CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Advanced Software Process

Part I: Software Process Maturity

2. The Principles of Software Process Change

Dave Garcia-Gomez
Faculty / Lecturer
Department of Computer Science

Course Roadmap

Part I: Software Process Maturity
- 1. A Software Maturity Framework
- 2. The Principles of Software Process Change
- 3. Software Process Assessment
- 4. The Initial Process

Part II: The Repeatable Process
- 5. Managing Software Organizations
- 6. The Project Plan
- 7. Software Configuration Management (Part I)
- 8. Software Quality Assurance

Part III: Defined Process
- 9. Software Standards
- 10. Software Inspections
- 11. Software Testing
- 12. Software Configuration Management (continued)
- 13. Defining the Software Process
- 14. The Software Engineering Process Group

Part IV: The Managed Process
- 15. Data Gathering and Analysis
- 16. Managing Software Quality

Part V: The Optimizing Process
- 17. Defect Prevention
- 18. Automating the Software Process
- *19. Contracting for Software*
- *20. Conclusion*

The Principles of Software Process Change

- Process in Perspective
- The Six Basic Principles
- Some Common Misconceptions about the SW Process
- A Strategy for Implementing SW Process Change

Process in Perspective

While process management provides …

- A powerful basis for assessing SW problems, and
- A consistent framework for organizational improvement,

… it is not a cure-all.

Two other key areas that need to be considered are

- People
- Design methods

The People

Better people clearly do better work.

Focusing only on talent, however, can lead into a blind alley for several reasons:

- The best people are always in short supply.
- You probably have about the best team you can get right now.
- With proper leadership and support, most people can do much better work than they are currently doing.

The People

While it is always desirable to recruit better people, it is also wise to focus on making better use of the talent we already have.

Design Methods

- There is convincing evidence that superior products have superior designs.
- Successes were always designed by people who understood the application.
- A program can be viewed as executable knowledge.
- Program designers should thus have application knowledge.
- With such good design, an orderly process can be of great help.

The Six Basic Principles of Software Process Change

- Major changes to the SW process must start at the top.
- Ultimately, everyone must be involved.
- Effective change requires a goal and knowledge of the current process.
- Change is continuous.
- SW process changes will not be retained without conscious effort and periodic reinforcement.
- SW process improvement requires investment.

Major Changes to the Software Process Must Start at the Top

Senior management leadership is required to launch the change effort and to provide continuing resources and priority.

Ultimately, Everyone Must Be Involved

Software engineering is a team effort, and anyone who does not participate in improvement will miss the benefits and may even inhibit progress.

Effective Change Is Built on Knowledge

To use a map, you must know where you are.

An effective change program requires a reasonable understanding of the current status.

An assessment is an effective way to gain this understanding.

Effective Change Is Built on Knowledge

The assessment is a learning process, aimed at gaining a clear definition of the key problems and the professionals' views of what should be done to resolve them.

Without such knowledge, priorities are hard to establish –and priorities are the most important part of an improvement program.

Change Is Continuous

Software process improvement is not a one-shot effort; it involves continual learning and growth.

One of the most difficult things for a management team to recognize is that human intensive processes are never static.

- Reactive changes generally make things worse.
- Every defect is an improvement opportunity.
- Crisis prevention is more important than crisis recovery.

Software Process Changes Won't Stick by Themselves

Precise and accurate work is so hard that it is rarely sustained for long without reinforcement.

This means that new methods must be carefully introduced and periodically monitored, or they too will rapidly decay.

Software Process Changes Won't Stick by Themselves

Human adoption of new process methods involves four stages:

- Installation

  This involves initial installation of the method, training in their proper use.

- Practice

  Here the people learn to perform the methods as instructed

- Proficiency

  With the traditional learning curve, people gradually improve their efficiency.

- Naturalness

  Finally the methods are so ingrained that they are performed without intellectual effort.

It Takes Time, Skill, and Money to Improve the SW Process

It takes planning, dedicated people, management time, and capital investment.

- To improve the SW process, someone must work on it.
- Unplanned process improvement is wishful thinking.
- Automation of a poorly defined process will produce poorly defined results.
- Improvements should be made in small, tested steps.
- Training!

To Improve the SW Process, Someone Must Work on It

No one questions the need to design a manufacturing process before they order the tools and go into production.

- They must consider raw materials handling, design the process flow, select the tools, specify the controls, and oversee ordering, installation, and operation.

The software process needs the same attention.

Unplanned Process Improvement Is Wishful Thinking

Software process improvement requires detailed improvement plans and the resources and dedication to follow them.

Plans define the resource needs and provide a framework for management tracking.

If process improvement is not rigorously planned and tracked, it will not happen!

Automation of a Poorly Defined Process Will Produce Poorly Defined Results

Automation should certainly be used to resolve the highest-priority problems, but little effort should be wasted until a clear definition of the process application is available.

Improvements Should Be Made in Small Steps

It is easy to design processes that are beyond peoples' capabilities.

