

Mastering IT Change Management Step Two: Moving from Ignorant Anarchy to Informed Anarchy

Ken Dietel

Duke University Health System
2200 West Main Street, Suite 450, Durham, NC 27705 USA
011.919.286.6322
ken.dietel@duke.edu

ABSTRACT

Where does one begin to address the multitude of issues surrounding implementing a Change Management process from scratch? We take a look at one IT support organization's beginning steps.

Yesterday: Each different IT support group received, processed, and implemented change requests differently. End users never knew when a change was coming. Other support teams were surprised when systems suddenly acted differently. Many incidents were a direct result of the IT organization breaking their own systems.

Today: Create and implement a standard process flow, identify interdependencies among systems, recognize relationships within the organization, and set up lines of communication.

Tomorrow: Interested parties are aware of when changes are scheduled to be implemented. Improvements to Change Management procedures are made reactively. Incidents resulting from changes are identified quickly. Ready for Step Three: Moving from Informed Anarchy to using a defined process for managing changes.

Categories and Subject Descriptors

D.2.9 [Software Engineering]: Management – *software configuration management*

General Terms

Management, Standardization, Documentation, Measurement.

Keywords

IT Change Management, production, implementation

1. INTRODUCTION

Everyone in the company knew there was opportunity for improvement. On a regular basis, a scenario like this would unfold:

The IT help desk would suddenly receive multiple calls from users reporting that a computer system was acting strangely, or not working at all. The help desk analysts would research the situation,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGUCCS '04, October 10-13, 2004, Baltimore, Maryland, USA.
Copyright 2004 ACM 1-58113-869-5/04/0010...\$5.00.

but find no previous record of the indicated behavior, so would forward the incident report on to the system developers. The system developers would report back that a change had just been implemented, and that the system was now supposed to be working that way. Or, it would eventually be discovered that another system had been changed, and that system's developers would be surprised to hear that their alteration had had an effect on this system.

I call this environment "ignorant anarchy". Each group (developers, help desk, users) was operating independently of each other. The Software Engineering Institute's (SEI) Capability Maturity Model (CMM) ranking for a software process exhibiting these characteristics is maturity level one (the lowest of five levels): "characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics." [1]

Management showed both its acknowledgement of the situation and a willingness to change it by creating a new job position within the computer operations group to tackle the task of implementing a change management process. With the other senior members of the operations group, a process improvement team was formed. Each person specialized in one IT support process, roughly as defined by the ITIL methodology. Change management, incident management, and the service desk function were the areas specifically targeted by this team.

This is about the limit of what may seem to be the obvious first steps toward solving this business communication problem. The rest of this paper will concentrate on what happened next. If Step Zero is acknowledgement of the problem and Step One is a willingness to do something about it, then Step Two becomes the next challenge: beginning the alteration of ingrained habits.

In this case we will examine one IT support organization's approach to implementing changes to their production systems. Not claiming to be The Answer, this paper will illustrate but one approach and identify some positive and negative aspects. This is an IT organization of approximately 300 people maintaining over 100 systems, implementing 60-75 changes per month. The issues addressed should apply to organizations of other sizes starting from the same level of process maturity.

2. TERMINOLOGY AND GOALS

As with any good business process re-engineering effort, initial steps included defining the terminology and setting the goals. The process improvement team defined a change as "Any event that alters the state of any production IT service". IT services were commonly thought of as collections of "systems", which included software, hardware, network, and facilities. Surprisingly, there was less debate

about the terms “service” and “system” than there was about the term “production”. Some groups didn’t realize that their systems would fall under the auspices of this new change management process. For example, the software that automatically backs up the standby database that an application accesses when the primary database is unavailable is indeed a production system, even though it may be three servers removed from the user and not currently in use. Also, every single work order for a network jack activation or user account creation fell under this umbrella definition. The fact that development systems were not included was important. By leaving them out, the multiple development teams retained ownership of their disparate development cycles, and let the operations group only become involved once an enhancement request (confusingly also referred to as a change request) was ready to be rolled out into production and needed to be supported.

