# Software Design and Engineering as a Social Process

**William A. Stubblefield**
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185
wastubb@sandia.gov

**Tania L. Carson**
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185
tlcarso@sandia.gov

## Abstract
Traditionally, software engineering processes are based on a formalist model that emphasizes strict documentation, procedural and validation standards. Although this is a poor fit for multidisciplinary research and development communities, such groups can benefit from common practices and standards. We have approached this dilemma through a process model derived from theories of collaborative work rather than formal process control. This paper describes this model and our experiences in applying it in software development.

## Keywords
Software process, agile methods, case study, situated cognition, activity theory, distributed cognition.


*CHI 2007*, April 28–May 3, 2007, San Jose, California, USA.
ACM 978-1-59593-642-4/07/0004.

## ACM Classification Keywords
D.2.0 Software Engineering - General, D.2.9 Management, D.2.10 Design.

## Introduction
Traditional software engineering practice assumes a fundamental distinction between design's messy, empirical explorations, and engineering's need to control complexity and change. This is evident in the accepted separation of design and engineering into distinct activities that communicate through formal, authoritative system requirements. In recent years, both theory and practice have undermined this distinction. Empirical studies of work reveal a rich, improvisational process that challenges the possibility of "capturing" user needs to levels of validity and stability assumed in formal requirements [1, 2]. Agile software methods [3-5] have shown the power of iterative prototyping to support concurrent design and engineering. The view of design and engineering as distinct activities has given way to an iterative process of refining design ideas across diverse representations: requirements, scenarios [6], storyboards, prototypes, and, ultimately, delivered software.

In spite of this, rethinking the relationship between design and engineering remains difficult, particularly in a corporate or governmental context. The separation of

"hard" engineering and "soft" design skills permeates our educational and work cultures. Management often places a premium on process models like the Capability Maturity Model (CMM) [7] that emphasize consistency and objective metrics over the often messy, empirical process of research and design. Although agile software methods challenge this orthodoxy, they often ignore such techniques as ethnography, storyboarding and usability testing to focus on coding as the vehicle for design. Where design and engineering do interact successfully, this success often depends upon implicit activities and social commitments, leading to communication problems, and difficulties in growing or even sustaining the development community.

This paper describes our experience in developing and applying a software development methodology that builds on the deeper similarities and interdependencies between design and engineering to address these problems.

## Background

The Systems Engineering and Analysis (SEA) business area at Sandia National Laboratories conducts applied research and development in three main areas: modeling and analysis of national infrastructure, knowledge management, and collaboration support. The complex, multidisciplinary nature of these problems has caused SEA to evolve into a loosely coupled community of computer scientists, modelers, infrastructure experts, ethnographers, usability specialists, social scientists, cognitive scientists and others who collaborate largely through informal, face-to-face means. In recent years, our success in delivering useful software and models has caused us to grow, bringing increased pressure to formalize our processes, both to address corporate requirements and to support our own need to maintain existing systems and manage our growth.

Recognizing that excessive formality could harm this productive social dynamic, our management chose not to impose disciplines such as CMM on the community, and challenged staff to take a grass-roots approach to the problem. Activities and Practices (AP) is a software process that has come out of this effort. AP builds on theories of cooperative work, situated cognition and our own sense of community to increase formality without restricting the interplay of disciplines – particularly design and engineering – that has proven so effective.

## Foundations

What design and engineering (as well as scientific research, another component of our community) have in common is that they are socially situated activities. Traditional software engineering takes a rationalist approach that emphasizes rigorous processes, formal representation, and quantitative metrics of engineering knowledge as a vehicle for managing development. We chose to base AP on theories of intelligent activity as a social process. Focusing on the social foundations of activity de-emphasized the methodological differences between design and engineering, and helped us to build on the common patterns of communication that already existed in our community.

We began with a study of current practices in the organization. We distributed a questionnaire asking our colleagues about the tools they used in their work. It is interesting to note that response was high (over 50%) even though this was not a management requirement. We believe this is because the survey came from other

staff members who shared their desire to preserve our culture while addressing process requirements. This survey revealed a sophisticated use of tools to manage source code, to automate tests and builds, and to track bugs. As would be expected, there was less automated support for fieldwork, interaction design, usability and other aspects of design. Also as we expected, there was little use of formal requirements management tools, with developers using text documents, screen sketches, rapid prototypes and other representations for design definition. Given this high-level of tool use, and what was already an informal integration of design and engineering activities, we made an early commitment to documenting existing practices, rather than imposing discipline on them. This "build the sidewalks where the grass is worn" approach quickly became the dominant constraint on our work.