When such processes are both ill-defined and near these capability limits, they easily become unstable and erratic.

Such processes must be changed with particular care.

Process changes should be made in small steps, and even then, they must be tested and adjusted before wide-spread implementation.

Train, Train, Train!

Many organizations proudly proclaim that their people are their most important asset.

Few software professionals are adequately prepared to use the languages and tools they are given, to understand the protocols and system services they need, to effectively use (or even know) the organization's software process, to appreciate the subtleties of the application they are implementing, or to plan and track their own work.

Train, Train, Train!

Without such training, the necessary learning is gained through trial and error.

- This not only wastes time and money; it also often involves a lot of error.

Training is expensive, but not nearly as expensive as not training.

- A plan is required, some specialists must be hired or retained, and the required courses must be defined and offered.

Some Common Misconceptions about the Software Process

- We must start with firm requirements.
- If it passes test, it must be OK.
- Software quality cannot be measured.
- The problems are technical.
- We need better people.
- Software management is different.

We Must Start with Firm Requirements

There is a widespread but fallacious view that requirements are the customers' job and that development should not start until they are explicitly defined.

With rare exceptions, the requirements must change as the software job progresses.

Generally the most practical answer is to use an incremental development process–that is, to gradually increase the level of detail as the requirement and the implementation are mutually refined.

We Must Start with Firm Requirements

Stable Requirement

- The solution is gradually evolved in a way that is more likely to meet the user's needs.
- Requirements problems can be uncovered during every development phase, and they must be properly resolved at the requirements level and not just patched in the design or the code.

We Must Start with Firm Requirements

Premature Requirements Freezes

- One argument against the incremental approach is that it takes too long.
- That is true when compared with projects with firm, accurate, and precise initial requirements, which is occasionally feasible, but not the general case.

If It Passes Test, It Must Be OK

- The converse is certainly true: If it does not pass test, it is not OK!
- As we build larger and more complex programs and their operating environments become more complex and unpredictable, testing is a progressively less reliable indicator of system quality.

Software Quality Can't Be Measured

- While not as well recognized, this too is false.
- There is no measure that comprehensively represents programming quality.
- Quality measures are valuable, however, so we must find some and use them.
- Discussions of software quality should use numbers, which requires measurement.
- Without measures, a quality program will produce little useful action.

The Problems Are Technical

- In spite of the many improved languages, tools, and environments, the problems of software cost, schedule, and quality remain.
- There are many important technical SW problems.
- Unfortunately, the highest-priority problems of most software organizations are not technical.

  Low-maturity SW organizations spend most of their energy in controlling the high volume of changes, recovering from late-discovered defects, and reacting to unanticipated crises.

We Need Better People

- Since the software professionals make the errors, some people erroneously feel that they should be blamed for them.
- While it is true that an occasional poor performer does bad work, the true cause of errors is the process, not the people.
- We must certainly get the most energetic and best-qualified people we can, but this alone will not solve all the problems.

  Most of the problems can only be fixed through management action.

Software Management Is Different

- While this is a new and unique field, traditional management methods can and should be used.
- Many factors make software different from other engineering fields.
- While there are many unique characteristics to software, they all require more management discipline, not less.

  Detailed plans, tracking systems, periodic technical and management reviews

A Strategy for Implementing Software Process Change

- While the software people will generally welcome a concerted effort to improve their processes, the changes must be handled properly or they will generate resistance.
- An effective change process has three phases:
  - Unfreezing
  - Moving
  - Refreezing

A Strategy for Implementing Software Process Change

- With the software process, unfreezing is best initiated by an effort to understand the true problems in the organization and to make people aware of the opportunities for change.
- With a software process assessment

Champions, Sponsors, and Agents

Champions are the ones who initiate the change process.

- Champions bring management's attention to the subject, obtain the blessing of a sponsor, and establish the credibility to get the change program launched.

The senior management role is equally crucial: someone in authority needs to recognize the value of the work and sponsor it.

- Provide both resources and official backing

Champions, Sponsors, and Agents

Once it is clear that changes are needed and management sponsorship has been obtained, the essential next step is to identify the change agents who will lead change planning and implementation.

The Elements of Change

Three key elements of effective change are

- Planning
- Implementation
- Communication

The Elements of Change

While the proper pace must depend on the problems encountered, it is essential to maintain a continuous stream of actions and successes.

- Public plans
- Periodic progress reports
- Early demonstration of success

These not only reassure the people; they also help to ensure continued executive support.

Refreezing

Refreezing can take many forms, but the basic objective is to ensure that an achieved capability is retained in general practice.

Several common techniques are:

- Retain the management team that instituted the change
- Modify the organization's procedure
- Establish measurements and incentives
- Set up a dedicated staff to monitor and support performance
- Establish an education and training program

References

Humphrey, Watts S., *Managing the Software Process*, The SEI Series in Software Engineering, Addison-Wesley, 1989. (29th Printing, May 2003) (ISBN 0-201-18095-2)