release Date (Not Available was OK)
7. SPOC Name

I was responsible for facilitating the triage process. This primarily consisted of some very long all-manager meetings, publishing many versions of a new Release/Change Request log in Excel, assigning segments of the list to the most knowledgeable workers for update, and numerous interventions. The sorting process consisted of IT managers agreeing on an initial High, Medium or Low priority for an “early” Release Target per Change Request. Customer VPs were also polled on their priority settings for Change Requests. The process was declared “finished” in 3 weeks. We achieved a stable state of Change Request information in the log and were ready to begin Release Management processes and the business cycle for control.

Conclusion

So at the end of this 6 week period, we had the following:

- A business cycle for gates/control meetings
- A set of definitions of work objects, deliverables, roles and processes
- A reviewed, organized list of work
- Understanding between the CIO and VPs on how the new processes should work

The saga of Scrum/Agile IT Release Management continues with THE CORE SOLUTION.

The Core Solution

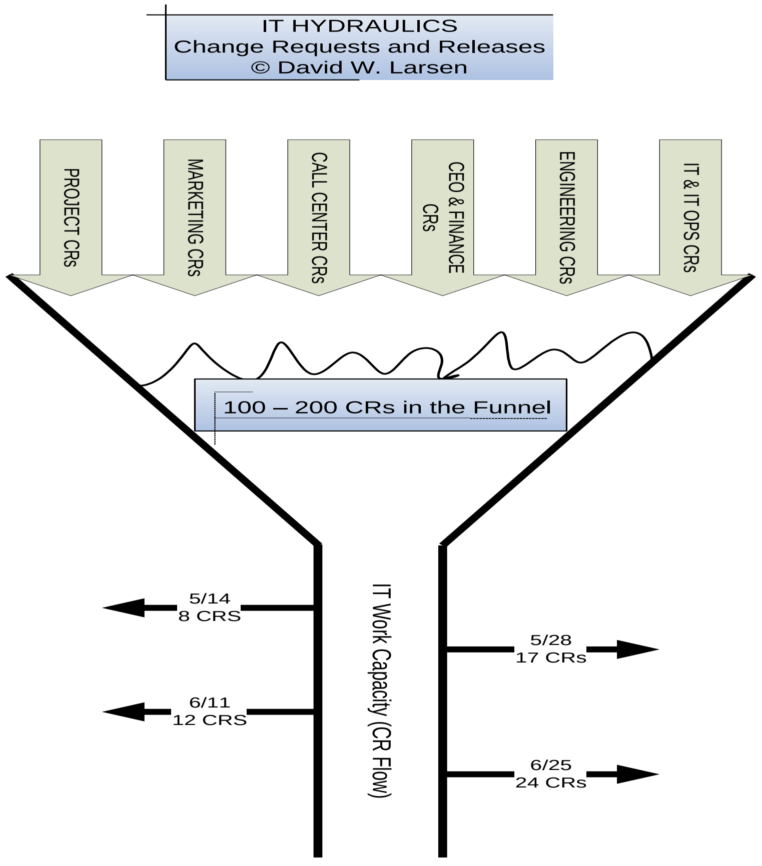
Overview:

What was the secret sauce that made Release Management processes a success? We thought we had taken the right first steps, but hadn't executed anything, really. The secret sauce begins with three foundation ingredients or elements – a paradigm shift, visualizing the whole problem domain, and knowledgeable collaboration.

Paradigm Shift – From Resource Balancing to IT Hydraulics and Kanban

The CIO and others initially assumed that the path to improved productivity would be an outcome of matching forecasted demand for service and forecasted supply (essentially of programming and QA staff), enabled by capturing a lot of size and capacity metrics, personal vacations, hours spent on work in progress, etc. In simple terms, each change request would be estimated (repeatedly) for hours of remaining effort from each source group or individual, run through a number cruncher that was time-sensitive, and out would come a resource-balanced schedule of releases with their associated change requests. Let's call this the MSProject model for short. In theory this works for standard parts and production lines, but the model fails miserably in the face of human nature, software craft, and changing business priorities and demands.