The objective of the new change management process was to “minimize disruption of IT services to customers through the use of a standard process to communicate and implement changes”. Goals to achieve that objective were: “mitigate and manage the risks associated with IT changes”, “proactively communicate changes and the expected impacts to customers and internal staff”, “establish a central change schedule that includes all scheduled IT changes”, and “establish a reporting process for change management that validates control of IT changes”. A grand vision, but some of those goals were difficult to measure to prove achievement.

3. PROCESS DEFINITION

The process improvement team created a standard change management process flow and associated procedures. The base flow started with a change request being submitted, then approved, reviewed, announced, and finally implemented. A minor, a moderate, and an emergency path were defined, based on expected impact and timeliness of the change. The difference in the three paths were that minor changes did not need to be reviewed by the change board, moderate changes would follow the full workflow, and emergency changes would both skip the review and allow notification communication to occur after implementation. Management readily agreed to this design, as it was an improvement over having no standard at all.

3.1 Submission

Using the resources most readily available, a repository to contain change requests was created in Lotus Notes. This mainly consisted of a template form for submission of new change requests, and a calendar view showing when the changes were to be implemented. Besides the basic fields of description, date/time of implementation, duration, scope, impacted customers, type of impact, implementor, implementor contact info, and manager, additional data was eventually found to be consistently asked for during the review step. So the form was updated to also include the change backout procedure and whether business continuity plans had been updated.

Everyone in the IT department had access to submit change requests and to view the calendar.

It might be worth noting that software development teams in this organization did not use software version tracking tools or well-defined requirements tracking procedures. Thus, it was up to developers to manually fill out change request forms.

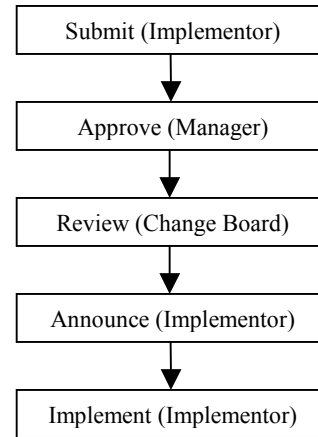


Figure 1: Initial Change Management Base Process Flow

3.2 Approval

When a change request was entered, an email notified the manager, who then needed to mark the request “approved” for it to show up on the calendar. Some managers were not used to tracking all the changes their teams made.

3.3 Review

A change review meeting was held on a weekly basis with representatives from all IT support groups: system administrators, database administrators, application development teams, network administrators, help desk analysts, etc. At each meeting, all upcoming changes were examined to determine who and what could be impacted should the change succeed or fail. Often, clarification of the initial change documentation was required (which resulted in the request form updates mentioned above). Participation was initially a problem, as people objected to yet another meeting keeping them from “real work”, until management announced that attendance was mandatory. Then the group began to discover and adjust for conflicts in scheduling, such as during fiscal year end, when a freeze was put on any changes with even a remote possibility of affecting financial processing.

3.4 Notification

Managers were identified as having the responsibility for advance communication of changes to customers and support personnel, though in practice that task fell to the change implementor. Application managers of smaller systems often had contact lists of all their users, but that was not the case for larger systems with hundreds of user accounts. For those systems, lists had to be constructed of primary contacts and power users who were then responsible for forwarding information to the rest of the user base. Notifying personnel within the IT department was also complicated. Though there was hope that everyone would consult the change calendar, targeted proactive notification was necessary and many groups discovered they didn’t know who would be affected downstream. For example, if a database server needed an operating system upgrade, the system administrators had to contact the database team who had to contact the application managers who might know which users would be affected during the proposed timeframe.

Other groups which had to be considered for notifications included IT upper management, vendors, and business area management who did not necessarily have system accounts. For major changes it was worth the effort to identify in advance the staff who would be on call for each IT support group during the change timeframe.

For some routine minor changes, email lists were put in place which people could subscribe to or not as they wished. These were not too effective, as a majority of the change announcements posted did not directly affect an individual list subscriber, so the entire list tended to be ignored. Targeted announcements sent directly and only to those individuals affected by changes became the preferred method.

