CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

Advanced Software Process

Part V: The Optimizing Process

17. Defect Prevention

Dave Garcia-Gomez
Faculty / Lecturer
Department of Computer Science

Course Roadmap

The Optimizing Process

- The organization in the optimizing process maturity level has a foundation for continuing improvement and optimization of the process.

The Optimizing Process

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

Advancing from the Managed Process to the Optimizing Process

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

Defect Prevention

- Defect Prevention Not a New Idea
- The Principles of SW Defect Prevention
- Process Changes for Defect Prevention
- Defect Prevention Considerations
- Management's Role

Defect Prevention

- Many approaches to defect prevention have been devised, such as formal specifications, improved design techniques, and prototyping, and these and other innovations will continue to improve our ability to produce quality software.
  - While the concepts of error cause analysis and defect prevention seem almost too simple to bear discussing, they are not easy to implement because they require that a precise process discipline be used by everyone in the SW organization.
  - As difficult as it is, however, it is the essential element of the optimizing process.

Defect Prevention Not a New Ideas

- Statistical quality control in Japanese industry shortly after World War II.
  - W. E. Deming (1986) and J. M. Juran (1981)
- With different details, these same principles can work for SE.

Why Defect Prevention is Crucial to the Software Process

- Finding and fixing errors accounts for much of the cost of SW development and maintenance.
- When one includes the costs of inspections, testing, and rework, as much as half or more of the typical development bill is spent in detecting and removing errors.
- What is more, the process of fixing defects is even
- more error-prone than original software creation.
- Thus with a low-quality process, the error rate
- spiral will continue to escalate.

Why Defect Prevention is Crucial to the Software Process

- Grady (1987) describes Hewlett-Packard's experiences with defect prevention.
  - More than a third of the errors were due to poor understanding of the users' interface requirements.
  - By establishing an extensive prototyping and design review program, the number of defects found after release was sharply reduced.
  - Their experiences on three classes of defects for four successive are shown in Figure 17.1.
- Figure 17.1 Number of Defects Found Before and After Release

When Errors Are Life-Threatening

- Perhaps the best way to look at defect prevention is to consider how you would act if a single error could threaten your life.
- It is instructive to think about our software problems as if they were highly critical.

The Principles of Software Defect Prevention

- The fundamental objective of software defect prevention is to make sure that errors, once identified and addressed, do not occur again.
  - Defect prevention cannot be done by one or two people, and it cannot be done sporadically.
  - Everyone must participate by faithfully executing the process – almost as if their lives depended on it.
  - As with any other skill, it takes time to learn defect prevention well, but if everyone on the project participates, it can transform an organization.

The Steps of Software Defect Prevention

- Every quality program must start with a clear management commitment to quality.
- This commitment must be explicit and all members of the organization must know that quality comes first.
- Until management delays or redirects a project to meet quality goals, the people will not really believe it.
  - Even then, the point must be reemphasized and the SW engineers urged to propose quality improvement actions, even at the potential cost of schedule delays.

Defect Reporting

- To implement defect prevention, a significant amount of information must be gathered at the time a defect is identified.
- Table 17.1 Data Gathered at the Time a Defect Is Found
- Table 17.2 Data Gathered at the Time a Defect Is Closed
- Table 17.3 Problem Report Form

Error Cause Categories

- When one embarks on defect prevention, one thinks about defects in a new way.
- Previously the emphasis was on how to fix them and, occasionally, on how to find them more effectively.
- This causes one to think about defect categories in terms of symptoms or repair actions.
- The need now is to understand what caused the problems.

Error Cause Categories

- Table 17.4 Percentage of Problems by Types
  - Soloway's research (1986) with novice programmers suggests that only a few error categories need be examined to cover a large percentage of the problems.
  - Somewhere between 10 percent to 20 percent of the bug categories included about half of all the errors.

Error Cause Categories

- Six categories of errors for defect prevention analysis [Endres 1975]
  - Technological
  - Organizational
  - Historical
  - Group dynamic
  - Individual
  - Other causes and inexplicable causes

Cause Analysis

- Cause analysis should be conducted as early as possible after a defect is found.
- Some delay may be necessary, however, to assemble the people required to do an effective job.
- Also, to be most efficient, a modest backlog is required for each analysis session.
- Since many SW defects generally stem from a relatively few causes, this grouping tends to make the cause analysis sessions much more efficient.

