Measuring 75 Million Lines of Code
A Report from the Field

METRIKON

Harry M. Sneed, ANECON, Vienna
Measuring 75 Million Lines of Code
A Report from the Field
by Harry M. Sneed, ANECON, Vienna

- Background of the Measurement Project
- Goals of the Project
- Metrics applied
  - Quantity Metrics
  - Complexity Metrics
  - Quality Metrics
- Tools employed
  - SoftAudit for Source Measurement
  - SoftEval for System Evaluation
  - SoftCalc for Cost Estimation
- Results of the Measurement
According to Chairman of German Software Initiative:

- 70% of all business applications are in a legacy language.
- Daily more than 30 billion transactions are made with legacy systems.
- In Germany 10 billion Euros are spent per year on the maintenance of legacy systems.
- The number of COBOL code lines in Germany is estimated to be 240 billion.
- To upgrade this code base would cost at least 960 billion Euros, more than the economy can afford.
- There is no alternative to maintaining this code.
Language Distribution in this Project

- **48 Million Statements**
- **75 Million LOCs**
- 7% Java
- 8% C++
- 15% Assembler
- 25% PL/I
- 10% 4GLs
- 8% COBOL
- 7% C++
- 15% Assembler
- 25% PL/I
- 10% 4GLs
- 35% COBOL

Language Distribution in this Project
Goals of the Measurement Project

• To compare the various application systems with one another in terms of the size, complexity and quality of their source
• To identify problem areas in the source
• To estimate the costs of alternative actions based on the sizes and complexities of their source code (Further maintenance and evolution, renovation, redevelopment, migration, wrapping)
QUO VADIS – Alternative Strategies

Legacy Application Systems

- Wrapped components in a new environment
- Transformed system in a new environment or language
- Rewritten components with the same design
- New solution

Replace with a newly developed or bought system

Status Quo

- Maintain
- Wrapping
- Migration
- Reimplementation
- Replacement
Product Evaluation Process according to ISO-9126

1. Started or Implied Needs
   - ISO 9126 & Other Technical Information

2. Quality Requirement Specification

3. Metric Selection

4. Rating Level Definition

5. Assessment Criteria Definition

6. Managerial Requirement
   - Requirement Definition
   - Prepare
   - Evaluation

7. Software Development
   - Products or Intermediate Product

8. Measurement
   - Measured Value

9. Rating
   - Rated Level

10. Assessment
    - Result (Acceptable or Unacceptable)
Quality Grading on a rational scale

- **excellent**
- **good**
- **satisfactory**
- **poor**
- **unacceptable**

Measure Scale:
- 1.0
- 0.8
- 0.6
- 0.4
- 0.2
- 0.0

Bewertung Scale:
- 0.0
- 0.2
- 0.4
- 0.6
- 0.8
- 1.0

Metric Scale vs Nominal Scale
Complexity Grading on a rational scale

- **excellent**
- **good**
- **satisfactory**
- **poor**
- **unacceptable**

**Bewertung**

**Metric Scale**

**Nominal Scale**
Quantity Metrics applied

Size Metrics

The following size metrics were extracted:

- Lines of Code
- Statements
- Data-Points
- Function-Points
- Object-Points for C++ and Java
Metrics from the Legacy Code Base

- 7,854 PLI Programs
- 13,530 PLI Includes
- 17,151 COBOL Programs
- 24,712 COBOL Copies
- 8,975 Assembler Programs
- 9,597 Assembler Macros
- 3,077 Easytrieve Programs
- 3,353 Pseudo Code Modules
- 3,270 IMS Datenbanken
- 4,416 DB2 Tables
- 31,476 IMS-DC Maps
- 22,090 JCL Procedures
- 2,977 C++ Classes
- 1,977 Java Classes