The first shift was to recognize that the IT environment behaved according to the principles of hydraulics. You may refer to the following graphic for this discussion:



We had a **funnel** or holding tank for Projects and Change Requests representing customer demand that was fed via "fire hoses" from the department heads. There was no limit on how many CRs could be pushed through a fire hose in a time period. The holding tank consisted of the 100-200 change request "forms" for the year of this case study.

Exiting the funnel was a **pipeline** containing a continuous stream of assigned work. The IT work capacity was a pretty steady state of 60 people working full time each week. Of these, about 40 were software developers and QA staff. These resources could work on any of the change requests from the holding tank, and it was management's job to direct them to work productively *and collectively* on the most important ones. Some change requests could be fulfilled with mere hours of effort. Many took person-weeks and person-months across IT organizational lines. Also, software developers operated with a limiting principle that they would **not** have more than 2 open "code branches" for an application. During this case study, IT software development capacity (pipeline) was held constant by a fixed budget, fixed headcount, and existing toolware.

The hydraulics system pipeline had **relief valves** – called releases that were scheduled for every 2 weeks. The system would thus "flush out" a variable number of change requests into production, preventing an overflow of the holding tank. Adding additional pressure from the holding tank above could result in some short-term flow improvements, but excess pressure (or push) invariably resulted in pipeline cracks and disruptive failures. Another primary limiting consideration is that only certain people could work productively on certain CRs due to application or toolware expertise. A prime example was that only 2 programmers knew how to implement change using Oracle Forms software, and had the appropriate licenses and training. There were a number of similar cases defining who worked on what. IT was teetering on the boundaries of failure because an important determinant of the success of production scheduling based on "pushing" the demand is the quality of the demand forecast which can receive such "push".

So IT management reconsidered its position regarding a "push" form of scheduling. In IT's environment we did not have standard production line equivalent inputs and outputs. As a consultant, I introduced a pull-based approach, based on my exposure to Kanban theories. To quote from an encyclopedia "Kanban, by contrast(to push), is part of an approach of receiving the "pull" from the demand. Therefore the supply, or production is determined according to the actual demand of the customers. In contexts where supply time is lengthy and demand is difficult to forecast, the best one can do is to respond quickly to observed demand."[1]   The theory sounded better than pushing stuff toward the programmers, but how do you pull it off?

Visualizing the Whole Problem Domain

The second ingredient for the secret sauce was absolutely crucial. We found that during the triage effort (working from Excel spreadsheets of 300 items and Change Request form documents) that it was literally impossible to use the many lists and pieces of paper, on projectors and laptops, and simultaneously make priority decisions in context of the whole set. It was a condition of too much information, too many people looking different places at once,

