Software size measures and their usefulness for software project estimation
Software Size Measures and their usefulness for software project estimation

Harold van Heeringen
Sogeti Nederland B.V., Senior Software Cost Engineer

San Diego (CA, USA) June 2015
Why this presentation?
Estimating maturity model

Level 0: Informal or no estimating, Manual effort estimating without a process

Level 1: Direct Task Estimation, Spreadsheets, Ad Hoc Process

Level 2: Formal Sizing (e.g. function points), Direct Task Estimation, Simple model (Size * Productivity), Some measurement & analysis, Informal Process

Level 3: Formal Sizing, Robust Parametric estimation, Estimate vs. actual capture, Rigorous measurement & analysis, Parametric planning & Control, Repeatable process

Level 4: Formal Sizing, Repeatable process, Robust parametric estimating, Rigorous measurement & analysis, Parametric estimation with tracking & control, Process improvement via lessons learned

Level 5: Formal Sizing, Repeatable process, Robust parametric estimating, Rigorous measurement & analysis, Parametric estimation with tracking & control, Continuous process improvement

Estimation Bias Mitigation Begins at Level 2, Solid at Level 3

ICEAA2015, San Diego, June 2015

* Developed by Dan Galorath, www.galorath.com
Software project estimation

- Size is main input parameter for many cost estimation models;
- Size should be measured, not guessed;
- Many sizing techniques available in the industry
  - IFPUG Function Points;
  - Nesma Function Points;
  - COSMIC Function Points;
  - Source Lines of code (Slocs);
  - Story points;
  - SNAP points.
- ... 
- How useful are these methods for project estimation?
Software project estimation (simplified)

Measures

Risk analysis

Risks

Consequences

Size

Gross hours

Effort hours

Adjustments

Productivity

Historical Data

Parametric tools
Software size and project estimation

- Size is one of the main **input parameters** for a software project estimate;
- Size is not enough, historical data and parametric tools are necessary in order to create accurate estimates;
  - Measuring the size in a method of which no **historical data** is available **does not make sense**;
  - Measuring the size in a method that is not supported by **parametric tools and/or models** **does not make sense**.
- The chosen combination of size measurement method, historical data and parametric tool drives the accuracy (**usefulness**) of the estimate.
What is software size?

- Size (surface) of a wall: **square feet**;
- Size (volume) of a bottle: **liters**;
- Size (weight) of a man: **pounds**.

Software size ??
- Number of screens?
- Number of modules?
- Number of objects?
- Number of source lines of code?

What is the square feet metric in software?
- We need **standards** to express the size of software.
Classification

- **Functional size measurement methods**
  - Measure the functionality the software offers to its users;
  - Independent of technology / implementation;
  - “What can the users do with the software”.

- **Non-functional size measurement methods**
  - Measure technical and/or quality attributes of the software;

- **Hybrid methods**
  - Measure both functionality and technical characteristics of the software.
## Size measurement methods covered

<table>
<thead>
<tr>
<th>Software size measure</th>
<th>Measurement scope</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source lines of code (slocs)</td>
<td>‘Physical’ software object</td>
<td>Technical size in slocs</td>
</tr>
<tr>
<td>IFPUG Function Points</td>
<td>Functional user requirements</td>
<td>Functional size in FP</td>
</tr>
<tr>
<td>NESMA Function Points</td>
<td>Functional user requirements</td>
<td>Functional size in FP</td>
</tr>
<tr>
<td>COSMIC Function Points</td>
<td>Functional user requirements</td>
<td>Functional size in CFP</td>
</tr>
<tr>
<td>SNAP Points</td>
<td>Part of the non-functional user requirements</td>
<td>Non-functional size in SNAP points</td>
</tr>
<tr>
<td>Usecase Points</td>
<td>Usecases, technical project characteristics, environmental project characteristics</td>
<td>‘Effort/Size combination’ in Usecase points</td>
</tr>
<tr>
<td>Fast Function Points</td>
<td>Functional user requirements and Configuration size</td>
<td>Gartner FFP</td>
</tr>
<tr>
<td>Story Points</td>
<td>Backlog items, functional and non-functional</td>
<td>Story points</td>
</tr>
</tbody>
</table>

ICEAA2015, San Diego, June 2015 |
Functional size measurement

- Independent of technical environment and implementation method
  - 200 FP Java project = 200 FP COTS package implementation.
- ISO/IEC standard for functional size measurement
  - Nesma, IFPUG, COSMIC, FiSMA, MK II.
- Objective, repeatable, verifiable measurement
  - Defendable!
  - Used in software estimation, project control, benchmarking, output based pricing contracts.
- Square meters of software.
  - Functionality = value
Historical data

  - IFPUG: 5244 projects;
  - Nesma: 187 projects (treated equal to IFPUG);
  - COSMIC: 488 projects;
  - FiSMA: 580 projects;
  - MK II: 37 projects;
  - SNAP: 0 projects;
  - LOC: 180 projects;
  - Usecase points: 0 projects.
- Other databases.
Parametric tools

- Most parametric tools and models support functional size measures as well as technical size methods:
  - QSM SLIM;
  - Galorath SEER-SEM;
  - Price True planning;
  - Cocomo II;
  - Others.

