CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Advanced Software Process

Part I: Software Process Maturity

1. A Software Maturity Framework

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*

Introduction

Companies have faced software problems for a long time.

- Open-ended requirements
- Uncontrolled change
- Arbitrary schedules
- Insufficient testing time
- Inadequate training
- Unmanaged system standards

Introduction

- Scale of development ranges from individuals to teams to armies of developers
- Hiring talented people is beneficial, but then what?
- Different programming languages, special conventions, code change coordination contributes to chaos
- Organizations still have same problems

Introduction

- Advanced technology tools would address the software issues.
- This is not only wrong, but is also dangerous.
- Diverts the need for better process management.
- Effective use of software technology is limited by ill-defined process, inconsistent implementation, and poor process management.

Introduction

This course will help you deal with the following questions:

- How good is my current software process?
- What must I do to improve it?
- Where do I start?

Introduction

This course will also provide you with the understanding of the following concepts:

CMM (the basis of this textbook)
ISO 12207 Software Lifecycle Processes
ISO 15504 Process Assessment (SPICE)
CMMI
SCAMPI

Introduction

This course will also provide you with the understanding of the following concepts:

Agile Processes: SCRUM, XP

Unified Process

Object Oriented Modeling

Introduction

- The SEI Software Capability Maturity Model (CMM) was built to provide guidance to the military services in selecting capable software contractors.

- The resulting method for evaluating their strengths and weaknesses has proved valuable for assessing other software organizations.

Introduction

The textbook consists of 5 parts, each of which corresponds to a stage of the software process improvement.

- Part 1: Software Process Maturity (Initial)
- Part 2: The Repeatable Process
- Part 3: The Defined Process
- Part 4: The Managed Process
- Part 5: The Optimizing Process

Introduction

Part 1: Software Process Maturity (Initial)

Describes a framework for software process management, the use of this framework in process assessment, and the steps required to initiate effective software process change

Introduction

Part 2: The Repeatable Process

Outlines the actions needed to establish basic control over the software process.

These provide the stability needed for orderly and continuous process improvement.

Introduction

Part 3: The Defined Process

Describes how to specify the development process and the technical and managerial concepts needed to control it.

Software development is a surprise-prone business, and a defined process permits more orderly reaction to unanticipated events.

Introduction

Part 4: The Managed Process

The method of quantitative software process control are introduced.

At this stage data is gathered and analyzed to support quantitative quality and process management.

These are the tools that can guide us to a better understanding of our work, a more precise way to control our actions, and a truly informed basis for sustained process improvement.

Introduction

Part 5: The Optimizing Process

Presents the final stage of software process evolution; it shifts the focus from fixing problems to preventing them.

At this level software managers and professionals learn to use quantitative process methods to harness technology for continuing quality and productivity improvement.

Introduction

Suggestion:

Outside of reading Chapter 1, also read Chapter 19 and Chapter 20, and the summaries of all chapters to get a good overview of the entire process.

A Software Maturity Framework

The software process is the set of tools, methods, and practices we use to produce a software product (what, who, where.)

The objectives of software process management are to produce products according to plan while simultaneously improving the organization's capability to produce better products

A Software Maturity Framework

The characteristics of a truly effective software process

- Fundamentally, it must be predictable.
- That is, cost estimates and schedule commitments must be met with reasonable consistency, and the resulting products should generally meet users' functional and quality expectations.

A Software Maturity Framework

The objectives of software process management are to produce products according to plan while simultaneously improving the organization's capability to produce better products.

A Software Maturity Framework

The basic principles are those of statistical process control, which have been used successfully in many fields.

A process is said to be stable or under statistical control if its future performance is predictable within established statistical limits.

When a process is under statistical control, repeating the work in roughly the same way will produce roughly the same result.

A Software Maturity Framework

- To obtain consistently better results, it is thus necessary to improve the process.
- If the process is not under statistical control, sustained progress is not possible until it is.
- The basic principle behind statistical control is measurement.

A Software Maturity Framework

There are several factors to consider in measuring the process.

- The numbers must properly represent the process being controlled.
- The numbers must be sufficiently well defined and verified to provide a reliable basis for action.
- The mere act of measuring human processes changes them.

It is thus essential to limit the measurements to those with a predefined use.

Software Process Improvement

Important first step in addressing problems in the software process, is to treat the entire software task as a process that can be controlled, measured, and improved.

Software Process Improvement

The improvement of software development organizations follows six steps:

1. Understand the current status of the development process.
2. Develop a vision of the desired process.
3. Establish a list of required process improvement actions in order of priority.
4. Produce a plan to accomplish the required actions.
5. Commit the resources to execute the plan.
6. Start over at step 1.

Process Maturity Levels

Five Levels of Process Maturity

1. Initial
   Basic Management Control
2. Repeatable
   Process Definition
3. Defined
   Process Measurement
4. Managed
   Process Control
5. Optimizing

Process Maturity Levels

CMM

- Initial
- Repeatable
- Defined
- Managed
- Optimizing

CMMI

- Initial
- Managed
- Defined
- Quantitively Managed
- Optimizing

Process Maturity Levels

Initial (Level 1)

- Until the process is under statistical control, orderly progress in process improvement is not possible.
- While there are many degrees of statistical control, the first step is to achieve rudimentary predictability of schedules and costs.

Process Maturity Levels

Repeatable (Level 2)

- The organization has achieved a stable process with a repeatable level of statistical control by initiating rigorous project management of commitments, cost, schedules, and changes.

Process Maturity Levels

Defined (Level 3)

- The organization has defined the process as a basis for consistent implementation and better understanding.
- At this point advanced technology can usefully be introduced.

Process Maturity Levels

Managed (Level 4)

- The organization has initiated comprehensive process measurements and analysis.
- This is when the most significant quality improvements begin.

Process Maturity Levels

Optimizing (Level 5)

- The organization now has a foundation for continuing improvement and optimization of the process.

Process Maturity Levels

Process maturity levels have been selected because they:

- Reasonably represent the actual historical phases of evolutionary improvement of real SW organizations
- Represent a measure of improvement that is reasonable to achieve from the prior level
- Suggest interim improvement goals and progress measures
- Make obvious a set of immediate improvement priorities, once an organization's status in this framework is known

The Initial Process (Level 1)

- Ad hoc
- Often even chaotic
- At this stage the organization typically operate without formalized procedures, cost estimates, and project plans.
- Tools are neither well integrated with the process nor uniformly applied.
- Change control is lax, and there is little senior management exposure or understanding of the problems and issues.

The Initial Process (Level 1)

- While organizations at this level may have formal procedures for planning and tracking their work, there is no management mechanism to ensure that they are used.
- The best test is to observe how such an organization behaves in a crisis.

  If it abandons established procedures and essentially reverts to coding and testing, it is likely to be at the **Initial Process Level**.

The Initial Process (Level 1)

Organizations at the Initial Process Level can improve their performance by instituting basic project controls.