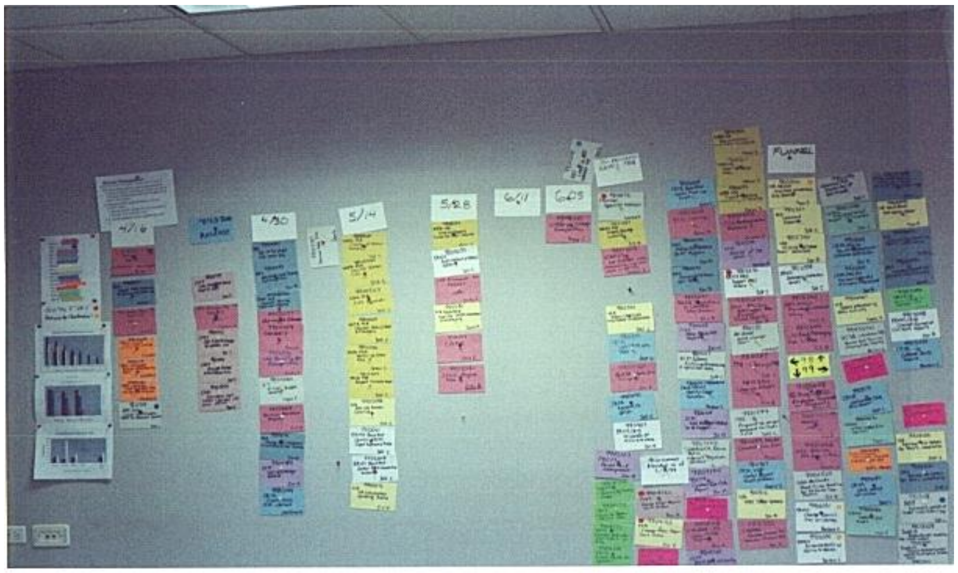
[1] http://en.wikipedia.org/wiki/Kanban

etc. I was failing to facilitate the priority-setting and release-bundling effort. Then we got really Scrum/Agile in technique. I decided to transform our favorite conference room into a "Visual Decision Space" using:

1. One ***colored*** 4X6 index card for each of the 125+ change requests – all info on a card was readable from 15 feet away
2. One 8'High X 15'Wide fabric-covered wall that allowed me to use stickpins to tack up the cards, and easily move them from spot to spot
3. Six Vertical columns on the wall each representing a 2-week release period
4. One overflow portion of the wall for CRs not assigned to a release period.

The puzzle of what should be positioned in each release could now be readily represented and dynamically manipulated by moving the cards from column to column or put in the overflow (funnel). Everyone in the room could simultaneously see and evaluate each move in context of the whole wall of CRs. Each manager could consider in real time how a move would affect the resources and throughput of their group in the context of 6 release cycles. All the lightbulbs flashed on at once.

This concept of a planning/production storyboard was certainly not new in the literature of production management, but the adaptation developed within our IT department in 1999 certainly preceded many of the authored works on enterprise requirements managements of the early Agile Alliance advocates. The Agile Manifesto itself can be dated around February 2001. Practitioners of Scrum techniques had published some good works on work planning, but they did not have widespread popularity at this juncture. With this second ingredient of "THE WALL" in place, and a very large DO NOT TOUCH sign on the wall of cards, the secret sauce was nearly complete. Here is a photo of the board itself:



Knowledgeable Collaboration – Making It Happen

The third ingredient of the sauce was the IT management staff's willingness and ability to collaborate on making decisions in an informed and friendly way. The wall of index cards provided the puzzle space to let managers propose how they could arrange the priority work in packages that made sense given their resource constraints and track records. No longer were we trying to sum up work estimates and do math problems and forecasts. Instead we could pick out 3 yellow cards from the overflow and say they should go together in the third release because they "hit the same code" or collectively made business sense for the client. Instead of agonizing over potentially crushing any one bottleneck, when a manager *felt* their unit had reached their max effort for a release, all managers would quickly adjust and move on. Sure, a lot of guesswork was involved, but we also knew if we were over- or under-committed in a release, our staff would read the wall and point out some possible adjustments that we could make in the next planning session. The process of rearranging THE WALL took place every week with all 5 IT managers, and took one hour or less, even when 10-20 new CRs arrived from the customer departments that week.

By blending knowledgeable collaboration with a paradigm shift to pull-based planning and a visual space for the whole problem domain, the secret sauce was ready to apply!

The Intake and Release Planning Cycle - IT

People abhor unproductive business meetings.
People love solving problems collectively.
Everyone likes efficiency.

Simply put, our intake and release planning cycle was a weekly cycle that went like this:

1. Every day the Release Manager would receive zero to many Change Requests on emailed forms.
2. That same day, the Release Manager assigned CR numbers, updated an Excel-based Change Request Log, acknowledged the author, and routed the CR form to the members of the Architectural Review Board.
3. Throughout the week it was the job of the Applications Architect and the Operations Architect to do a first cut investigation and resource estimate of these new CRs for use in an ARB meeting.
4. On Tuesday mornings at 9AM the Architecture Review Board met to consider the CRs submitted from the previous 5 business days, hear from the Architects, discuss the client's priorities, assess risks, and assign a SPOC for each CR. The meeting ended faithfully on or before 10AM.
5. Tuesday between 10 and 10:30, the Release Manager made up new CR index cards for the wall, choosing the right color (colors were unique to each application area, plus some special colors for projects and IT internal stuff).
6. Tuesdays at 10:30AM, the members of the Release Planning Group convened in the conference room holding THE WALL of CR index cards. The first order of business

was for the Release Manager to ask for an immediate consensus on where each *new* CR should be placed. The default position was in the holding tank, but some could quickly be slotted for a pending release package. As a second order of business, the RPG would then begin adjusting the content of the six release columns on the wall, based on known problems or opportunities. Members would propose moving cards back and forth, discuss what they thought, make a decision and move on.

