Issues in Process Architecture

Dewayne E. Perry
Software and Systems Research Center
AT&T Bell Laboratories
Murray Hill NJ 07901

Abstract

I consider the problems of process system architecture in the context of the Perry-Wolf model of software architecture: process elements are executed in process systems by both machines and people; data elements tend to be informal documents in process systems rather than formal, machine manipulatable objects; and connecting elements are much more complex in process systems, involving both automated, social and organizational structures.

1 Introduction

The fact we refer to *the* software development process and provide *a process* support environment is a measure of the (im)maturity of much of process research. We do not have *a* software development process, we have a large set of processes. We have multiple, interdependent processes in much the same way that software products have multiple, interdependent components.

Because of this component structure of our development processes, we have architectural considerations that are both similar to and different from product architectures. The similarities stem from the fact that a process architecture is made up of data, processing and connecting elements [6] in much the same way that a product architecture is. The differences stem from the facts that 1) people are the primary (essential, dominant) processing agents that manipulate the data and perform the process activities and 2) the connecting elements are often organizational and social rather than just technological.

In the discussion that follows, I will consider these multiple, interdependent processes first within their architectural context and second within their organizational and social context.

2 The Architectural Context

While software development processes are clearly a domain-specific set of business processes, the actual set of elements in this domain-specific process architecture varies widely along several dimensions. The actual set of elements in a development process architecture will vary dependent on the size of the project (1-5, 20, 200, 2000 people, etc), the type of product developed (safety critical, prototype, etc), the type of approach (entrepeneurial, waterfall, spiral, evolutionary, etc), or the level of process maturity. These dimensions partition the process architectures into different process architectural styles.

More fundamental than the determination of architectural elements is the problem of defining what an architectural element is. In building software, we have a relatively good understanding what the various component granularities are. In defining processes, however, we do not yet understand and agree upon the various granularities needed to create a process architecture and we do not yet agree on the principles of process element decomposition [2]. We have a small vocabulary to describe these components, but no well-defined meanings or well-understand sizings. For example, the term process is used to mean anything from a small activity that may or may not be automated to the entire set of activities that is done in the development of a product.

In the context of a very large development project and organization that I have been working with, the term "process" is used to denote a large set of (perhaps concurrent) activities. Some of these activities include the use of tools but many are often done by people independent of a supporting technological context. In fact, even in the most tool intensive of these activities — for example, doing a system build [7] — the human element is critical and dominant over the technological. Certainly the large-grained processes are not of the automated or even automatable type. The critical difference between process processing elements and product processing elements is that they are entities that are performed by people — that is, humans, not tools are the processing agents. They may be formally modeled [3], but much of the processing that is done by people is done informally and outside the context of technological support. Moreover, much of what is

done even in the context of technological support is actually independent of that support.

Most of the data elements in process architectures are unlike those in product architectures in a way similar to the way that process processing elements are unlike their product counterpart. These data elements are generally informal documents. At best, various instances of the same kind of data element have the same general structure (often derived from the use of document templates). While these data elements can be formally modeled [3], they are not automatedly manipulatable by process elements as they are in product architectures. Again, as with the process elements, these data elements are manipulated by people as the processing agents.

We are on firmer ground, however, in considerations of architectural form. Here, at least at one level of abstraction, there is an analogy between product architecture and process architecture. This level of abstraction is that of the basic relationships between architectural elements [1]. As in product architectures, we have process elements that have are *independent* of each other — they do not depend on each other in any way. We have process elements that are *input dependent* — they depend on another element for some form of input necessary to successful processing. We have process elements that function much the same way that subroutines do — they are *nestable* within another element. For example, the bug reporting process is performed within the context of some other process element (often coding or testing) causing the suspension of that element while it is performed and returning to the suspension point when it has been completed.

It is with elements that are both *concurrent and dependent* on each other where the analogy begins to break down. We have some concurrent dependencies that function along the lines of cooperating processes with well defined points of interactions, or even that function like co-routines. However, we also have instances of concurrent dependencies that are much more unstructured. For example, it is often the case, especially in large projects, that the design and coding processes interact in arbitrary ways — dependent on the circumstances rather than on well the defined modes of interaction that we have in inspections, etc. It is in this informal and unstructured form of interaction that we need further work on descriptive notation.

We find one other analogue relationship in product architecture — namely, that of one element *monitoring* another. The most appropriate case of the kind of relationship is that of the progress monitor in a database system. Its function is to monitor the transactions and determine when progress cannot be made for some subset of the transactions because of resource contentions. Its job is then to reallocate resources so that at least some transactions can progress. The project management process element functions analogously within the context of a process architecture. Note, however, that we do not have the analogous supporting infrastructure that we have in database architectures. Neither do we have supporting notations to describe this kind of relationship between process elements.