- Project management
- Management oversight
- Quality assurance
- Change control

The Initial Process (Level 1)

Project Management

- The fundamental role of a project management system is to ensure effective control of commitments.

  Adequate preparation, clear responsibility, a public declaration, a dedication to performance

- Starts with an understanding of the job's magnitude

  In any projects, a plan must be developed to determine the best schedule and the anticipated resources required.

The Initial Process (Level 1)

Management Oversight

- A suitably disciplined SW development organization must have senior management oversight.
  - Review and approval of all major development plans prior to their official commitment
  - A quarterly review should be conducted of facility-wide process compliance, installed quality performance, schedule tracking, cost trends, computing service, and quality and productivity goals by project.
  - The lack of such reviews typically results in uneven and generally inadequate implementation of the process as well as frequent over-commitments and cost surprise.

The Initial Process (Level 1)

Quality Assurance

- A quality assurance group is charged with assuring management that SW work is done the way it is supposed to be done.
- To be effective, the assurance organization must have an independent reporting line to senior management and sufficient resources to monitor performance of all key planning, implementation, and verification activities.
- This generally requires an organization of about 3 percent to 6 percent the size of the SW organization.

The Initial Process (Level 1)

Change Control

- Change control for SW is fundamental to business and financial control as well as to technical stability.
- To develop quality SW on a predictable schedule, requirements must be established and maintained with reasonable stability throughout the development cycle.
- Requirement changes; design and code changes
- If changes are not controlled, then orderly design, implementation, and test is impossible and no quality plan can be effective.

The Repeatable Process (Level 2)

The **Repeatable Process** provides control over the way the organization establishes its plans and commitment.

Organizations at this level face major risks when they are presented with new challenges.

- New tools and methods will affect the process.
- When the organization must develop a new kind of product, it is entering new territory.
- Major organizational changes can also be highly disruptive.

The Repeatable Process (Level 2)

Key actions required to advance from the Repeatable to the next stage, the Defined Process, are to

- Establish a process group (EPG, SEPG)
- Establish a software development process architecture (or development life cycle)
- Introduce a family of SE methods and technologies

The Repeatable Process (Level 2)

Establish a Process Group

- A process group is a technical resource that focuses exclusively on improving the SW process. (EPG, SEPG)
- In SW organizations at early maturity levels, all the people are generally devoted to product work.
- Until some people are given full-time assignments to work on the process, little orderly progress can be made in improving it.

The Repeatable Process (Level 2)

Establish a SW development process architecture, or development life cycle

- … that describes the technical and management activities required for proper execution of the development process
- The process architecture is a structural decomposition of the development life cycle into tasksitecture is a, each of which has a defined set of prerequisites, functional descriptions, verification procedures, and task completion specifications.

  The decomposition continues until each defined task is performed by an individual or single management unit.

The Repeatable Process (Level 2)

Introduce a Family of SW Engineering Methods and Technologies

- These include design and code inspections, formal design methods, library control systems, and comprehensive testing methods.
- Prototyping should also be considered, together with the adoption of modern implementation languages.

The Defined Process (Level 3)

With the **Defined Process**, the organization has achieved the foundations for major and continuing progress.

- For example, the software team, when faced with a crisis, will likely continue to use the process that has been defined.
- The foundation has now been established for examining the process and deciding how to improve it.

The Defined Process (Level 3)

The key steps required to advance from the Defined Process to the next level:

- Establish a minimum basic set of process measurements to identify the quality and cost parameters of each process step.
- Establish a process database and the resources to manage and maintain it.
- Provide sufficient process resources to gather and maintain this process data and to advise project members on its use.
- Assess the relative quality of each product and inform management where quality targets are not being met.

The Managed Process (Level 4)

In advancing from the previous levels to the **Managed Process**, organizations should expect to make substantial quality improvement.

- The greatest problem with the Managed Process is the cost of gathering data.
- There are an enormous number of potentially valuable measures of the software process, but such data is expensive to gather and to maintain.

The Managed Process (Level 4)

The two fundamental requirements for advancing from the Managed Process to the next level are:

- Support automatic gathering of process data.
- Use process data both to analyze and to modify the process to prevent problems and improve efficiency.

The Optimizing Process (Level 5)

In varying degree, process optimization goes on at all levels of process maturity.

- With the step from the Managed Process to the Optimizing Process, however, there is a paradigm shift.
- Up to this point SW development managers have largely focused on their products and will typically gather and analyze only data that directly relates to product improvement.

The Optimizing Process (Level 5)

- In the Optimizing Process, the data is available to tune the process itself.
- With a little experience, management will soon see that process optimization can produce major quality and productivity benefits.
- With the Optimizing Process, the organization has the means to identify the weakest elements of the process and to fix them.

People in the Optimizing Process

Clearly any SW process is dependent on the quality of the people who implement it.

The Optimizing Process enhances the talent of quality people in several ways.

- It helps managers understand where help is needed and how best to provide people with the support they require.
- It lets the professionals communicate in concise, quantitative terms.
- It provides a framework for the professionals to understand their work performance and to see how to improve it.

The Need for the Optimizing Process

- Unless we dramatically improve our error rates, the greater volume of code will mean increased risk of error.
- The complexity of our systems is increasing, which will make the systems progressively more difficult to test.
- As well as being a management issue, quality is an economic one.

  It is always possible to do more tests, but it costs time and money to do so.

The Need for the Optimizing Process

- It is only with the Optimizing Process that the data is available to understand the costs and benefits of such work.
- The Optimizing Process provides the foundation for significant advances in SW quality and simultaneous improvements in productivity.

The Need for the Optimizing Process

Transition from Level 1 to Level 2 or from Level 2 to Level 3 take from one to three years, even with a dedicated management commitment to process improvement [Humphrey 1989].

Process Levels Comparison

	CMM Maturity	CMMI Maturity	CMMI v1.2 Capability	CMMI v1.3 Capability	ISO 15504 Capability
Level 5	Optimizing	Optimizing	Optimizing	-	Optimizing
Level 4	Managed	Quantitatively Managed	Quantitatively Managed	-	Predictable
Level 3	Defined	Defined	Defined	Defined	Established
Level 2	Repeatable	Managed	Managed	Managed	Managed
Level 1	Initial	Initial	Performed	Performed	Performed
Level 0	-	-	Incomplete	Incomplete	Incomplete

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)

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)

CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Advanced Software Process

Part I: Software Process Maturity

3. Software Process Assessment

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*

Software Process Assessment

- Assessment Overview
- Assessment Phases
- Five Assessment Principles
- The Assessment Process
- Assessment Conduct
- Implementation Considerations

Assessment Overview

Process assessment helps software organizations improve themselves by identifying their critical problems and establishing improvement priorities.

The basic assessment objectives are:

- To learn how the organization works
- To identify its major problems
- To enroll its opinion leaders in the change process

Assessment Overview

A software process assessment is not an audit but a review of a software organization to advise its management and professionals on how they can improve their operation.

It is conducted by a team of SW professionals who typically have assessment experience or training.

