

V5.1 as of 11/10/00 (3teams cacm 00.wpd)  
(3999 words)

## **Software Development Teams: Three Archetypes and their Differences**

Steve Sawyer  
School of Information Sciences and Technology  
The Pennsylvania State University  
513 Rider I Building  
University Park, PA 16801  
(o) 814-865-4450  
(f) 814-865-5604  
[sawyer@ist.psu.edu](mailto:sawyer@ist.psu.edu).

Please do not cite or quote without my permission

Earlier portions of this paper were presented at the 1999 OASIS workshop in Charlotte, NC and at the 2000 America's Conference on Information Systems ( as Sawyer, S. (August, 2000) "A Social Perspective on Software Development: Three Models," *The 2000 America's Conference of the Association of Information Systems.*)

# Software Development Teams: Three Archetypes and their Differences

## **Abstract** (for reviewers):

In this paper we describe the three archetypal social structures embedded in the sociotechnical activity called software development. We do so to pursue two questions: What can we learn about software development by focusing on its social aspects and what insight does a social perspective give us regarding the tasks, production methods and tools used in software development? For each of three models of social structure we outline, compare, and comment on issues with the way the software development tasks, methods and tools are conceptualized. We include a brief discussion of hybrid models such as those used at Microsoft and other packaged software vendors. The paper concludes with a discussion of the implications of these models on the relationship between team structures and tasks, development methods and development tools.

## **Structure of Paper** (for reviewers):

The Socio-technical nature of Systems Development

Three Archetypes of the Social Structures of Software Development Teams

    The Sequential Archetype

    The Group Archetype

    The Network Archetype

Hybrid Forms

Insights and Opportunities

    Sequence Perspective

    Group Perspective

    Network Perspective

Privileging the Social Perspective

References

Sidebar on conceptual bases for archetypes

## **Software Development Teams: Three Archetypes and their Differences**

What can we learn about software development by taking a social perspective? By social perspective we mean explicitly focusing on the way a work group's members organize themselves and interact in ways that both respond to and shape the production methods and tools they use and the tasks which they are charged to perform. A social perspective on software development differs from the more traditional production perspective by highlighting the social milieu in which the work done to make code takes place. In the production perspective, social action is often implicitly or explicitly discussed as an outcome of production. A social perspective reverses this and highlights how production methods, techniques and tools are enmeshed in and enacted through the structures and interactions of the professionals who work together to build software. In taking a social perspective we focus on the social unit (as an aggregate of people), not an individual perspective on software development.

The paper continues in five sections. We first explore the socio-technical nature of software development to highlight the interdependencies between the social and technical (or production) aspects of this effort. Then we outline three archetypes of software development teams that a social perspective helps to illuminate. By archetype we mean an idealized form that is premised on an internally consistent set of assumptions. A software team archetype represents the implicit beliefs about human behavior and work organization that structure the explicit activities of development. In the third section we discuss some of the issues and insights that arise from comparing these three archetypes of software development. In section four we discuss common hybrid forms of development. We conclude with a discussion of the different software team archetypes's implications for tasks, development methods, and software development tools.

### **The Socio-technical Nature of Software Development**

We conceptualize software development as a set of activities comprised of people interacting with each other and with (primarily information and communication) technologies (ICT) that they use to do their work. That is, software development is a socio-technical activity (Bijker, 1995). Characterizing software development as a socio-technical activity helps to highlight the analytic distinction between how people work and the technologies they use. Social aspects of software development include how people interact, behave and organize/structure. Technical aspects of software development include the use of production methods, techniques and tools. However, in practice it is very difficult to disentangle the way people do things

from the technologies which they use. Nor is it easy to gain insight into the use of a technology if it is removed from the social context of its use. That is, while the social and technical aspects of a socio-technical system can be explicitly characterized, they are mutually interdependent.

This mutual interdependence suggests that it is difficult to understand either aspect independently. Interdependence also suggests that different social contexts will shape the use of the same technical artifact in differing ways (Sproull and Goodman, 1989). One example of this is variations in use of computer-aided software engineering (CASE) tools among software development teams (Guinan, Coopriker and Sawyer, 1997). This view often contrasts with the more common perspective on software development which first focuses on the means of production (methods, techniques and/or tools) - a techno-social approach. Both perspectives on software development implicitly or explicitly embed certain views of the social aspects of this effort into the production view. For example, the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) focuses primarily on the production aspects of software development. The People-Capability Maturity Model (P-CMM) complements this by explaining how people can best change their behaviors to fit the CMM approach. Together, the CMM and P-CMM reflect the traditional production first, people second, approach to software development. In contrast, in this paper we ask: what can we learn about software development by privileging the social perspective?

