Barriers to Effective Process Architecture — An Experience Report

Ashok Dandekar
Fujitsu Network Transmission System
Transmission Development Division
Richardson TX 75082
{avdandektddcae99.fnts.com}

Dewayne E. Perry
Systems and Software Research Laboratory
Bell Laboratories
Murray Hill, NJ 07974
{depbell-labs.com}

Abstract

In trying to understand the architecture of the processes governing the development of a large software product, we used various techniques for describing, analyzing and visualizing that process system. A "big picture" visualization of the software development processes yielded a number of cogent observations: I/O mismatches, large fan ins/outs, no clear path through the project, inconsistency in the level of detail, no clear architectural organizational structure. We report the results of a quality improvement team (QIT) put together 1) to determine how the process architecture got to this state, 2) to delineate the base measures by which we plan to measure architectural improvement, 3) to establish surface and root causes for the current state of the architecture and define their interrelationships, and 4) to derive primary and secondary process architecture drivers and to establish counter measures that will yield a more coherent and appropriate process architecture. As a result of these studies, we offer some principles learned about process architectures and generic processes.

Keywords: process system architecture, generic process principles, process architecture principles, process system architecture problems

1 Introduction

Software development organizations are moving away from software development as a purely craft approach towards software development as an engineering approach. In doing so, we move from implicit processes and mystical rites passed from master to apprentice to defined and written processes amenable to scrutiny and measured improvement by the software engineering community.

The current state of practice is generally a set of defined processes informally described, often created as a result of a management edict. Eventually a process management group is created to manage the emerging ad hoc process system. The barriers we found to an effective process system architecture are typical of those one finds in similar software development organizations.

Process system architecture is a very young field and in the main a very immature one, poorly understood with little experience and little understanding. Our attempts to create a coherent process system architecture in many cases push the state of the art and in some cases push the state of research as well.

We describe the current state of the process architecture and delineate the base measures by which we measure architectural improvement. We then discuss our analysis of surface and root causes and define their interrelationships. From this analysis we present the primary and secondary process architecture drivers and report the counter measures that we have established and discuss the progress that we have made on these countermeasures to establish a more coherent and appropriate process architecture. Finally, we summarize some principles about generic processes and process architecture that we have learned from our study.

2 Current State and Base Measures

Our current process architecture is implicit in the set of (more or less) independently defined processes that govern the production of a large, real-time system from initial marketing interactions with customers through software development and verification through to final customer support. These processes are defined in a set of on-line manuals in which the processes are described informally in a highly structured format.

In trying to understand the architecture of the processes governing the development of a large software product, we used various techniques for describing, analyzing and visualizing that process system [Perry 1994] [Carr, Dandekar and Perry 1995]. The "big picture" visualization of the software development processes yielded a number of cogent observations: I/O mismatches, large fan ins/outs, no clear path through the project, inconsistency in the level of detail, and no clear architectural organizational structure. A quality improvement team (QIT) was put together to address the lack of a coherent process system architecture — an architecture with components that do not fit together, components that are unnecessary, components that either duplicate or overlap each other, and a structure that is not well controlled.

We first drafted a problem statement to summarize the current state of the process architecture:

> The existing process architecture is complicated, incomplete, and contains redundancy, thus making interactions between processes difficult to understand and control. This limits the potential for reductions in cost and interval.

Further details of the current state of our process system architecture are found in the discussions below.

Given the problem statement, we then defined a set of metrics to be used as the basis from which to measure any improvement achieved by our projected countermeasures. These base measures are those provided by `pichk` [Carr, Dandekar and Perry 1995], a suite of tool fragments that analyze the process interface descriptions and their interrelationships. The kinds of analyses and summaries produced include data on the average interface structure of the processes, the consistency of the interface specifications, the interface errors found, and data on the production and use of the process artifacts.

3 Root Causes and their Effects

The quality improvement process (QIP)[1] requires that we first determine the underlying root causes to explain why the process system architecture is in the state it is in. Given that set of root causes, we then prioritize them with the aid of interrelationship diagrams (IRDs) and determine their interdependencies. The root causes in their prioritized order are

- no global management of the process system,
- no coherent process system structure,
- no support for process execution,
- no standard definitions and terminology,
- process inflexibility,
- inappropriate process documentation,
- inadequate customer-added-value focus, and
- processes represent organizational structure.

We will consider each of these root cause and discuss their effects on the process system architecture.

3.1 No global management of the process system

The initial result of this root cause is that the process architecture came into being in an uncontrolled way. It was not planned and no one laid out what the process architecture ought to be. Management decided (rightly) that processes were important and ought to be documented. What was documented was what was going on at the time of the documentation, not by a process engineering staff allocated for the job, but by the current staff as an additional task. Ameliorating this effect is the fact that process system architecture as a whole is poorly understood, even in research: it is an immature field with little extant experience and understanding.

