



Conway's Law Revisited: The Evidence for a Task-Based Perspective

Irwin Kwan, Marcelo Cataldo, and Daniela Damian

A LONG-STANDING APPROACH to managing complex product development is to divide the system into smaller parts and assign responsibility for developing each part to a particular team. Conventional wisdom suggests that this will reduce the communication overhead by making the development teams as independent as possible. This approach follows “the mirroring hypothesis” or “Conway’s law,” based on Melvin Conway’s 1968 argument that “development organizations ... are constrained to produce designs which are copies of the communication structures of these organizations.”¹

Other researchers have since refined Conway’s law^{2–4} and, furthermore, shown that alignment between technical and organizational structures improves productivity and product quality in the development of physical systems.^{4,5} However, the benefits of Conway’s law in software projects are less clear.

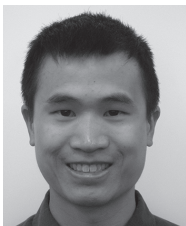
Software systems differ from physical systems in many ways. Their interfaces are easy to change, but the impact of making a change isn’t easily understood, especially in large software systems. Changes in software organizations are complicated by the tacit knowledge that can easily depreciate or disappear when engineers leave the company or change positions. The evidence for align-

ment in physical systems doesn’t necessarily apply to software systems. On one hand, many studies of software projects—including our own—have identified sociotechnical alignment. However, a recent cross-disciplinary review of the mirroring hypothesis in product development found that many software development projects lacked significant alignment. Moreover, the authors found no evidence that the lack of alignment between product and organizational structures was detrimental to those projects.

Does Conway’s Law Matter in Software Development?

The short answer is yes—Conway’s law still matters. However, what also matters is how you examine Conway’s law in a software development context. Software practitioners tend to conceptualize the product’s structure as “the software architecture” and the organization’s structure as “the organizational chart,” just as you might in a physical system. This view isn’t sufficient for project success.

One issue with taking a high-level view of the organization and software architecture is that developers don’t see their work from that perspective. They usually view their work as a set of goals the system must



accomplish. Thus, a developer's tasks involve modifying the system or coordinating with others to modify it to accomplish these goals.⁶

A second issue is that taking an architectural-level view misses the nuances of the developers' work. In some situations, the software architecture can work in the organization's favor as a coordination mechanism,⁷ but we know of no evidence suggesting project-wide benefits, such as better productivity or quality, from achieving alignment. On the contrary, problems have occurred despite architectural alignment with the organization. In fact, two studies reported that achieving architectural-level alignment wasn't sufficient to enable the coordination required to avoid issues such as major quality problems.^{8,9}

This evidence, as well as our past work, suggests that Conway's law matches the unique characteristics of software development when we consider alignment at a task level rather than an architectural level.

Task-Level Alignment

The task-level view of Conway's law conceptualizes a unit of work as whatever task a developer is engaged in, such as a defect in a bug-tracking system, a feature, or a software build. When developers work on a task, they modify artifacts such as files. Previous work has shown that files modified together are often interrelated.¹⁰ If multiple developers work on the same tasks, then the rationale for the alignment is that people working on the same tasks and files should coordinate their actions so they know what's changing in the system.

Evidence shows that task-level alignment benefits a software project. For example, two studies of corporate software development found that maintaining congruence between communication and coordination actions and task dependencies significantly reduced resolution times of development tasks.^{11,12}

A study of open source projects showed the same benefit.¹³

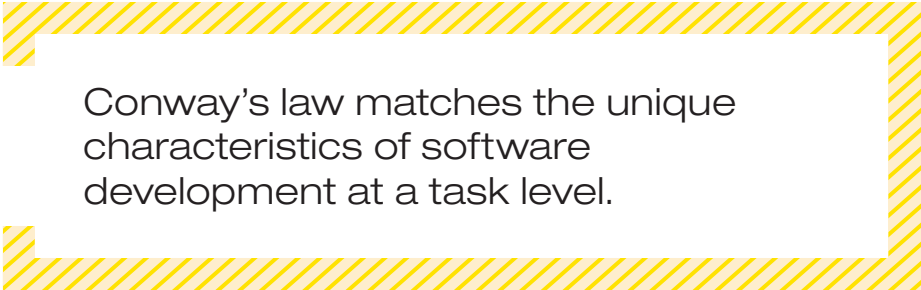
One study of a software company found that architectural-level product dependencies failed to predict 43 percent of person-to-person communications, suggesting that the task level might be more appropriate for seeking such alignment.¹⁴ A later study produced similar evidence, showing that small teams working on a large project were successful in software builds when sociotechnical alignment was high.¹⁵ Paradoxically, this study also found low alignment to be related to high performance during large integration builds.