Assessment Phases

Assessment typically conducted in three phases:

- Preparation
- Assessment
- Recommendations

Assessment Phases

Preparation

- Senior management becomes committed to the process, agree to participate personally, and commits to take action on the resulting recommendations or explain its reasons for not doing so.
- Phase one concludes with a brief one-or two-day training program for the assessment team.

Assessment Phases

Assessment

- Phase two is the on-site assessment period.
- Assessment activity typically takes several days, although it can take two or more weeks, depending on the size of the organization and the assessment technique used.
- Phase two concludes with a preliminary report of the findings to local management.

Assessment Phases

Recommendations

- In phase three the findings and action recommendations are presented to the local managers.
- A local action team is then assembled to plan and implement the recommendations.
- The assessing organization may provide assistance during this period and may participate in a subsequent follow-up assessment.

Five Assessment Principles

As in many activities, the basic requirements for a good assessment are a competent team, sound leadership, and a cooperative organization.

Because the software process is human-intensive, however, some special considerations should be kept in mind.

Five Assessment Principles

- The need for a process model as a basis for the assessment
- The requirement for confidentiality
- Senior management involvement
- An attitude of respect for the views of the people in the organization being assessed
- An action orientation

Start with a Process Model

- An assessment implies a standard.
- The organization's process is reviewed in comparison with some vision of how such processes should be performed.

Observe Strict Confidentiality

- The assessment's purpose must be to support the organization's improvement program and not to report its problems to higher management.
- Even when initiated with this intent, it is extraordinarily difficult to maintain confidentiality, particularly when a chief executive demands to see the results.

Observe Strict Confidentiality

- If assessment team provides such data, however, people will learn that they can not speak in confidence.
- As this becomes widely known, the assessment group will find it increasingly difficult to conduct assessments that uncover the real issues.
- Confidentiality is required at all organizational levels.

Involve Senior Management

- The senior manager sets the organization's priorities.
- Site manager
  - Senior manager of total organization
  - The site manager must be personally involved in the assessment and its follow-up action plans.
  - If not, the work will not be given sufficiently priority.

Keep an Open Mind and a Level Head

- Even when the assessment team is appropriately supportive, some local people will be resentful and not cooperate.
- If the team members' actions clearly demonstrate their desire for active collaboration with the on-site professionals, however, this will be recognized and people generally will respond positively.

Focus on Action

- The assessment must be directly toward improvement.
- An action orientation keeps the questions focused on current problems and the need to solve them.
- If the assessment turns into a general exploration, it will not focus on the priority issues or produce recommendations that will be implemented.
- Management must either focus on taking action or not do an assessment.

The Assessment Process

- The first step in any assessment is to identify the organization to be assessed and the team to do it.
- This typically requires the site manager's commitment to doing the assessment and willingness to assign sufficient skilled resources to get it done.

Forming an Assessment Team

- The assessment team leader is selected first.
- The assessment team members should all be experienced SW developers, and one or more should have experience in each phase of the SW process.
- Four to six professionals typically form an adequate team, although more can be used if desired.

Guidelines for Selecting Assessment Team Members

Each assessment team member should:

- Have at least eight to ten years professional software experience
- Be well respected in the organization
- Be able to deal with people in an informal and non-threatening manner
- Be a team player
- Have attended assessment training with this team

No assessment team member should:

- Be currently serving in an audit or review capacity for any of the projects being assessed
- Be a line manager over any of the projects being assessed or people being interviewed
- Be working directly on any of the projects being assessed or working on their direct support

Self-Assessment Considerations

While it is possible for organizations to assess themselves, they should be aware of several potential problems.

- Few organizations can afford a staff of assessment experts.

Assessment Ground Rules

It is desirable to have a written set of assessment ground rules for the organization being assessed and for the assessment team members.

For an external assessment, the site manager and the assessment team leader should sign a written agreement covering these ground rules.

A copy of the standard agreement used by the SEI is shown in Appendix B.

Assessment Team Training

As the assessment team is formed, the members must agree to participate fully during the training period, on-site review, and wrap-up meetings.

The On-Site Period

The assessment starts with a presentation to the site manager and staff.

The assessment ground rules are discussed, as well as the assessment principles and the overall schedule.

An overview meeting is then held with all the site participants, including the project managers and the professionals to be interviewed.

Assessment Conduct

The assessment should focus on what the projects actually do, how they do it, the problems encountered, and the results obtained.

A selected set of questions should be prepared in advance of the actual assessment period.

Meetings are next held with small group of selected professionals who have expertise in various facets of the software process.

Probing Questions

In conducting assessments, it is hard to obtain really accurate information.

- Questions are often misunderstood.
- The respondents may have a different understandings of some common terms.
- The respondents may not be broadly aware of the work in their own organization.
- Occasionally people are unwilling to risk the truth.

Assessment Conclusions

- At the assessment conclusion, the team prepares a report on its initial findings.
- The report should be a composite summary of site status, together with more detailed findings in key areas.
- Prior to reviewing this material with the site manager, the team should review it with the project managers.

  This should identify any overlooked problems or any misstated or overemphasized topics.

Assessment Conclusions

- The last site actions is to review the composite findings with the site manager and staff.
- Any questions are addressed, and the schedule for follow-up work is reviewed.
- If possible, all the site personnel who participated in the assessment should attend this meeting.

Assessment Report

The final assessment team action is the presentation of a written final report and recommendations to the site manager and staff.

- The recommendations should highlight the three or four items of highest priority.
- Since no organization can handle more than a few priority tasks at a time, the total number of items requiring attention should be limited to around ten.
- These should be clearly explained with the assessment team's views on implementation priority.

Action Plans

- The action plans are next prepared by the local site organization, generally under the guidance of the team member named for this purpose.
- If properly chosen, this member is now fully knowledgeable on the issues and is able to start quickly.

Assessment References

- Typical SEI-Assisted Assessment Schedule [Table 3.2]
- Guidelines for Selecting Functional Area Representatives [Table 3.3]
- Guidelines for Finding Formulation [Table 3.4]
- A Typical Outline of the Findings Presentation [Table 3.5]
- A Typical Final Report Outline [Table 3.6]

Reassessments

Organizations should generally conduct follow-up assessment one to two years after the initial action plans have been developed and approved.

This is important for several reasons:

- To assess the progress that has been made
- To provide a visible future milestone for completion of the actions from the prior assessment
- To establish new priorities for continued improvement

Implementation Considerations

- The greatest assessment risk is that no significant improvement actions will be taken.
- Without proper management focus a few superficial efforts may be made, but soon everything will revert to business as usual.

Implementation Considerations

A catalyst is needed to maintain the improvement priority, such as goals and management reviews.

- Long-term goals are first established, and then sub-goals are defined for intervening two-or three-month periods.
- A senior management quarterly review then maintains high-level checkpoint visibility, crystallizes the plans, and creates the periodic crises required to get things accomplished.

Risks

Some of the other key risks and potential actions to alleviate them are schedule conflicts, inadequate support, and lack of follow-through.