As a result, processes developed independently created by teams acting in isolation. This localization was encouraged by two factors: there was no overall strategy or general goal, and organizational boundaries encouraged such isolation. Moreover, this localization spreads to cost reductions, optimization of the processes, the definitions of process artifacts, and the management of process tasks (that is, process control).

An inevitable result of the lack of global management is that the process system only expands, it does not contract. Process teams are created but are not destroyed, nor do their processes die. There is no incentive to reduce the bureaucracy, only to expand it.

Because of this balkanization, there is no management of the interfaces and no global considerations of process artifacts that are the means of communication and coordination between processes. In fact, there is no formal

[1] This is the standard quality process used throughout this development organization.

view of what a process interface is. The process management teams (PMTs) develop and act locally and create their own interfaces from a local view.

With no control of the "big picture", it is not surprising that there is no planned evolution. Process changes are made because they are due, generally as a result of low priority, and tend not to be well-planned and well-considered. Moreover, there are no process metrics to serve as the basis for improvement.

An important effect is that the fidelity of execution to the defined processes is not enforced. There is no means of enforcement and no priority provided to guide the process executors when conflicts arise.

Compounding all these problems are two further factors. First, the current architecture is the merger (in many cases, the union) of the process systems of two organizations who build similar products. Second, the current system is in a state of dynamic change in which interval, cost and quality factors are changing to respond to market demands.

3.2 No coherent process system structure

Since the process system grew in an an hoc manner, it is not surprising that there is no coherent process system structure. Nor is it surprising that the entire structure is flat — that is, there is no hierarchy to the process system. Similarly there is no difference in the levels of detail, even though it is quite clear that a single level does not meet all the varied needs in process support. Yet despite this flatness, the various process descriptions are written at different levels of abstraction.

Consistent with the flatness of the system is the fact that all process executors and all process customers are consider to be equal — that is, there is no sense of priority or ranking of importance. There is neither a distinction between essential and supporting tasks, nor any difference in the ways process artifacts are treated or used.

Another result of the ad hoc growth is the duplication of process tasks: just as there is duplication in large software systems, so is there duplication in large process systems. In some cases this duplication results from lack of proper decomposition; in other cases it is merely the result of inattention to what else has been done.

Finally, there is no end to end traceability of either the processes executed or the artifacts produced.

3.3 No support for process execution

The effects of this root cause appear in three different aspects of process: lack of automated support, lack of appropriate descriptions, and lack of buy-in by those who are supposed to execute the processes.

Process descriptions are weak in defining who is responsible for doing what — that is, roles are not associated with particular aspects of the processes. Compounding this problem is the fact that in many cases, the process descriptions do not reflect the actual process work.

It is often the case that process are viewed as intrusive rather than supportive, something that hinders rather than augments the effectiveness of the developers. Moreover, it is often the case that the most productive people do not follow the written processes, either because they are inappropriate or because they have more effective processes.

3.4 No standard definitions and terminology

Given the informal nature of the process definitions, the number of processes and the size of the organization, it is not surprising that there is no standard set of definitions or terminology. Each process defined its processes in terms of its local vocabulary — for example, there might be five different ways of referring to a single artifact. While there is a certain amount of commonality centered around the primary product and the core processes that produce it, those processes are a relatively small part of the entire process system. Beyond that basic core, there are many disagreements about the definitions of the process artifacts that are the inputs to and outputs from the various processes.

The most profound effect of this lack of standard definitions and terminology is the lack of a well-defined distinction between what is a task, a subprocess, and a process. We have no underlying model that provides a rationale for distinguishing one from the other. Consequently, we have no crisp way of distinguishing between procedures and guidelines either. Nor do we have either a crisp definition of what a role is or a description of them.

The fact that we do not have agreed upon definitions of processes implies that some of the process management teams (PMTs) are not really PMTs, but links to other groups or links to tasks or procedures.

3.5 Process inflexibility

There are a number of ways in which process descriptions are inflexible or not adaptable. First is the problem of scale. The factor of scale affects both the relationships between processes and they way they interact. We try to make 'one size fits all' both in the way processes relate to each other and they way in which they interact. This is a much too simplistic approach. We need a rich set of interactions and relationships as part of our arsenal to successfully attack the problems of scale. Second, we are unable to deal with new technology easily. This is related to the lack of abstraction levels in that we bind tools into our processes rather than abstract them to a lower level. Third, process descriptions incorporate project plans and are not easily adaptable to different projects or to projects of different sizes. Finally, processes are often too complicated to execute flawlessly and as a result are also too complicated to be adapted easily.

