The Large System Problem

The TSP Symposium

September 22, 2009

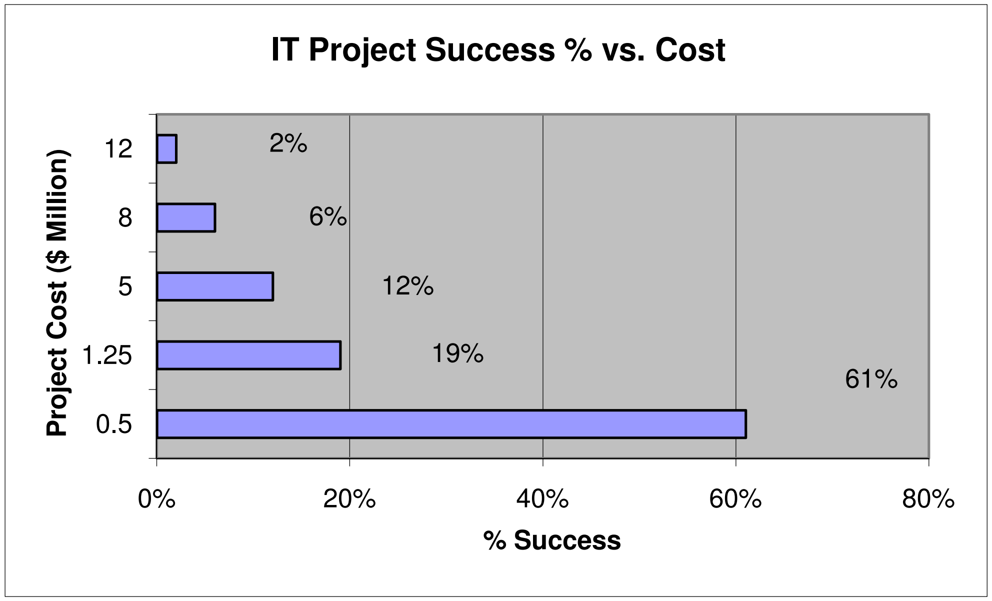
Watts S. Humphrey
The Software Engineering Institute
Carnegie Mellon University

Carnegie Mellon
Software Engineering Institute

Why Can't We Develop Large Systems?

Software development projects often fail.

The larger they are, the more likely they are to fail.



Adopted from the Standish Chaos Report - 2009

The FAA System

The FAA Advanced Automation System

- contracted to a CMM level-3 organization in 1989
- contractor later appraised at CMMI level 5

Costs

- original cost estimate, 1989: $4.3 B
- interim cost estimate, 1994: $6.9 B
- when cancelled, total spent: $2.6 B

Clearly, being CMMI level 5 does not solve the cost and schedule problems.

Other Examples

The IRS system – finally started to use in 2005

- 5 years of delays
- costs exploded to $2 B

FBI system killed

- 3 years late
- $150 M spent
- 5 CIOs, 9 program managers

Clearly, changing managers did not solve the FBI's problems.

This Is a Major Problem

A GAO study of 72 weapons programs

- Projected costs up by 26%
- Development costs up by 40%

The New York Times

- Two thirds of largest weapons over budget last year
- Total extra cost: $296 B
- Programs averaged 2 years behind schedule

Finding a Better Way

For the last 25 years we have tried changing

- managers
- acquisition strategies and regulations
- incentive systems
- auditing procedures

None of these changes has fixed the problem.

There must be a better way!

The Problem

Development work has changed in the last 50 years.

Project management methods have not.

In the past, development work concerned things.
- We produced items one could touch and feel.
- The managers could understand the work by watching it.

These older methods were designed for supervising factory workers.

Traditional Management

These management methods were defined by Frederick Winslow Taylor 100 years ago.

They were designed for

- largely uneducated workers
- relatively simple manual labor

Even though the work and workers have changed, Taylor's methods are still used.

Carnegie Mellon
Software Engineering Institute

Taylor's Management Principles

Taylor's methods rest on three principles.

1. Management knows the best way to do the job.
2. The managers can monitor the work by watching it.
3. The workers cannot be trusted to do good work unless they are watched.

Knowledge Work

Traditional management methods do not work for software because it is knowledge work.

Drucker's definition of knowledge work.

- It involves concepts, ideas, designs.
- It is done in the workers' heads or on computers.
- The workers often know more about the work than their managers.

Knowledge work cannot be tracked and managed by just watching it.

Traditionally Managing Software

With Taylor's methods, the manager's job is to

- define the job
- plan the work
- tell the workers how to do their jobs
- monitor their performance
- correct them when they do something wrong

With these methods, the workers and managers have different objectives.