7. By 11:20 the Release Manager would call a halt to the discussions and do a “session wrap”. It was common to see 15 to 30 cards change position in the course of a single meeting. No one kept any minutes, but the new WALL clearly showed the Release Plan for 6 more Releases.
8. By 1:00PM on Tuesday, the Release Manager made all corresponding Release date changes in the Excel file (Change Request Log/Release Plan) and routed it to the entire IT department and the business VPs.
9. By 5:00PM on Tuesday, the individual new CR authors were emailed on the status of their submitted requests. For “older” CRs, the assigned SPOCs were responsible for communicating any Release date adjustments.

Intake and Release Planning Cycle – Client

The commitment to keep the customers apprised of the status or disposition of the requests took several forms. As noted above, on every Tuesday the Release Plan was distributed as an Excel document to the VPs of the organization. All the new CRs submitted that week also had the individual feedback emails. The second part of the client cycle was a monthly review meeting scheduled with each VP, with the Project Management Manager and Release Manager attending. This meeting was facilitated with Change Request Log (Excel) reports which exposed solely the departments’ own Change Requests. The goal of this meeting was to clearly identify that department’s current “Top 5” Change Requests. Month to month these items changed with completion of work and shifting business interests. In turn, the Release Manager updated the Change Request Log ***and*** pasted a “Top 5 dot” on those CR index cards on THE WALL. This proved to be a great way to let all IT managers visualize the top priority work during Release Planning sessions. These monthly client meetings were frequently held in the conference room where the VPs could easily see the scope of the IT workload and where their own CRs were queued up.

Conclusion

The Intake and Release Planning cycle operated with friendly precision. People did not miss meetings. Everyone actively participated within their role *because it was efficient*. The Release Manager performed all record keeping, and was acknowledged for the accuracy and timeliness of the Change Request Log as a “routable” version of THE WALL. But whenever anyone in the organization wanted a comprehensive appreciation of the whole business, they would sneak into the conference room and read THE WALL. The CIO would occasionally bring in the President to demonstrate where everything stood. This was truly an information radiator (thank you for the term, Alistair Cockburn).

Final Quality Control

Overview:

It has been said that the Plan is nothing, Execution is everything. No Release Management process is complete without the correct steps to implement changes into IT production safely, securely and with acceptance by the client community. The IT organization still had some work to do in this area at the beginning of the consulting engagement. Here is how we improved, enhanced, and succeeded in the final quality control and implementation steps of releases.

Objective Setting:

In the first weeks of the consulting engagement I was encouraged by the progress already made by the Configuration Management team in beginning to exert discipline on what changes got implemented in production and how source and object code management tools (Clearcase) were being applied to support developers. What I didn't find were any metrics or objectives for these processes. Were they being applied with great consistency or not? How could targets in this area be set? The best processes in the world must be executed successfully. Based on interviews, it seemed that about 80% of the changes that went into production were going through the Configuration Management team, and perhaps 20% were still being done "programmer-direct". A lot of reasons (excuses) were offered.

Given this backdrop, I prompted the IT management team to set a new goal for the disciplined configuration-managed deployment of software to production at 95% of the deployments. Setting this higher objective allowed for some middle-of-the-night direct patches for emergency fixes, but IT made a resolute commitment to keep the source and object code integrity through quality-driven deployments. The results were pretty remarkable. The new objective itself caused behavioral changes in the programming staff, better collaboration with the configuration management team, and the outcome could actually be monitored and reported. In the course of 10 months, 98% of the deployments were done through the configuration management team. The Software Development Manager insisted that even "emergency" changes in his total control should follow the better path to production.

