# Organizational Patterns: Looking Back Nearly 30 Years

Neil B. Harrison
Department of Computer Science
Utah Valley University
Orem, Utah, USA
neil.harrison@uvu.edu

***Abstract*—Since patterns of the dynamics of organizations were introduced, they have had an enduring impact on the practices of software development. They have influenced the notable software practices such as Scrum and Extreme Programming, and Agile software development in general. The main work, Organizational Patterns of Software Development has been widely cited. It continues to be cited in academic works even to this day. The patterns appear to have had the greatest impact on agile development processes and the dynamics of communication and collaboration. However, they also appear to have had some influence in a broad range of other areas.**

***Keywords—Patterns, Organizational Patterns, Agile Software Development, Scrum***

## I. Introduction

The human dynamics of software development organizations are an important factor contributing to or hindering the success of the team. These dynamics include the roles individuals play, the patterns of communication, the amount of communication, leadership and mentoring, and others. Successful practices in these areas have been captured as *Organizational Patterns*.

Patterns of the dynamics of organizations were presented at the first conference of Pattern Languages of Programs in 1994 [1]. The patterns were based on numerous studies of software development organizations, including a notable study of Borland's software development organization [2]. These studies revealed that the structure of organizations and the communication dynamics in the teams play a major role in the productivity of the teams.

Continued studies of organizations yielded further patterns and information about organizational dynamics [3, 4]. Jim Coplien and Neil Harrison brought all the patterns together in "Organizational Patterns of Agile Software Development" [5].



The Organizational Patterns book presents 104 patterns, divided into four distinct, but not completely separate pattern languages. They reflect different ways in which human dynamics play roles in organizations. These languages are "Project Management", "Piecemeal Growth", "Organizational Style", and "People and Code". The first deals with how to set up an organization, including roles and interactions among them. The second is about the fact that an organization evolves during the course of the project. People come and go, roles emerge, and the functions of roles may change. The third language is all about how social forces impact the code – and vice versa.

The robust sales of the book indicated that it has had some influence on the software development community at large, and perhaps in realms beyond software development. In this paper, I have attempted to understand the nature of their impact. I examined the references to the patterns as well as current and past software methodologies. In the following sections, I discuss how these patterns appear to have influenced the software development landscape. Organizational patterns from the aforementioned book [5] are given in italics, and thumbnail definitions are provided in an appendix to this paper.

## II. Historical Background

Patterns never claim to be novel work. Indeed, they focus on capturing practices that have been proven to work. As such, the organizational patterns build on a legacy of effective teams.

Since its inception, software development has likely often been done in groups of two or three people working closely together. Richard Gabriel reports an early record of programming in pairs that dates from 1957 [6].

Mel Conway studied how committees invent new things and found that the organizational structure of the team tended to match the structure of the artifact being invented [7]. Coplien and Harrison captured this as the organizational pattern, *Conway's Law*. This pattern has been much discussed and debated.

One of the early works that explored the impact of people and organizations on the production of software was "The Mythical Man-Month." It was the source of "Brooks' Law," which is commonly portrayed as, "adding people to a late project makes later." Coplien and Harrison found organizational patterns to deal with the reality that people are not

interchangeable commodities which can be added to or removed from projects at will. The most notable of these was probably *Moderate Truck Number*; other patterns included *Apprenticeship* and *Sacrifice One Person*.

In 1971 Thomas Allen published the results of years of studies of engineering organizations in, "Managing the Flow of Technology" [8]. This work explores the flow of information in engineering teams, including the impact of physical distance on information flow. Coplien and Harrison observed similar things, including the value of informal places for conversations (*The Water Cooler*.) Allen also described the value of "gatekeepers", which was the inspiration for the pattern of the same name.

