The Ideal Software Job

Watts S. Humphrey
The Software Engineering Institute
Carnegie Mellon University
Great Projects

We hear a lot about death-march projects.

What about those truly great projects?

These are the projects that we remember and talk about.

They were interesting, exciting, and fun.

Why couldn’t every job be like that?

This talk is about turning ordinary jobs into memorable ones.
Agenda

Typical projects

Critical success factors

People management

Knowledge work

A knowledge-working process

Industry experience
Typical Software Projects

Typically, software projects are not successful.

In engineers’ opinions, these projects
• were not achievable from the outset
• had excessive management pressure
• required unreasonable overtime
• were technically frustrating
• had lots of team conflict
• operated in a chaotic environment
Successful Projects

The general view is that a successful project
• meets the users’ expectations
• is delivered on or near the committed schedule
• cost what it was supposed to cost

These are management’s typical views of success.

The developers’ views are quite different.
The Developers’ Views

Developers view projects as successful if
- they were a technical challenge
- the product worked as it was supposed to work
- the team was high performing and cohesive

In general, developers feel that, on such projects
- the product was well designed, implemented, and tested
- the team members enjoyed working together
- the team did not feel pressured by management
Comparing Views of Success

Developers typically feel that
• the key to success is technical challenge and team chemistry
• cost, schedule, or user reaction alone do not make projects memorable

Management feels that
• the key is cost and schedule performance and user reaction
• projects can be successful almost regardless of the technical challenge or team chemistry
Critical Success Criteria

The key to truly great projects is to align the engineers’ and managers’ views of success.

This, however, involves changes in
• management style
• engineering behavior
People Management

For knowledge workers to consistently meet management’s expectations, they must be trained in
• personal and project planning
• personal quality-management methods
• project tracking and reporting

For knowledge workers to consistently work in this way, they must be competently
• led
• coached
• recognized and rewarded
Historical Management Styles

Management systems have evolved to meet ever more demanding needs.

The principal phases of this development have been

• Body management – people as oxen
• Task management – people as machines
• Knowledge management – people as individuals
Body Management

In body management, people are valued for their muscles.

They must be driven and directed.

Historically, body management has treated people as slaves.

In body management, motivation is through fear.
Task Management - 1

As defined by Frederick Winslow Taylor, there are three principles to task management.

• Work incentives are required.

• Management knows the best way to do each job.

• Management requires that the workers follow this best way.
Task Management - 2

These principles rest on several assumptions.

• The workers cannot be trusted to manage themselves.
  - Managers and workers have different motivations.
  - Unless the workers are managed, they will be unproductive.

• The managers closely monitor and control the work.

• The managers understand current job status and can take prompt corrective action.
Knowledge Management -1

All development is knowledge work.

Knowledge work
- requires assimilating and integrating concepts
- produces intangible intellectual products
- involves working with people

Managing knowledge workers fundamentally differs from managing other workers.
Knowledge Management - 2

The key rule in managing knowledge work is: the workers must manage themselves.

To manage themselves, these professionals must be properly
• trained
• guided
• supported
Managing Knowledge Work -1

The four principles of knowledge management are
• Only the workers understand the work.
• Knowledge workers must manage themselves.
• The workers must be trusted to manage their work.
• Knowledge workers need motivation, leadership, and coaching.
Managing Knowledge Work -2

To manage themselves, knowledge workers must behave like responsible managers. They must
• make accurate plans
• negotiate commitments
• track their work
• consistently do what they say they will do

Software is the first knowledge industry.

It won’t be the last!
Knowledge Work Challenges

There are four management challenges in large-scale knowledge work.

- Development management
- Commitment management
- Project management
- People management
Development Management

Large-scale knowledge products are extraordinarily complex.

Many critical decisions involve design details.

Only the knowledge workers understand those details.

That means that decision-making must be decentralized.

That also means that management responsibility must be decentralized.
Commitment Management

Commitment management has two objectives.
• To make commitments that can be met
• To strive to meet all commitments

This requires detailed development plans.

For knowledge work, detailed plans can only be made by the people who will do the work.
The objectives of project management are to
• produce quality products
• meet program commitments

This requires that project management consistently
• understand project status
• maintain process discipline
• rebalance program resources
• protect the development teams
For knowledge work, this further requires that the developers and their teams
• regularly and accurately report project status
• do quality work

This further requires that the knowledge workers and their teams
• precisely know their project status
• know how to do quality work
• consistently work the way that they know they should
A Knowledge-Work Process

The Software Engineering Institute (SEI) has developed the Team Software Process (TSP)\textsuperscript{SM} for knowledge work.

The TSP shows knowledge workers how to manage themselves.
- Define their own processes
- Make their own plans
- Negotiate their commitments with management
- Track, manage, and report on their own work

\textsuperscript{SM} Team Software Process and TSP are service marks of Carnegie Mellon University.
TSP Launch Meetings

In starting a TSP project, management asks the team to produce its own plan.

In TSP launch meeting 1, management
• describes what they want done
• explains why the job is important
• answers the team member’s questions

The team then produces a plan to address management’s goals.
An Example TSP Project

The project was to develop communications test equipment.

Management described the product they wanted.
• a new communication-line tester
• required new hardware and software

The finished product had to be available in nine months.
During the four-day TSP launch, the team members
• define a process and strategy for doing the job
• produce detailed team and personal plans
• assess the risks of their plans
• prepare a plan presentation to management

The team and team leader are guided through the launch process by a qualified TSP coach.
TSP Launch Meeting 9

At the end of the TSP launch, the team presents its plan.
  • best plan
  • alternative plans

Management then
  • probes the team’s plan
  • assesses the plan
  • approves the plan if it is suitable
The Example TSP Project

The project was to develop communications test equipment.