3 The Organizational and Social Context

The analogy between process and product architecture breaks down even further when we consider connecting elements. Where processing and data elements exhibit some automated aspects, connecting elements exhibit even less automation. Practically all of the connecting elements are organizational and social in structure — and certainly informal and non-automated.

At the fine-grain level of connecting elements are various forms of communication between people such as electronic mail, phone-mail, phone conversations and face-to-face interchanges. We found in one of our process experiments [4] that the automated forms of communications (specifically electronic mail) were used only only for broadcast messages. Technical interchanges were invariably done either in person or over the telephone.

At the large-grain level of connecting elements are various forms of meetings, document handoffs, test laboratories, etc. I will consider several such elements and indicate some interesting experiments with these connecting elements that have illustrated significant improvements in both their structure and their performance.

Probably the most common process architecture connecting element is the *document handoff* . This element is as prevalent in process architectures as the procedure call is in product architectures. Unlike a procedure call, however, it is a very *lossy* channel of communication. The resulting documents represent but a pale shadow of the knowledge gained in constructing them. This uncommunicated knowledge is needed in understanding those documents properly. Without it, the documents become a source of error injection. In one process experiment [5], the project was organized in such a way as to minimize these handoffs and thus to minimize the error insertion aspects of the element. This was done by using interdisciplinary

teams to maintain the continuity of knowledge while handing off the document from one process element to the next. In other words, the underlying processing structure of the process and connecting elements was changed organizationally to incorporate previously local data elements into the combined structure. Both the performance and quality of the resulting process architecture improved dramatically.

Another common connecting element is that of relying on management for technical decisions. This is a two-way connecting element where a request for a decision is sent up the hierarchy to the appropriate level of responsibility and eventually the response comes back down the hierarchy. The typical problem with this kind of connecting element is that it is suffers from a wide degree of variance in response time. In a another process experiment [5], the project was organized in such a way to minimize the time factor of this connecting element. This was done, again, by using interdisciplinary teams, with the added factor of decision empowerment. The time variance factors associated with the connecting element were removed. Again, both the performance and the quality of the resulting process architecture improved dramatically.

One of the problems that we have uncovered in our visualization and analysis experiments [1] is the large amount of complexity introduced into the process architecture because of the lack of appropriate connecting elements. In particular, project management requires the production of a large number of artifacts (data elements) to be produced by the managed processes for them only. This increases the number of data elements significantly and the number of connections between the project management process element and other process elements. What is needed is a monitoring capability to serve as a connector between project management and various process and data elements as well as the underlying process state.

Yet another problem arises with respect to the constraints on the process elements. We found the following classes of process elements in our analysis of our current process architecture: processes, organizations and roles. While some roles and organizations are necessary, the primary problem is that they represent undefined process elements. Clearly, we must refer to customers as part of our process architecture. However, the primary class of process element should be processes. They in turn are decomposed into various subprocesses and tasks where the actual consumption and production of the process artifacts takes place. These processes are performed by organizations and their internal task steps are performed by people in various roles. Thus the distribution of the various data elements via the connecting elements should be between processes. Organizations (as groupings of people) and roles (as performed by people) are the processing agents of the process elements, not appropriate architecture elements.

4 Summary

Process architecture differs from product architecture in a number of important ways. First, where product architecture assumes computational processing agents for the various architectural elements, process architecture has in addition two radically different type of processing agents — namely, people and organizations. Second, where product architecture assumes components that have computationally well-defined semantics, process architecture usually has components that, while they may be modeled formally, are at heart informal with loosely defined semantics and unenforceable execution.

References

[1] David C. Carr, Ashok Dandekar and Dewayne E. Perry. "Experiments in Process Description, Visualization and Analysis", *European Workshop on Software Process Technology 1995* , Leiden, The Netherlands, April 1995.

[2] Ashok Dandekar and Dewayne E. Perry. "Experience Report: Barriers to an Effective Process Architecture — Extended Abstract", AT&T Software Symposium, Holmdel NJ, October 1994.

[3] Dewayne E. Perry. "Policy-Directed Coordination and Cooperation", *7th International Software Process Workshop* , Yountville CA, October 1991.

[4] Dewayne E. Perry, Nancy A. Staudenmayer and Lawrence G. Votta. "Understanding Software Development Processes, Organizations and Technologies", *IEEE Software* , July 1994.

[5] Dewayne E. Perry and Lawrence G. Votta. "A Tale of Two Projects", July 1993. Technical Memorandum.

[6] Dewayne E. Perry and Alexander L. Wolf. "Foundations for the Study of Software Architecture", *ACM SIGSOFT Software Engineering Notes* 17:4 (October 1992), pp 40-52.

[7] Alexander L. Wolf and David S. Rosenblum. "Process-Centered Environments (Only) Support Environment-Centered Processes", *8th International Software Process Workshop* , Dagstuhl Germany, March 1993.