In 1980 Barry Boehm published "Software Engineering Economics" [9]. Boehm had studied many large software projects in order to determine how to predict software development effort. The book presented a model of software development effort based on project size as well as numerous other factors. However, while Boehm's research showed team capability to be a highly significant factor in the effectiveness of software development, it did not explore the role of human dynamics of the teams in software development. The organizational pattern research picked up where Boehm left off: Cain, Coplien and Harrison found that human dynamics could play at least as big a role as any of the factors in Boehm's model. Studies found several organizations with extremely high productivity – designated as "hyperproductivity" (see [2], [4].) These levels of productivity were far above the productivity measured by Boehm, and were rooted in what became the organizational patterns, such as levels of communication and leadership style.

Much of the information in the patterns came from studies of many organizations, both inside and outside of Bell Labs, where we worked. In addition, we saw in our own experiences at Bell Labs many beneficial practices that became patterns. Many of the practices we used became patterns in the "People and Code" pattern language. A notable example is *Named Stable Bases*.

At the time of the writing of the organizational patterns, there was a groundswell of movement away from the traditionally perceived waterfall approach to software development to more iterative approaches. The Agile Manifesto was written [10]. Scrum [11] and Extreme Programming (XP) [12] were being developed. These practices and the organizational patterns influenced each other, although the patterns predated and influenced many of the practices (see later sections.) At about the same time, Ward Cunningham wrote the Episodes Pattern Language [13], which was part of the inspiration for the pattern, *Development Episodes*.

## III. Influence on Agile Practices

We can see the imprints of many of the organizational patterns in many Agile practices, both at their start and even today. As noted above, Agility was coming of age in the mid-1990s and early 2000s; about the same time the organizational patterns book was being written, and slightly after the earliest organizational patterns had been published. While it is difficult to definitively quantify the effect of the organizational patterns on Agile methodologies, we see evidence of their impact.

### A. The Agile Manifesto

The Agile Manifesto for Software Development, published in 2001, is a statement of comparative values with respect to approaches to software development. It is clearly influenced by the rejection of large methodology-heavy software projects and the emergence of small episodes of development.

While there is some evidence of influence of the organizational patterns on the Agile Manifesto, it is limited primarily because they address different things. The Agile Manifesto is a statement of values and principles of software development, while the patterns are solutions to people-related problems in teams. The two are complementary, but not strongly overlapping. Note that several of the creators of the Manifesto were familiar with the organizational patterns, including Jeff Sutherland, Mike Beedle, Ward Cunningham, and others. (Jim Coplien was absent from the Manifesto meeting because of a scheduling conflict.)

There are, however, specific synergistic areas. Here are some examples where the patterns provide answers or implementations of the values or principles in the Agile Manifesto:

- "Individuals and interactions" is the premier value of the Manifesto. A large number of patterns implement this value, including *Community of Trust, Team Pride, Matron Role, Engage Quality Assurance, The Water Cooler, Lock 'Em up Together*, and others.
- Customer collaboration is a value. This is reflected in the pattern, *Engage Customers*.
- The Manifesto states that comprehensive documentation is less valued. Where comprehensive documentation is still required, the *Mercenary Analyst* pattern provides an answer.
- A principle is early and continuous delivery. Patterns that facilitate this include *Incremental Integration* and *Named Stable Bases*.
- A principle is to trust the team to get the job done. This is reflected in the patterns *Community of Trust* and *Self-Selecting Team*, among others.

### B. Extreme Programming

Extreme programming shows some influence of the organizational patterns. The influence is most apparent in two values and one practice:

- Communication: This shows the same influence as "individuals and interactions" of the Agile Manifesto.
- Courage: This is only possible in a team that has the pattern *Community of Trust*. Courage is also embodied in the pattern, *Wise Fool*.
- Pair Programming: This practice is based on *Developing in Pairs*.

A few practices in XP are somewhat at odds with some organizational patterns. This indicates that they should be examined carefully.

