Re-Engineering a Software Development Organization as a Complex Adaptive System

Steven Gollery
Software Engineer
Collaborative Agent Design Research Center
Cal Poly State University
San Luis Obispo, CA, USA

Abstract

This paper proposes that it is possible and desirable to restructure a software development organization as a complex adaptive system. Such a structure would increase the possibility that the organization would be able to survive and prosper in a rapidly-changing market place by shortening response time, increasing flexibility in the face of new and unexpected circumstances, and allowing the organization to learn quickly from its experiences.

The paper shows how a common information repository can be used to provide decision support to developers and managers, and discusses the relationship between decision support systems and complex adaptive systems.

Introduction

As software systems become more complex, the processes needed to construct these systems become more complex as well. Simultaneously, market forces are tending to shrink the time available for building a successful system. Software development organizations are faced with rising complexity and falling and falling time-to-market.

This reality drives the need for software development organizations to find methods of responding to and preparing for rapid change.

Complex adaptive systems have the ability to evolve rapidly in response to changes in their environment, and hence to survive in situations where a rigidly-structured system would fail (Holland 1995, Kauffman 1992). This appears to be a desirable trait for organizations that are subject to stimuli from many points, both internal and external. However, complex adaptive systems also have the property that large unpredictable changes can result from small inputs. The impossibility of either foreseeing or controlling the total results of a change can be seen as a negative, both for software systems and for organizations.

This paper presents some thoughts about the implications of applying complex adaptive systems notions to a corporate entity whose primary occupation is the development of computer software systems. In particular, the paper addresses a strategy for the reorganization of such a business so that it will gain the ability to respond rapidly and flexibly in the presence of new stimuli. This paper also considers some of the cultural changes that may be necessary in some organizations in order to adapt to a complex adaptive systems model.

An Analogy: Software Development as a Cooperative Decision-Making System

The general structure of a software development project mirrors the architecture of an agent-based, distributed object system (Porzcak et al, 1999). In this view, we can consider developers, testers, system administrators, project leaders, and so on, as agents in the system. The various

artifacts of the project (e.g., requirements documents, architecture and design, source code, bug reports, schedules, etc.) are the objects on which the system works. The object distribution system is represented by a combination of many tools: the development environment; the bug reporter; the version control system; that programs that are used to create project schedules; and many more.

It is in this last area (i.e., that of tools to create, manipulate, and query the artifacts of software development) that the analogy between actual development processes and true cooperative decision-making systems fails in the current structure of many organizations. While integrated development environments exist for a subset of the activities comprising the development process, in general development artifacts are isolated from each other, and no tool can serve as a single source for this information, as the object distribution network would do in a cooperative decision-making system.

Without the equivalent of an object distribution system for the development process, there is an increased reliance on humans to perform actions that could potentially be automated: data produced by multiple tools in not associated in ways that allow programmatic access and reasoning. In order for the development process to become a true cooperative decision-making system, we must find ways to automate more of the process itself, so that all aspects of development become more of a partnership between human and computer.

The Common Information Repository

In order to move towards a software development process that is supported by a cooperative decision-making system, the first requirement is that all artifacts of the development process are stored in a single common repository. This does not mean that all information is stored in a single database. Instead, it means that all artifacts can be accessed through the same interface, and that links between artifacts are maintained by the repository, enabling automated navigation and queries.

As an example: we might like to ask the repository about the development status of a specific requirement. This query can be performed only if the repository contains the relationships between requirements, design documentation, test cases, schedule, and so on. Each of these objects may have been constructed using a specific tool, and may be stored in that tool's unique file format. It is likely that none of the objects involved in this relationship can be accessed using the same query mechanism as any of the others.

In order to represent and exploit the associations among development process artifacts, we need a higher-level model of these objects, one that abstracts away from the physical storage mechanism and allows connections between data from disparate origins. This higher-level model must be extensible both in the information represented and in the operations that can be applied to the information. Extensibility is the key to the ability of the information repository to support continuous and rapid evolution: the repository must be able to respond to changing information needs within a very limited amount of time. This precludes the possibility that all (or even a large subset) of the requirements of the repository can be known in advance.

From Individual Projects to a Single Development Organization

While placing more project information in a common repository will tend to improve the quality of the development effort, the repository by itself will not enable a software development

organization to transition to a corporate architecture that supports the concept of the organization as a complex adaptive system. The common information repository is only a necessary first step in this restructuring effort.

One traditional model for the structure of a software development company has been to assign personnel to specific projects, generally for a large portion of the development life-cycle. As a complex adaptive system, however, the company as a whole must become capable of shifting resources swiftly in response to changes in its environment (Pohl, 1999). This suggests that personnel assignments would become more fluid, with individuals attached to a project only for the duration of specific tasks, then migrating to another project in order to perform other tasks that require their specific skills.