Risks: Schedule Conflicts

- Despite the best intentions, crises that conflict with assessment plans often arise.
- The most damaging of these require the site manager to miss either the opening or the closing meeting or both.
  - Such unfortunate schedule conflicts have happened in about one-third of the assessments.
  - A solution -substitute executive who can speak for the site manager and then arrange for a later private meeting to cover the issues

Risks: Inadequate Support

- The reason of inadequate support is that the assessment commitment is made at too low a management level.
- Often only a very senior executive can take a sufficiently long-term view to avoid becoming defensive.
- Also, even fairly high-level managers are often only responsible for portions, so they cannot provide adequate organization-wide priority.

Risks: Lack of Follow-through

- Frequently management change or other high-priority issues reduce the focus on action plan implementation.
- The most important determinants of success is the presence of an aggressive manager to lead the change efforts, a capable process improvement staff, and a clearly stated improvement goal.

Staffing

- Staffing is generally the most serious implementation problem.
- In addition to staffing the assessment itself, the other staffing needs will generally include:
- A small, full-time staff to focus and to guide the improvement efforts
- Part-time project participation in the action plan working group
- Project review and implementation of the resulting actions

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)

CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Advanced Software Process

Part I: Software Process Maturity

4. Software Process Assessment

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 Initial Process

- The Nature of the Initial Process
- A Case Study of a Chaotic Project
- Why Software Organizations Are Chaotic
- Software Process Entropy
- The Way Out

The Level of Software Process Maturity

The Levels of Software Process Maturity (Table 4.1)

- Level 1 – Initial
- Level 2 – Repeatable
- Level 3 – Defined
- Level 4 – Managed
- Level 5 – Optimizing

The Level of Software Process Maturity

Level 1 – Initial

- Characteristics: Chaotic – unpredictable cost, schedule, and quality performance
- Needed actions: Planning (size and cost estimation and schedules), performance tracking, change control, commitment management, Quality Assurance

The Level of Software Process Maturity

Level 2 – Repeatable

- Characteristics: Intuitive – cost and quality highly variable, reasonable control of schedules, informal and ad hoc process methods and procedures
- Needed actions: Develop process standards and definitions, assign process resources, establish methods (requirements, design, inspection, and test)

The Level of Software Process Maturity

Level 3 – Defined

- Characteristics: Qualitative – reliable costs and schedules, improving but unpredictable quality performance
- Needed actions: Establish process measurements and quantitative quality goals, plans, measurements, and tracking

The Level of Software Process Maturity

Level 4 – Managed

- Characteristics: Quantitative – reasonable statistical control over product quality
- Needed actions: Quantitative productivity plans and tracking, instrumented process environment, economically justified technology investments

The Level of Software Process Maturity

Level 5 – Optimizing

- Characteristics: Quantitative basis for continued capital investment in process automation and improvement
- Needed actions: Continued emphasis on process measurement and process methods for error prevention

The Initial Process

Most software organizations operate in the Initial Process Level at least some of the time; many organizations never leave it.

The Nature of Initial Process

- In the Initial Process, the professionals are driven from crisis to crisis by unplanned priorities and unmanaged change.
- From the outside, such groups are often hard to identify, at least in the short term.
- Over time, however, they are easy to recognize because they generally do not meet commitments.

The Nature of Initial Process

- While their managers often present a convincing and impressive story and they may even meet their interim checkpoints, there is often some last-minute crisis that *blows the plan out the window*.
- Such groups occasionally even deliver on schedule, but these accidents are the exception and are generally *due to herculean individual efforts rather than the strength of the organization*.

Chaotic Project

- With the Initial Process, inadequate project management may nearly destroy years of good technical work.
- Without a plan, you have no idea how big the job really is, no basis to know where you stand, and no way to justify added resources when they get into trouble.

Chaotic Project

- A chaotic project is seriously late.
- The schedule does not match actual.
- No plan even with a good technical shape
  - Lack of documented planning
  - A small amount of additional function expected turns out to be hundred thousands of code.
  - No idea how big the project really is
  - No basis to know where they stand
  - No way to justify added resources when they get into trouble
  - Even though the technical work would be sound, the customer was convinced the project would be a disaster.

Why Software Organizations Are Chaotic

Without some work we generally have no idea how much code a given function will require or how much effort it will take.

While lack of a commitment discipline is the most common reason for chaotic behavior, there are other forces.

Why Software Organizations Are Chaotic

1. Under extreme pressure, software managers often make a guess rather than a plan.

2. When the going gets rough, there is a strong temptation to believe in magic.
   - Savior, new technology, excuse for not planning – merely postpones the problem

Why Software Organizations Are Chaotic

3. The scale of software projects follows an escalating cycle:
   - Programs generally take far more code than expected.
   - As the programs become larger, new technical and management issues come up.
   - Since these are unlike previous experience, they are a surprise.

Why Software Organizations Are Chaotic

4. Even after a higher maturity level is reached by an organization, new management, increased competition, or new technical challenges put pressure on the process.
   - With the maturity of our field and the general lack of training for our people, this often means that organizations under pressure revert to the Initial Process.

Chaotic Forces – Unplanned Commitments

The most insidious and probably most difficult software management problem is the unplanned commitment.

There are many reasons why commitments are made before a work plan can be developed, and they have considerable justification.

Chaotic Forces – Unplanned Commitments

Minor commitments can also cause problems.

- Simple-looking functions often have hidden trap.

The unplanned commitment trap

- Lack of an orderly plan generally means there are insufficient time and resources to do the job.

Chaotic Forces – Gurus

The technical wizard can be a powerful asset.

Unfortunately, gurus sometimes believe they can do no wrong.

After they led a small team to produce a superbly crafted program, they sometimes believe they can develop anything.

Chaotic Forces – Magic

Human beings are inherently repelled by complexity.

- System developers also search for simple solutions to complex problems.
- There is no silver bullet.

While there are many cases in which improved technology can help, there are many more that need effective management.

Chaotic Forces – Problems of Scale

One of the fundamental reasons for problems of scale is the large size of many programs.

Software size is insidious because of its impact on the development process.

Having learned to build a small program, we are not fully prepared to build a large one, though we often think we are

Chaotic Forces – Problems of Scale

As software products become larger, they are much more difficult to understand.

Progressive levels of scale

- One person knows the details of the program.

  …
- A large software product is defined and understood at the product management level, but a separate team understands the characteristics and design of each its component programs.

  …
- When the system is very large and has evolved through many versions, there may be no one who understand it.

Chaotic Forces – Problems of Scale

As software knowledge is more widely distributed:

- Common notations are needed for precise communication.
- These standards must be documented, interpreted, and updated.
- Conflicts in standards must be identified and resolved.
- Standards changes must be controlled and distributed.

With larger-scale software, similar control is needed for requirements, design, code and test.

