MIT Sloan School of Management

MIT Sloan Working Paper 4637-07
January 2007

A Coordination-Theoretic Approach to Understanding Process Differences

Mark Klein, Steven Poltrock and Mark Handel

A Coordination-Theoretic Approach to Understanding Process Differences

Mark Klein
Massachusetts Institute of Technology
Cambridge, MA 02139
617-253-6796
m_klein@mit.edu

Steven Poltrock and Mark Handel
The Boeing Company
Seattle, WA 98124
425-373-{2729; 2883}
{steven.poltrock; mark.j.handel}@boeing.com

Abstract—*Supporting human collaboration is challenging partly because of variability in how people work. Even within a single organization, there can be many variants of processes which have the same purpose. When distinct organizations must work together, the differences can be especially large, baffling and disruptive. Coordination theory provides a method and vocabulary for modeling complex collaborative activities in a way that makes both the similarities and differences between them more visible. We illustrate this, in this paper, by analyzing three engineering change management processes and demonstrating how our method compactly highlights the substantial commonalities and precise differences between what are on first glance are extremely divergent approaches.*

1. INTRODUCTION

Multiple organizations that must work together (e.g. corporations working on a product, military organizations working on a joint operation, non-profits working on an aid mission) often find themselves hindered by the large apparent differences in the processes they use to accomplish what may be very similar goals. A method for identifying the root causes of their differences may help such coalitions eliminate, reconcile, or at least understand how and why their processes differ, and lead to more effective joint efforts.

This paper presents a top-down approach for achieving this goal, based on the insight that much collaborative work involves activities for coordinating inter-dependent "core" tasks. Different groups, even if they have the same core tasks, may chose different ways for coordinating them. These choices can yield work processes that appear widely divergent. Our approach is based on making these shared core tasks and differing coordination choices readily visible.

In the following sections we introduce coordination theory and a method, based thereon, for modeling processes. We apply this method to *change management*, a complex collaborative process frequently performed in engineering organizations. Although the basic change management process is relatively constant, there is great variation in its implementation in different contexts. We compare three different change management processes, identifying their differences in terms of the coordination mechanisms they invoke.

2. COORDINATION THEORY

Coordination theory [4, 5, 6] is the general body of theory about how people or software agents coordinate their activities, and it has been the subject of research in both computer science where the focus is on coordinating software agents, and the social sciences, where the focus is on describing how people coordinate.

A key concept in coordination theory is that collaboration occurs in order to manage the dependencies between tasks. A *flow* dependency exists when one person creates a product required by another person. A *sharing* dependency exists when a task requires a shared resource such as the labor of people who are involved in other tasks. A *fit* dependency exists when two or more people create products that must integrate. There are, of course, many ways to manage each type of dependency. People communicate, share information with one another, and use collaboration technologies in order to manage these dependencies. Variation in complex activities is due largely to different choices regarding how to manage these dependencies.

Malone and his colleagues [2,3,5] have developed a top-down approach to modeling complex activities. In this approach, one defines a process by identifying the core tasks and key dependencies in that process, and then selecting the coordination mechanisms that will be used to manage each dependency. These mechanisms may introduce new dependencies and exceptions that will in turn require additional mechanisms and handlers. This decomposition can continue to any desired level of detail. A key element in this approach is a large taxonomically-organized repository, known as the Process Handbook,

which captures the substeps of these mechanisms, the exceptions commonly encountered with each mechanism, as well as handlers for resolving these exceptions. These mechanisms represent, as we shall see, high-level building blocks for creating models of collaborative processes.

3. CHANGE MANAGEMENT

Change management is a key process in engineering organizations. A large aerospace program, for example, may have hundreds of change management processes that govern changes to software applications, plans, requirements, costs, schedules, configurations, and any other attribute of importance to the program. There is widespread agreement, depicted in Figure 1, about the basic change management process, though details vary widely from one instance of the process to another. In brief, a change is proposed and, if authorized, it is then implemented.

Figure 1 - Basic change management process

A Coordination Theoretic Model for Change Processes

In Figure 2 we illustrate the first steps of developing a model of the change process using coordination theory. We first must identify the "deep structure" for the process, i.e. the core tasks and key dependencies. The change process consists of three core tasks (propose changes, authorize changes, and implement changes) as well as two key dependencies (a change request (CR) flows from the first task to the second, and an authorizing change notice (CN) flows from the second task to the third). Next, we define a coordination mechanism for the first flow dependency. Any flow is managed by some variation of the generic "manage flow" building block in the Handbook repository. This template captures the fact that managing a flow always involves managing the timing, usability, and location of the resource that is flowing. Each of these subtasks, furthermore, have their own characteristic exceptions (the manage usability step has, for example, the exception "flow wrong thing"), and each of these exceptions has a range of processes (not shown) for handling them.

