CALIFORNIA STATE UNIVERSITY
FULLERTON™

COLLEGE OF ENGINEERING
AND COMPUTER SCIENCE

# Advanced Software Process

## Part IV: The Managed Process

### *15. Data Gathering and Analysis*

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 Managed Process

# The Managed Process

- Once the topics of Part III (the Defined Process) are in place, the software organization will have a firm foundation for introducing measurement and statistical management.
- While the topics from prior sections must continue to be refined and improved, management's attention should now turn to quantitative planning and process control.
- Software Process Data Gathering [Ch.15]
  - This data meets the basic management needs for quantitative analysis and establishes the foundations for software quality planning and management.
- Software Quality Plan [Ch.16]
  - Quality measures, quality estimates, the problems of tracking and controlling quality

# The Managed Process

- The key steps required to advance from the Defined Process to the Managed Process
  - Establish a minimum basic set of process measurement to identify the quality and cost parameters of each process step.
  - Establish a process DB 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.

# Data Gathering and Analysis

- The Principles of Data Gathering
- The Data Gathering Process
- Software Measures
- Data Analysis
- Other Considerations

# Data Gathering and Analysis

- We gather and analyze software data to help us improve the SE process.
- As we face increasingly demanding SW projects, we need to understand more precisely what we are doing and how to improve our effectiveness.
  - Data gathering is expensive and time-consuming.
  - If affects the busiest people and may even be viewed as personally threatening.
  - There is also considerable confusion on what data to gather and how to use it.

# The Principles of Data Gathering

- The data is gathered in accordance with specific objectives and a plan.
- The choice of data to be gathered is based on a model or hypothesis about the process being examined.
- The data gathering process must consider the impact of data gathering on the entire organization.
- The data gathering plan must have management support.

# The Objectives of Data Gathering

Four Basic Reasons for Collecting Software Data

- Understanding
  - As part of a research or development study, data can be gathered to learn about some item or process.
- Evaluation
  - The data can be used to study some product (or activity) to see if it meets acceptance criteria.
- Control
  - The data can be used to control some activity.
- Prediction
  - The data can be used to develop rate or trend indicators.

# Data Gathering Models

- To measure the software process successfully, one must almost start by knowing the expected results.
- With a poorly understood process, it is hard to be precise about the data that is needed.
- After some study and evaluation, however, assumptions can usually be made about the key process events and their relationships.
- Data is then gathered to verify these assumptions.
- These measures are related to this conceptual model, the variations are examined, and changes are made to the process or the model to bring them into closer agreement.

# The Impact of Data Gathering on the Organization

- There are two fundamental issues that data gathering plan must consider:
  - The effects of measurement on the people
    - When people know they are being observed, their performance will generally change.
  - The effects of the people on the measurements
    - It is essential to motivate the professionals to gather the needed data in a timely way.

# Management Support

- Because it is so expensive and because there is often considerable delay before the benefits of data gathering are apparent, data gathering must be viewed as an investment.
- As with any other investment, senior management support is essential to keep the needed resources in place.
- Once useful results have been produced, however, the project managers generally will be more willing to support the work.

# The Data Gathering Process

- Because of the issues of resources, cost and accuracy, people often see data gathering as an expensive waste of time.
- Project managers are reluctant because they do not see the direct value to their projects and are hesitant to insist on any long-term efforts that may affect immediate priorities.
- This is why the data gathering program must start with management support and be thoroughly planned and carefully validated.

# The Data Gathering Process

- The SE Lab at the NASA Goddard Space Flight Center has gathered SE data for a number of years.
  - Cost of Data Collection (1982) [Table 15.1]
  - 747 different measures on a number of projects
- Their experience shows that data gathering can be expensive, particularly if it is done as part of a research program.

# Managing the Data Gathering Process

- Cost of Data Collection (1982) [Table 15.1]
  - Overhead to tasks (experiments) 3 – 7%
    - Forms, meetings, training, interviews, cost of using tools
  - Data processing 10 – 12%
    - Collecting/validating forms, archiving/entering data, data management and reporting
  - Analysis of information Up to 25%
    - Designing experiments, evaluating experiments, defining analysis tools

# Managing the Data Gathering Process

- The reason it is so expensive to gather and analyze software data is that this is manual work by people who are inexperienced at data gathering.
- The effort of recording time spent per task, results produced, or defects found is time-consuming.
- With increasing experience and improved tools, however, these costs can be reduced.

# Managing the Data Gathering Process

- Since data gathering is expensive its process must be carefully planned and managed.
- The data must be precisely defined to ensure that the right information is obtained and it must be validated to ensure that it accurately represents the process.
- The data must be retained by someone who owns and maintains the process DB.

# Managing the Data Gathering Process

- Summary of Data Gathering Problems [Alter 1980] [Table 15.2]
  - Problem, typical cause and possible solutions
    - Data is not correct.
    - Data is not timely.
    - Data not measured or indexed properly
    - Too much data needed
    - Needed data does not exist.

