# TOPICS

Overview

Metrics

Estimation

Planning

# SOFTWARE METRICS

Measuring Software

Why Measure Software?

Two Types of Measurements

Categories of Metrics

Size-Oriented Metrics

Function Points

Feature Points

Function-Oriented Metrics

Measuring Software Quality

Relationship of LOC to FP

Use of Productivity Data

Integrating Metrics into the Software Engineering Process

Collecting Software Metrics

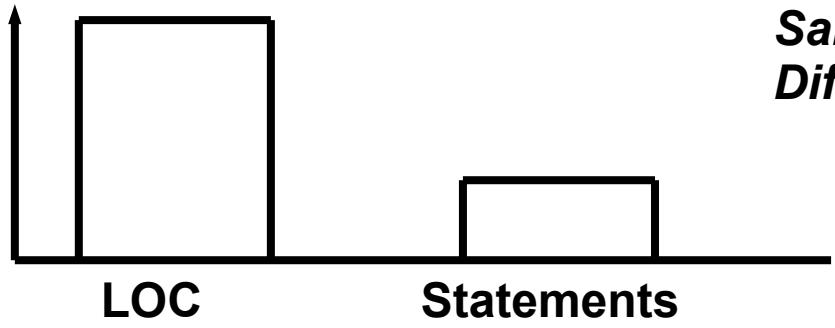
# Measuring Software

**Objectively measuring software is difficult.**

- **Most projects use only "lines of code" (LOC) for metrics.**
- **Much disagreement exists on what and how much to measure.**

## but

**Accurately measuring software is vitally important to tracking and controlling software development.**




***Same Project***
***Different Metrics***

# Why Measure Software?

To --

1. identify quality of the software product
2. assess productivity of the software developers
3. assess benefits of using development processes and tools
4. form a baseline for estimation
5. justify requests for tools and training

# Two Types of Measurements

## Direct

- -- cost
- -- LOC
- -- execution speed
- -- binary code size
- -- memory used

- easy to make

## Indirect

- -- functionality
- -- quality
- -- "-ilities"

- not easy to make

# Categories of Metrics

# Size-Oriented Metrics

Let *KLOC* = "thousand lines of code"

Then we can define

*productivity = KLOC / person-months*

*quality = defects in code / KLOC*

*cost = dollars / KLOC*

*documentation = pages of documents / KLOC*

Efforts and costs include all elements of software development (analysis, design, code, test, *etc.*).

# Size-Oriented Metrics - Examples

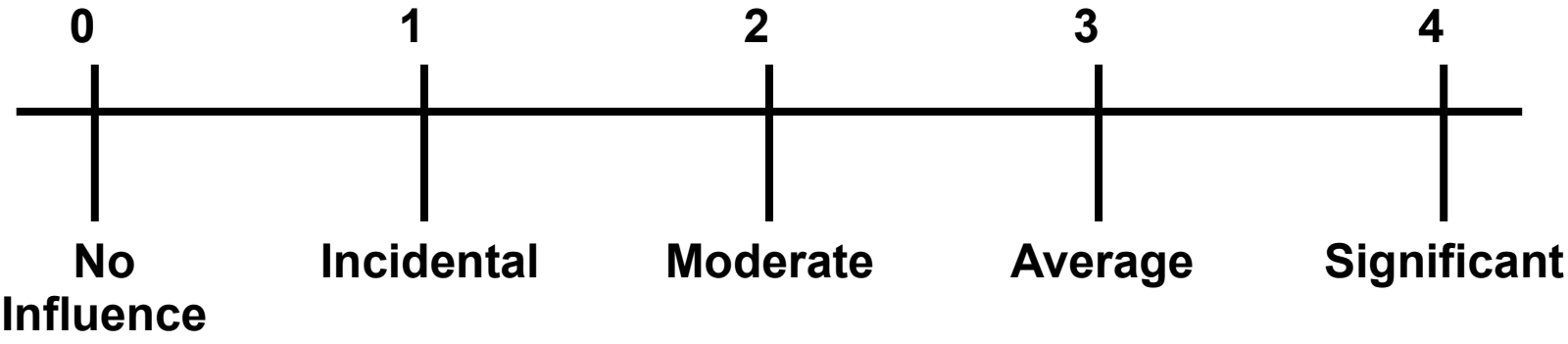
| Project | Person-Months | Cost | KLOC | Pages of Doc | Errors |
|---|---|---|---|---|---|
| A | 24 | $168,000 | 12.1 | 365 | 29 |
| B | 62 | $440,000 | 27.2 | 1224 | 86 |
| C | 43 | $314,000 | 20.2 | 1050 | 64 |