3.6 Inappropriate Process documentation

It is often the case that informally descriptions tend to be overly verbose. Our process descriptions are not exceptions this tendency. The template we use encourages verboseness. Further, this verbosity is the result of several factors. First, the process descriptions tend to be too detailed in that they define what to do as well as how to do it. This is a particularly thorny research issue: the balance between what to do and how to do it. Second, processes tend to be all at one level, the lowest one possible, rather than structured in levels of abstraction. Both of these factors are compounded by the fact that process descriptions tend to describe how to write a document.

In addition to being too detailed and wordy, they also tend to contain more than is necessary and do not succeed in separating concerns properly. For example, process inputs and outputs tend to reflect benchmarks in a project rather than just I/O in a process — that is, project management plans are embedded into the process descriptions and they should be kept separate. Another aspect of this is that production and management tasks are often mixed together in the same process rather than being separated into their appropriate spheres of concern.

3.7 Inadequate customer-added-value focus

Harrington [Harrington 1991], in considering the problem of simplifying processes, advocates characterizing the various parts of processes as adding either customer value, business value, or no value. Parts that add no value should be eliminated; parts that add business value should be scrutinized to make sure they are necessary; and parts that add customer value should be emphasized.

One effect of this lack of customer focus is that processes are often developed without knowledge or understanding of customer requirements, and without an awareness of business and market imperatives. Most processes add at least business value (that is, are necessary to maintain a product, etc.) but there is still insufficient focus on customer value added activities.

3.8 Processes represent organizational structure

Processes are (often) aligned with existing organizational boundaries partly because that is the easiest way of doing it. The organizational boundaries were already in place when the processes were defined and, in fact, provided the implicit process structure before we had formal process documents. Moreover, budgets are distributed over organizational boundaries, not process boundaries.

It is not surprising then that process teams reflect organizational structure as well and not the actual work that needs to be done. Given this initial seeding of the process teams along organizational lines, there also has not been much effort or incentive to extend the teams beyond these organizational lines. Furthermore, there is not much incentive to belong to a process team outside your own organization.

Of course, there is the age-old problem of empire building that contributes to this problem. It is the organization that controls what people, and this control is implicitly part of their process.

As with most software development organizations, the organizational structure is a product organization. Strengthening this tie is the fact that products tend to be tied to particular markets. Still, the utility of this kind of organizational basis has not been questioned and needs to be addressed in the context of both process and product structures.

As with the root cause of no global management, the process system merge complicates the effects of this root cause. While some processes were in fact merged, many were not. This lack of merging was due to the fact that the corresponding similar organizational parts did not merge.

4 Process Architecture Drivers and Countermeasures

Using interrelationship diagrams derived from the root causes, we determined that there were three primary architectural drivers and two secondary ones. The primary drivers are

- processes represent organizational structure,
- no global management of the process system, and
- no standard definitions and terminology.

The secondary drivers are

- no coherent process system structure and
- inappropriate process documentation.

The first primary driver is something that is exceedingly difficult to do anything about at the level of the process management team, except to be aware of the impact of organizational structure on processes and process structure. Ideally, one would like to see a process driven organizational structure — an approach that is extremely unlikely to happen since an organizational structure is driven from the top levels of management downward and the process structure tends to be driven from the bottom levels upward.

The other two primary drivers and the secondary drivers are within the realm of the process management team's control. It is in the context of these remaining drivers that we propose eleven countermeasures that we considered to be effective in combating the fundamental problems that underlie our lack of a coherent process system architecture.

1. Define and control process interfaces
2. Model and define the overall process system structure
3. Introduce life-cycle management for processes
4. Continuously monitor and evaluate the process management teams
5. Improve documentation of processes
6. Relate process to product
7. Reflect the work done
8. Have short and long term goals for process improvement
9. Focus on process in addition to product
10. Develop goals, staff, and empowerment
11. Reward process work

Countermeasures 1 through 4 are aimed at countering the lack of global management of the process system and the lack of a coherent system structure by establishing four important global strategies. These countermeasures will be the responsibility of a Global Process Management Team — a level of process management that has yet to be created, but which will have global authority over the process system. The definition of process interfaces and the overall process architecture is central to the global control of the process system, while life-cycle management, in the context of continuous monitoring and evaluation, is central to the evolution of both the process system and the individual processes. Moreover this team is responsible for standardizing the definitions and terminology as a necessary precondition to the control of process interfaces.

Countermeasures 5 through 7 are aimed at countering the lack of appropriate process documentation. Getting the documentation of the processes to be at the right level of abstraction and detail is fundamental to this goal. In achieving this desired level, it is important that the processes are directly correlated to the product which the processes govern — that is, the processes should be product-centered since its production is the entire *raison d'etre* for the processes. While there is a fine line between the processes as prescriptions rather than descriptions, it is important that the processes reflect the work that must be done. This implies that there are two factors at work: first, the processes are changed when they do not reflect necessary work; second, people are changed (that is, they are encouraged, guided, managed, etc) when they do not do the necessary work. These countermeasures are carried out by the existing individual process management teams.