Open source development is a special case in which research has found evidence of low alignment with no detrimental effects on project performance.⁵ These projects involve independent, geographically distributed contributors that work in the absence of a well-defined organizational structure and make highly independent contributions to the same design. Mutually

task level and practitioners opportunities to benefit from aligning the dependencies developers face in their tasks with their communication and coordination activities. To realize the advantages of Conway's law in practice, we outline a procedure for using this information.

Leverage software repositories for technical dependencies as well as developers' coordination patterns. Augmenting software repository data with a team's knowledge of social interaction patterns provides valuable information to understand whether alignment exists and how it evolves over a development project's lifetime. Communication data in the form of bug reports exists alongside technical artifacts such as code and opens the door for analytics and tool support that can significantly improve an organization's ability to coordinate effectively, particularly in a globally distributed setting.

Identify task-based logical dependencies. Logical dependencies provide valuable information about a software



Conway's law matches the unique characteristics of software development at a task level.

compatible motivations, lack of major economic conflicts of interest, and an established common ground about the evolving design are among the factors that could explain this exception to Conway's law.

Realizing the Benefits of Conway's Law

The wealth of digital information available in software project repositories gives researchers opportunities to further understand Conway's law at the

system's technical relationships and complement the information that the more traditional syntactic dependencies provide.¹⁰ Using logical dependencies, developers can better understand the technical relationships between different source code files or modules as well as a particular change's potential impact. Tools for collecting this information can display dependencies in the appropriate context for developers. However, traditional infrastructure tools also support inquiries about

changes in a particular file or how an artifact affects other files.

Maintain awareness of task dependencies. Design and implementation changes in interrelated software entities could significantly affect developer re-

cilitating sociotechnical alignment by keeping developers aware of relevant development tasks.¹⁶

Discover if team coordination and tasks align. After you identify task dependencies and developers' coordina-

tion relationships, a manager should examine them to identify where alignment exists among team members. This means identifying whether people who are coordinating with each other are also working together on dependent tasks, or vice versa. If a close match exists, then all is well.

vestigation. Are there gaps in knowledge among developers? Do they know which clients to talk to? If coordination involves developers you didn't expect to see, do they represent expertise on a specific part of the system? You want to be aware of hidden experts and ensure that this connection is not lost.¹⁷ Encouraging team members to inform you of the people they contact outside the scope of their current task or who aren't part of your software team expands your awareness of critical knowledge sources for future modifications.

Software development calls for a task-level perspective on the idea that an organization's designs will inevitably mirror its communication structure.

sponsibilities. One common approach to maintaining awareness of other development activities is to subscribe to a particular component or team task in task- or defect-tracking systems. However, this approach becomes ineffective pretty quickly in large projects. Using information about specific dependencies, such as logical or work-related dependencies, helps manage the information volume by filtering it, thus fa-

tion relationships, a manager should examine them to identify where alignment exists among team members. This means identifying whether people who are coordinating with each other are also working together on dependent tasks, or vice versa. If a close match exists, then all is well.

Mismatches suggest potential issues to address. In most projects, a lack of alignment warrants further in-

The evidence for Conway's law suggests that managers should take a step back from looking at a system with respect to inputs, outputs, and call graphs and, instead, examine the tasks that people must perform and how software modules influence these tasks. James Coplien advises that the architect should "serve as the long-term keeper of architectural style" and the developer should control the process.¹⁸ By taking a task-level

Silver Bullet Security Podcast




In-depth interviews with security gurus.
Hosted by Gary McGraw.

www.computer.org/security/podcasts
*Also available at iTunes

The Silver Bullet Security Podcast with Gary McGraw

Sponsored by 

view of alignment, your organization can experience real benefits in a team's ability to work together. 