- Pair Programming may not be the most expedient or effective approach in all cases. Sometimes it is better to use the pattern, *Solo Virtuoso*.
- XP tends to encourage emergent architecture through refactoring. However, an intentional system architecture is more robust; see *Architecture Team*, *Architect Controls Product*, and *Architect Also Implements*.
- XP encourages detailed unit testing. However, developers are often blind to their own mistakes. Be sure to have independent quality assurance, but use the pattern, *Engage Quality Assurance*. Also, testing at the system level matches customers' needs best. See *Application Design is Bounded by Test Design*.
- *Onsite Customer* can lead to feature creep. It may also be impractical. The organizational pattern *Surrogate Customer* is a workable alternative.

Test-Driven Development (TDD) suggests writing (unit) tests before writing the associated code. The pattern *Application Design is Bounded by Test Design* recommends establishing the application by designing the tests a usage-centric system-wide approach.

### C. Scrum

Scrum has some of its roots in organizational patterns. In "Scrum : the Art of Doing Twice the Work in Half the Time" [11], Jeff Sutherland explains how the organizational studies and patterns helped shape some of the practices in Scrum. Daily Scrum meetings came from this work (see pattern: *Standup Meeting*).

Sutherland and others have written patterns of Scrum [14]. The book took shape over numerous workshops ("ScrumPLOP") held over several years. In these workshops, the authors formed and reviewed the patterns. The authors frequently discussed the influence of organizational patterns in Scrum, and agreed that the organizational patterns are part of the foundation of Scrum. The Scrum book lists forty of the organizational patterns part of the complete set of Scrum patterns. A sample of the organizational patterns that form part of the foundation of Scrum follows:

- *Conway's Law* gives guidance for organizing the workforce, "partitioned according to the most important concerns for the creation of value by the enterprise." It was rewritten and enlarged in the Scrum patterns.
- *Community of Trust* is a starting point for all Scrum organizations. Without it one cannot have functional relationships between the Scrum Team and the Product Owner, or between the Scrum Master and the Development Team.
- It is essential that Scrum Team Members all work together toward a common Sprint Goal. This unified approach is built on the organizational pattern, *Unity of Purpose*.
- The Scrum pattern Small Teams has roots in the organizational pattern, *Size the Organization*.
- The Scrum pattern Self-Organizing Team is closely related to *Self-Selecting Team*, although they are not exactly the same. It also follows the organizational pattern, *Informal Labor Plan*. A related pattern is *Developer Controls Process*.
- The Scrum Patterns Product Backlog and Sprint Backlog are implementations of the organizational pattern, *Work Queue*.
- The Scrum pattern Daily Scrum is built on the *Standup Meeting* pattern. *Standup Meeting* describes the purpose and structure of frequent small meetings; Daily Scrum additionally specifies the frequency of such meetings.
- In Scrum, problems that prevent the team from moving forward are kept in an Impediment List. Individuals who are able to resolve these items do so. This is similar to the organizational patterns, *Interrupts Unjam Blocking*, *Don't Interrupt an Interrupt*, and *Someone Always Makes Progress*. While there is not strong evidence that these patterns inspired the Impediment List practices, it is clear that they are related.to each other.

### D. Other Use in Practice

The organizational patterns may have had influence in other software development practices, as well as practices in other industries. For example, other areas in which social interaction is significant, such as education, may be rich in the use of organizational patterns. This is an area for possible future exploration.

## IV. Academic Work

The organizational patterns have been the subject of some attention in academic research as well. The organizational patterns book has been widely cited in academic papers. We studied these citations to understand the nature of the academic impact of the organizational patterns. From this we learned that the organizational patterns have had a moderate impact on the field and continues to have some impact.

I limited the investigation to the references to the organizational patterns book; it is likely by far the best-known source. Other publications of organizational patterns are much smaller, and virtually all the patterns in those articles appear in the book. The citation count of the organizational patterns book from Google Scholar is currently 525. The most recent citation (as of May 10, 2023) was March 2023.

In order to understand the areas of influence, I investigated the topics of the citing papers; see Table 1. This is not an exhaustive list of paper topics, but it indicates the most common research topics that reference the book.

