Software Reuse: Metrics and Models
WILLIAM FRAKES
Virginia Tech
and
CAROL TERRY
INCODE Corporation
Metrics and Models

Overview

- Introductory Material
- Cost Benefit Analysis
- Maturity Assessment
- Amount of Reuse
- Software reuse failure modes
- Reusability Assessment
- Reuse Library Metrics
1. Introductory Information

- What is Software Reuse?
  - Use existing ______ or ______ to create new software
Information

• What is Software Reuse?
  • Use existing artifacts or knowledge to create new software

• Resusability?
  • The degree to which a thing can be reused

• HOW DO YOU MEASURE Reuse?
Metrics and Models

- Metric = Measurement
- Model = Specifies relationships among metrics

From paper: “a quantitative indicator of an attribute of a thing.”
Metrics and Models

Figure 1. Categorization of reuse metrics and models.
Table 1. Reusable Aspects of Software Projects

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>architectures</td>
</tr>
<tr>
<td>2</td>
<td>source code</td>
</tr>
<tr>
<td>3</td>
<td>data</td>
</tr>
<tr>
<td>4</td>
<td>designs</td>
</tr>
<tr>
<td>5</td>
<td>documentation</td>
</tr>
<tr>
<td>6</td>
<td>estimates (templates)</td>
</tr>
<tr>
<td>7</td>
<td>human interfaces</td>
</tr>
<tr>
<td>8</td>
<td>plans</td>
</tr>
<tr>
<td>9</td>
<td>requirements</td>
</tr>
<tr>
<td>10</td>
<td>test cases</td>
</tr>
</tbody>
</table>
1.1 Types of Reuse
Refer to Table 2

- Development Scope – Is reusable component external or internal to a project?
- Modification: How much is reusable asset changed?
- Approach: Technical methods for implementing reuse
- Domain Scope: Does reuse occur within a Family or Families of systems
- Management: Degree reuse done systematically
- Reused Entity: Type of reused object
2. Cost Benefit Analysis

- Economic cost-benefit models
- Quality and Productivity payoff analyses
- “None of these models are derived from data, nor have they been validated with data.”
2.1 Cost / Productivity Models

- Gaffney and Durek (1989) – two models
  - Simple Model—shows cost of reuse
  - Cost-of-Development Model—builds on simple modes and reflects cost of developing reusable components
2.1 Simple Model

Favaro: “What price reusability? a case study”

Step 1 - Determine percentage \( R \) of code contributed by reusable components.

Step 2 - Estimate cost \( b \) of reusing component instead of simply developing from scratch.

Step 3 – Determine relative cost \( RC \) can be expressed as

\[
RC = (1 - R)1 + Rb
\]

FAVARO: “We found it surprisingly difficult to estimate the quantity \( R \) (the amount of reused code in the application).” Unclear which measure use – size of source code or relative size of the load modules
Example

Step 1 - Determine R = .50

Step 2 - Estimate cost (b) of reusing component instead of simply developing from scratch = $1000

Step 3 – Determine relative cost (RC) can be expressed as

\[ RC = (1 - R)1 + Rb \]

\[ RC = (1-.50) 1 + .50(1000) \]

\[ RC = $500.50 \]
2.1 Cost of Development Modes

- $E =$ cost of developing a reusable component relative to the cost of producing a component that is not to be reused
- $N =$ number of users over which code cost will be amortized
- Cost ($C$) = $(b + \frac{E}{N-1}) R + 1$
2.1 Applications of Cost Development Models

- Margono/Rhoads – 1993 – 100K lines of ADA code – RESULT – Cost of reuse often 2X more than cost of developing an equivalent nonreusable component

- Favaro – 1991 – ADA-Based project – RESULT – cost of reuse increased as complexity of component increased
2.2 Quality of Investment

- Barnes/Bollinger – 1991
  - Divide reuse components into
    - Producer Activity—reuse investments / costs incurred while making one or more work products easier to reuse by others
    - Consumer activities – reuse benefits – measures in dollars how much reuse has helped/hurt effectiveness
  - Quality of Investment \((Q) = \frac{\text{reuse benefits } B}{\text{reuse investments } R}\)