Chaotic Forces – Problems of Scale

As software size increases, prototypes or multiple releases are needed because:

- The total function cannot be implemented in time.
- Some needs cannot be understood without operational experience on a partial system.
- Some design issues cannot be resolved until a preliminary system has been built and run.
- A release discipline helps sort out user priorities.
- No one successfully builds large software systems in one shot anyway.

Chaotic Forces – Problems of Scale

With multiple releases, new complications arise:

- The requirements must be phased to meet end user needs.
- Each software component design must synchronize with these needs.
- Product interdependencies are orchestrated to meet release functional prerequisites.
- The build and integration plans are scheduled around these interdependencies.
- Early system drivers are scheduled to meet component test needs.

Chaotic Forces – Problems of Scale

- The tight schedule requires subsequent release development to start before the prior releases are finished.

There undoubtedly will be new surprises when we reach a new level, as we ultimately must.

The Implications of Software Scale

The problems of software scale affect the individual, the management system, and the technical methods and tools that we use.

Software Process Entropy

There are many forces on the software process that push us toward disorganization (or increased entropy).

Even when we have established a sound project management system, three classes of forces tend to disrupt it:

- Dynamic requirements
- Increasing system size
- Human nature

The Way Out

By examining many organizations that have worked their way out of the chaos trap, however, the outlines of a general solution are clear:

- Apply systematic project management

  The work must be estimated, planned, and managed.

- Adhere to careful change management

  Changes must be controlled, including requirements, design, implementation, and test.

- Utilize independent software assurance

  An independent technical means is required to assure that all essential project activities are properly performed.

The Way Out

The basic principles for controlling the chaos in software organizations

- Plan the work.
- Track and maintain the plan.
- Divide the work into independent parts.
- Precisely define the requirements for each part.
- Rigorously control the relationships among the parts.
- Treat software development as a learning process.

The Way Out

- Recognize what you don't know
- When the gap between your knowledge and the task is severe, fix it before proceeding.
- Manage, audit, and review the work to ensure it is done as planned.
- Commit to your work and work to meet your commitments.
- Refine the plan as your knowledge of the job improves.

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)

CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Advanced Software Process

Part II: The Repeatable Process

5. Managing Software Organizations

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*

Part II Summary
The Repeatable Process

Ch.05 – Managing SW Organizations
- Deals with the way SW commitments are made and managed

Ch.06 – The Project Plan
- Reviews SW planning principles and goals, the Work Breakdown Structure, size estimating, resource estimating, and project tracking

Ch.07 – Software Configuration Management (Part I)
- Explains why configuration management is important and describes the capabilities with highest initial priority.

Ch.08 – Software Quality Assurance
- Describes the benefits and goals of SQA, how it is organized, and some of the key considerations in establishing and managing an SQA organization.

Part II Summary
The Repeatable Process

The reason the four topics of Part II are addressed at this point is not that they cover all important issues but that they likely represent the highest-priority areas for organizational improvement at this point.

Part II Summary
The Repeatable Process

The reason these four topics are selected for attention at this point is that they establish a sound basis for project planning and process management.

- Until the organization effectively handles commitments, planning, configuration management, and Quality Assurance, the process will be too erratic to permit orderly improvement.

Managing Software Organizations

- Commitment Discipline
- The Management System
- Establishing a Project Management System

Managing Software Organizations

- The role of the management system is to ensure that projects are successfully completed.
- This implies some organization-wide agreement on the meaning of the terms "success" and "completion".
- It also requires a continuing management focus on the progress of each project.
- Project management starts with a definition of the job to be done and the plan to do it.

Managing Software Organizations

Basic principles of project management:

- Each project has a plan that is based on a hierarchy of commitments.
- A commitment is an agreement by one person to do something for another.
- A management system resolves the natural conflicts between the projects and between the line and staff organizations.
- An oversight and review system audits and tracks progress against the plans.

Commitment Discipline

The foundation for software project management is the commitment discipline.

This is supported by plans, estimates, reviews, and tracking systems, which focus on ensuring that the organization meets its commitments.

Commitments are not met by reviews, procedures, or tools, however; they are met by committed people.

Making a Commitment

A commitment involves a planned completion date and some consideration or payment.

When the coordinated efforts of many professionals are involved, mutual commitments are essential.

Making a Commitment

The elements of an effective commitment

- The person making the commitment does so willingly.
- The commitment is not made lightly.
  - The work involved, the resources, and the schedule are carefully considered.
- There is agreement between the parties on what is to be done, by whom, and when.
- The commitment is openly and publicly stated.
- The person responsible tries to meet the commitment, even if help is needed.
- Prior to the committed date, if it is clear that it cannot be met, advance notice is given and a new commitment is negotiated.

The Commitment Hierarchy

While commitments are met by committed individuals, the work is not done in a vacuum.

As long as the professionals feel they are part of an organization that treats its commitments seriously, they will strive to do their part.

This calls for a management team that takes care in making commitments and then insists on extraordinary efforts to meet them.

The Software Commitment Process

To be effective, the software commitment process must reach to the top of the organization.

This requires that:

- All commitments for future SW delivery are personally made by the organization's senior executive.
- These commitments are made only after successful completion of a formal review and concurrence process.
- An enforcement mechanism ensures that these reviews and concurrences are properly conducted.

Establishing a Commitment Process

A commitment process can be established very quickly.

The basic requirement is a senior executive who is willing to insist that the required planning be done before any commitment is made.

In addition, since the people must know how to make schedules and estimates, training courses are required, as are specific estimating, review, and approval procedures.

The Management System

While every organization will have unique objectives, there are generally four components:

- To have a technical and business strategy that aims at such long-term goals as growth rate or market position
- To provide quality products that meet customer's needs in a timely and effective way
- To perform the assigned mission competitively and economically
- To improve continually the organization's ability to handle more challenging work

Product and Period Plans

To resolve the inherent conflict and to establish a framework for operations, most organizations produce annual operating plans.

These specify the tasks to be performed and assign the responsibilities and resources to accomplish them.

Since skilled resources are the single most important need of every project and staff, their allocation is the essential first step in producing these plans.

Management Oversight

An effective management system typically uses reviews and a contention system to resolve product and period conflicts and establish the balance between line and staff.

Each line and staff organization prepares its annual plan and reviews it with all involved parties.

Management Oversight

The issues are then resolved, the separate plans are consolidated into a total organization plan, and this total plan is incorporated in the plan for the next higher organizational level.

Similarly, each project area establishes its own plan prior to project initiation and periodically reviews and updates it.

The Contention Process

An effective review system requires a parallel contention system to encourage the open expression of differences and their rational resolution.

The principle behind the contention system is that the best decisions are based on a full understanding of the relevant issues.

There are many ways to solve most problems and there is rarely agreement on the best approach.

The Contention Process

Most of successful organizations have found that the contention system helps them arrive at better decisions.