### **Three Archetypes of the Social Structures of Software Development Teams**

We present three generic archetypes of software development teams: the sequential, the group and the network (each of these are defined in Table 1). The conceptual bases of these three archetypes are highlighted in the sidebar. As archetypes, these represent idealized views. We use them to help understand the more practical issues of hybrid models of software development: models where elements of different archetypes are combined. In the rest of this paper we address two questions that arise from this characterization. Firstly, in what ways do these archetypes differ? Secondly, what insight can these differences provide? Responding to the first question, in Table 2 we present a summary of the contrasts among these three social process archetypes of software development and discuss each archetype in the following sub-sections.

**Table 1: Three Archetypes of the Social Processes of Software Development**

Archetype	Definition
Sequence	Software development is a production effort based on a linear set of discrete tasks. People work in specialized functions with formalized interactions across functions. People are valued for their particular specialized skills.
Group	Software development is seen as a combination of development and production where a set of discrete tasks may need to be repeated until the product is complete. Developers are organized into interdependent groups and are valued for both their particular skills and for their ability to work with others.
Network	Software development is seen as a process of constant development with a specific focus on the outcome/product. Tasks are not seen as sequential and are also tied to individuals (or small groups) whose participation is based on interaction. Group members are valued for what they can produce. This implies a complex net-work of ties between people and a hub-and-spoke management approach.

**Table 2: Aspects of the Three Archetypes of Software Development Team Structure**

Archetype Aspect	Sequence	Group	Network
Perspective	Process first	Process first	Product first
Belief mode	Control	Conflict	Interaction
Orientation	Prescriptive	Normative	Descriptive
View of task	Production	Production & Development	Development
Implied Method	Linear & sequential	Iterative & sequential	Emergent & non-linear
Tie to context	Prescribed boundary	Permeable boundary	Embedded
People's actions	Prescribed	Role & goal driven	Individual& linked
Examples	SDLC, SEI/ CMM	Spiral, RAD, JAD	Open source, Chief programmer

## The Sequential Archetype

The social structure embedded in the sequential archetype of software development is linear and supposes a known and pre-specified ordering of the requisite tasks. This social structure is seen as independent of the larger social and organizational context. The social structure is also hierarchical, with little intra-functional discussion needed (save through formal channels). Within a task, each person's role is discrete, specialized and identifiable. A pre-specified task ordering further implies a prescriptive view of the production process. The sequence archetype explicitly enacts the belief that a good

process leads to a good product.

The social interactions in a sequence archetype of software development are based on the concepts of control. That is, people's interactions are seen solely as a function of the work they do. Thus, this work can be more easily measured and differences analyzed and the roles are stable. The task/specialization orientation also suggests that any one member can be replaced as needed by another person with the same functional level of skill.

The work emphasis is on embedding the required information in the work product and/or associated documents to pass on to each downstream task. This implies that the required inter-team and extra-team interactions can be defined, formalized and perhaps automated. These characteristics suggest that a sequence team archetype would benefit from automation: for discrete tasks (such as coding or testing) capital in the form of computer systems can replace (or at least augment) labor. The control orientation, formalized interactions among team members, and push for automation suggest that there is very little need to create strong social bonds. Examples of the sequence archetype include the traditional waterfall (such as the systems development life cycle or SDLC), the CMM and the SPICE approach.

## **The Group Archetype**

The social structures in the group archetype reflect an interactive and collaborative effort. In this archetype, software development processes are based on a set of predefined tasks but this must also take into account (and build on) the individual skills and weaknesses of the group's members. The tasks, while often sequential, are seen as iterative and there is explicit attention to process improvement by the members of the group. The group archetype also explicitly recognizes, in its iterative nature, that software development and production are often intimately linked. Thus, a group archetype is normative. Further, a group archetype also implies that the boundary between the team and the social context is permeable and that boundary-spanning (to key stakeholders, say) is an important part of development. Still, the implicit belief reflects the process-first approach also present in the sequential archetype.

