THE MYTHICAL MAN-MONTH

Many software projects exceed their due dates for several reasons. These reasons can be badly developed estimating techniques, effort with progress approach assuming that men and months are interchangeable, manager hesitation about estimations, badly monitored scheduling and schedule slippage.

Optimism: One of the bad approaches is optimism. It includes famous quotes that are “all will go well” and “each task will take only as long as it ought to”. These unsupported ideas are worthless during software development. For example, design or code have some bugs and incomplete components during implementation. Also, tasks have nonzero probabilities of failure. Therefore, probability that all will go well is about zero. The existence of those indicates that optimism is unjustified.

The Man-Month: The second bad approach is about man-month calculation in estimating and scheduling. Progress does not directly relate to man-month calculation. The bad approach states that man and month are interchangeable. This is only true when each task can be divided correctly, and workers have no communication. However, this is not the case in software development since the tasks cannot be divided due to sequential constraints. This means that more effort does not lead to more effect on scheduling. Moreover, training and intercommunication between workers are necessary in software development. When the number of workers increase, intercommunication time increases as well, and communication takes more time when we compare to decrease in task time due to division.

Systems Test: Due to optimism, the developers does not give enough importance to tests regarding scheduling. The writer has stated that scheduling of task should be 1/3 planning, 1/6 code, 1/4 component test and early system test and 1/4 system test, all components in hand. It differs from conventional divisions in a way that while planning, debugging of completed code has larger portion than normal. Due to failure to time for system test, the delay happens at the end of the schedule and no enough action time left to recover this. Furthermore, delay leads to bad financial consequences.

Gutless Estimating: Urgency of boss forces developers to agree to unrealistic schedules. Also, it is very tough to defend an estimate without well-defined estimations. The writer has suggested solutions that are productivity figures, bug incidence figures and estimating rules.

Regenerative Schedule Disaster: Software project may be in behind its schedule. There are four options for manager. The first one is that only the first part of the task was estimated incorrectly and add more men to make difference for first part. The second one is that the entire project estimate is incorrect and add more men for all phases of the project. The third one is reschedule and the last one is trimming the tasks. Furthermore, we can get some insights from regenerative schedule disaster. For instance, adding workers needs retraining, retraining is not in planned and cycle regenerates itself. Thus, according to Brooks’s Law, adding manpower to a late software project makes it later.

To conclude, the study is related to demolish the man-month myth.