Quality Gates

There were 3 principle quality gates that improved during the engagement. They were:

Passed QC Testing

The Quality Control group in IT had a strong leader and pretty talented and experienced testers. They had a firm grip on their processes and knew what they were doing. Viewed as the end of the chain, they often got the short end of the stick for the proper timeframes to do their job. Their work got supported and strengthened by three key things:

1. The QC group was able to insist that any code they were to test had to flow through the configuration management team first. This was great discipline to apply.
2. The QA Manager attended all the Release Planning meetings and added a key piece of information on all CRs slated for the next release. She would place a green dot on all CRs that had passed the QC steps, denoting her team's signoff or not. Also, she would place an orange dot on any CR that was late in getting their code into QA, based on project or CR-specific target dates. "Dot-placing" was done immediately prior to the Tuesday 10:30AM RPG meetings.
3. During the RPG, the IT managers held focused discussions about the quality state of CRs and often made decisions to defer items to the next release or apply more resources to get a "Green-Dot" status. The QA manager wielded a lot of power in a short amount of time because the conditions were correct for these discussions.

Passed Change Control Board (Checklist)

As a reminder, earlier we stated about the CCB:

"This group was chaired by the Configuration Management leader, and had the responsibility to review and approve or defer the completed Change Requests for implementation in production. The Operations Manager and QA Manager played strong roles within this forum. The SPOCs for each Change Request were questioned for preparedness items, including the advance notification of the client communities. The CCB made a consensus decision on each Change Request and the outcome of these decisions allowed the Configuration Management Team to prepare the scripts and code packages for production upgrades."

At the onset of the engagement the Change Control Board met infrequently (mostly to review implementation of only major projects). With a firm commitment to release production changes every 2 weeks, this role and the execution of its duties was firmly reinforced. A standard agenda was prepared and meetings were facilitated by the Release Manager. As Release Manager, I also proposed that the SPOC present answers to a checklist of items for every CR and to bring the checklist(s) to the CCB to aid in its decision. The checklist approach was useful and helped keep the key discussions focused. If only 2 CRs were being proposed, these meetings were mercifully short. If we had 15-20 CRs in a release, the meeting was suitably extended, but we never had a single meeting exceed 90 minutes.

As a practical matter, the CCB meetings were conducted on alternate Wednesday afternoons, just prior to the Releases done Thursday night. That left the right amount of time for the Configuration Management team to do its staging job effectively. On rare occasions, the CCB agreed to grant a SPOC (and the developers) another 8 hours to "get ready" for CCB approval, but these cases were true exceptions to the rule.

Passed Deployment Testing

We mentioned previously that the QC group included 2 functions – software testing and also help desk/customer support. The 3 people assigned to the Help Desk were extremely knowledgeable in the client's use of applications and were always assigned as the post-deployment testers for application changes. They would know if anything didn't work or "looked funny" and could order a rollback to the previous production code. The Help Desk staff also covered the non-prime-time shifts of the Call Center to resolve problems. They didn't want bad code going into production.

We enhanced the Deployment Testing process in a major way with the simple act of scheduling releases at 2 week intervals. The previous practice had been for developers or project managers to individually negotiate with the QA manager and team for releases on any night of the week or on weekends. The Help Desk wanted to be helpful, but was forsaking any semblance of a real life with the requests to have production deployments and testing happening 2 to 3 times per week.

We also made a smart move by scheduling deployments on Thursday evenings. This allowed the Help Desk to work late on alternate Thursdays, but also "earn" Friday afternoons off for a long weekend. Simple way to build morale and improve execution! This scheduling pattern also had benefits for the client community as the routine was familiar and consistent. In selected cases, they would perform end-user testing as well on Thursday evenings.

One oddity I should mention. I was the Release Manager, but I played a very hands-off role on the practical decisions needed on Thursday night deployments. Our Help Desk Team was very accountable for the benefit of their users and made all the decisions you typically encounter on busted deployments or smoke tests of production. This was an outstanding example of a self-managing team.

Conclusion

These improvements in the Final Quality Control steps built a lot of credibility for the IT organization. We saw a marked decrease in the number of Change Requests that failed the Deployment Testing gate and a marked upswing in customer confidence in implementation of change. We also found ourselves much more capable of backing out, recovering and then re-executing a configuration-managed deployment when necessary.