A group archetype of software development makes explicit the need for social interactions among the group's members. Social structures are oriented to resolving the inevitable conflicts that arise from people collaborating. This means that there is the potential for *partial*-automation of production tasks and that additional tools and methods are needed which explicitly support and/or enable collaboration among the group members. Examples of group archetypes are the spiral/ evolutionary

approach to software development, rapid application development (RAD) and joint application development (JAD).

## The Network Archetype

The social structure of the network archetype is premised on the collection of ties among the members of a network . These ties reflect the frequency and value derived from interaction. Stronger ties reflect common interests, significant information sharing and high levels of interaction. Weaker ties reflect less frequent interactions and different forms of information transfer. In the network archetype, the development process typically emerges and reflects the network ties developed by the participants. Unlike the other two archetypes, the product is the central focus and production process are secondary. The network archetype implies that the individual members, and the social ties that connect them, define the software development effort. Further, this network is fully embedded in a larger social context, a context that may not easily map to any organizational or geographic boundary.

An emergent process is constrained by tasks but these constraints are a function of the social ties that create the social network. So, even if the process is not central, there is some form of version control, of testing and of documentation. One belief underlying the network archetype is that a good product comes from having good people. This people-first approach recognizes that it is very difficult (if not impossible) to replace key members (or a member) of a network as they represent important hubs. These key people provide the connections that define the network.

To support the network it becomes critical that the software development tools provide for interconnection. Simply, any software development tool is valued for how it helps the individual member and/or for how well it enables sharing among the network. A second implication is that a network archetype, being both emergent and descriptive, reflects a developmental view (as opposed to production) of software. Further, given the centrality of the social structures and individual members interaction to a network archetype, the group's effort are often contentious (Zachary, 1994). From this perspective, the interactions between the members are focused on product features, functions or actions. There are few procedural details and the expectations among the members is "show and tell." That is, the resolution of disagreements is often rooted in providing code that delivers on the concepts discussed.

The chief programmer team model of software development (Baker, 1972) is one example. The network structure is hub-and-spoke: strong ties between members and the chief programmer and weak ties between individual team members. The

recent growth of open source development efforts (Raymond, 1999) reflects a second form of the networked group archetype. This network has multiple hubs (of varying importance) and multiple ties among many members of the network.

## Hybrid Models

A more typical scenario is that an actual development team will adopt a hybrid social structure, drawing elements from several archetypes. For example, both Baker (1972) and Brooks (1975) write about the IBM System 360 development effort. Brooks highlights the importance of a project development structure (essentially the SDLC) while, as we said above, Baker explains how the chief programmer team works. The sequential archetype advocated by Brooks provides a stylized view of development while Baker's view provides insight into how the chief programmer can create a hybrid social structure underlying the sequence of the SDLC.

One proposition that arises from this is: *for software development to work, a hybrid network structure of relations is needed among the developers*. Existing evidence suggest that this may be true. For instance, Weinberg (1971) noted how developer productivity fell when vending machines were removed from programmer's break area and they, subsequently, spent less time interacting informally with each other. Removing the machines deprived the developers of their reliance on the informal network that had emerged around the vending machines.

Carmel and Sawyer (1998) document how packaged software (software made for sale/license to others) development approaches differ from more traditional custom/internal software development. They note that developer interaction is greater, there is less reliance on formal processes, and more reliance on product construction. Both Cusamano and Selby (1997) and Zachary (1994), in their studies of Microsoft's development practices, highlight that a product focus allows for internal competition among developers. In essence, the Microsoft development approach is a hybrid between a group and network archetype.

From this overview come two additional propositions: *for each development effort, a small number of critical people (the "core team") serve as the hubs in the evolving network of programmers. And: as developer networks become more formally structured, issues of conflict and interdependence (aspects of the group archetype) become increasingly important*. Sawyer (2000) highlights these issues relative to packaged software developers but provides little evidence.

## Insights and Opportunities

Responding to the second question we discuss some of the insights and opportunities that a social perspective on software development provides. Table 3 summarizes this discussion.