Alfred P. Sloan, the founder of GM

> *"No important decision should be made unless there is some contention."*

The Principles of the Contention System

- All major decisions are reviewed with the involved parties in advance, and the parties are requested to agree.
  - When possible, any issues are resolved before proceeding.
- When the time comes for the decision, all dissenting parties are present and asked to state their views.
- When there is no disagreement, the senior manager determines if there is knowledgeable agreement, if any disagreeing parties are absent, or if more preparation is needed.
  - In the latter two cases, the decision is deferred until the necessary homework has been done.

The Quarterly Review

The quarterly review provides a forum for resolving conflicts and monitoring progress against period and product objectives.

The topics should typically include an assessment of project performance against plan and the organization's performance against its goals.

The Quarterly Review

TABLE 5.1
EXAMPLE OF A QUARTERLY REVIEW AGENDA

Opening Comments	**Site Manager**
Project A Review	
Milestone Status	Project A Manager
Financial Status	Finance Manager
Issues	Project A Manager
Staff Comments	Finance and SQA
Project B Review	
Milestone Status	Project B Manager
Financial Status	Finance Manager
Issues	Project B Manager
Staff Comments	Finance and SQA
Project C Review	
Milestone Status	Project C Manager
Financial Status	Finance Manager
Issues	Project C Manager
Staff Comments	Finance and SQA
Remaining Project Overviews	
Milestone Status	Development Manager
Financial Status	Finance Manager
Issues	Development Manager
Staff Comments	Finance and SQA
Computing Support	
Performance Measures	Computing Manager
Issues	Computing Manager
Project Comments	Development Manager
Process Status	
Assessment Update	SEPG Manager
Action Plan Status	SEPG Manager
Technology Plan Status	SEPG Manager
Organization Performance	
Productivity	Finance/SEPG
Quality	SQA/SEPG
Action Item Summary	Site Manager

Example of a Quarterly Review Agenda (Table 5.1)

The Quarterly Review

Typical Software Lab Organization (Figure 5.1)

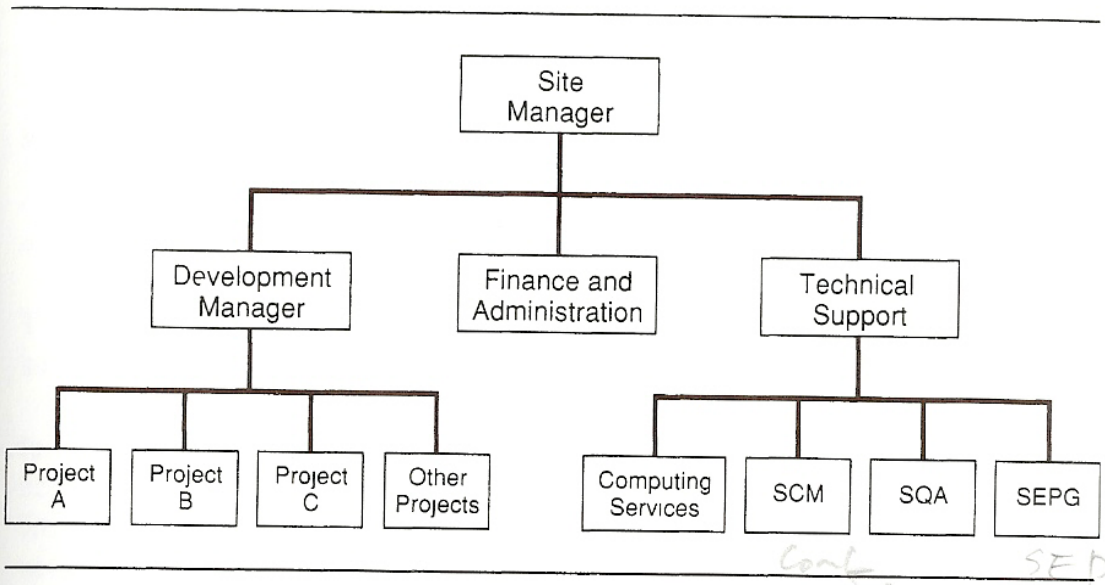

FIGURE 5.1
Typical Software Laboratory Organization

The Quarterly Review

Simple Project Schedule Checkpoints (Table 5.2)

TABLE 5.2
SIMPLE PROJECT SCHEDULE CHECKPOINTS

Project: _____ Manager: _____ Date: _____

Item	Planned	Projected	Actual	Difference
Functional Rqts. Approval				
Phase 1 Review Approval				
Design Reviews Complete				
Phase 2 Review Approval				
Code Reviews Complete				
Unit Test Complete				
Phase 3 Review Approval				
Integration Test Entry Complete				
System Test Complete				
Phase 4 Review Approval				
Customer Shipment				

Project Phase Reviews

Management must also assess project progress periodically.

This is typically accomplished through a sequence of reviews at key points in each project.

Project Phase Reviews

ISO/DoD Standards

- DoD standards MIL-STD-2167
  - Establishes a uniform software development process which is applicable throughout the system life cycle
  - http://www.stsc.hill.af.mil/
  - http://www.crosstalkonline.org
- MIL-STD-498: SW Development and Documentation
- ISO/IEC 12207: Information Technology – SW Life Cycle Processes
- ISO/IEC 15504: Information Technology – Process Assessment (SPICE)
- ISO/IEC 15288: Systems and Software Engineering – System Life Cycle Processes
- ISO 9000 Series – standards related to quality management systems

Establishing a Project Management System

The most important management review system is that conducted by the project management team itself.

If project management is not aware of and not actively involved in project issues, no other management system can be fully effective.

With effective and involved project managers, however, a management system is still needed to resolve conflicts between the projects and between the line and the staff organizations.

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)

CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Advanced Software Process

Part II: The Repeatable Process

6. The Project Plan

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 Project Plan

- Project Planning Principles
- Project Plan Contents
- Size Measures
- Estimating
- Productivity Factors
- Scheduling
- Project Tracking
- The Development Plan
- Planning Models
- Final Considerations

The Project Plan

The project plan defines the work and how it will be done.

- A definition of each major task
- An estimate of the time and resources required
- A framework for management review and control

When a project plan is properly documented, it is a benchmark to compare with actual performance.

Project Planning Principles

1. While requirements are initially vague and incomplete, a quality program can only be built from an accurate and precise understanding of the users' needs.

   The project plan thus starts by mapping the route from vague and incorrect requirements to accurate and precise ones.

Project Planning Principles

2. A conceptual design is then developed as a basis for planning.

   This initial structure must be produced with care since it generally defines the breakdown of the product into units, the allocation of functions to these units, and the relationships among them.

   Since this provides the organizational framework for planning and implementing the work, it is almost impossible to recover from a poor conceptual design.

Project Planning Principles

3. At each subsequent requirements refinement, resource projections, size estimates, and schedules are also refined
4. When the requirements are sufficiently clear, a detailed design and implementation strategy is developed and incorporated in the plan.

Project Planning Principles

5. As various parts of the project become sufficiently well understood, implementation details are established and documented in further plan refinements.
6. Throughout this cycle, the plan provides the framework for negotiating the time and resources to do the job.