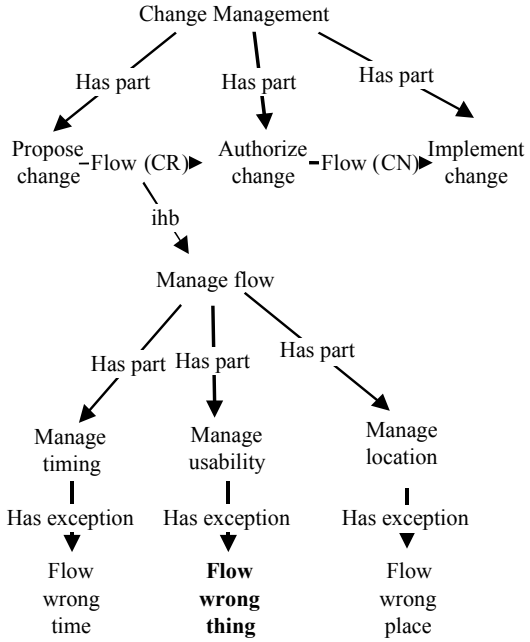

Figure 2 - Top level of the change process

Avoiding Inappropriate Changes

The key challenge in change management is to avoid implementing the wrong change request or, in other words, to avoid the "flow wrong thing" exception shown in Figure 2. Table 1 below lists some of the mechanisms in the Handbook repository suited for handling this exception.

Table 1 - Handlers for "flow wrong thing"

Avoided by	Filter out unwanted elements (by individual or team judgment)
Resolved by	Filter out bad agents
Detected by	Monitor agents for commitment violations
Anticipated by	Track reputation information
Avoided by	Provide incentives

Most engineering organizations use variants of the "filter out unwanted elements" handler. One variant (filter by individual) involves asking an individual to reject the change requests that in his or her judgment are not viable, while the second involves asking a team to serve this role. The Handbook repository includes tradeoff tables that describe the relative strengths and weaknesses of alternative processes for fulfilling a given function. The tradeoffs for the two variants of "filter out unwanted elements" are shown in Table 2.

Table 2 - Tradeoffs for specializations of filtering out unwanted elements

Alternative	**Best for**	**Cost**	**Quality**	**Speed**
Filter by individual	Initial pruning of easy-to-find problems, such as missing data	Low	Low	Fast
Filter by team	Careful evaluation of resource from multiple perspectives	High	High	Slow

Filtering by individual is fast and cheap but of low quality, whereas filtering by teams is slow and more expensive but higher in quality. In many engineering organizations, filter-by-individual is used as an initial screening step, followed by filter-by-team. The generic "filter by team" handler in the Handbook repository consists of several steps (Figure 3):

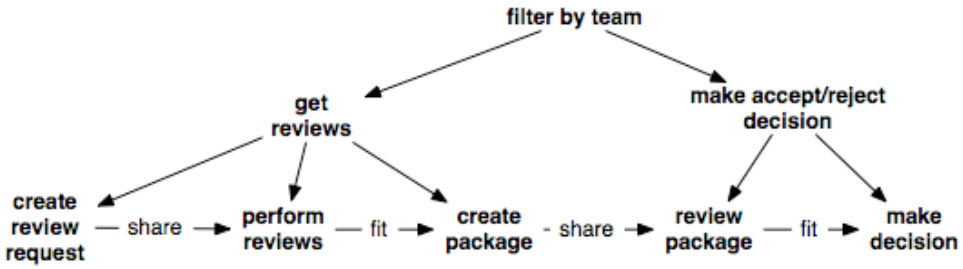

Figure 3.-The "filter by team" exception handler.

The first is to get reviews or assessments of the impact of a CR, and this requires creating a review request that is sent to all reviewers, performing the reviews, and then consolidating the reviews into a coherent package. The filter-by-team handler also includes making an accept/reject decision, which requires first reviewing the completed package and then making a decision. Note that there are dependencies between these lowest level parts, and each dependency in turn requires a coordination mechanism. Figures 2 and 3, taken together, represent a coordination-theoretic model of the change management process used by many engineering organizations.