# The Data Gathering Plan

- The data gathering plan should be produced by the SEPG with the help of the projects and the participation of some of the professionals who will gather the data.
  - What data is needed, by whom, and for what purpose?
  - What are the data specifications?
  - Who will gather the data?
  - Who will support data gathering?
  - How will the data be gathered?
  - How will the data be validated?
  - How will the data be managed?

# Data Validation

- Software data is highly error-prone and special validation provisions are generally needed.
  - The data must either be gathered automatically or extensively validated.
- When people have been trained on the definitions and data gathering methods, periodic spot checks will generally suffice.
- People who do this validation must be able to explain the data definitions, the data gathering methods, and the program's objectives.

# Software Measures

- With this background we next look at some specific software measures:
  - Characteristics of several useful software measures
  - Examples of software data

# Data Characteristics

- To be of most value, software process measures should have the following characteristics:
  - The measures should be robust.
  - The measures should suggest a norm.
  - The measures should relate to specific product or process properties.
  - The measures should suggest an improvement strategy.
  - They should be a natural result of the process.
  - The measures should be simple.
  - They should be both predictable and trackable.

# Data Characteristics

- Software measures can be classified in several ways:
  - Objective/subjective
  - Absolute/relative
  - Explicit/derived
  - Dynamic/static
  - Predictive/explanatory
- The SE objective for data gathering and analysis to use an increasing number of objective, absolute, explicit and dynamic measures to control and improve the way the work is done.

# Software Size Measures

- Size measures are important in SE because the amount of effort required to do most tasks is directly related to the size of the program involved.
  - Unfortunately, there is no generally accepted measure of program size that meets all the previously described criteria.
  - There is no simple measures of software size because software size is not a simple subject.

# Lines of Code

- The line of code (LOC) measure is probably most practical for measuring program size.
- The major risk is that its unthinking use can motivate maximization of LOC counts.
- The only solution is to use size measures as guides and not for evaluation of people or organization.
- In any case they must be used carefully and with judgment.

# Lines of Code

- Alternative ways of counting LOC:
  - Executable lines
  - Executable lines plus data definitions
  - Executable lines, data definitions, and comments
  - Executable lines, data definitions, comments, and JCL
  - Physical lines on an input screen
  - Logical delimiters, such as semicolons

# Normalized LOC Measures

- Another problem with LOC counts is that with high-level languages they do not recognize the increased functional content of the source lines of codes.
- Some organizations compensate for this by using factors to generate equivalent assembly language.
- To be truly representative, such factors must reflect both the source and assembly languages being used, as well as the particular program functions involved.

# Error Data

- Errors, Defects, Bugs, Faults, and Problems [Table 15.3]

| Category | Item Measured | Causes |
|---|---|---|
| Errors | Human actions | Programmer mistakes |
| Defects | Program properties | Errors |
| Bugs | Program malfunctions | Program defects |
| Failures | System malfunctions | Bug and other malfunctions |
| Problems | Human perceptions | Failure, human errors, human misconceptions |

# Error Data

- Errors, Defects, Bugs, Failures, and Problems [Figure 15.1]

# The Process of Problem Analysis

- Basically the software error measurement problem is one of distinguishing between causes and effects.
- Since our intent is both to correct and to prevent the errors that programmers make, we must work our way up this chain of possible measurable events.

# Classes of Defect Measures

- Since software defects are a major concern in both testing and operation, it is natural to use them as one key process measurement.
- Software defects (and the bugs that identify them) can be categorized as follows:
  - severity, symptoms
  - where found, when found, how found
  - where caused, when caused, how caused
  - where fixed, when fixed, how fixed

# Classes of Defect Measures

- Error Categories [Table 15.4]
- Problem Discovery and Resolution Data [Table 15.5]
- Frequency of Error Occurrence [Table 15.6]
- Types of Design Errors [Table 15.7]
- Types of Coding Error [Table 15.8]

# Productivity Data

- How much effort is spent in each task of the software process?
- For each product element, what effort was devoted to each process task?
- What is the relative cost of defect removal by the various tests and inspections and by each major error category?
- How effective are various methods and technologies in improving the productivity of each process task?
- What is the resource history by major process task, and how should the planning factors be adjusted for subsequent projects?

# Organizational Productivity

- Productivity measures are most prone to misuse when they are applied at the highest project and organizational level.
- It is, however, desirable to have some measure of how the organization is performing over time.
- For software unfortunately, the only simple measures available are lines of code and effort expended.

# Sensitivity of Productivity Measures

- Why software productivity numbers vary so widely among several companies?
  - There are significant differences among the organizations' definitions of lines of code and labor expended.
- It is clear that productivity data can be highly misleading unless all the parameters are strictly defined and controlled.