| Project | Productivity ($/LOC) | Quality (KLOC/p-months) (pages/KLOC) | Cost | Documents (errors/KLOC) |
|---|---|---|---|---|
| A | 0.504 | 2.40 | $13.88 | 30.17 |
| B | 0.439 | 3.55 | $16.18 | 45.00 |
| C | 0.470 | 3.67 | $15.54 | 51.98 |

# Problems with Size-Oriented Metrics

- Definition of "lines of code"
  - Programming language dependent
  - Penalize well-designed shorter programs
  - Cannot easily accommodate non-procedural languages
  - Difficult to assess LOC before a program is written
- Only known errors can be counted
- Types, skill levels, and productivity of personnel varies

# Function Points - Fi Values




1. Does the system require reliable backup? 8. Are the master files updated on-line?
2. Are data communications required? 9. Are the inputs, outputs, files, or inquiries complex?
3. Are there distributed processing functions? 10. Is the internal processing complex?
4. Is performance critical? 11. Is the code designed to be reusable?
5. Will the system run in an existing environment? 12. Are conversion and installation included in design?
6. Does the system require on-line data entry? 13. Is the system designed for multiple installations in
7. Does the on-line data entry require the input organizations?

# Function Points - Computation

| Measurement Parameter | Count | | Weighting: Simple | Average | Complex | | Factor: Product |
|---|---|---|---|---|---|---|---|
| Number of user inputs | | x | 3 | 4 | 6 | = | |
| Number of user outputs | | x | 4 | 5 | 7 | = | |
| Number of user inquiries | | x | 3 | 4 | 6 | = | |
| Number of files | | x | 7 | 10 | 15 | = | |
| Number of external interfaces | | x | 5 | 7 | 10 | = | |

Count - Total →

$$FP = count - total(0.65 + 0.01 \sum F_i)$$

# Feature Points

## Function Point Extensions for Technical Software

Function points were originally designed for business information systems applications.

Extensions called *feature points* apply to technical software applications.

Algorithms are a bounded computational problem that is included within a specific computer program.

# Feature Points - Computation

| Measurement Parameter | Count | Weight | | Product |
|---|---|---|---|---|
| Number of user inputs | | x 4 | = | |
| Number of user outputs | | x 5 | = | |
| Number of user inquiries | | x 4 | = | |
| Number of files | | x 7 | = | |
| Number of external interfaces | | x 7 | = | |
| Algorithms | | x 3 | = | |

Count - Total →

$$FP = count - total(0.65 + 0.01 \sum F_i)$$

# Problems with Function Points and Feature Points

1. These metrics are based on subjective data.
2. Parameters can be difficult to obtain after-the-fact.
3. Function and Feature Points have no direct physical meaning.

# Function-Oriented Metrics

Focus is on "functionality" or "utility"

Both Function Points and Feature Points support the derivation of potentially useful data for the comparison of one project to another:

Productivity = FP / person-month

Quality = defects / FP

Cost = $ / FP

Documentation = pages / FP

# Measuring Software Quality

- Before Delivery
  - Program complexity
  - Effective modularity
  - Program size
- After Delivery (most widely used)
  - Number of defects uncovered in the field
  - Maintainability of the system

# "After Delivery" Quality Metrics

***Correctness*** **- defects/KLOC or defects/FP over a one-year period**

***Maintainability*** **- mean-time-to-change (MTTC), which is the time required to:**

**analyze the change request,**

**design a modification to the software,**

**implement the change,**

**test the changed software and the system as a whole, and**

**distribute the changed system to the users**

# "After Delivery" Quality Metrics, Continued

*Integrity* - based on threats and security

*Threat* - probability that a specific attack will take place within a given period of time

*Security* - probability that the attack of a specific type will be repelled

$$\text{Integrity} = \sum_{\text{allthreats}} (1 - \text{threat}(1 - \text{security}))$$

*Useability* - based on several perceptions of the users:

skill required to use the program

time required to learn the use of the program

the increase in productivity from using the program

the user's attitude towards the program

# Relationship of LOC to FP

The relationship of lines of code to feature points is a function of the programming language used and the quality of the design.