<table>
<thead>
<tr>
<th>Plan</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Size - KLOC</td>
<td>110</td>
</tr>
<tr>
<td>Effort - hours</td>
<td>16,000</td>
</tr>
<tr>
<td>Schedule - weeks</td>
<td>77</td>
</tr>
<tr>
<td>Defects per KLOC</td>
<td></td>
</tr>
<tr>
<td>Integration and system test</td>
<td>1.1</td>
</tr>
<tr>
<td>Field trial</td>
<td>0.0</td>
</tr>
<tr>
<td>Customer use</td>
<td>0.0</td>
</tr>
</tbody>
</table>
The Example TSP Project

The project was to develop communications test equipment.

<table>
<thead>
<tr>
<th></th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size - KLOC</td>
<td>110</td>
<td>89.9</td>
</tr>
<tr>
<td>Effort - hours</td>
<td>16,000</td>
<td>14,711</td>
</tr>
<tr>
<td>Schedule - weeks</td>
<td>77</td>
<td>71</td>
</tr>
<tr>
<td>Defects per KLOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration and system test</td>
<td>1.1</td>
<td>0.6</td>
</tr>
<tr>
<td>Field trial</td>
<td>0.0</td>
<td>0.02</td>
</tr>
<tr>
<td>Customer use</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Why the TSP Works

The TSP addresses the fundamental issues of knowledge management.

• Provides the developers the skills to manage themselves.
• Enables management to trust the developers to manage themselves.
• Guides teams in making responsible commitments.
• Builds teams that own their processes and plans.
TSP Industrial Experience

Hundreds of organizations are using the TSP.

Microsoft uses the TSP in its world-wide IT organization.

Intuit has widely adopted TSP for their software work.

Oracle is introducing TSP in its product-development groups.

IBM has started a product-development effort.
<table>
<thead>
<tr>
<th>Product size (KSLOC)</th>
<th>Defect density (defects/KSLOC)</th>
<th>Number of defects</th>
<th>Cost of addressing defect</th>
<th>Cost of addressing all defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>WARP (before TSP)</td>
<td>443</td>
<td>1.13</td>
<td>501</td>
<td>$8,330</td>
</tr>
<tr>
<td>AVJMP (after TSP)</td>
<td>443</td>
<td>0.59</td>
<td>261</td>
<td>$8,330</td>
</tr>
<tr>
<td>Cost saving from reduced defects</td>
<td></td>
<td></td>
<td></td>
<td>$1,992,663</td>
</tr>
<tr>
<td>Cost of TSP training &amp; support</td>
<td></td>
<td></td>
<td></td>
<td>$225,300</td>
</tr>
<tr>
<td>Total cost savings from reduced defects</td>
<td></td>
<td></td>
<td></td>
<td>$1,767,363</td>
</tr>
</tbody>
</table>

KSLOC: One thousand source lines of code
# Microsoft Results

<table>
<thead>
<tr>
<th></th>
<th>Non-TSP Projects</th>
<th>TSP Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Released On Time</td>
<td>42%</td>
<td>66%</td>
</tr>
<tr>
<td>Average Days Late</td>
<td>25</td>
<td>6</td>
</tr>
<tr>
<td>Mean Schedule Error</td>
<td>10%</td>
<td>1%</td>
</tr>
<tr>
<td>Production Defects/KLOC</td>
<td>1.8</td>
<td>0.5</td>
</tr>
<tr>
<td>Sample Size</td>
<td>80</td>
<td>15</td>
</tr>
</tbody>
</table>
A Microsoft TSP Study

Team performance improved with experience. In 4 releases, one team
• delivered on time
• increased productivity by 81%

<table>
<thead>
<tr>
<th>Defect Data - Five Releases of the Same Product</th>
<th>Non-TSP</th>
<th>TSP</th>
<th>TSP</th>
<th>TSP</th>
<th>TSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release number</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Defects/1000 LOC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Test</td>
<td>13.3</td>
<td>2.5</td>
<td>0.8</td>
<td>1.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Acceptance Test</td>
<td>0.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Customer Production</td>
<td>1.2</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Intuit Results

From data on over 40 TSP teams, Intuit has found that
• post code-complete effort is 8% instead of 33% of the project
• for TSP projects, standard test times are cut from 4 months to 1 month

Twice the function in less time with higher quality.
Intuit TSP Survey Results

Engineers love it... Once they adopt it they can’t imagine going back
Conclusions

Because they require knowledge work, software projects have not typically been successful.

Knowledge work requires a new management paradigm.

In the future, all development work will be knowledge work. Such work requires
• developers who can manage themselves
• a knowledge-working management style
For More Information

Visit the PSP/TSP web sites
General: http://www.sei.cmu.edu/tsp

Contact SEI customer relations
Phone, voice mail, and on-demand FAX: 412/268-5800
E-mail: customer-relations@sei.cmu.edu

See the Watts Humphrey books
Winning with Software: an Executive Strategy, Addison-Wesley, 2002
PSP: A Self-Improvement Process for Software Engineers, Addison-Wesley, 2005
TSP: Leading a Development Team, Addison-Wesley, 2006
TSP: Coaching Development Teams, Addison-Wesley, 2006