Software M&M – II
Process & Project Metrics
A Good Manager Measures

What do we use as a basis?
- size?
- function?
Why Do We Measure?

• assess the status of an ongoing project
• track potential risks
• uncover problem areas before they go “critical,”
• adjust work flow or tasks,
• evaluate the project team’s ability to control quality of software work products.
• We measure the efficacy of a software process indirectly.
  – That is, we derive a set of metrics based on the outcomes that can be derived from the process.
  – Outcomes include
    • measures of errors uncovered before release of the software
    • defects delivered to and reported by end-users
    • work products delivered (productivity)
    • human effort expended
    • calendar time expended
    • schedule conformance
    • other measures.

• We also derive process metrics by measuring the characteristics of specific software engineering tasks.
Process Metrics Guidelines

• Use common sense and organizational sensitivity when interpreting metrics data.
• Provide regular feedback to the individuals and teams who collect measures and metrics.
• Don’t use metrics to appraise individuals.
• Work with practitioners and teams to set clear goals and metrics that will be used to achieve them.
• Never use metrics to threaten individuals or teams.
• Metrics data that indicate a problem area should not be considered “negative.” These data are merely an indicator for process improvement.
• Don’t obsess on a single metric to the exclusion of other important metrics.
Software Process Improvement

Process model

Improvement goals

Process metrics

SPI

Process improvement recommendations
Process Metrics

- **Quality-related**
  - focus on quality of work products and deliverables

- **Productivity-related**
  - Production of work-products related to effort expended

- **Statistical SQA data**
  - error categorization & analysis

- **Defect removal efficiency**
  - propagation of errors from process activity to activity

- **Reuse data**
  - The number of components produced and their degree of reusability
Project Metrics

• used to minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks
• used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality.
• every project should measure:
  – inputs—measures of the resources (e.g., people, tools) required to do the work.
  – outputs—measures of the deliverables or work products created during the software engineering process.
  – results—measures that indicate the effectiveness of the deliverables.
Typical Project Metrics

• Effort/time per software engineering task
• Errors uncovered per review hour
• Scheduled vs. actual milestone dates
• Changes (number) and their characteristics
• Distribution of effort on software engineering tasks
Typical Size-Oriented Metrics

- errors per KLOC (thousand lines of code)
- defects per KLOC
- $ per LOC
- pages of documentation per KLOC
- errors per person-month
- errors per review hour
- LOC per person-month
- $ per page of documentation
Typical Function-Oriented Metrics

- errors per FP (thousand lines of code)
- defects per FP
- $ per FP
- pages of documentation per FP
- FP per person-month
# Comparing LOC and FP

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>LOC per Function point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg.</td>
</tr>
<tr>
<td>Ada</td>
<td>154</td>
</tr>
<tr>
<td>Assembler</td>
<td>337</td>
</tr>
<tr>
<td>C</td>
<td>162</td>
</tr>
<tr>
<td>C++</td>
<td>66</td>
</tr>
<tr>
<td>COBOL</td>
<td>77</td>
</tr>
<tr>
<td>Java</td>
<td>63</td>
</tr>
<tr>
<td>JavaScript</td>
<td>58</td>
</tr>
<tr>
<td>Perl</td>
<td>60</td>
</tr>
<tr>
<td>PL/1</td>
<td>78</td>
</tr>
<tr>
<td>Powerbuilder</td>
<td>32</td>
</tr>
<tr>
<td>SAS</td>
<td>40</td>
</tr>
<tr>
<td>Smalltalk</td>
<td>26</td>
</tr>
<tr>
<td>SQL</td>
<td>40</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>47</td>
</tr>
</tbody>
</table>

Representative values developed by QSM
Why Opt for FP?

- Programming language independent
- Used readily countable characteristics that are determined early in the software process
- Does not “penalize” inventive (short) implementations that use fewer LOC than other more clumsy versions
- Makes it easier to measure the impact of reusable components
Object-Oriented Metrics

- Number of **scenario scripts** (use-cases)
- Number of **support classes** (required to implement the system but are not immediately related to the problem domain)
- Average number of **support classes per key class** (analysis class)
- Number of **subsystems** (an aggregation of classes that support a function that is visible to the end-user of a system)
WebApp Project Metrics

- Number of **static Web pages** (the end-user has no control over the content displayed on the page)
- Number of **dynamic Web pages** (end-user actions result in customized content displayed on the page)
- Number of **internal page links** (internal page links are pointers that provide a hyperlink to some other Web page within the WebApp)
- Number of **persistent data objects**
- Number of **external systems interfaced**
- Number of **static content objects**
- Number of **dynamic content objects**
- Number of **executable functions**
Measuring Quality

• **Correctness** — the degree to which a program operates according to specification
• **Maintainability** — the degree to which a program is amenable to change
• **Integrity** — the degree to which a program is impervious to outside attack
• **Usability** — the degree to which a program is easy to use
Defect Removal Efficiency

\[ \text{DRE} = \frac{E}{E + D} \]

where:

- \( E \) is the number of errors found before delivery of the software to the end-user
- \( D \) is the number of defects found after delivery.
Metrics for Small Organizations

- time (hours or days) elapsed from the time a request is made until evaluation is complete, $t_{queue}$.
- effort (person-hours) to perform the evaluation, $W_{eval}$.
- time (hours or days) elapsed from completion of evaluation to assignment of change order to personnel, $t_{eval}$.
- effort (person-hours) required to make the change, $W_{change}$.
- time required (hours or days) to make the change, $t_{change}$.
- errors uncovered during work to make change, $E_{change}$.
- defects uncovered after change is released to the customer base, $D_{change}$. 
Establishing a Metrics Program

- Identify your business goals.
- Identify what you want to know or learn.
- Identify your subgoals.
- Identify the entities and attributes related to your subgoals.
- Formalize your measurement goals.
- Identify quantifiable questions and the related indicators that you will use to help you achieve your measurement goals.
- Identify the data elements that you will collect to construct the indicators that help answer your questions.
- Define the measures to be used, and make these definitions operational.
- Identify the actions that you will take to implement the measures.
- Prepare a plan for implementing the measures.