Rough estimates of the number of lines of code to create on feature point are:

| *Language* | *LOC/FP* |
|---|---|
| Assembly | 300 |
| COBOL | 100 |
| FORTRAN | 100 |
| Pascal | 90 |
| Ada | 70 |
| Object-Oriented Languages | 30 |
| Fourth Generation Languages | 20 |
| Automatic Code Generators | 15 |

# Use of Software Productivity Data

**Do not use LOC/person-month or FP/person-month to:**

- **Compare one group of developers to another**
- **Rate the performance of an individual**

**Many factors affect productivity:**

| *Factor* | *Approximate % Variation in Productivity* |
|---|---|
| People (number, experience) | 90% |
| Problem (complexity, number of changes) | 40% |
| Process (language, CASE) | 50% |
| Product (reliability, environment) | 140% |
| Resources (CASE, hardware, software) | 40% |

# Integrating Metrics into the Software Engineering Process

A historical baseline of metrics data is needed:

- Company, department, or unit should be identified in the scope of this data.
- Resistance to data collection should be expected in many corporate cultures.

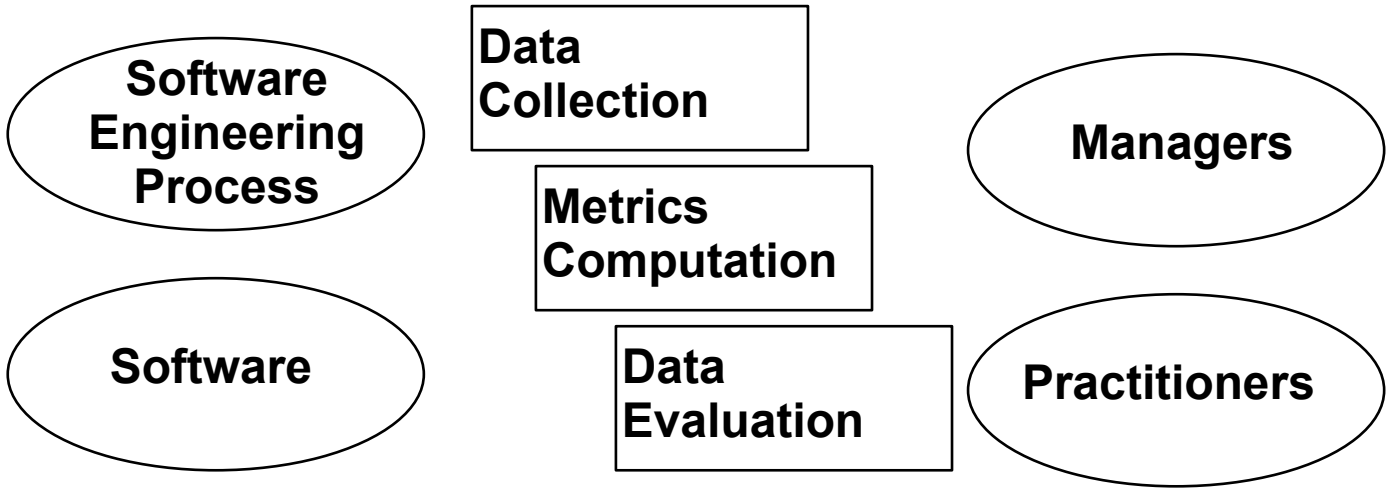
At least three years of accurate, standardized metric data collection is needed to produce accurate planning estimates.

# Collecting Software Metrics

The process of collecting and using software metrics includes the following steps:

1. data collection
2. metrics computation
3. data evaluation

The following slides show a spreadsheet model for the collection and computation of historical software baseline data.