**Table 3: Insights and Opportunities**

Insight \ Archetype	Sequence	Group	Network
<b>Team/Task Issues</b>	Cohesion	Consensus	Contribution
<b>Opportunities</b>	Cross-training, Process management	Teamwork skills Conflict management	Evaluation Project management
<b>Method Issues</b>	Adherence	Regulation	Repeatability
<b>Opportunities</b>	Components Feedback mechanisms	Connections Iteration controls	Taskings Interdependencies
<b>Tool Issues</b>	Integration	Collaboration	Connection
<b>Opportunities</b>	Automation Process control	Shared tools Process support	Interoperability Interaction support

**Sequence Perspectives** Given the degree of task specialization and decoupling between tasks, the sequence archetype's work is supposed to be routine. However, this can make it hard for participants to see the value of their individual contribution to the whole. This decoupling can also lead to limited interaction with other members and further reduce the likelihood that project team cohesion will develop. A lack of cohesion shows up in projects where analysts won't speak with developers and testers stand independent of the rest of the team. Typical approaches to alleviating this are cross-training (for multiple roles) and more management intervention (through formal meetings and walkthroughs).

The sequence archetype is predicated on method adherence. This is often difficult since the process orientation and related task specialization make it easy for an individual member to ignore cross-task work (such as documentation) as the penalty of poor performance shows up in a different part of the process from where it was introduced. For instance, bad requirements do not (directly) affect the analyst, nor does bad code (directly) affect the programmer. This concern for adherence leads to opportunities for techniques to help ascertain between-task adherence. Examples of these techniques include walkthroughs, check-lists and sign-offs. Often techniques such as walkthroughs serve two purposes: a forum for management intervention and a means for cross-assigning members of one task to be part of another.

Since the sequence archetype is premised on routinization, automation is often a goal. This trend is seen in the emergence of integrated development environments and full suite CASE tools. Additional tool development would be to focus on embedding process control into the tool suites to help reinforce method adherence.

**Group Perspectives** The group archetype recognizes the importance of team-member interaction and is predicated on developing consensus among team members. Combined with the cyclic and integrated nature of the production tasks, this suggests team members must be trained and supported in learning group process and conflict management skills. That is, in the group archetype, the task/team issue is to improve member's teamwork skills. This is one reason why total quality management approaches focus on improving the social skills of the participants. They do so in order to make effective use of group efforts.

From a group perspective regulating the iterative nature of the project is important. That is, how do teams know when to stop iterating? The most blunt form of iteration control is money (or some other resource constraint), though others may be more valued (such as user feedback or functional compliance assessments). Another opportunity for method development in the group archetype would be to provide cross-iteration task links such as change and version tracking, release control, and release planning.

The group archetype would value tools support team-member collaboration. This suggests developing shared tools that allow for group access. The proliferation of Lotus Notes' databases to support software development teams exemplifies this attention. Further, the differences between sequence and group archetype perspectives on CASE tool use are highlighted in the ways they support collaboration (Vessey and Sravanapudi, 1995).

**Network Perspectives** In this archetype it is often difficult to reward contributions without stifling links.. Given the interdependent nature of the work combined with the individualized nature of the way the work is done, evaluating contributions is often outcome or deliverable-based. Such an approach demands strong project management. Often this project management seems centralized in one or in a small set of hubs. Thus, the network archetype, dispersing work actually seems to concentrate management. That is, the control of the product is with a person (hub) and, while this control may shift over time, it is rarely shared (Raymond, 1999). This also implies that a member of a network can choose to leave the effort if their contributions are not being rewarded.

The network archetype starts with an idiosyncratic set of individuals linked together in an evolving set of social ties. In this environment one method issue is repeatability and stability: the effort to ensure a common process for repetitive activities

(such as task assignment, progress reporting, issue tracking, etc.). And, while they are issues with all forms of software development, they are particularly central to the network archetype (and its reliance on social interaction as the dominant form of structure). One method opportunity for network archetypes is automated tasking mechanisms. This is, in essence, a public project tracking board. Such a public tracking mechanism would also help to maintain and track the interdependencies among the members of the network. Another issue is the move to outcome measures such as “does your module run and does it interact with the larger product?” Outcome-oriented approaches are common aspects of the Microsoft models of development (Zachary, 1994; Cusamano and Selby, 1995).

The interdependent nature of work in a network archetype means that the tools to support this approach must provide for interoperability and interaction. That is, in the from a network perspective, it must be easy (if not seamless) to share files and even to pass useful utilities and tools. This helps to explain why stable, free and well-known platforms (such as Unix/Linux) are the base for much open source (network archetype) development (Raymond, 1999).