4. PROCESS VARIATION

When activities are collaborative, the way they are performed varies greatly from organization to organization, from one team of participants to another, and from one time to another [1]. This is certainly true for change management. We investigated whether such differences can be explained as a consequence of selecting different coordination mechanisms and/or different exception handling mechanisms and, if so, whether this perspective on process variation can help us understand and, if necessary, eliminate it.

To do so, we compared three change management processes used within one aerospace program: one for managing change to cost and schedule (BMW), another for managing change to product configuration (CCP), and a third for managing change to processes and tools (SIP). These are complex processes involving many people in varying roles. Applications that automate the flow of work have been implemented for all three processes, and a portion of the BMW workflow management process model is depicted in Figure 4 (below) to illustrate their complexity. The complexity of these processes prevents us from presenting a complete coordination theory model; the simplest model (BMW) includes 48 steps. Even though the processes all have essentially the same goal, the way they were modeled by their users and managers were widely divergent, the workflow applications that support them are different, and commonalities are far from obvious.

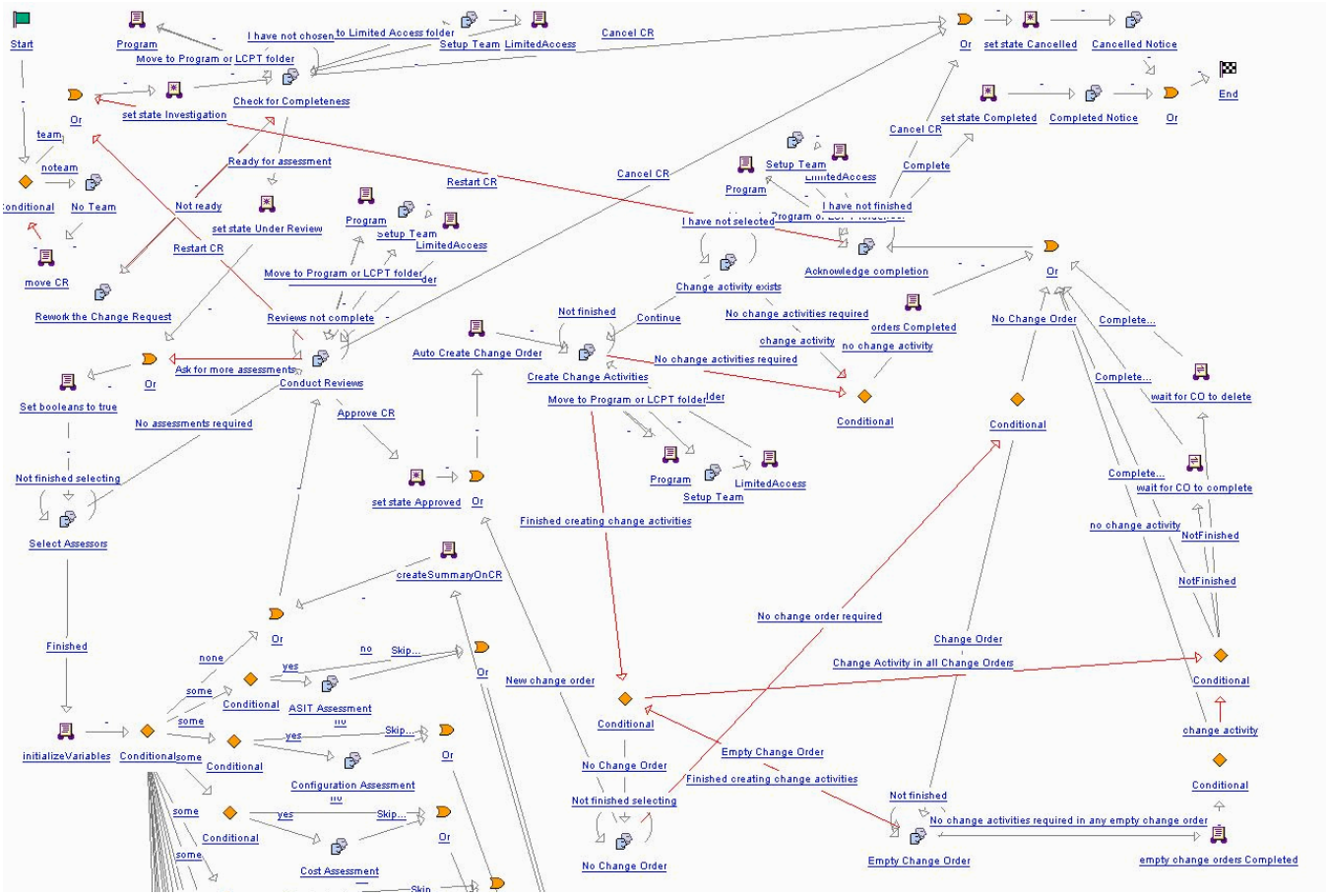

Figure 4 – A portion of the workflow model that implements the BMW process

Our first key finding was that most of the steps in these processes involved coordination. Of the 48 tasks in BMW, for example, 41 are coordination mechanisms (e.g. sending change requests to the reviewers, collecting and consolidating the reviews, distributing them, holding a review meeting, and notifying the requestor about the outcome) or exception handlers (e.g. filtering CRs and handling review requests that are sent to the wrong person or not returned on time).