TABLE I. CITING PAPER TOPICS

| Research Topic | Frequency of Occurrence |
|---|---|
| Agile software development | 31 |
| Group dynamics and collaboration | 27 |
| Architecture, Design, and Modeling | 19 |
| Management, especially project management | 15 |
| Technical topics such as tools | 15 |
| Distributed software development | 15 |
| Communication | 13 |
| Scrum | 8 |
| Software Engineering | 8 |
| Education and teaching | 8 |
| Business | 7 |
| Ontology of organizational patterns | 7 |
| Requirements | 6 |
| Organizational structure, including organization hierarchy | 6 |
| Experience | 6 |
| Free and Open-Source Software | 6 |
| Conway's Law | 6 |
| Systems engineering | 5 |
| Pair programming | 5 |
| Large-scale agile software | 5 |
| Extreme Programming | 3 |
| Customer interaction | 3 |
| Startups | 3 |
| Other miscellaneous topics, including general papers on patterns | 43 |

Note that some of these categories overlap some. For example, papers on Agile software development may also discuss Scrum. Some papers may fit into more than one category. Therefore, these categories should be considered to indicate only general trends. We make the following observations:

- It is not surprising that the most frequent topic observed was agile software development. Note that in spite of the title of the book, the organizational patterns can be used in non-agile software development organizations, as well as non-software organizations.
- The categories of collaboration and communication certainly have considerable overlap. Collaboration relates to individuals and groups working together, while communication refers to the transmittal of information between individuals and groups. However, we cannot be sure that the authors of these papers share the same definitions.
- Distributed development was a common topic among the papers. At the beginning of the organizational patterns work, distributed development was still young and relatively uncommon. There are a few organizational patterns that specifically apply to distributed teams, namely *Face to Face before Working Remotely*, *Standards Linking Locations*, and *Organization Follows Location*. The recent explosion of remote work changes the dynamics of teams. This is an important area for further study.
- Open-source software has become very popular in recent years. Like remote work, it changes the human dynamics of software development. There has been some investigation into this topic, but there is no doubt much to be learned.
- The number papers on software architecture that cited the book was significant. One notable paper that described the key role of organizational patterns in software architecture [15] mentioned ten different organizational patterns and explained how they impact software architecture.

The organizational patterns may have influence in fields other than software development. They certainly apply to many different types of organizations, particularly the patterns on project management, piecemeal growth (or the organization), and organizational style. The patterns on people and code relate specifically to software. I did not extensively search literature outside of the software field but observed little to no evidence of influence outside of software organizations.

## V. CONCLUSIONS AND FUTURE WORK

While this study is limited in scope, it indicates that the patterns have played a meaningful role in software. I am pleased with the continuing influence of the organizational patterns on software development. Of special note is their influence in agile software development and Scrum.

Further study of the role of organizational patterns in these areas may be fruitful:

- Distributed software development has changed the way teams interact with each other. These new ways of communicating and collaborating demand careful study. There are certainly new patterns of organizational dynamics to be mined. However, distributed development is certainly at odds with some patterns. Practices of distributed development should be examined and perhaps changed in order to attain the benefits of some of the patterns.
- Open-source software presents a new paradigm of teams. These practices should be examined to understand the best practices and the patterns used. This is likely a particularly rich area of study.
- Software architecture continues to have interesting interactions with teams. It too, is changing, with the advent of cloud computing. There may be some implications for the organizational patterns.
- It may be useful to explore the references to organizational patterns in academic works in more depth. For example, the papers may reference specific patterns or sets of patterns.

## REFERENCES


[1] Coplien, James O., "A Generative Development-Process Pattern Language." In Pattern Languages of Program Design, Addison-Wesley, Reading, MA 1995, pp. 183-237.

[2] Coplien, James O., "Borland Software Craftsmanship: A New Look at Process, Quality, and Productivity," in Proceedings of the 5th Annual Borland International Conference, Orlando, FL, USA, 1994.