## **Privileging the Social Perspective**

Taking a social perspective on software development helps us identify three general archetypes of the social organization. Comparing these three archetypes, in turn, leads us to insights on the socio--technical act of software development. And, given the predominance of the production-focused perspective of software development, a social focus provides a means to explore the current literature’s findings from a different perspective.

The differences between these three archetypes of software development leads to asking which is the best approach? This is both an empirical and philosophical question and demands more attention, certainly beyond the limited space of this paper. However, we do provide evidence that hybrid social structure archetypes exist and that these archetypes can be decomposed into some combination of the three base social structure archetypes. Further, this analysis of the social structures of software development suggests that the socio-technical nature of this effort is often under-represented by focusing on the technical (or production) side.

Moreover, a social perspective of software development deeper insight into the relationship among methods, techniques, tools and the people’s use of these. For example, the brief analysis in this paper suggests that most software development efforts are a hybrid of the three archetypes. Given that, it is clear that informal social networks are important to developers,.

Further, certain members of each development effort become “hubs” and are not easily replaced. Further, that a belief in a sequence approach may understate the iterative nature of software development and falsely raise expectations that people can be added and removed from a project with impunity. The iterative nature and complexity of social interaction implied in hybrid models also suggest that software development tools and methods which both enable collaboration and support production would be highly valued.

## References

1. Baker, F. Chief Programmer Team Management of Production Programming. *IBM Systems Journal*, 11(1), 1972, 56-73.
2. Bijker, W., "Of Bicycles, Bakelites and Bulbs: Toward a Theory of Socio-technical Change," MIT Press, Cambridge, MA, 1995.
3. Brooks, F., The Mythical Man-Month, *Datamation*, 1974. 44-52.
4. Carmel E and Sawyer S. Packaged software development teams: What makes them different? *Information Technology & People*, 11(1), 1998. 7-19
5. Cusamano M and Selby R How Microsoft builds software. *Communications of the ACM*, 40(6), 1997, 53-61.
6. Guinan, P. Coopriider, J., and Sawyer, S., "The Effective Use of Automated Application Development Tools," *IBM Systems Journal*, 36(1), 1997, 124-139.
7. Raymond, E., *The Cathedral and the Bazaar*, Available online at <http://www.tuxedo.org/~esr/writings/>, 1999.
8. Sawyer, S. (2000) "Packaged Software: Implications of the Differences from Custom Approaches to Software Development," *European Journal of Information Systems*, 9(1), 2000, 47-58.
9. Sawyer, S., Farber, J. and Spillers, R.. Supporting the social processes of software development teams. *Information Technology & People*, 10(1), 1997, 46-62.
10. Vessey, I., and Sravanapudi, P. CASE Tools as Collaborative Support Technologies. *Communications of the ACM*, 38(1), 1995, 83-95.
11. Zachary, G. *Showstopper: The breakneck race to create Windows-NT and the next generation at Microsoft*. New York: The Free Press, 1994.

### Sidebar: Conceptual bases of the three archetypes

The conceptual basis for the sequential team archetype of software development team social structure comes from the work design tradition in industrial engineering (e.g., Nadler, 1963), based on work first laid out by Frederick Taylor (and often known as scientific management). Work is seen as a set of discrete tasks that can be measured.

The group archetype draws its intellectual roots from theories of social psychology and/or small group theories such as “work redesign.” (Hackman and Oldham, 1980). Work redesign arose in response to issues such as personnel motivation, retention and productivity that typically arose in a work design approach. Work is seen in a more holistic and integrative sense, and specific features of tasks can be identified.

The networked group archetype draws on concepts of social network theory (Granovetter, 1982; Wellman, et. al., 1996). In this archetype a group of people are linked by the relative “strength” of the social ties among them. Work is seen as the use of ties to deliver and receive information and these uses both span and define tasks.

See:

Granovetter, M. The strength of weak ties: A network theory revisited. In N. Lin and P. Marsden (Eds.), *Social structure and network analysis*, Sage: Beverly Hills, CA, 1982, 105-130.

Hackman, J., and Oldham, J. *Work Redesign*. Addison-Wesley, Boston, 1980.

Nadler, G. *Work Design*. Richard D. Irwin, Homewood, IL, 1963.

Wellman, B., Salaff, J., Dimitrova, D., Garton, L., Gulia, M., and Haythornthwaite, C. Computer Networks as Social Networks: Collaborative Work, Telework and Virtual Community. *Annual Review of Sociology*, 22, 1996, 213-238.