- The managers want the maximum amount of work for the least cost.
- The workers want maximum pay for the least amount of work.

Observations on Knowledge Work

Knowledge work does not fit the principles for Taylor's methods.

With Taylor's methods, the knowledge workers and managers have different views of project success.

- The workers view projects as successful if they were technically interesting and rewarding.
- The managers view projects as successful if they met their cost and schedule targets.

Since groups rarely succeed when members and managers work to different objectives, software projects keep failing.

It is also why large systems programs are unmanageable with today's management methods.

The Large Project Problem

Large programs are typically composed of multiple small projects.

The small projects can usually be managed with informal and unmeasured methods.

When large programs are managed informally, they typically fail because of

- unanticipated problems
- unmet interdependencies
- the blame culture

Unanticipated Problems

Fred Brooks best described the software management problem.

“Schedules slip a day at a time.”

To do his or her job, the manager must

- know job status
- recover from the daily schedule slips every day
- keep management informed

Anticipating Problems

To anticipate project problems, the managers must ask the knowledge workers who

- have no measures of job status
- see schedule management as the manager’s job

They typically make vague statements like

- “I’m 90% through coding.”
- “Just a couple more bugs and I will finish testing.”

As a result, the manager

- does not know job status
- cannot anticipate problems
- cannot correct problems before they are big enough to see

Unmet Dependencies

In large programs, all the parts must come together to produce a complete operational system.

To meet the overall program schedule, every part of the program must meet its dependency commitments.

Any delay in any part of the work can distort the entire dependency network.

Effectively managing this web of interdependencies is essential for program success.

Managing Program Dependencies

To manage the interdependencies among all the parts of a large program, the team managers

- must know their team’s status
- anticipate problems in other teams
- warn other teams of commitment delays
- update plans for program changes
- dynamically negotiate and rebalance team commitments

This dependency management process requires

- accurate status information
- timely problem identification
- a dynamic and cooperative replanning process

The Blame Culture

In today’s blame-based culture, nobody wants to speak up.

- The knowledge workers first sense trouble.
- They see the schedule as a management problem and are reluctant to get involved.
- Eventually, the problems are serious enough for the managers to see.
- By then, many parts of the program are in schedule trouble.
- No lower-level manager wants to be first to admit to problems.
- Finally, the cost and schedule problems are so serious that someone must speak up.

At this point, the cost and schedule commitments are unrecoverable and everybody upstairs is surprised.

Carnegie Mellon
Software Engineering Institute

Large System Management

The first step in managing large systems programs is to recognize that these programs involve knowledge work.

The second step is to adopt a fact-based knowledge-working process like the Team Software Process (TSP)SM.

The third is to involve the knowledge working teams in managing their own work.

Finally, management must support and coach the teams when they need help.

SM Team Software Process and TSP are service marks of Carnegie Mellon University.

Managing Knowledge Work -1

The four principles of knowledge management

- Only the workers understand the work.
- Knowledge workers must manage themselves.
- The workers must be trusted to manage their work.
- Knowledge workers need leadership and coaching.

Responsible Project Behavior

When following a knowledge-working process like the TSP, the knowledge workers

- still seek interesting and rewarding work
- continue to value a rewarding team environment
- feel responsible for project cost, schedule, and quality performance

In doing their jobs, TSP teams

- plan, track, and manage their own work
- measure schedule and quality performance
- promptly identify schedule slips
- strive to meet all their commitments
- provide early warning when they cannot

Large Project Consequences

When the project's knowledge-working teams know their status

- they promptly seek help when they need it
- their managers have the facts and data to help them
- a fact-based attitude fosters cooperation across the program

The teams, their managers, and the customers can then

- identify problems in time to resolve them
- work cooperatively to make the program successful

Carnegie Mellon
Software Engineering Institute

Conclusion

Today, large system programs

- almost never meet their cost and schedule commitments
- are often expensive failures

A key reason is an outdated and inefficient management system.

A knowledge-working process like the TSP would enable

- objective fact-based management
- a cooperative working environment
- consistently successful programs

For More Information

Visit the TSP web site: http://www.sei.cmu.edu/tsp/

Contact a PSP transition partner
http://www.sei.cmu.edu/collaborating/partners/trans.part.psp.html

Contact SEI customer relations
Software Engineering Institute, Carnegie Mellon University
Pittsburgh, PA 15213-3890
Phone, voice mail, and on-demand FAX: 412/268-5800
E-mail: customer-relations@sei.cmu.edu

Read the book
Winning With Software: an Executive Strategy, by Watts Humphrey, Addison-Wesley, 2002