- Summary – Showed if reuse effort resulted in financial gain or loss
2.3 Business Reuse Metrics

- Poulin et al. - 1993 – IBM Metrics – measured benefits vs. expenditures for reuse
- More business oriented statistical approach to measurement
- See Table 5 in paper
2.4 Relation of Reuse to Quality and Productivity

- Limited empirical evidence on if software reuse improves quality and productivity
- Agresti/Evanco – 1992 – study to predict defect density – RESULT - “High level of reuse correlates with a low defect density”
- Brown et al. - 1990 – Reusability-Oriented Parallel Programming Environment (ROPE)--RESULT--”high correlation between the measures of reuse rate, development time, and decreases in number of errors”
2.4 Relation of Reuse to Quality and Productivity (Contd)

- Card et al. - 1986 – FORTRAN-based with goal of analysis to ID types of software reuse and to quantify benefits – RESULTS –
  - Modules reused without mod were small/simple;
  - High mod = largest of all reused software in terms of numbers of executable statements;
  - 98% modules resued w/out mod were FAULT free
2.4 Relation of Reuse to Quality and Productivity (Contd)

- Matsumura – 1991 – Toshiba reuse – RESULTS – 60% ratio of reused components and 20-30% decrease in errors – Reuse profitable if component reused 3X

- Chen/Lee – 1993 – OO Reuse of C++ components – RESULTS – 30-90% improvement in LOC / hr

- Gaffney/Durek – 1989 – Cost/Productivity Model – cost of reuse must be shared to achieve higher payoffs
3. Maturity Assessment

- Assess How advanced reuse programs are in implementing systematic reuse
- Maturity Model – core of planned reuse – help organizations understand past, current and future goals for reuse activities
Maturity Models

- 3.1 Koltun/Hudson Reuse Maturity Model – 1991 – Table 9 – Phases of reuse identified and tracked along ordinal scale – 1 – 5.
  - 10 Dimensions of reuse
  - Amount of organizational involvement increases from lowest – initial/chaotic to highest level ingrained
  - Organizations use this model to guide reuse efforts
3.2 SPC Reuse Capability Model

- Software Productivity Consortium – 1993
  - Two Components
    - Assessment model – set of success factors / goals
    - Implementation Model – used to prioritize goals
  - No formal validation
4. Amount of Reuse

- Used to monitor reuse improvement effort
  - Tracks percent of reuse of life cycle object over time
  - LOC Reused / Total LOC in system
4.1 Reuse Level

- Level of Reuse - “Basic” dependent variable
- Assumes system is composed of parts at different levels of abstraction that must be designed to measure reuse
  - For example—C-based system: Modules, Functions, LOC
- Software Component
  - Internal Lower Level – developed for higher level component
  - External Lower Level – used by higher level component but created for different item or general use
4.1 Reuse Levels (contd)

- Calculated Quantities
  - L – total number lower level items in the higher level item
  - E – Number of lower level items from external repository in the higher level item
  - I – number of lower level items in the higher level item that are not from external repository
  - M – number items not from an external repository that are used more than once

- Software – rl – calculates reuse in C Code
4.2 Reuse Metrics for Object-Oriented Systems

- Bieman – 1992 – proposed reuse metrics for OO systems
  - Three perspectives: server, client, system
- Bieman/Karunanithi – 1993
  - Prototype tool to collect data from Ada programs
- Chidamber/Kemerer – 1994
  - Metric suite for OO design
    - Depth of inheritance tree Metric: calc length of tree
    - Manage reuse by measuring inheritance
4.3 Reuse Predictions for Life Cycle Objects

• Frakes/Fox – 1995
  • Predict reuse levels for on life cycle object based on that of other life cycle objects – RESULT – significant correlations between reuse levels of life cycle objects
5. Software Reuse Failure Modes Model