The Cause Analysis Meeting

- Cause analysis meetings are similar in many ways to inspections.
  - They have a leader and a recorder and include the involved programmers and some of their peers.
  - The meeting should not be too large (five to eight professionals at most) and should not include managers.
  - The result of the meeting is a report on each defect, and a report on the meeting itself.
  - Also, a summary cross index should be prepared of the defect report numbers with the action recommendation numbers.

The Cause Analysis Meeting

- Table 17.5 Defect Analysis Report
- Table 17.6 Causal Analysis Meeting Agenda
- Table 17.7 Action Plan Analysis

The Cause Analysis Meeting

- Gray and Caswell (1987) report an interesting example of the cause analysis process on a compiler project.
- Figure 17.2 Compiler Defects
- Figure 17.3 Compiler Design Defects
- Figure 17.4 Cause/Effect Diagram

The Action Team

- Up to this point the main concerns have been with understanding problem causes and how they might be fixed.
- Next, the actual preventative actions are initiated.
- This is no longer a study by a small team of technical professionals.
- Managers are involved, responsibilities are assigned, and implementation progress is tracked.

Action Team Meeting

- The action team meeting starts with an initial review of the recommendations made to date and a decision on the priority of the items to be addressed.
- Since there are probably many more items than can be reasonably handled at one time, the most important areas are addressed first, and actions are promptly launched to handle them.
- As success is achieved with these initial actions, additional items can be handled.

Examples Actions

- While defect prevention has been used in various branches of manufacturing and engineering for many years, few realize how much it has driven the field of SW.
- Table 17.8 Example Defects and Preventive Actions
- Some actions are simple and can be implemented quite quickly.
- Others, such as initiating or modifying a training program, might take more effort.

Tracking Action Progress

- Every action must be logged and tracked and its status periodically reported.
- There should be follow-up reports, aging statistics, open item lists, and priority hot lists.
- Every delay in prevention action permits more errors to be made.
- Once a prevention action has been approved, it thus should be instituted with all possible speed.
- Table 17.9 Action Records

Prevention Feedback

- Perhaps the most important single action to come out of any defect prevention program is the awareness effort.
- This ensures that all SW professionals are aware of the major error categories and the actions they can take to prevent or detect them at the earliest possible time.
- The awareness effort should involve training programs for the new tools and methods and special seminars on the most recent prevention experiences and plans.

Process Changes for Defect Prevention

- The process changes required to incorporate defect prevention are shown in Figure 17.5.
- Figure 17.5 Defect Prevention Process
  - Kickoff meeting
  - Task
  - Causal analysis review
  - Improvement suggestion
  - Priorities
  - Feedback

Process Changes for Defect Prevention

- With defect prevention there are several process levels to consider.
  - Actual work of developing the SW
  - The process model that provides the framework for project analysis and control
  - To change the SW development and maintenance process

Defect Prevention Considerations

- Some questions frequently asked about defect prevention are:
  - Where should one start?
  - What is the role of tools and technology?
  - What does it cost?
  - What is management’s role?

Where to Start?

- While there is no magic answer to the question of where to start, there is compelling evidence that software modification is the most fruitful area for improving software development and maintenance.
  - Collofello and Buck (1987)
    - 62 Percent of the defects in their systems were due to incorrectly implemented changes.
  - Table 17.10 Distribution of Defect Origin
    - 52% of the errors were regressions to functions that had previously worked.

The Role of Tools and Technology

- While the subject of tools and technology is covered in more detail in the next chapter, cause analysis provides an informed way to select parts of the software process for automation.
  - This approach requires a precise definition of the tasks to be automated, why they are important, and their current problems.
  - When this work is done in the context of a structured and quantified process, the resulting tools are more likely to be effective in assisting the professionals to do their jobs.

The Costs and Benefits of Defect Prevention

- Most organizations worry about the cost of defect prevention.
- While this is a reasonable concern, it only makes sense when considered in light of the costs of not preventing them.
- For software, much as with hardware, this generally includes the cost of testing, repair, customer dissatisfaction, and warranty and field service.
- Table 17.11 Hardware Prevention, Appraisal, and Failure Costs.

Management's Role

- Table 17.12 gives Crosby's list of questions for managers and supervisors to help them understand their quality situation and whether they are prepared to take action [Crosby 1979].
- Table 17.12 Quality Improvement Self- Evaluation
  - Software managers should answer these questions, and then, unless satisfied with the result, proceed to implement the actions discussed in this chapter.

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)