A second key finding is that the differences between these processes concerned how they perform coordination and exception handling. In order to make these differences readily visible, we created a process representation we call a "derivation tree". This tree captures the refinements (i.e. additions of coordination mechanisms or exception handlers used to create a process model using the top-down analysis introduced above. The derivation tree for the generic change management process described above, for example, is presented in Figure 5.

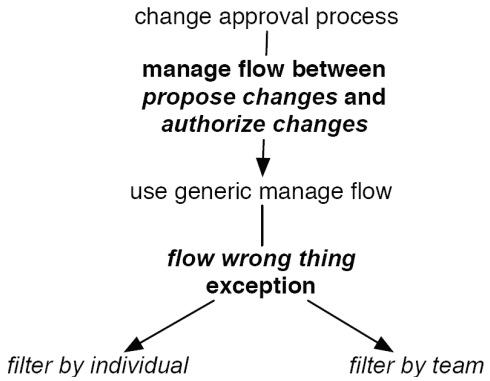

Figure 5.-Derivation tree for the generic change management process model.

Each arrow describes (in bold text) what aspect (dependency or exception) of the model is to be refined. The target of each arrow captures the coordination or exception handling process selected for this purpose. These processes can in turn have dependencies and exceptions that need to be refined. A derivation tree is generally quite compact, because a refinement often represents the addition of a relatively large building block (i.e. a coordination mechanism or exception handler) from the Handbook repository. The derivation tree for the BMW change process, for example, consists of 11 refinement operations, while the conventional flowchart model for this process includes 48 steps.

Derivation trees can be used to highlight the similarities and differences between related processes. The trick is to consolidate, into a single tree, the derivation trees for the processes being compared. Differences between the processes become immediately evident as alternative refinements for a given dependency or exception. Figure 6 shows a consolidated derivation tree for the BMW, CCP, and SIP change processes. Much of the tree, we can see, is shared by all three processes. All start with the generic change approval process, and use individual and team reviews to avoid authorizing the wrong change requests.