# Spreadsheet Data Collection Model

| *Description* | *Units* | *Sample Data* |
|---|---|---|
| Cost Data Input | | |
| Labor cost | $/person-month | $7,744 |
| Labor year | hours/year | 1560 |
| Data for Metrics Computation | | |
| Release type | alphanumeric | maintenance |
| Number of staff members | people | 3 |
| Effort | person-hours | 4800 |
| Elapsed time to complete | hours | 2000 |
| Source code | KLOC | |
| Newly developed | | 11.5 |
| Modified | | 0.4 |
| Reused | | 0.8 |
| Delivered | | 33.4 |

# Spreadsheet Data Collection Model

| *Description* | *Units* | *Sample Data* |
|---|---|---|
| Data for Metrics Computation, Continued | | |
| Documentation | pages | |
| Technical | | 265 |
| User | | 122 |
| Number of errors to date | numeric | |
| Critical errors | | 0 |
| Level 1 errors | | 12 |
| Level 2 errors | | 14 |
| Documentation errors | | 40 |
| Maintenance to date | person-hours | |
| Modifications | | 3550 |
| Error correction | | 1970 |

# Spreadsheet Data Collection Model

| *Description* | *Units* | *Sample Data* |
|---|---|---|
| Project Data | % of total | |
| Analysis and specification | | 18% |
| Design | | 20% |
| Coding | | 23% |
| Testing | | 25% |
| Other - Describe | | 14% |

# Spreadsheet Data Collection Model

| *Description* | *Units* | *Sample Data* |
|---|---|---|
| **Function-Oriented Data** | | |
| **Information Domain** | | |
| **1. No. of user inputs** | **inputs** | **24** |
| **2. No. of user outputs** | **outputs** | **46** |
| **3. No. of user inquiries** | **inquiries** | **8** |
| **4. No. of files** | **files** | **4** |
| **5. No. of ext. interfaces** | **interfaces** | **2** |
| **Weights** | | |
| **1. No. of user inputs** | **3, 4, 6** | **4** |
| **2. No. of user outputs** | **4, 5, 7** | **4** |
| **3. No. of user inquiries** | **3, 4, 6** | **6** |
| **4. No. of files** | **7, 10, 15** | **10** |
| **5. No. of ext. interfaces** | **5, 7, 10** | **5** |

# Spreadsheet Data Collection Model

| *Description* | *Units* | *Sample Data* |
|---|---|---|
| **Function-Oriented Data, Continued** | | |
| **Processing Complexity Factors** | **0-5** | |
| 1. backup and recovery required | | 4 |
| 2. data communication required | | 1 |
| 3. distributed processing function | | 0 |
| 4. performance critical | | 3 |
| 5. heavily utilized operating environment | | 3 |
| 6. online data entry | | 5 |
| 7. input transaction with multiple screens | | 4 |
| 8. master files updated online | | 4 |
| 9. input, output, files, queries complex | | 3 |
| 10. internal processing complex | | 3 |
| 11. code designed to be reusable | | 2 |
| 12. conversion/installation included in design | | 2 |
| 13. system design for multiple installation | | 4 |
| 14. maintainability/ease of use | | 5 |

# Spreadsheet Data Collection Model

| *Description* | *Units* | *Sample Data* |
|---|---|---|
| **Size-Oriented Metrics** | | |
| **Productivity and Cost** | | |
| **Output** | **KLOC/p-month** | **0.905** |
| **Cost - all code** | **$/KLOC** | **$22,514** |
| **Cost - exclude reuse** | **$/KLOC** | **$24,028** |
| **Elapsed time** | **months/KLOC** | **1.0** |
| **Documentation** | **pages/KLOC** | **30** |
| **Documentation** | **pages/p-month** | **10** |
| **Documentation** | **$/page** | **$739** |
| **Quality** | | |
| **Defects** | **errors/KLOC** | **2.0** |
| **Cost of errors** | **$/error** | **$376** |

# Spreadsheet Data Collection Model

| *Description* | *Units* | *Sample Data* |
|---|---|---|
| **Function-Oriented Metrics** | | |
| **Productivity and Cost** | | |
| **Output** | **FP/p-month** | **378** |
| **Cost - all code** | **$/FP** | **$700** |
| **Elapsed time** | **FP/month** | **31.4** |
| **Documentation** | **pages/FP** | **0.9** |
| **Quality** | | |
| **Defects** | **errors/FP** | **0.064** |