In order to give expression to people's informal practices, we drew on a variety of social models of work. Although space precludes a more detailed discussion, theoretical elements we built on include:

- *Embodied cognition* [8-10] holds that intelligent activity depends fundamentally on elements of our biological embodiment, including perception, emotion, aesthetics, and innate learning biases. This led us to recognize that much of skills a process must support are implicit in people's experience and social context. Embodied theories challenge the common goal of making implicit knowledge explicit with evidence for the deeply embodied, concrete, situational nature of practice. This led us to avoid the common goal of trying to make this implicit knowledge explicit in the process.
- *Situation-action theory* [2] was important in emphasizing the role of improvisation in carrying out plans and procedures. It led us to avoid constraints on the order of activities and focus on the products (code, models, documentation) these activities produced.
- *Activity theory* [11] not only reinforced this focus on the object of work, rather than specific behaviors, but also emphasized the importance of tools in mediating collaboration. In leading us to look at tools as mediating collaboration, rather than simply as easing work, this led to a key element of AP: the use of a wiki to moderate communication and documentation alike (see below).
- *Distributed Cognition* [12] further reinforced our approach by emphasizing information artifacts as the focus of understanding collaborative work.

## Details of Activities & Practices (AP)

Taking social models of cognition as a foundation for our process, rather than the formal/rationalist model underlying traditional software engineering, provided a common foundation for managing design and engineering. It also influenced our process in more specific ways. The key structural commitment was to work at a higher level of abstraction than most software processes. We defined two major elements in our process: activities and practices. Activities are relatively self-contained units of work we provide customers. Each activity has a set of practices associated with it. As a first approximation, we can think of activities as "what we do," and they are mainly characterized by the product they produce for our customers. In contrast, practices define "how we do it." Practices do not have products in the sense of something delivered to a customer, but contribute to creating the activity's product. However, practices do produce various artifacts and require documentation.

An example of an activity is "Fieldwork and Problem Analysis." The object of this activity is a documented understanding of the customer problem and its context. Practices that support it include qualitative interviews, observational studies, focus groups, and problem analysis. Similarly, the activity "Project Management" has as its object the documentation of development work for customers and our own management. Practices include planning, budget, issue tracking, etc. The activity "Iterative Software development" produces code as its object and includes such practices as architecture and algorithm design, design and code reviews, unit & regression testing, and frequent integration. There are a total of ten activities with an average of five practices each.

This higher-level focus enabled us to integrate design and engineering easily. For example, the activity "Interaction, interface and functional design" has as its product a product definition expressed through requirements, scenarios, storyboards and information models. Note that we use the same framework for both engineering and design. Design and Engineering are both activities; AP makes no structural distinction between them, leaving these specifics to the individual developer. Note also the de-privileging of requirements: although they are an object of the design activity, they are treated on the same level as scenarios, storyboards and other less formal design representations.

Another feature of AP is its support for improvisation. This comes from two features: 1) individual project teams specify which activities and practices they will implement. Generally, this is the result of discussion between teams and customers. 2) There is no order on the conduct of practices. Individuals and teams can carry them out in any order that makes sense in context. This idea, drawn from situation-action theory allows maximum flexibility while affording standards on tools and delivered artifacts.

Perhaps the most important aspect of AP is its emphasis on tools and media. Our goal was to eliminate after the fact documentation and capture all relevant information as a by product of collaboration. To this end, we use a wiki-based project management tool, Trac, for all project interactions. This includes meeting notes, management of design documents, and assignment of project tasks. Tickets not only assign software tasks like bug fixes or new features, but also track design ("interview Jane Doe") and project management tasks.

An important feature of AP is the use of the Trac tool to capture meeting minutes and assign tasks in real time. Project meetings, generally held weekly, use a networked computer and projector to display a new Trac-wiki page for the meeting. As decisions are made, the project leader documents them on the wiki. Since everyone participates in both the decision and its documentation, later misunderstandings are reduced. Finally, the wiki and associated tickets provide a history of the project for corporate and other requirements.

**Example: The Integrated Stockpile Evaluation (ISE) Project**

Our customer, the ISE team is responsible for overseeing a large number of projects involved in maintaining the United States' weapons stockpile. Their responsibilities include planning, funding, tracking and reporting on these diverse efforts. Although they had a

process for doing so, they wanted to both automate and improve it. We were brought in as software developers, and also participated in their process redesign. One of this paper's authors (Carson) was Project Manager (PM) on this effort for its duration; the other (Stubblefield) worked as a designer in the early stages but left as it moved into implementation.

