Software Development Myths

Pressman (1997) describes a number of common beliefs or myths that software managers, customers, and developers believe falsely. He describes these myths as ``misleading attitudes that have caused serious problems.'' We look at these myths to see why they are false, and why they lead to trouble.

Software Management Myths. Pressman describes managers' beliefs in the following mythology as grasping at straws:

- *Development problems can be solved by developing and documenting standards.* Standards have been developed by companies and standards organizations. They can be very useful. However, they are frequently ignored by developers because they are irrelevant and incomplete, and sometimes incomprehensible.
- *Development problems can be solved by using state-of-the art tools.* Tools may help, but there is no magic. Problem solving requires more than tools, it requires great understanding. As Fred Brooks (1987) says, there is no silver bullet to slay the software development werewolf.
- *When schedules slip, just add more people* This solution seems intuitive: if there is too much work for the current team, just enlarge it. Unfortunately, increasing team size increases communication overhead. New workers must learn project details taking up the time of those who are already immersed in the project. Also, a larger team has many more communication links, which slows progress. Fred Brooks (1975) gives us one of the most famous software engineering maxims, **which is not a myth**, ``adding people to a late project makes it later.''

Software Customer Myths. Customers often vastly underestimate the difficulty of developing software. Sometimes marketing people encourage customers in their misbeliefs.

- *Change is easily accommodated, since software is malleable.*

  Software can certainly be changed, but often changes after release can require an enormous amount of labor.
- *A general statement of need is sufficient to start coding*

  This myth reminds me of a cartoon that I used to post on my door. It showed the software manager talking to a group of programmers, with the quote: ``You programmers just start coding while I go down and find out what they want the program to do.'' This scenario is an exaggeration. However, for developers to have a chance to satisfy the customers requirements, they need detailed descriptions of these requirements. Developers cannot read the minds of customers.

Developer Myths. Developers often want to be artists (or artisans), but the software development craft is becoming an engineering discipline. However myths remain:

- *The job is done when the code is delivered.*

  Commercially successful software may be used for decades. Developers must continually maintain such software: they add features and repair bugs. Maintenance costs predominate over all other costs; maintenance may be 70% of the development costs. This myth is true only for *shelfware* --- software that is never used, and there are no customers for next release of a shelfware product.

- *Project success depends solely on the quality of the delivered **program***.

  Documentation and software configuration information is very important to the quality. After functionality, maintainability, see the preceding myth, is of critical importance. Developers must maintain the software and they need good design documents, test data, etc to do their job.
- *You can't assess software quality until the program is running.*

  There are *static* ways to evaluate quality without running a program. Software reviews can effectively determine the quality of requirements documents, design documents, test plans, and code. Formal (mathematical) analyses are often used to verify safety critical software, software security factors, and very-high reliability software.