Using COSMIC-FFP to Quantify Functional Reuse in Software Development

Vinh T. Ho, Alain Abran, Serge Oligny

Presented at

ESCOM-SCOPE 2000, Munich, Germany

April 18-20, 2000
Agenda

- Context
- Functional reuse
- Measuring functional reuse
- Examples
- Conclusion
**Goal**: Using COSMIC C-FFP, explore functional reuse measurement as a quantitative basis for evaluating alternatives designs

- **Exploratory work**
- Reuse *mostly* measured at code level
- Reuse at earlier stages is of interests
Functional reuse

Kinds of reuse:

- Reuse without modification: black-box reuse
- Reuse with modification: white-box reuse

This exploratory research concentrates on black-box reuse.
Functional reuse

- **Functional reuse**: identifying “avoided” functions, that is functions which do not need to be re-developed.

- **Goal**: How much such re-development could be avoided?
Measuring functional reuse

- COSMIC-FFP: a functional size measurement method
- Candidates for reuse
- Quantifying functional reuse
Measuring functional reuse

COSMI C-FFP:

**Principle**
Software functional size is directly proportional to the number of its data-movement sub-processes.

**Base Functional Components**
COSMI C-FFP recognize 4 types of data movements: Entry, Exit, Reads and Writes.

**Unit**
The standard unit of measurement (1 C_{fsu}) is defined by convention as equivalent to one single data movement at the sub-process level.
Measuring functional reuse

COSMIC-C-FFP:

Measurement process

© 2000, V. T. Ho, A. Abran, S. Oligny
Software Engineering Management Research Laboratory
Measuring functional reuse

Candidates for reuse:

- **COSMI C-FFP layer** differentiates F.U.R. allocated at different levels of functional abstraction

- Each layer encapsulates **functionality useful to other layers** using its services

- Identifying layers facilitate **identification of reused functionality**.
Measuring functional reuse

Quantifying functional reuse:

**Principle**

Amount of functional reuse is proportional to the product of the size of the functional processes reused and the number of functional processes using their services.

\[ F_{RU} = \sum_{i=1}^{n} \text{Size(Func. Process}_i) \times U_i \]

Where:  
- \( F_{RU} \) represents the total amount of functional reuse, in \( C_{fsu} \),  
- \( i \) represents the total number of reused functional process identified,  
- \( U_i \) represents the number of functional processes using functional process \( i \)
Measuring functional reuse

Quantifying functional reuse:

**Reuse index**

The relative amount of reused functionality within a piece of software.

\[
R_I = \frac{F_{RU} \times 100}{\text{Size(Software)}}
\]

Where:
- \(R_I\) represents the reuse index, in %,
- \(F_{RU}\) represents the total amount of reuse within a piece of software, in \(C_{fsu}\)
- \(\text{Size(Software)}\) represents the size, in \(C_{fsu}\), of the piece of software without reuse
Examples

- ISDN Loopback tester, detailed example
- Summary of three other software
Examples

ISDN Loopback tester:

- Device used to test the integrity of four wires ISDN circuits at a remote location
- Users dial in to connect to ISDN device and performs testing and maintenance programming functions
ISDN Loopback tester:

- 1 functional process
- 37 functional processes (total)
- 32 functional processes used by Access layer

User → Access Layer → Application Layer → ISDN Device
ISDN Loopback tester:

Scenario 1 - Assuming Access functionality is reused whenever possible

\( i = 1 \) (1 reused functional process - Access)

\( U = 32 \) (32 functional processes using Access)

Size (Func. process\(_1\)) = 10 \( C_{fsu} \)

\[
F_{RU} = \sum_{i=1}^{1} 10 \cdot C_{fsu} \cdot 32
\]

\( F_{RU} = 320 \cdot C_{fsu} \)

Size of software with functional reuse: 136 \( C_{fsu} \)
ISDN Loopback tester:
Scenario 2 - Assuming Access functionality is not reused at all

Size of Access layer: 10 $C_{fsu}$
Size of Application layer: 126 $C_{fsu}$

Size of duplicated access functionality (31 * 10 $C_{fsu}$): 310 $C_{fsu}$

Total size of software: 446 $C_{fsu}$
ISDN Loopback tester:
Comparing scenario 1 and 2

Scenario 1 - maximum reuse

\[ F_{RU} : 320 \text{ C}_{fsu} \]

Size (software without reuse): \( 446 \text{ C}_{fsu} \)

\[
R_l = \frac{320 \text{ C}_{fsu} \times 100}{446 \text{ C}_{fsu}}
\]

\[ R_l = 72\% \]

Scenario 2 - no reuse

\( 446 \text{ C}_{fsu} \)

\( 136 \text{ C}_{fsu} \)

3,28 ratio
Examples

What have we done?

Functional Design no. 1
Functional Design no. 2
Functional Design no. 3

COSMIC C-FFP Measurement Process

Functional Size no. 1
Functional Size no. 2
Functional Size no. 3

Quantitative comparison

© 2000, V. T. Ho, A. Abran, S. Oligny
Software Engineering Management Research Laboratory
Examples

Three other software:

<table>
<thead>
<tr>
<th></th>
<th>Size((F_{p1}))</th>
<th>U</th>
<th>(F_{RU})</th>
<th>Size (A^1)</th>
<th>Size (B^2)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control software A</td>
<td>50 (C_{fsu})</td>
<td>9</td>
<td>450 (C_{fsu})</td>
<td>807 (C_{fsu})</td>
<td>357 (C_{fsu})</td>
<td>2.26</td>
</tr>
<tr>
<td>Control software B</td>
<td>25 (C_{fsu})</td>
<td>8</td>
<td>200 (C_{fsu})</td>
<td>359 (C_{fsu})</td>
<td>159 (C_{fsu})</td>
<td>2.26</td>
</tr>
<tr>
<td>Surveillance sub-system</td>
<td>29 (C_{fsu})</td>
<td>3</td>
<td>87 (C_{fsu})</td>
<td>131 (C_{fsu})</td>
<td>44 (C_{fsu})</td>
<td>2.98</td>
</tr>
</tbody>
</table>

1: size of software without functional reuse
2: size of software with complete functional reuse
Conclusion

- Important variance in size with and without functional reuse
  - must be taken into account in productivity and cost analysis
  - could be used as index to evaluate quality of design
  - could be used as index to evaluate integration of application portfolio

- Reuse must be recorded in data collection for benchmarking
Conclusion

- COSMIC-FFP layer concept useful to identify functional reuse
- Measuring reuse provides a quantitative basis for evaluating alternatives designs
- Enlarge scope from black-box/complete reuse to white-box/partial reuse
Further readings

- Bootsma F., “Applying Full Function Points to drive strategic business improvement within the real-time software environment”, Annual IFPUG Conference, New-Orleans, 1999


Downloadable at www.lrgl.uqam.ca/ ffp.html
Acknowledgments

- The Software Engineering Management Research Laboratory of the Université du Québec à Montréal is supported through a partnership with Bell Canada.

- Additional funding is provided by the Government of Canada.