To summarize the improvements achieved, our story continues with LESSONS LEARNED.

Lessons Learned

Overview:

This case study is entitled Proven, Practical Tactics for Agile IT Release Management. Now is the time to assess how "Agile" were we? This Release Management process was implemented in 1999, without benefit of access to the thoughts and ideas published following the Agile Manifesto. We also have some basic metrics to consider and explain, and thoughts on the lessons along the road.

How Agile Was This Work?

I willingly concede that there are experts in the Agile community who are far better qualified to render an opinion on how closely this work conforms to the principles of Agile Software development and the complementary Scrum approaches to Product and Enterprise Requirements management. On the one hand this was not a discussion of software development. The Agile Manifesto states

"We are uncovering ways of developing software by doing it and helping others do it. Through this work we have come to value:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan"[2]

Our process work was very steadfast, disciplined and critical to success. Our interactions were frequent and very focused. We used plain cheap tools, but exceedingly well. Individuals – we used everyone's strengths to succeed. I'd give us a grade of a B on item 1.

The Release Management process itself took virtually no notice of the interim steps of software development. In fact we stripped out tracking of interim dates, then put back in the importance of starting QA. The only thing we worried about was production-ready software. I'd give us a grade of A.

On customer collaboration, we certainly improved communications about what was being worked on (and what wasn't also was obvious). We definitely showed the VPs that we were trying to slide their Top 5 requests in at the earliest juncture in the overall plan. The Release Management process did not operate at the level of the software's requirements, design and functionality. In essence we just did a great job of clearly starting and stopping work. I'd give us a B+ on item 3.

This process excelled at responding to change over following a plan. Every week we would build a firm Release Schedule for 6 Releases, and the very next week we would re-work the whole thing due to circumstances and reality. We did that with clarity, collaboration, understanding and high levels of communication. I'd give us an A+ here.

The basic work planning and release business cycles were closely aligned with the Scrum techniques advocated in the industry literature of the time by Ken Schwaber, Mike Cohn, and others, but the basic pattern of software development at this organization remained a waterfall, with at best an iterative approach to meeting clients requirements.

Metrics

I will state my personal opinion that 99% of the published material regarding IT processes lacks meaningful statistical indicators. There is a lot of "crowing" about methods and tools, but not a lot of believable concrete information. Also, keep in mind that IT headcounts were held constant for the periods in question.

[2] Manifesto for Agile Software Development © 2001 at http://agilemanifesto.org, 17 signatories

Here is a sample of the data and metrics we collected:

	A	B	C	D
1	Release Management Selected Metrics			
2				
3		Year End 1998	Year End 1999	Improvement
4	Customer Satisfaction - IT	2.5	4.0	60.0%
5				
6		4/1/98 - 3/31/99	4/1/99 - 3/31/2000	
7	CRs Completed or Cancelled	210	373	
8	(including Project Releases)	222	410	84.7%
9				
10		4/1/98 - 3/31/99	4/1/99 - 3/31/2000	
11	Major Projects Completed	12	18	50.0%
12				
13		As Of 5/31/1999	As Of 5/31/2000	
14	Average Age of CR Backlog	97 days	154 days	-59%
15				
16		As Of 5/31/1999	As Of 5/31/2000	
17	Avg Cycle Time-Completed CRs	85 days	76 Days	10.6%
18				
19		As Of 5/31/1999	As Of 5/31/2000	
20	Size of CR Backlog	118	111	5.9%

Customer Satisfaction – IT

The CIO conducted a very simple poll of the senior managers in the organization each year, asking for an overall degree of satisfaction with the IT performance for the prior year. On a scale of 1 to 5, the 10 managers selected from Very Unsatisfactory (1) to Outstanding (5). This simple scoring did not differentiate between performance in Operations, or on Projects or on implementing Change Requests. It was the simple view of their Overall Satisfaction. We believe that the efforts on release management were a major factor in raising the score from the prior year. Another major win was that the IT organization turned the corner on the Year 2000 without mishaps.