[3] Harrison, Neil B., "Organizational Patterns for Teams", in Pattern Languages of Program Design 2, Addison-Wesley, Reading, MA, 1995, pp. 345-352.

[4] Cain, Brendan G, Coplien, James O, and Harrison, Neil, "Social Patterns in Productive Software Development Organizations." (vol. 2). Annals of Software Engineering, 1996.

[5] Coplien, J. and Harrison, N., Organizational Patterns of Agile Software Development, Prentice-Hall, 2005.

[6] Gabriel Richard P., personal communication, sometime in the last 20 years.

[7] Conway, Melvin E., "How do Committees Invent?" in Datamation, 14 (5), 1968, pp. 28-31.

[8] Allen: Managing th eflow of Technology

[9] Boehm, Barry, Software Engineering Economics, Prentice-Hall 1981.

[10] Agile Manifesto: agilemanifesto.org, 2001.

[11] Sutherland, Jeff and Sutherland, J. J., Scrum: the Art of Doing Twice the Work in Half the Time, Penguin Random House, New York, 2014.

[12] Beck, Kent, Extreme Programming Explained, Addison-Wesley, Reading, MA, 2000.

[13] Cunningham, Ward, "Episodes: A Pattern Language of Competitive Development," in Pattern Languages of Program Design 2, Addison-Wesley, Reading, MA, 1995.

[14] Sutherland, Jeff, et al., A Scrum Book: the Spirit of the Game, Pragmatic Bookshelf, 2019.

[15] Booch, Grady, "On Architecture", in IEEE Software, May-June 2008, pp. 18-19.

## VI. Appendix: Patterns referenced

The following organizational patterns were referenced in this paper:

- Application Design is Bounded by Test Design: Design the system according to tests that define it.
- Architect Also Implements: System architects should have recent practical implementation experience.
- Architect Controls Product: The product is defined by the architecture.
- Architecture Team: Use a team of architects representing different views of the system.
- Community of Trust: Team member must trust each other and feel safe in their interactions.
- Conway's Law: The organization of a team and the organization of the artifact it invents are isomorphic.
- Developer Controls Process: Developers control the way they work.
- Developing in Pairs: Developer should often work in pairs.
- Development Episodes: Divide work periods into separate dedicated episodes.
- Don't Interrupt an Interrupt: Fix one critical blocking issue at a time; don't get distracted by another issue.
- Fact to Face Before Working Remotely: Remote teams should meet each other personally first.
- Engage Customers: Closely work with customers during all phases of the project.
- Engage Quality Assurance: Involve quality assurance in test design right from the start.
- Gatekeeper: An individual who looks for future trends
- Incremental Integration: Use frequent, small system builds.
- Informal Labor Plan: Individuals devise their own short-term plans.
- Interrupts Unjam Blocking: Address blocking problems immediately.
- Lock 'Em Up Together: Design teams should physically work together to craft initial designs.
- Matron Role: A person concerned with the human and social needs of the team.
- Mercenary Analyst: Engage someone expert in writing to write documentation.
- Named Stable Bases: Stabilize system interfaces periodically.
- Organization Follows Location: Distributed teams organize along geographic boundaries.
- Self-Selecting Team: Allow the team to select its members.
- Size the Organization: Begin with no more than ten people on the team.
- Solo Virtuoso: Sometimes a person needs to work alone.
- Someone Always Makes Progress: When fixing a blocking issue, try to keep people working on the main development.
- Standards Linking Locations: Distributed teams should have unified standards.
- Standup Meetings: Have short meetings; enforce it by having them standing up.
- Surrogate Customer: In the absence of a real customer, a team member plays the role of customer.
- Team Pride: The team must feel pride in workmanship.
- The Water Cooler: Provide an informal gathering place where innovative ideas are born.
- Unity of Purpose: All team members must understand and share the same goals.
- Wise Fool: A person not afraid to state uncomfortable facts.
- Work Queue: provide a prioritized list of work items.