The last set of countermeasures (8-11) are aimed at providing an appropriate cultural setting for process work and improvement. Rewarding process work, with the same reward structure as exists for work on the software system being developed and evolved, is fundamental to an effective culture for process work. Having an appropriate set of both short and long term goals, sufficient staff together with the empowerment to implement them, is also necessary for an effective supporting culture. These countermeasures are the responsibility of the leadership team and are fundamental in achieving success in the other countermeasures.

5 Progress

We are systematically working towards a coherent process system architecture. We have created a hierarchical process structure by creating domain-specific clusters of processes as the top level in the hierarchy. The general goal for each of the clusters is to simplify both the internal structure (that is, the interfaces and interrelationships between the processes within the cluster) and the external structure (that is, the inter-cluster interfaces). As part of that effort we have been experimenting with various ways of doing process streamlining and simplification. Concurrently with these efforts, we are creating a new form of presentation and representation of the process descriptions.

6 Some Principles

As a result of our work in trying to understand process system architecture, we have distilled a small set of principles for process descriptions and for process system architectures.

Given the fact that the current process system was the result of the merging of two process systems and the fact that there is a trend within most companies for the consolidation of product lines, moving from individual products to product families with interchangeable parts, we have concentrated on the problems of generic processes. We offer the following principles for achieving what we consider to be the appropriate level of descriptions for generic processes.

1. *Define process fragments.* Instead of monolithic process descriptions, use process fragments that can be combined in various ways according to the needs of the specific project.

2. *Define fragments in terms of their goals.* Rather that describe the implementation of a particular process fragment, define what the fragment is meant to achieve. The person executing the process can supply the appropriate implementation.

3. *Use appropriate means of abstraction (or generalization).* One of the standards forms of abstraction is that of parameterization: abstracting values, types, objects, functions, etc. There are several other forms of abstraction that are also useful: primitivation and stratification. Primitivation is that form of abstraction that requires elaboration before execution: the actual implementation is left to the process executor and is bounded either by a formal grammar or by the existence of the available building blocks. Stratification is that form of abstraction by which we layer a system. In this case we recommend abstracting the methods and tools layers from the generic descriptions.

4. *Align activities with their appropriate processes.* All too often, you find project management activities described as part of technical processes. For example, estimating coding effort is usually a part of the description of the coding process. Note, however, it is a project management activity and should be part of the project management process description. Moreover, it is probably a generic activity that could be parameterized for the appropriate development phase.

5. *Separate project structure from process structure .* Nothing makes a process less applicable to another project that to embed project specific data into the process descriptions. We note that there are three project related aspects that are often included in process descriptions:

   - project milestones and schedules,
   - project roles, obligations and permissions, and
   - the project's organizational structure.

Based on our work here together with our work in process system visualization and analysis [Carr, Dandekar and Perry 1995], we offer the following principles for process system architecture. Not surprisingly, these principles are akin to those useful for software product systems as well.

1. *Modularize processes.* In much the same way we modularize software systems, we should modularize process systems. Only by segmentation into manageable parts can we hope to cope with the entire system.

2. *Encapsulate domain-related activities.* Rather than just arbitrarily cutting up the processes into pieces, we need a rationale for the segmenting of the process system. How finely you divide the domains is a design decision but one that should be guided by comprehensibility.

3. *Decompose processes hierarchically.* As with software systems, we decompose our process systems into layers of increasing detail, moving from the more general to the more particular.

4. *Explicitly define the relationships among processes.* It is in this principle, that we differ most from the principles needed for software systems. Where there, we have only a limited number of formally defined relationships that are supported by our programming and system model languages, here we have an unlimited number of informally defined relationships. Making those relationships explicit is extremely important.

Acknowledgements

This work would not have been possible without the unflagging support of Al Barshefsky and the active contributions of the other members of the quality improvement team: Dave Smith, Mike Meuer, Tave Lamperez, Dave Carr and Janel Green.

References

[Carr, Dandekar and Perry 1995] David C. Carr, Ashok Dandekar, and Dewayne E. Perry. "Experiments in Process Interface Descriptions, Visualizations and Analysis", *Software Process Technology — 4th European Workshop, EWSPT'95* Wilhlem Schaefer, ed. Lecture Notes in Computer Science, 913, Springer-Verlag, 1995. pp 119 - 137.

[Perry 1994] Dewayne E. Perry. "Issues in Process Architecture", *Proceedings of the 9th International Software Process Workshop — The Role of Humans in the Process*, October 1994, Airlie VA. IEEE Computer Society Press. pp 138 - 140.

[Harrington 1991] H. James Harrington. *Business Process Improvement.* New York: McGraw Hill, 1991.