References

1. M. Conway, "How Do Committees Invent?" *Datamation*, vol. 14, no. 4, 1968, pp. 28–31.
2. D.L. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules," *Comm. ACM*, vol. 15, no. 12, 1972, p. 12.
3. J.D. Herbsleb and R.E. Grinter, "Architectures, Coordination, and Distance: Conway's Law and Beyond," *IEEE Software*, vol. 16, no. 5, 1999, pp. 63–70.
4. C.Y. Baldwin and K.B. Clark, *Design Rules: The Power of Modularity*, MIT Press, 2000.
5. L. Colfer and C.Y. Baldwin, "The Mirroring Hypothesis: Theory, Evidence and Exceptions," working paper, Harvard Business School, 2010.
6. M.E. Nordberg III, "Managing Code Ownership," *IEEE Software*, vol. 20, no. 2, 2003, pp. 26–33.
7. C.R. de Souza and D.F. Redmiles, "The Awareness Network: To Whom Should I Display My Actions, and Whose Actions Should I Monitor?" *IEEE Trans. Software Eng.*, vol. 37, no. 3, 2011, pp. 325–340.
8. P. Ovaska, M. Rossi, and P. Marttiin, "Architecture as a Coordination Tool in Multi-Site Software Development," *Software Process Improvement and Practice*, vol. 8, no. 4, 2003, pp. 233–247.
9. M. Bass et al., "Architectural Misalignment: An Experience Report," *Proc. IEEE/IFIP Conf. Software Architecture*, IEEE CS Press, 2007, pp. 17–25.
10. H. Gall, K. Hajek, and M. Jazayeri, "Detection of Logical Coupling Based on Product Release History," *Proc. Int'l Conf. Software Maintenance (ICSM 98)*, IEEE CS Press, 1998, pp. 190–198.
11. M. Cataldo et al., "Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools," *Proc. Computer-Supported Cooperative Work (CSCW 06)*, ACM Press, 2006, pp. 353–362.
12. M. Cataldo, J.D. Herbsleb, and K.M. Carley, "Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity," *Proc. Empirical Software Eng. Measurement*, ACM Press, 2008, pp. 2–11.
13. P. Wagstrom, J. Herbsleb, and K. Carley, "Communication, Team Performance, and the Individual: Bridging Technical Dependencies," *Proc. Academy of Management Ann. Meeting*, Academy of Management, 2010; available at <http://academic.patrick.wagstrom.net/publications>.
14. M. Sosa, "A Structured Approach to Predicting and Managing Technical Interactions in Software Development," *Research in Eng. Design*, vol. 19, no. 1, 2008, pp. 47–70.
15. I. Kwan, A. Schroter, and D. Damian, "Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project," *IEEE Trans. Software Eng.*, vol. 37, no. 3, 2011, pp. 307–324.
16. J.M. Costa, M. Cataldo, and C.R. de Souza, "The Scale and Evolution of Coordination Needs in Large-Scale Distributed Projects: Implications for the Future Generation of Collaborative Tools," *Proc. 2011 Ann. Conf. Human Factors in Computing Systems (CHI 11)*, ACM Press, pp. 3151–3160.
17. I. Kwan and D. Damian, "The Hidden Experts in Software-Engineering Communication," *Proc. 33rd Int'l Conf. Software Eng. (ICSE 11)*, IEEE CS Press, 2011, pp. 800–803.
18. J. Coplien, "Organizational Patterns," *Enterprise Information Systems VI*, Springer, 2006, pp. 43–52.

IRWIN KWAN is a postdoctoral researcher at Oregon State University. Contact him at kwan@eecs.oregonstate.edu.

MARCELO CATALDO is a senior research engineer at Robert Bosch's Research and Technology Center. Contact him at marcelo.cataldo@us.bosch.com.

DANIELA DAMIAN is an associate professor at the University of Victoria, Canada, and leader of the Software Engineering Global InterAction Laboratory (<http://segal.uvic.ca>). Contact her at danielad@cs.uvic.ca.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

THERE NOW IS A QUICK AND EASY WAY TO FIND OUT ABOUT OUR COLLECTION OF TRANSACTIONS



Our new "What's New in Transactions" webpage provides an overview of our 14 peer-reviewed scholarly journals.

- Read free articles from each title,
- View and listen to new multimedia,
- Learn about recent special issues and special sections, and
- Discover open calls for papers.

It is ever changing like the research areas our journals cover, so make sure to come back frequently. Visit <http://www.computer.org/whats-new> today!



IEEE  computer society