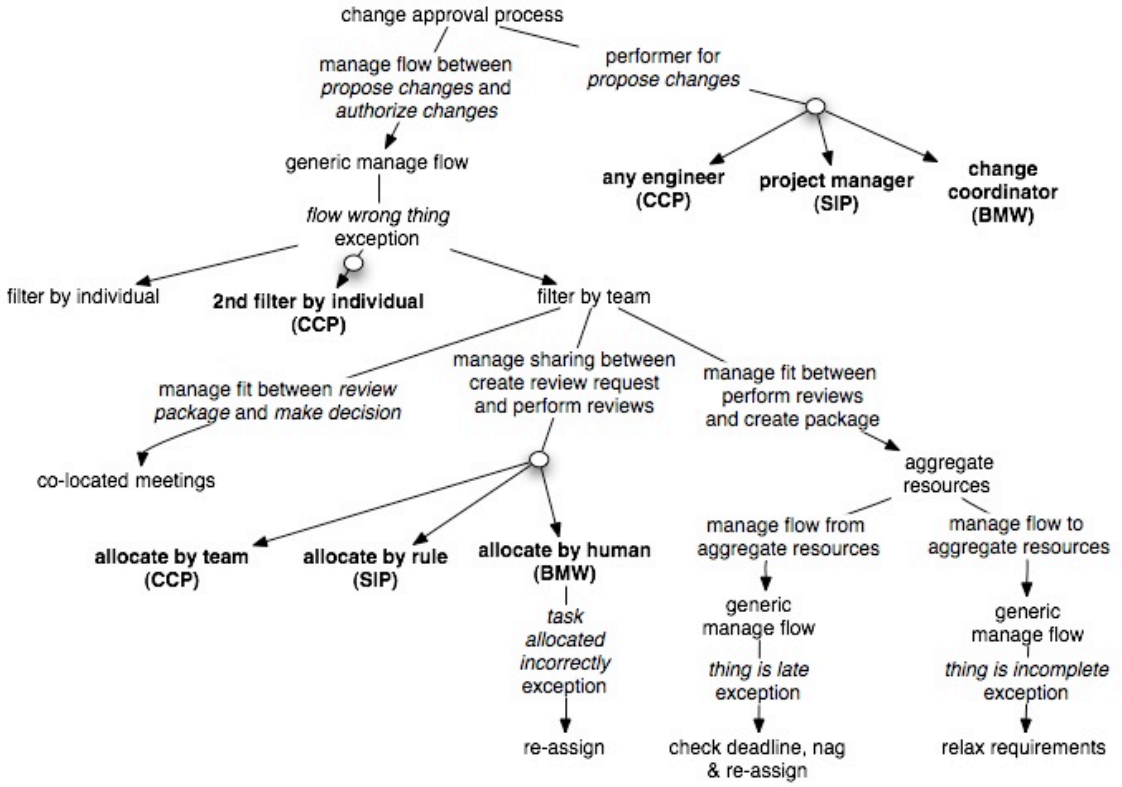

Figure 6 - Consolidated derivation tree comparing the BMW, CCP and SIP engineering change processes.

The small ovals represents the points of differences between the three processes, with the alternatives presented in bold font. One difference concerns *who* can generate change requests. In the CCP process, any engineer can request a change. In the BMW process, proposed changes must be submitted by engineers via a change coordinator, so the coordinator can filter/revise inappropriate change requests before they enter the change process. In SIP, change requests must be submitted via the project manager. A second difference is that the CCP process includes a second filter-by-individual step, performed by the requestor's manager. This additional step avoids the cost of a team review for CRs that are unlikely to be authorized. The processes also differ, finally, in how people are assigned to review change requests (i.e. in how we refine the sharing dependency between "create review request" and "perform reviews"). A fixed set of reviewers are expected to review every CR in the SIP process, which corresponds to an allocate-via-rule mechanism. The change coordinator in the BMW process decides who reviews CRs, which corresponds to an allocate-via-human-judgment mechanism. For the CCP process there are thousands of potential reviewers, and no one individual can be expected to have the knowledge required to determine who should review each CR. Instead, they use an 'allocate by team' mechanism, wherein members of a team make suggestions concerning how to allocate each resource (change request), and these recommendations are consolidated somehow (e.g. concatenated) to produce the final list of recipients (change request reviewers).

5. CONCLUSION

In this research we studied just three of hundreds of change management processes within a single aerospace program. Although all three processes are intended to accomplish the same thing, they use different tools and procedures, and the documentation that guides participants in these three processes looks radically different. This makes it difficult to understand and, if desired, eliminate differences between process variants. By applying top-down coordination-theoretic modeling, supported by a Handbook of generic coordination mechanisms and exceptions handlers, we were able to create derivation trees for all three processes that made the source of their similarities and differences much easier to identify. With this information in hand, we can then begin asking *why* the processes differ in these ways, and whether/how we want to change them in order to make the processes more effective and more consistent.

REFERENCES

[1] Dourish, P. *Where the action is: The foundations of embodied interaction.* MIT Press, 2001.

[2] Klein, M. and Dellarocas, C. Designing robust business processes. *Organizing business knowledge: The MIT process handbook.* The MIT Press, 2003, 423-439.

[3] Klein, M. and Petti, C. A handbook-based methodology for redesigning business processes. *Knowledge and Process Management, 13,* 2006, 108-119.

[4] Malone, T.W. and Crowston, K. What is coordination theory and how can it help design cooperative work systems? *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'90),* Los Angeles, October 7-10, 1990, pp. 357-370.

[5] Malone, T.W., Crowston, K. and Herman, G.A. *Organizing business knowledge: The MIT process handbook.* The MIT Press, 2003.

[6] Malyankar, R.M. and Findler, N.V. A methodology for modeling coordination in intelligent agent societies. *Computational and Mathematical Organization Theory, 4,* 1998, 317-345.

ACKNOWLEDGMENTS

MIT's contribution to this work was funded by The Boeing Company, award MIT-BA-GTA-1.

Research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.