<table>
<thead>
<tr>
<th>Code Quantity Metrics</th>
<th>Number of Sources analyzed: 93321</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Source Lines in all</td>
<td>70752999</td>
</tr>
<tr>
<td>Number of Genuine Code Lines</td>
<td>52048897</td>
</tr>
<tr>
<td>Number of Comment Lines</td>
<td>26349104</td>
</tr>
<tr>
<td>STRUCTURAL QUANTITY METRICS</td>
<td></td>
</tr>
<tr>
<td>Number of Modules</td>
<td>33616</td>
</tr>
<tr>
<td>Number of Copy/Includes</td>
<td>430028</td>
</tr>
<tr>
<td>Number of Entry Points</td>
<td>102645</td>
</tr>
<tr>
<td>Number of Exit Points</td>
<td>209614</td>
</tr>
<tr>
<td>Number of Sections/Procedures</td>
<td>633900</td>
</tr>
<tr>
<td>Number of Labels/Paragraphs/Code Blocks</td>
<td>1365186</td>
</tr>
<tr>
<td>Number of Reusable Code Blocks</td>
<td>583818</td>
</tr>
<tr>
<td>Number of Data Structures/Objects</td>
<td>2098429</td>
</tr>
<tr>
<td>Number of Reusable Data Objects</td>
<td>778125</td>
</tr>
<tr>
<td>PROCEDURAL QUANTITY METRICS</td>
<td></td>
</tr>
<tr>
<td>Number of Statements</td>
<td>42677337</td>
</tr>
<tr>
<td>Number of Macro Statements</td>
<td>591353</td>
</tr>
<tr>
<td>Number of Procedural Statements</td>
<td>22760888</td>
</tr>
<tr>
<td>Number of Convertable Statements</td>
<td>18913227</td>
</tr>
<tr>
<td>Number of Input Operations</td>
<td>88742</td>
</tr>
<tr>
<td>Number of Output Operations</td>
<td>266829</td>
</tr>
<tr>
<td>Number of Foreign Modules called</td>
<td>114236</td>
</tr>
<tr>
<td>Number of Call Statements</td>
<td>644132</td>
</tr>
<tr>
<td>Number of Perform Statements</td>
<td>1738020</td>
</tr>
<tr>
<td>Number of Selections (If &amp; Case)</td>
<td>3567905</td>
</tr>
<tr>
<td>Number of Loop Statements (Until/While)</td>
<td>398877</td>
</tr>
<tr>
<td>Number of GOTO Branches</td>
<td>789364</td>
</tr>
<tr>
<td>Number of all Control Statements</td>
<td>7498184</td>
</tr>
<tr>
<td>Number of Control Flow Branches</td>
<td>7103089</td>
</tr>
<tr>
<td>Number of Literals in Statements</td>
<td>5362729</td>
</tr>
<tr>
<td>Number of Constants in Statements</td>
<td>4723285</td>
</tr>
<tr>
<td>Number of Test Cases required</td>
<td>1173467</td>
</tr>
<tr>
<td>Number of Function-Points</td>
<td>704918</td>
</tr>
</tbody>
</table>
Basic Rule of Complexity Measurement

Complexity = 1 - (Entities / Relationships)

Entities =
- Modules, Procedures, Classes, Methods, Stmts
- Files, Tables, Records, Objects, Attributes
- Business Objects, Business Processes, Business Rules, Use Cases, Steps, Actors, Triggers, Requirements, etc.

Relationships =
- Module2Module, Class2Class, Stmt2Stmt, Stmt2Data, Object2Object
- Decision2Action, Program2File, Method2Method, Method2Param
- BusinessRule2BusinessObject
Complexity Metrics applied

Complexity Metrics

The following source complexities were measured:

- Data Complexity (Chapins Q-Metric)
- Data Flow Complexity (Elshoff ref frequency)
- Data Access Complexity (Card access density)
- Interface Complexity (Henry fan-in/fan-out)
- Control Flow Complexity (McCabe graph comp)
- Decisional Complexity (McClure decision density)
- Branching Complexity (Sneed branch density)
- Language Complexity (Halstead Volume)
Complexity of the Mainframe Software

<table>
<thead>
<tr>
<th>Complexity Metrics</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA COMPLEXITY (Chapin Metric)</td>
<td>0.358</td>
</tr>
<tr>
<td>DATA FLOW COMPLEXITY (Elshof Metric)</td>
<td>0.632</td>
</tr>
<tr>
<td>DATA ACCESS COMPLEXITY (Card Metric)</td>
<td>0.664</td>
</tr>
<tr>
<td>INTERFACE COMPLEXITY (Henry Metric)</td>
<td>0.399</td>
</tr>
<tr>
<td>CONTROL FLOW COMPLEXITY (McCabe Metric)</td>
<td>0.577</td>
</tr>
<tr>
<td>DECISIONAL COMPLEXITY (McClure Metric)</td>
<td>0.338</td>
</tr>
<tr>
<td>BRANCHING COMPLEXITY (Sneed Metric)</td>
<td>0.382</td>
</tr>
<tr>
<td>LANGUAGE COMPLEXITY (Halstead Metric)</td>
<td>0.494</td>
</tr>
<tr>
<td>AVERAGE PROGRAM COMPLEXITY</td>
<td>0.480</td>
</tr>
</tbody>
</table>
### Basic Rule of Quality Measurement

