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

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)