Planning Considerations

- With rare exceptions, initial resource estimates and schedules are unacceptable.
- It is best to reach early agreement on the essential functions and to defer the rest until later.
- Quality products are an artistic blend of needs and solutions, requiring harmonious teamwork between the users and the software engineers.

The Planning Cycle

Iterative plan negotiation process [Figure 6.1]

1. The cycle starts with the initial requirements.

2. The response to every demand for a commitment must be “I understand your requirement and will produce a plan with that objective, but without a plan, I cannot make a commitment.”

The Planning Cycle

3. The plan is produced by first breaking the work into key elements, called a Work Breakdown Structure (WBS).
   This implies that a conceptual design has been developed.
4. The size of each product element is estimated.
5. The resource needs are projected.
6. The schedule is produced.

The Planning Cycle

The Software Development Planning Cycle [Figure 6.1]

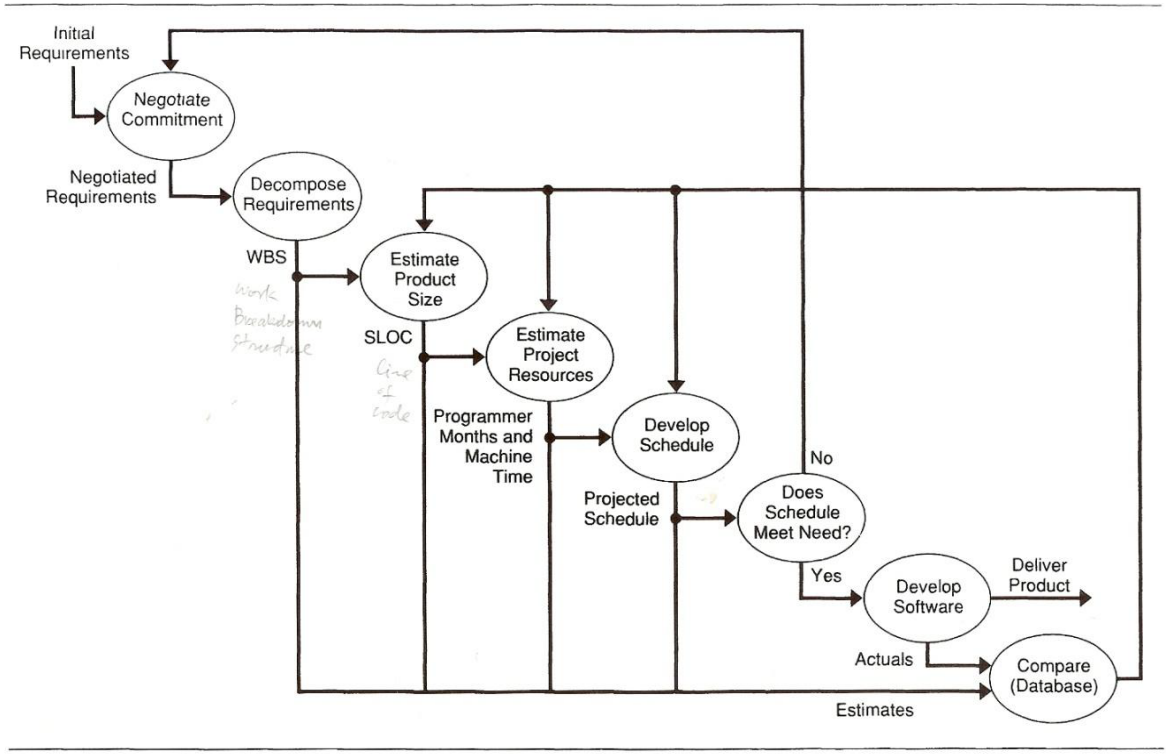

FIGURE 6.1
The Software Development Planning Cycle

Project Plan Contents

- Goals and objectives
- Work Breakdown Structure (WBS)
- Product size estimates
- Resource estimates
- Project schedule

Goals and Objectives

- The project's goals and objectives are established during the requirements phase.
- This is also a negotiation period between the software engineers and the users on what is to be done, how success will be measured, and how much time and resources are needed.

The Work Breakdown Structure (WBS)

- Project planning starts with an estimate of the size of the product to be produced.
- This estimation begins with a detailed and documented breakdown of the product into work elements.
- The breakdown generally has two parts:
  - The product structure
  - The software process
- The WBS then provides the framework for relating them.

Size Measures

- The measure used in program size estimation should be reasonably easy to use early in the project and readily measurable once the work is completed.
- Subsequent comparisons of the early estimates to the actual measured product size then provide feedback to the estimators on how to make more accurate estimates.
- Function points start from the user's perspective and estimate the size of the application [Albrecht 1979].

Line-of-Code Size Estimates

- Line-of-code (LOC) estimates typically count all source instructions and exclude comments and blanks [Jones 1978].
- Automated tools, for example, are often designed to count semicolons.
- Even though it is difficult to estimate lines of code from high-level requirements statements, this measure does facilitate a learning process that leads to improved estimating.

Line-of-Code Size Estimates

- The finer the product structural detail, the more likely relevant examples are available, and the more accurate the subsequent estimates.
- Perhaps the most important advantage of the LOC is that it directly relates to the product to be built.

Function Points

- With function points (FP), initial application requirements statements are examined to determine the number and complexity of the various inputs, outputs, calculations, and databases required.
- By using values that Albrecht (1979) has established, points are assigned to each of these counts.

Function Points

- These points are then summed to produce an overall function point rating for the product.
- Based on prior experience, this final function point figure can be converted into a reasonably good estimate of the development resources required.

Estimating

- Once a size measure has been established, an estimating procedure is needed.
- Probably the most accurate one is the Wideband Delphi Technique originated by the Rand Corporation.
  1. A group of experts is each given the program's specification and an estimation form.
  2. They meet to discuss the product and any estimation issues.
  3. They then each anonymously complete the estimation forms.

Estimating

4. The estimates are given to the estimate coordinator, who tabulates the results and returns them to the experts.
5. Only each expert's personal estimate is identified; all others are anonymous.
6. The experts meet to discuss the results, revising their estimates as they feel appropriate.
7. The cycle continues until the estimates converge to an acceptable range.

Estimating Inaccuracies

- Regardless of the method used, software estimates will be inaccurate.
- The inaccuracies must be compensated for, however, because a poor size estimate results in an inappropriate resource plan and a generally understaffed project.
- To date, at least, software size estimates have almost invariably erred in being too low.
- These low estimates typically result in a last-minute crisis, late delivery, and poor quality.

Estimating Inaccuracies

- There are two types of estimating inaccuracies, and they can both be compensated for, at least to some degree.
- Normal statistical variations can be handled by using multiple estimators.
- The other source of error is estimating bias caused by the project stage at which the estimate is made.
  - The earlier the estimate, the less is known about the product and the greater the estimating inaccuracy.