Change Requests Completed or Cancelled

The consulting engagement began in early April of 1999. At that point in time the definition of Change Requests did *not* include production software changes caused by major projects. Using the new definitions of what Release Management considered an in-scope Change Request, the base of Change Requests Completed was expanded for the earlier time period so that a fair comparison can be drawn. The IT organization, using Release Management, dispatched about 85% more Change Requests over 12 months.

As a parallel metrics observation, the IT group set annual targets for completing change requests. Their goal for the year 1999 was 240 (this was considered an aggressive target at the time). On a calendar year basis, IT completed 336 Change Requests in 1999.

Major Projects Completed

In case people wonder if IT just re-directed effort to do more change requests, thus short-changing the efforts on projects, the numbers for major projects are shown. We do not know what % of total resources were used year over year, as project-hour accounting was weak. Given that the Year 2000 Project was a major endeavor, I offer that it is safe to assume that there was no disproportionate shift of resources that favored better Change Request results.

Average Age of Change Request Backlog

We decided to consider how well we were doing in terms of reducing the amount of time the clients were waiting to get their Change Requests taken care of. For the period in question, we were amazed at first to see the age of the backlog increase dramatically! On reflection, we realized that our data proved that lower-priority change requests were always being bumped back in their potential implementation, so the department could handle the new incoming higher priority changes. We could never empty the funnel, we could only work on the most important stuff first.

Average Cycle Time of Completed Change Requests

We measured the overall turnaround on items that were being completed and saw that it was trending favorably – a 10% reduction in "Wait Time" as seen by the user community. While this was not a targeted metric, we had a baseline that would be interesting to watch.

Size of Change Request Backlog

In a similar vein, we kept an eye on the total change requests not yet completed. We saw minor fluctuations, but the client community was always submitting more improvements. As a very general statement, we always had about 3 months of work capacity either as work in process (WIP) or requested, but not WIP. There was no budgetary chargeback mechanism from the IT department to the VPs, so asking for more IT work had no direct consequences for them.

5 Key Lessons Learned

1. First and foremost, the direct investment made in Release Management implementation brought better than expected results for the stakeholders.

2. I was amazed and delighted to see the Wall of Index Cards morph over time to be a more elaborate information radiator for the organization. One example was the addition of colored dots to the cards for Top 5 and also QC status. We also got tricky with positioning cards above and below certain horizontal lines to convey new information. We also started to display the thermometer of completed change requests versus the annual target (it was uplifting). There is a lot of truth to the adage that you learned everything you need to know in kindergarten…..

3. The solutions we applied were just about perfect for a collocated organization with the configuration management and QC processes in place and an organizational commitment to release management.

4. The CIO, seeing that the process was successfully embedded, at the end of 1999 asked the consultant to undertake 2 items:
    a. Do a fresh study of the commercial software market for supporting tools in the Release Management arena – ***none were found*** that could match the team effectiveness we achieved with cards on a wall.
    b. Prepare a transition plan to bring in an internal manager for the ongoing position of Release Manager. It took over 4 months to locate and train a replacement candidate for the permanent position. The first chosen candidate just couldn't keep the pace of detailed item management that was required.

5. If this problem had involved 600 Change Requests, it might not have worked at all. As long as we had fewer than 200, we could handle it on one wall and you could read every card from about 15 feet away. There are limits to this media/storyboard approach.

Conclusion

There is a huge amount of value to creating a Visual Decision board that covers the whole problem for an organization. This general finding can be applied in a low-cost manner to many problems. In my research to find an adequate software package solution for Release Management, all products stumbled on the problem of scale on a 21" computer monitor. To this day, the tenth anniversary of this endeavor, only very sophisticated hardware systems and conference room environments begin to match THE WALL and the practices we used. I smile each time I see a modern spy movie, or "24" or "CSI Miami" characters dazzle the audience with technology for the virtual space, index card objects and puzzle manipulation approach, It doesn't have to be rocket science.

If you would like to contact me to discuss down-to-earth process improvement in your organization, email dwlarsen1946@gmail.com