# Data Analysis

- While many kinds of analysis are possible, a few examples here indicate some ways data can be used to support software development and maintenance.
- As software organizations gather data and begin to use it, they will find many more ways to apply it.

# Error Data Analysis

- An exhaustive study of data analysis on the IBM DOS OS performed by Endres (1975) includes considerable detail on their error cause analysis results.
  - They concluded from this data that only about half of the mistakes can be avoided with better programming tools and techniques.
  - They found that an important benefit of gathering and analyzing such data is the focus it provides on the areas with the greatest likelihood of causing problems.

# Error Data Analysis

- [Table 15.9] Questions to be Answered by IBM DOS Error Study
- [Table 15.10] Number of Modules Affected by An Error
  - Three largest modules had the largest numbers of errors.
- [Table 15.11] Number of Errors Per Module
  - The 21% of the modules that had more than one error accounted for 78% of the total errors.
- [Table 15.12] Frequency of Errors
  - The changed code had a 60% higher error rate than new code (7.8 vs. 4.8).

# Test Data Analysis

- Many different analysis can be made of test data.
  - Typically, one can look at defect data rather than examining the error causes for programming defects.
  - Some example questions:
    - What is the relationship between test coverage by program module and the number of defects detected and remaining in the program?
- Since test data can be obtained relatively easily, testing is one of the best place to start gathering and analyzing the SE process.

# Analysis of Inspection Rates

- Wenneson (1985) describes using statistics for tracking, estimating, planning, and scheduling development and QA work.
- Nominal Inspection Rates (FORTRAN design and code inspection) [Table 15.13]
  - Significant deviation from these values were cause for review of the way the inspection was conducted.

# Analysis of Inspection Rates

- Wenneson (1985) describes using statistics for tracking, estimating, planning, and scheduling development and QA work.
- Nominal Inspection Rates (FORTRAN design and code inspection) [Table 15.13]
  - Significant deviation from these values were cause for review of the way the inspection was conducted.

# Analysis of Inspection Rates

- Errors Detected Versus Inspection Rate [Figure 15.2]
  - The errors found per KLOC decline with increasing inspection rate.
  - While it is true that code with a large number of errors is harder to review, this effect probably tails off at about 400 LOC/hour.
  - A rough upper limit of 0.3 to 0.4 KLOC/hr was probably about optimum for these FORTRAN programs.

# Analysis of Inspection Rates

- Inspection Rate Versus Preparation Rate [Figure 15.3]
  - Inspection preparation time should roughly equal review time.
- Upper and Lower Ranges of Inspection Rates Versus Preparation Times [Figure 15.4]
  - The upper and lower limits of inspection rates with various preparation times are plotted.
  - The LOC/hour of inspection plus preparation is thus between 60 and 100.

# Inspection Data Analysis

- Schulmeyer and McManus (1987) have provided some data on design and code inspections that can be analyzed with statistical techniques.
  - Selected Design Inspection Data [Table 15.14]
- To facilitate analysis, the statistical control charts are next constructed.

# Constructing Statistical Control Charts

- Inspection Control Chart – Lines of Code Inspected per Hour [Figure 15.5]
- Inspection Control Chart – Defects per Total Inspection Hours [Figure 15.6]
- Inspection Control Chart – Defects per Thousand Lines of Code [Figure 15.7]
- Inspection Control Chart – Hours of Preparation per Hour of Inspection [Figure 15.8]

# Constructing Statistical Control Charts

- Percentage of Values Within Control Limits [Table 15.15]
  - Confidence interval
    - 68% within average +/- 1*SD
    - 95% within average +/- 2*SD
  - SD: standard deviation
    - [average of $(values)^2$ – $(average\ of\ values)^2]^{1/2}$
  - UCL: average + 2*SD
  - LCL: average – 2*SD (0 if negative)

# Using Control Charts

- The reason UCL and LCL values are important is that we need to identify unusual conditions and control them as we try to improve our processes.
- It is thus important to screen out those values that are probably due to purely random variation and examine the rest.

# Statistical Analysis of Test Data

- Card (1987) has reported an example of how these same statistical techniques can be applied to test data.
- Process Control Results [Figure 15.9]

# Other Considerations

- Since it is easy to draw erroneous conclusions from poorly conducted measurements, it is important to remember some key points.
  - Since data can generally be interpreted in many ways, listen to what the data is saying rather than using it to reinforce preconceived opinions.
  - Provide resources to validate and analyze the data.
  - Make sure that process data is never used to evaluate people.
  - The simple act of gathering data will change the process.

# Other Considerations

- Start carefully and don't attempt to do too much at one time.
- Seek the views of the data collectors on the data collection process and any modifications, enhancements, or support they need.
- Provide the initial results to the people the data concerns before giving it to anyone else.
- Start with a comprehensive plan.
- The debate on data definitions and formats can be endless.

# 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)