Estimating Contingencies

- When programmers estimate the code required to implement a function, their estimates are invariably low.
- While there are many potential explanations, the important point is that their optimism is a relatively predictable function of project status.
  - Code may grow at the completion of each project phase.
  - These amount should be added as an estimating contingency at the end of the indicated project phase.

Estimating Contingencies

- Every software size estimate should include a contingency, and the only debate should be over the amount to use.
- Because contingencies inflate the resource estimates, add to the schedule, and increase costs, they are energetically resisted by almost everyone on the project.
- Experience shows, however, that until enough is known to partition the product into modules of a few hundred LOC each, substantial contingencies are required.

A Size Estimating Example

Example Program Size Estimate [Table 6.2]

- Hypothetical satellite ground station control program estimated during the requirements phase

TABLE 6.2
EXAMPLE PROGRAM SIZE ESTIMATE—SATELLITE GROUND STATION CONTROL PROGRAM

Date: 5/25/87 Estimator: WSH Program: Satellite

Component	Base KLOC*	Contingency %	Contingency KLOC*	Total KLOC*
A Executive	9	100	9	18
B Function Calculation	15	100	15	30
C Control/Display	12	100	12	24
D Network Control	12	200	24	36
E Time Base Calculation	18	200	36	54
Total	66	145	96	162

*KLOC = thousands of lines of source code

Productivity Factors

- Once we have an estimate of the amount of code to be implemented, the amount can be converted into the number of programmer months and time required.
- While some programmers can intuitively estimate their work, most of us need a more orderly procedure, particularly when the projects are large and many people are involved.

Productivity Factors

- This involves productivity factors, or standard factors for the number of lines of code that, on average, can be produced per programmer month.
- There are many kinds of potentially useful software productivity factors, and each organization should select those most relevant to their situation.

Productivity Factors

- Productivity data
  - Organization productivity data
  - Developing productivity data
  - Using productivity data
  - Resource estimating
- Organizations can gather their own data and compare it with some of these factors.

Organization Productivity Data

- In productivity calculations, it is important to use factors that relate to the specific organization doing the work and not to use more factors than the available data warrants.
- Since all productivity factors are at best averaged data from several projects done by different people with various levels of ability, it is not wise to try to account for more than a few of the most significant variables.

Organization Productivity Data

- Because so many factors affect productivity and because there is no way to adjust standardized productivity numbers to account for all these variations, each organization should gather its own data to use as a baseline for productivity calculations.
- Adjustment can then be made to account for those variations that are considered most significant.

Organization Productivity Data

- Organizations can develop their own productivity factors by examining a number of recently produced programs, counting their lines of code in a consistent and defined manner, and calculating the programmer months (PM) required to do the job.

Developing Productivity Data

- Since software organizations generally have some records of the work they have produced, the first step in developing productivity data is to see what is available.
- With this information, the following approach should produce the needed data:

Developing Productivity Data

1. Identify a number of recently completed programs that are as similar to the new program as possible in size, language used, application type, team experience, and so forth.
2. Get data on the size, in LOC, of each project. Try to use the same counting scheme for each program.

Developing Productivity Data

3. For modified programs, note the percent of code modified and count only the number of new or changed LOC in the productivity calculations.
   Special management emphasis on the advantages of reuse is needed to counteract the bias toward new code that this measure might cause.

Developing Productivity Data

4. Obtain a count of the programmer months expended for each project, but be sure to include or exclude the same labor categories for each program.
   Generally the categories to include are direct design and implementation programmers, test, and documentation.
   To minimize the number of variables, it is often wise to exclude SQA, managers, clerical, requirements, and computing operations.
   The requirements effort in particular should often be excluded because it is highly dependent on customer relationships and application knowledge.

Using Productivity Data

- With the basic data in hand, it is now possible to derive base productivity numbers.
- Since the data gathered will be for various sizes and classes of programs, some adjustments may be needed.
- With some local data, it is thus reasonable to examine some of the published productivity figures to see which ones might apply to your organization.

Resource Estimating Example

- The actual process of making a resource estimate is quite simple once the size estimate and productivity factors are available.
- Program Resource Calculation Example [Table 6.6]

Scheduling

- Once the total resource needs have been calculated, the project schedule can be developed by spreading these resources over the planned product development phases.
- This is best done by using data on the organization's historical experience with similar projects.
- When such data is not available, publicly available factors can be used for guidance.

Scheduling

- Once the project resource distribution is known, an overall project schedule can be produced as follows:
  - Based on the overall project schedule objective, a staffing plan is developed.
  - A preliminary schedule for each phase is next established by comparing the cumulative resource needs with those expected to be available. An initial schedule is then made.
  - This preliminary plan is then reviewed to ensure that reasonable staffing assignments can be made consistent with this schedule and resource profile. Adjustments are generally required

Project Tracking

- One requirement for sound project management is the ability to determine project status.
- The planning process should thus produce a schedule and enough check-points to permit periodic tracking.
- One way to do this is with earned-value project scheduling [Snyder 1976].
- Schedule Tracking Example [Table 6.9]
- Module Checkpoint Plan [Figure 6.5]

The Development Plan

- After completing the estimates and schedule, the full development plan is assembled in a complete package and submitted to management for review and approval.

The Development Plan

Software development plan [Figure 6.6]

- Project purpose and scope
- Project goals and objectives
- Organization and responsibilities
- Management and technical controls
- Work definition and flow
- Development environment
- Software development methodology
- Configuration management
- Verification and validation
- Quality assurance provisions

Planning Models

- Software cost models can be helpful in making product plan, but they should be used with care.
  - COCOMO (Constructive Cost Model) [Boehm 1981]

    http://sunset.usc.edu/Research_Group/barry.html

    http://cost.jsc.nasa.gov/COCOMO.html
  - SLIM (SW Life Cycle Mgmt) [Putnam 1978]

    http://www.qsm.com/tools/index.html

    http://yunus.hun.edu.tr/~sencer/cocomo.html

Planning Models

- Experience has shown that models, if not calibrated to the specific organization's experience, can be in error by 500 percent or more.
- Models should thus be used to augment the estimating process and not to replace it.

Planning Models

- No model can fully reflect the product's characteristics, the development environment, and the many relevant personnel considerations.
- Once the size and resource estimates have been made, it is necessary to spread these resources over the planned schedule.

Planning Models

- If the organization does not have a good historical basis, a number of modeling techniques can be of help.
- Models such as COCOMO and SLIM, for example, can be used effectively to check the estimates for errors or oversights and to help in assessing risks.
- Under no circumstances, however, should such models be relied on to produce the final estimates.

Final Considerations

- The most important single prerequisite to good software cost estimating is the establishment of an estimating group.
- The group ensures that the cost estimate and actual performance data are retained in a database, that the available data is analyzed to establish the productivity factors and contingencies, and that the development groups are competently assisted in preparing and documenting their plans.

Final Considerations

- When development teams are so supported, they are in the best position to capitalize on their planning experience and to progressively improve their estimating accuracy.
- While experience is the key, experienced SW groups rarely improve their estimating accuracy to this degree without the support of an orderly planning process.

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)