Brady Field

On Building Software Process Models Under the Lamppost

**Important points**

“We conducted a field study of large system development projects to gather empirical information about the communication and technical decision-making processes that underlie the design of such systems. The findings of this study are reviewed for their implications on modeling the process of designing large software systems. The thesis of the paper is that while there are many foci for process models, the most valuable are those which capture the processes that control the most variance in software productivity and quality.” [96]

“The fundamental problem is that the techniques and tools that underlie alternate approaches to the development process must be substantially more powerful to offload from project personnel the burden of managing the complexity of large systems development. As the size of the system grows from programming-in-the-large to programming-in-the-gargantuan, the factors that control productivity and quality may change in their relative impact.” [97]

“If software process models are to offer more than illusory comfort to managers that the project really is under control, then we must focus them on something other than phase-ending events and activity descriptions that are useful when there is little uncertainty. It is our primary thesis that this focus should be on the activities that account for the most variation in software productivity and quality. If we can represent these processes, then it may become easier to know how to improve them and use them as indicators of project status.” [98]

“It is not surprising that we observed substantial differences in individual talents and skills on the projects that we surveyed. Project and division general managers consistently commented on how these differences related to project performance.” [99]

“In an idealized model of the team process leading to the creation of a design, the process would begin with each team member holding their own model (a mental representation) of what should be done. These mental models could differ on their representation of the behavior the application system should exhibit, the structure of the environment in which it would operate, the computational model most appropriate for creating the behavior expected of the application system, etc. In the next stage of the group process, individuals sharing similar models would form coalitions to argue for their point of view. In the final stage, the differences are resolved between the conflicting models of various coalitions and a team consensus is reached. This process is similar to the group process for Japanese projects described by Belady (1986).” [100]

“Belady (1979) and Brooks (1986) argue that change is inherent in software projects. Internal company environments change due to new organizations, technologies, personnel, policies, procedures, etc. External environments change due to economic factors, market conditions, political events, etc. Change stimulated from within the development organization appeared easier to manage, because the parameters of its effects are easier to determine. External change was more perplexing, because the context for technical decisions had changed at the same time that the project was struggling to maintain consistency among the decisions already made.” [101]

“Current software process models are prescriptive in describing how the software development process should proceed. However, if the variability of results within a process phase is large, then greater visibility into the activities and events underlying the process is needed. We are suggesting a different type of process model for representing software development at a deeper level of understanding than has been characteristic of most management accountability models. These models will initially be descriptive. However, they should provide empirically-based recommendations for tools and methods that are needed to augment the development process. If new technology does not affect the factors controlling the greatest variance in project outcomes, its benefit will not show up in productivity and quality results.” [102]

“Existing models of the software development process do not provide enough insight into actual development processes to guide research on the most effective development technologies.” [103]

**Disagreements**

“The major shortcoming of the waterfall model was that it failed to treat software development as a problem solving process. Not only is the developer attempting to solve the problems presented by the stated requirements and the constraints of available technology, but customers are also trying to solve a problem for which they believe the requirements will yield a solution. Yet, since customers often don't understand the subtleties of their problem, and even more often don't understand the limits of technology, software development becomes a problem solving process involving multiple agents.” [96]

The first sentence states that the Waterfall method does not solve problems that arise in development. The authors then state that the customers don’t properly understand the problems they are attempting to solve, and this is something the Waterfall method is not equipped to deal with. While customers may not understand their needs completely at times, the Waterfall method clears these up before any development begins. Waterfall is very strong in this area. The entire project must be designed and documented before it can be implemented and delivered. A customer not understanding a problem properly will lead to complication, but this is not something specific to Waterfall. This would be methodology-independent and therefore not solvable by simply implementing something other than Waterfall. This is a poor argument of the problem the new approach to methodologies is presented to solve.

**Questions**

I understood everything in the article.