3.5 Implementation

It was difficult to get people to change their habits to plan ahead far enough to submit change requests in enough time for them to be approved and reviewed. Changes were often implemented regardless of the status of the request. Sharing the outcome of a change, even those completed successfully within the timeframe allocated, was also a new procedure to instill.

4. PROCESS ROLLOUT

The moderate and major changes were targeted first, leaving the minor changes to be added into the process at a later time. The process improvement team wanted to address the larger issues first, resulting in a bigger impact on service levels.

A gradual implementation of the change management process across teams was chosen over a quick transition, to iron out rough spots in the new process before everyone adopted it. This helped identify some additional information that needed to be collected on initial request forms, and made it easier to roll out access to a common calendar tool. But overall this may have lead to more confusion than it saved, as people were not sure which groups were using the new process, and when they should start using it themselves.

5. LESSONS LEARNED

It didn't take long to discover some improvements were necessary to get even this basic process to work. The Change Review meeting was not an effective forum to review changes. There were too many participants for everyone to understand the implications of the requested changes, and the initial change request documentation was not sufficient to answer everyone's questions. A Core Change Team was defined, which met the day before the Change Review for a pre-review to ensure that documentation was sufficient and prerequisites had all been met. They then set the agenda for the full Change Review.

Another improvement to the original process was the addition of a post-implementation review step. Managers would update the change request form indicating a successful or unsuccessful change implementation. For changes that resulted in incidents (yes, they continued to occur), a formal post-mortem analysis became required.

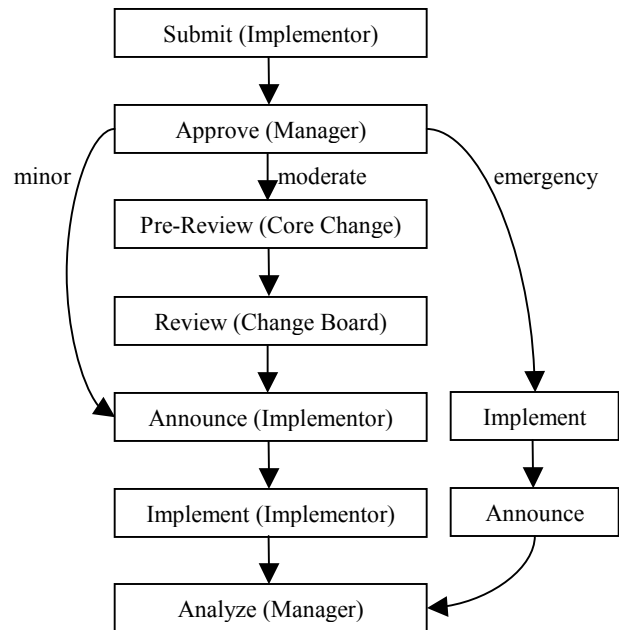


Figure 2: Revised Change Management Process Flow

6. CONCLUSION AND NEXT STEPS

All this work only brought this organization to a state I call "informed anarchy". Information about upcoming changes was beginning to be collected in a centralized location, available for anyone to go look up, but proactive communication was still occasionally lacking. Incidents still occurred with root causes which traced back to changes that affected something unanticipated, which could have been prevented if the right people knew about the change ahead of time. The process was still being significantly adjusted as it was being rolled out to more groups. Adherence to following the process was not always enforced.

As use of the change management process became more consistent throughout the IT department, the next aspect to be addressed was formalizing measurements and reports. Additional challenges included expanding the use of the process to include minor changes and also periodic, regularly scheduled maintenance. I consider that moving on to Step Three.

Further refinements down the road that the organization looked forward to included migrating the Lotus Notes change repository to a workflow application that would route requests to appropriate approvers and automatically distribute notifications to affected parties. That second part depended upon the creation of a database identifying interdependencies of hardware, software, applications, support personnel, and user groups. But that is getting into Step Four of Mastering IT Change Management.

7. REFERENCES

- [1] Carnegie Mellon University Software Engineering Institute. *The Capability Maturity Model*. Addison-Wesley, 1998.