For this model to succeed, the repository must be capable of linking personnel information, including skills and interests, the schedule information for all current projects, and the requirements and schedule of all proposed projects. This will enable scheduling agents to check for conflicts when multiple projects seek to incorporate the services of the same developer, and to report on the availability of developers with requisite skills.

An Example Scenario

When a software development organization is centered around a shared distributed object repository system, opportunities for new workflow models are created. These models may be formally defined, allowing a wider scope for process automation than currently exists. In particular, it may become possible to shift the responsibility for some activities from humans to their (software-based) agents.

We can imagine an environment where a new project is defined as an object which is placed in a commonly visible location. Various agents would then respond to and perhaps alter the new project object. There might be, for instance, an agent that maintains the requirements database, one that performs cost estimation, and one that is responsible for relating the new project to designs produced for previous projects with similar requirements.

As a simple example, consider a possible sequence of events for the creation of a new project idea. In the following the list, the word "agent" should be understood to mean either a human being or a software construct.

1. The concept for a new product is entered into the system. As with all other artifacts related to the organization, this concept is placed into the repository in a structured format that allows automated processing.
2. All agents that have registered interest in new product ideas are notified.
3. Individual agents then register their interest or non-interest in this particular idea. This forms the equivalent of a mailing list for further discussions.
4. Agents post responses to this idea. Some agents, for example, might be able to identify commonalities between the new product proposal and existing or previous projects.
5. If there is enough interest, the originating agent requests a project brain-storm meeting. This request also has a standard format, so that agents can register interest in learning about new meetings.
6. Agents indicate interest in attending.

7. The initial meeting request has also alerted an automated meeting agent to monitor this conversation. As agents state that they want to attend, the meeting agent checks their schedule and the schedule of meeting rooms to find the most accessible time and place. When all agents have responded, the meeting agent posts a proposed schedule, including indications of level of conflict for various times.
8. Interested agents select a time and place, and the meeting agent reserves that room at that time.

This is a small example: many standard organizational interactions could be envisioned as consisting of a combination of human and automated actions. As in other cooperative decision-making systems, the advantage comes from moving the responsibility for keeping track of information from the human agents to the software system. This increases both the reliability and the quality of the resulting decisions, as well as supporting the repeatability of the process.

Such a system would be able to help convert data into information: two pieces of data (e.g., work request and personal schedule) become information exploitable by software agents, once a relationship has been established between them.

As software development organizations incrementally construct support systems based on a common repository, we will begin to discover what parts of the project life-cycle can be automated and what must remain a human responsibility. Increasing the amount of work that the development system can do for us will reduce the amount of information that humans need to monitor in order to manage and guide the development process. At the same time, enabling automated reasoning on the information in the repository will decrease the possibility that information will be lost or overlooked. Both of these results increase the amount of time that people will be able spend in creative endeavors, improve the quality of products, and decrease the organization's response time.

How Does this Proposal Relate to Complex Adaptive Systems?

Under the proposed reorganizational concepts, the human resources of an organization are considered as small teams or individuals. Each individual has the ability to move to new areas of the organization based on his or her set of skills as conditions change. This gives the organization the ability to reconfigure itself in response to new and unexpected stimuli.

Complex adaptive systems rely on feedback as a means of control. As a development organization uses the complex adaptive system model to restructure itself, personnel will migrate to areas where the amount of work is greatest. This migration will reshape the organization based on external stimuli. In turn, the new shape will tend to guide the selection and creation of future projects. The ability of the organization to place personnel at the point where their skills will have the greatest advantage is in itself a form of feedback.

The common information repository becomes the memory of the organization. This results in a library of plans, designs, and solutions, along with the measurements of the success of previous projects. In planning new projects, project leaders will be able to start for earlier work while avoiding pitfalls into which other projects have previously stumbled. Both of these benefits will tend to decrease the organization's response time and increase the probabilities of success.

The availability of an organizational memory also serves to enable the organization to change its response to a repeated stimulus, and corresponds to an organism learning from experience. The

ability to change a response is in marked contrast to the behavior of a typical computer program which is, almost by definition, intended to produce the same output whenever it receives the same input. For a system to change over time, it must not only remember its previous responses, but incorporate mechanisms for relating the response to the result, and to evaluate the result. This is equally true for software systems and for organizations.

Complex systems cannot be controlled from "above" in a hierarchical fashion, so the importance of feedback as the only real method of control cannot be overstated. Small changes in input tend to translate into large, mostly unpredictable changes in the system; this means that an organization needs to be able to take every opportunity to measure and evaluate the current state of the system, so that it can make necessary (small) corrections. The combination of the repository to contain the information and software agents to monitor the information serves to provide a useful and necessary level of feedback.

Some Implications of the Trend towards Organization-based Resources

In an organization that has multiple simultaneous projects, all projects must have access to all the available information. Again: this does not imply that information will be stored in a single database or file format. Rather, it means that any artifact storage mechanism must include the ability to programmatically access the information in the repository in a distributed fashion.