One of our motivations for developing AP was the PM's need for a more consistent and automated way to manage, track and oversee projects. She was responsible for 6 projects plus miscellaneous teams to track proceedings for at some level. All had similar needs that included project meetings, report, partner status, management status, budget and financial tracking, team resources and general information. She needed communications to be in one place for each project and ease of use with consistency for these projects. When the project began, AP was in its conceptual stages, so this effort became both a focus and test for the effort.

From the beginning, the PM used a tablet PC to document the details of the meeting as it occurred, and projected these results for the participants. She captured details such as the date, time and place of the meeting; who attended; the agenda or purpose; the body of the meeting and finally the action items. After each meeting, the report would be sent via email to the team. This approach effectively made our customers partners in a participatory development effort.

In keeping with our goal of basing AP on what had proved successful in practice, we retained this approach as a project management practice we called real-time, media-based (RTMB) documentation. At the suggestion of a colleague, we adopted the Trac Wiki and Subversion tools as a common framework for this. Nothing really changed about how meetings, reports, status, etc. were conducted for any of these projects with the exception of the PM.

As we developed the AP, we chose the ISE project to "pilot" it using the new set of tools, because it was the basis of much of the process design. Interactions with the partners of this project were on a weekly basis. Everyone discussed and reviewed the week's work. We negotiated and agreed upon the next week's goals, which we captured in an "as-you-go" fashion. A benefit was that the entire team participated, clarified, asked questions and owned any action items created during these meetings. This avoided later confusion.

We recorded action items on Trac tickets for documentation and tracking purposes. When the tickets or action items were entered into the system during the meeting, a team member would take ownership of the item. It would then be waiting for them in their email inbox at their office. The team defined all types of requests or items tracked: requirements, changes, bugs, acceptances, milestones, etc. Other meetings occurred as needed, for example, between the programmers and one or more of the partners. Although the PM was not present, these meetings were conducted in the same fashion.

## Conclusion

In retrospect, the most valuable decision we made was to avoid imposing hierarchy or process on the effort and to let the team's collaboration evolve naturally and to use this to define the process. Although everyone recognized the need for formality in project

management, this evolution kept an emphasis on communication.

The strengths of this real-time approach to documentation are that it is written, visual, verbal and participatory. Because everyone saw what was being captured, interpreted and discussed as it happened, trust and ownership within this team, spilled over to the relationship with the partners and visa versa, and over time, open and honest negotiations occurred naturally. This approach, of developing AP while our partners developed their own process, and we developed the software proved fortunate. Comments from the partners: "I think the ISE project is already going very well…I really like the iterative method we've been using this year due to all the changes in requirements…"

Having formalized this in the Activities and Practices definition, we have applied it successfully to other projects. In keeping with the multidisciplinary nature of our organization, we have used AP on development, assessment, and pure design projects alike. We have shared these ideas with our colleagues, and adoption is beginning to spread. Colleagues who have not adopted AP use similar methods and tools. We are negotiating refinements to incorporate their ideas into future versions of the standard.

## Acknowledgements

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94aAL85000.

## Citations

[1] Forsythe, D.E., *Studying Those Who Study Us: An Anthropologist in the World of Artificial Intelligence*. 2001, Stanford, CA: Stanford University Press.

[2] Suchman, L.A., *Plans and Situated Actions: The Problem of Human Machine Communication*. 1987, Cambridge: Cambridge University Press.

[3] Beck, K., *Extreme Programming Explained: Embrace Change*. 1999: Addison-Wesley.

[4] Cockburn, A., *Crystal Clear: A Human-Powered Methodology for Small Teams*. 2005, Boston, MA: Addison Wesley.

[5] Highsmith, J.A.I., *Adaptive Software Development*. 2000, New York: Dorset House Publishing.

[6] Carroll, J.M., *Making Use: Scenario-based design of human-computer interactions*. 2000, Cambridge, Mass: MIT Press.

[7] SEI, C.M.U.S.E.I., *The Capability Maturity Model: Guidelines for Improving the Software Process*. 1994, Reading, MA: Addison Wesley.

[8] Barkow, J.H., L. Cosmides, and J. Tooby, eds. *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*. 1992, Oxford University Press: Oxford.

[9] Lakoff, G. and M. Johnson, *Philosophy in the Flesh: The Embodied Mind and Its Challenge to Western Thought*. 1999, New York: Basic Books.

[10] Varela, F.J., E. Thompson, and E. Rosch, *The Embodied Mind: Cognitive science and human experience*. 1991, Cambridge, Mass.: MIT Press.

[11] Nardi, B.A., *Context and Consciousness: Activity Theory and Human-Computer Interaction*. 1996, Cambridge, Mass.: MIT Press.

[12] Hutchins, E., *Cognition in the Wild*. 1995, Cambridge, Mass.: MIT Press.