UNIQUE CHALLENGES OF SOFTWARE PROJECT MANAGEMENT

Software development is one of the most unique engineering efforts undertaken by humans. The raw materials are based on ideas and concepts, which by their very nature are fluid and often ill defined. The first step in the development process consists of capturing and converting those ideas to text and diagrams. These items, or "artifacts," are models meant to interpret and represent requirements. But no model is ever completely accurate in representing requirements. Models inherently reflect some vision of "reality" by exposing some details about that reality while suppressing others. Next, these models are given to designers and developers who interpret them in an intangible "virtual" world where few physical constraints exist that would limit how models are implemented. Software development therefore proceeds from intangible requirements reflected by imperfect models to design and construction in almost any way conceivable. All this in an engineering discipline supported by incomplete and often debatable methodologies!

In a series of articles to appear over the next few months, we confront the possible misconception that software development can simply be managed like any other engineering or manufacturing project. In this first installment, we introduce at a high level the key aspects and unique challenges of software development. Subsequent articles will explore the implications of those characteristics and the unique demands they place on software project managers. Later articles will explore detailed aspects of specific tools and methods, such as Rational Unified Process (RUP), Extreme Programming (XP) and other "agile" methods, and the added value they bring to general project management knowledge, such as PMBOK®.

The following is a list of key challenges and characteristics that make software development unique and distinct and its management potentially more difficult than other types of engineering.

1. Software is irregular, intangible, invisible.
Software cannot be directly "experienced" in the physical world, using any of the five senses. It does not abide by laws of nature. Progress is difficult to see, experience and measure. Software project managers must often rely on indirect evidence of progress. This frequently means resorting to asking team members for their perceptions and other highly subjective measures of progress.

2. There is little agreed upon guidance or standard process.
Civil, mechanical, electrical and other engineering fields have painstakingly developed guidelines and handbooks that capture proven processes. This is not true for software development. Despite recent progress, we cannot reliably predict when a specific process or methodology is likely to lead to project success, or conversely, cause development problems. There is little consensus and few standards on how software development should be carried out and managed.

3. No two parts or two systems are alike.
Software projects are often unique. Even within a single industry or within a single organization, project managers can be asked to manage a wide variety of projects using very different technologies and tools. Neither software engineer nor manager can assume that specific expertise they established on one project will be applicable, or even relevant, to the next.

4. Software construction often requires innovation.
Software systems are often new and technically innovative. This poses tremendous challenges in development and management, leading to known software project symptoms, such as cost and schedule overruns and a final product that falls short of meeting customer needs and expectations. Interestingly, innovative projects in an established engineering field (such as a new transport system) often exhibit and suffer from similar difficulties and exhibit similar symptoms.

5. The myth that software is malleable.
Because of the flexibility inherent in software development, we have the ability to modify the product very late in the project life cycle, even when it is inadvisable to do so. A common misperception among non-technical managers and customers is that "you can always change the code." Experience proves, however, that change can lead to unexpected side effects and costly adverse impacts.

6. Software construction is human-intensive.
Unlike many other engineering disciplines, the software development effort is largely a manual, human process. Although helpful automation is available in areas, such as code generators and automated testing, these tools still require human effort and technical expertise to set up and execute. The opportunity for automation is minimal in comparison to manufacturing of physical, tangible products.

7. Software application horizons expand with hardware capabilities.
As the underlying physical platform upon which software is built and executed changes, it can have a dramatic impact on the performance and capabilities of the software. Users' perceptions and expectations of software capabilities continue to evolve, thereby placing continued pressure on software projects.

8. Software solutions require unusual rigor.
Software problems are unprecedentedly complex with the potential for seemingly "minor" changes having significant repercussions in other parts of the system. A defect in a single line of code can halt operation of an entire system; this is experienced in military and space software as well as medical and commercial enterprise systems.

9. Scaling up the system causes a nonlinear increase in complexity.
As we move to more complex systems distributed across organizational or functional boundaries, the design of those systems becomes increasingly complex as well. We can no longer rely upon a single system architect to comprehend and design the entire system. The unique difficulty in software is that complexity arises between modules, i.e., in module interfaces and relationships, and therefore increases in non-linear fashion as the number of modules increases.

In summary, one could interpret the marriage of "software" and "engineering" as representing "engineering envy." Software developers, managers, and users alike wish to experience the high level of product success and reliability and project predictability and repeatability commonplace in fields such as automobile manufacturing and civil, mechanical, and electrical engineering.

Software engineering therefore represents a desire, an aspiration to be like other engineering fields, thereby setting highly ambitious, lofty goals for software engineers. In addition to being a younger, less established field, software development suffers from the unique characteristics and challenges described above. It is the unique character of software that project managers must be versed in, indeed immersed in, to successfully manage software development projects.

Hadar Ziv, PhD
University of California, Irvine
ziv@ics.uci.edu

Craig D. Wilson, PMP
IT Management Consultant
craigdwilson@matincor.com
Copyright 2004 Hadar Ziv/Craig D. Wilson