It is likely that each project will have unique information needs. This argues against centralizing the responsibility either for creating information storage (e.g. defining database tables) or for entering data. Such centralization would tend to increase the time it takes to add information to the repository, which in turn would reduce the ability of the organization to react to the new information in a timely manner.

Instead, it is likely that the organization would create a repository team whose responsibility would be to construct standard tools that other project teams would use to define, enter, modify, and search for information. This approach would enable each team to determine what information it needs to maintain, without preventing access to this information by the rest of the organization.

An information repository is built incrementally. As each development team defines and enters its own information, the individual project receives closely targeted immediate benefits. Over time, the repository will evolve in its capabilities as teams enter more information, thus improving the repository's benefits for the entire organization.

As more data and information become part of the repository, it will become important to create a common vocabulary for describing artifacts and their parts, so that tools to mine the repository can become more sophisticated and can be written at a higher level of abstraction. This development artifact vocabulary may be the result of an organization-wide effort, or may evolve over time based on the information that each project decides to put into the repository.

The information within the repository should be seen as a primary organizational resource whose maintenance and expansion are vital to the continued health of the company.

Some Implications of the Trend towards Organization-based Personnel

The organization will need to make a commitment to the idea that personnel will not be permanently attached to a given project, but will instead attach themselves to a given unit of work based on the requirements of the work and their own skillset. This may require a shift in

attitude for project leaders: many of them will no longer have full use of a particular developer over a long period of time.

For managers accustomed to direct control based on authority, this attitude shift will be even more profound. Complex adaptive systems cannot be controlled from above: management must rely of feedback mechanisms. The culture of the organization must become oriented towards indirect means of control based on the free flow of information. Since complex adaptive system can have a tendency to translate small inputs into large and unexpected results, feedback must be available at all relevant levels of the organization as early as possible, when problems are still small enough to be correctable.

In a complex adaptive system, teams will become much more fluid, coming together for the purpose of implementing some set of functionality and then breaking up and reforming for other purposes. At the team level, this is similar to the difference between "partnership" and "relationship" at the organizational level. The idea of teams created for a single purpose is also similar to the emerging model of the virtual corporation: a "corporation" that exists for the purpose of fulfilling a single contract or creating a single product, and which is composed of units from several existing corporations. This concept matches the strengths of each team member to the needs of the specific unit of work.

One danger in this concept is that, when a team works will together, the team is stronger than the sum of its members. To break up such a team arbitrarily would be detrimental to the organization. Therefore, the process of creating a new team should include giving weight to the history of the people in the team, and favor team members who have worked together before.

The skills and knowledge necessary to create software products will continue to change. In order to anticipate this change, the organization must support further development of individual skillsets. This support may come in the form of specified time for exploration of new technologies, or a financial aid for professional training both on and off-site.

The organization must enable and encourage personnel to move from one area of development to another. The best developers are those who want to learn new techniques and to expand their experience. No software development company can survive if it places these developers in a single niche and does not support their professional growth. Further, a developer whose experience is limited to a single area will become less valuable as the organization's internal and external environments change.

Conclusion

In order to survive and grow as part of an increasingly fluid and unpredictable environment, software development organizations must find ways to become more like a complex adaptive system: they must be able to respond quickly to unexpected stimuli; to evolve in order to compete successfully in the face of rapid change; and to construct and deliver systems on shorter timelines.

By re-imagining themselves as examples of complex adaptive systems, development companies can benefit from all the strengths of such systems. However, they may also experience the problems of complex adaptive systems: unpredictability, chaotic behavior, and lack of explicit control mechanisms (Waldrop 1992, Holland 1998).

This paper has presented the conceptual underpinnings that support a view of the software development company as a complex adaptive system. It has shown the advantages and

disadvantages of such a view, and briefly examined some of the consequences of this type of organization on the culture of the company. The paper has also discussed the supporting relationship between cooperative decision-making systems and complex adaptive systems.

References

Holland J.H. (1995); 'Hidden Order: How Adaptation Builds Complexity'; Addison-Wesley, Reading, Massachusetts.

Holland J.H. (1998); 'Emergence: From Chaos to Order'; Perseus Books, Reading, Massachusetts.

Kaufmann S.A. (1992); 'Origins of Order: Self-Organization and Selection in Evolution'; Oxford University Press, Oxford, England

Pohl J. (1999); 'Some Notions of Complex Adaptive Systems and Their Relationship to Our World'; InterSymp-99, focus symposium on Collaborative Decision-Support Systems, Baden-Baden, Germany, August 3

Porczak M., K. Pohl, R. Leighton, A. Davis, H. Assal and L. Vempati (1999); 'IMMACCS: Urban Warrior Advanced Warfighting Experiment After Action Report'; CAD Research Center, Cal Poly, San Luis Obispo, California, April.

Waldrop M. (1992); 'Complexity: The Emerging Science at the Edge of Order and Chaos'; Simon and Schuster, New York.