- Provides measurement based on the way reuse processes fail
- Frakes/Fox – 1996 – Models can evaluate quality, determine impediments, and devise improvement strategy
- This is the study of why Reuse Fails
Failure Modes

- No Attempt to Reuse
- Part Does not exist
- Part is not available
- Part is not found
- Part is not understood
- Part is not valid
- Part can not be integrated
6. Reusability Assessment

- Estimation of reusability for a component
- Key Question: “Are there measurable attributes of a component that indicate its potential reusability?”
- If yes, then the measured attributes = goals for reuse design and reengineering
  - NASA Study
  - Data-Binding Study in Ada
  - Abstract measurement of reuse in Ada components
7. Reuse Library Metrics

- Reuse Library – repository for storing reusable assets with an interface for searching library
- Assets – Reengineered, Made from scratch, or bought – then they are certified
- Components are classified so they can be found during library search
  - Enumerated, faceted, free text indexing
- Reuse components must be understood
- Reuse library must be efficient
Summary

- Reuse must be:
  - Planned
  - Deliberate
  - Systematic

- Organizations must be able to measure progress and ID most effective reuse strategies

- Metric=quantitative indicator of an attribute

- Model=Specifies relationship between metrics
QUESTIONS

1. Is REUSE done by the corporations individually or by the software reuse community?
   ANS: Both
Is the analysis results shared to the public?
   ANS: Yes, when published

2. With the six models/metrics proposed, is any of them more frequently used in real practice (proven to be most enabling)?
   RPC?
Questions:

3. What other Reuse Metrics are used

Orbotech, as part of the Israeli Software Reuse Industrial Consortium (ISWRIC), explored the possibilities of software reuse in a three-year project

Only metric used was a measure of man days used in development – interpreted this as only metric needed
4. Can these techniques be applied to the open source community?

“Metrics of software reuse for free and Open Source software,”

Author RAMEL Sophie

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Granularity for Metric</th>
<th>Origin of projects</th>
<th>Number of projects</th>
<th>Average External Reuse Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>packages</td>
<td>randomly selected on sourceforge.net</td>
<td>54</td>
<td>0.45</td>
</tr>
<tr>
<td>Java</td>
<td>packages</td>
<td>projects from <a href="http://www.java.net">www.java.net</a></td>
<td>58</td>
<td>0.52</td>
</tr>
<tr>
<td>C/C++</td>
<td>shared libraries</td>
<td>SuSE 9.2 packages</td>
<td>396</td>
<td>0.48</td>
</tr>
</tbody>
</table>
5. Does the data that is collected leave room for interpretation?

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Measurable Impacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality</td>
<td>• Error density (errors per LOC) [5]</td>
</tr>
<tr>
<td></td>
<td>• Fault density (faults per LOC) [6]</td>
</tr>
<tr>
<td></td>
<td>• Ratio of major errors to total faults [4]</td>
</tr>
<tr>
<td></td>
<td>• Rework effort [6]</td>
</tr>
<tr>
<td></td>
<td>• Module deltas [10]</td>
</tr>
<tr>
<td></td>
<td>• Perceived by developers [10]</td>
</tr>
<tr>
<td>Productivity</td>
<td>• LOC per effort</td>
</tr>
<tr>
<td></td>
<td>• Per day per person [4]</td>
</tr>
<tr>
<td></td>
<td>• Per engineering month [5]</td>
</tr>
<tr>
<td>Time-to-market</td>
<td>• Cycle time (months) [4]</td>
</tr>
</tbody>
</table>

Seems like each metric & model measure things from different aspects so am I right that normally we need to use more than 1 metric & model per project?

ANS: METRIC is a MEASUREMENT – If you can measure it --- you can use it.

Can we use these metrics & models with other lifecycle model such as agile which normally has very short period of iteration?

Could you please provide some numerical examples to explain the different formulas and interpret the results (the meaning of numbers in real world)

What level of the Capability Maturity Model (CMM) an institution should acquire to be able to adopt software reuse? Paper stated that most companies are between “INITIAL CHAOS” and “MONITORED”