**Quality Metric** = Ist / Soll

<table>
<thead>
<tr>
<th>Quality Level</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowest</td>
<td>&lt;0</td>
</tr>
<tr>
<td>Acceptable</td>
<td>0,5</td>
</tr>
<tr>
<td>Highest Feasible</td>
<td>&gt;1</td>
</tr>
</tbody>
</table>

=> soll
Quality Metrics applied

Quality Metrics
The following source qualities were measured
- Modularity (Cohesion & Coupling)
- Portability (Environmental Dependency)
- Testability (Number of paths & control data)
- Reusability (Degree of Code Dependency)
- Convertibility (Ease of statement conversion)
- Flexibility (Proportion of hard coded data)
- Conformity (Compliance with coding rules)
- Maintainability (CodeQuality – CodeComplexity)
# Quality of the Mainframe Software

<table>
<thead>
<tr>
<th>Quality Metrics</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree of Modularity</td>
<td>0.586</td>
</tr>
<tr>
<td>Degree of Portability</td>
<td>0.741</td>
</tr>
<tr>
<td>Degree of Testability</td>
<td>0.680</td>
</tr>
<tr>
<td>Degree of Reusability</td>
<td>0.336</td>
</tr>
<tr>
<td>Degree of Convertibility</td>
<td>0.395</td>
</tr>
<tr>
<td>Degree of Flexibility</td>
<td>0.528</td>
</tr>
<tr>
<td>Degree of Conformity</td>
<td>0.813</td>
</tr>
<tr>
<td>Degree of Maintainability</td>
<td>0.554</td>
</tr>
<tr>
<td>Average Program Quality</td>
<td>0.579</td>
</tr>
</tbody>
</table>
SoftAudit processes 12 Programming, 4 Database and 9 Interface Languages. The User selects them.
The User creates a macro or function table to identify special operations like DB-Accesses and TP-Interactions.
The User selects the rules to be checked.
The User can weight the metrics.
The User selects the sources to be analyzed
The tool marks the rule violations.
The tool produces aggregated metric reports at the component, system and product level.
The tool produces a metric export file in XML for the metric database.
The User imports the metrics into the metric database.
The user displays the metrics of the individual systems.
The User displays the size distribution of the member systems.
The user displays the relationship of the quality or complexity metrics to one another.
The User has the systems ranked by size, quality or complexity.
## Cost Estimations for one Set of Systems

### Geschäftsbereich XYZ
- 29.3 Mio. Lines of Code
- 22.2 Mio. Source Statements
- Languages: COBOL, ASM, PLI, CSP, IMS-DB, DB2, IMS-MFS, OS-JCL

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Jährliche Wartung</td>
<td>294</td>
<td>250</td>
<td>280</td>
<td>239</td>
<td>298</td>
</tr>
<tr>
<td>Neuentwicklung</td>
<td>4.113</td>
<td>3.017</td>
<td></td>
<td>2.955</td>
<td>3.656</td>
</tr>
<tr>
<td>Sanierung</td>
<td>88</td>
<td>92</td>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Migration</td>
<td>412</td>
<td>424</td>
<td>329</td>
<td>311</td>
<td>372</td>
</tr>
<tr>
<td>Kapselung</td>
<td>82</td>
<td>58</td>
<td>71</td>
<td></td>
<td>89</td>
</tr>
</tbody>
</table>
Lessons learned

- The most important prerequisite to a measurement project is to define the goals.
- The metrics used must be adjustable to the measurement goals.
- The metrics must fit to different programming languages with different paradigms.
- The tools must be able to handle multiple languages and to process database and interface as well as programming languages.
- The cost estimations must be made with at least three different methods and then compared.
Software ist unsere Leidenschaft