- ISO/IEC standard Functional size measurement methods + historical data + parametric tools should result in accurate project estimates.
Slocs

▶ How many lines is this?

```c
for (i = 0; i < 100; i++) printf("hello"); /* How many lines of code is this? */
```

▶ Same functionality

```c
/* Now how many lines of code is this? */
for (i = 0; i < 100; i++)
{
    printf("hello");
}
```

▶ Different code counters, different results;
▶ Code counters often count 1 programming language.
Source Lines of Code (slocs)

- Technical size measure;
- Not standardized
  - Different code counters result in different slocs;
  - Eslocs / Slocs / Locs / Physical locs / etc;
- Measurement only possible to measure after completion;
  - More slocs is not better... or more value;
  - Use in project estimation → very risky;
  - Large error range in main input parameter;
  - Project estimates are probably not very accurate.

- Why is it used? Easy / Fast / Automatic / No expertise.
Advantages
- ISO/IEC standards;
- Used in many countries worldwide;
- Many certified analysts available;
- Lots of historical data available;
- Supported by most parametric tools and models;
- Powerful static test of requirements;
- Very useful for software project estimation.

Disadvantages
- Documentation requirements;
- Available expertise/skills, Usually takes a few days.
Nesma derived methods

- **Global Nesma**
  - Applicable earlier in the software project lifecycle;
  - Less effort while not losing accuracy (-8 / +15%);
  - Less strict documentation requirements;
  - Faster to carry out.

- **Indicative Nesma**
  - Only data model is needed;
  - Even faster to carry out;
  - Enough accuracy for ROM estimate (-30% / +50%)

- **Nesma is very useful for software project estimation**
COSMIC

- Open standard, material for free;
- Possible to measure more than business software:
  - Realtime software;
  - Infrastructure software;
  - Mobile software.
- Possible to measure the size per ‘layer’ and ‘peer component’.
- No size limit per functional process, possibly more accurate sizing than other methods.
- Early methods available, but accuracy may be low.

- COSMIC is very useful for software project estimation
SNAP points

- **Software Non-Functional Assessment Process**
  - Should be complementary to IFPUG;
  - Connected to ISO 9126 and ISO 25010;
  - Not a standard;
  - Often no information on SNAP categories available;
  - Not all non-functional requirements are measured;
  - Relationship between classes is questionable, e.g.
    - UI Complexity – Nr. of properties added or configured;
    - Batch processes: 4*, 6* or 10* nr. of data attributes.
SNAP categories

1. Data operations
   1.1. Data Entry Validations
   1.2. Logical and Mathematical Operations
   1.3. Data Formatting
   1.4. Internal Data Movements
   1.5. Delivering Added Value to Users by Data Configuration

2. Interface Design
   2.1. User Interfaces
   2.2. Help Methods
   2.3. Multiple Input Methods
   2.4. Multiple Output Formats

Often also Functional
SNAP Categories

- 3. Technical Environment
  - 3.1. Multiple Platform
  - 3.2. Database Technology
  - 3.3. Batch Processes

- 4. Architecture
  - 4.1. Component Based Software
  - 4.2. Multiple Input / Output interfaces

- Performance / Security / Maintainability ??
SNAP points

- **Other issues**
  - Published proof is reported statistically invalid;
  - There is no historical data available that can be used publicly;
  - Not supported by parametric tools.
  - Project documentation usually does not explicitly state the NFR’s measured by SNAP → Not clear what to do….

- The idea may be good, but SNAP is **not yet useful** for project estimation.
Use case Points

- Developed by Gustav Karner
  - UML/RUP projects.
- Measure the size per Use case, e.g. nr of transactions;
- Measure the number and complexity of actors;
- Determine ‘Technical Complexity Factor’ and ‘Environmental Factor’
  - Highly subjective activities;
  - Multiplier effect result in huge differences between people.

- \( UCP = (UUCW + UAW) \times TCF \times EF \)
TCF and EF

\[ TCF = 0.6 + \left( \frac{TF}{100} \right) \]

\[ ECF = 1.4 + (-0.03 \times EF) \]

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Distributed system</td>
<td>2.0</td>
</tr>
<tr>
<td>T2</td>
<td>Response time/performance objectives</td>
<td>1.0</td>
</tr>
<tr>
<td>T3</td>
<td>End-user efficiency</td>
<td>1.0</td>
</tr>
<tr>
<td>T4</td>
<td>Internal processing complexity</td>
<td>1.0</td>
</tr>
<tr>
<td>T5</td>
<td>Code reusability</td>
<td>1.0</td>
</tr>
<tr>
<td>T6</td>
<td>Easy to install</td>
<td>0.5</td>
</tr>
<tr>
<td>T7</td>
<td>Easy to use</td>
<td>0.5</td>
</tr>
<tr>
<td>T8</td>
<td>Portability to other platforms</td>
<td>2.0</td>
</tr>
<tr>
<td>T9</td>
<td>System maintenance</td>
<td>1.0</td>
</tr>
<tr>
<td>T10</td>
<td>Concurrent/parallel processing</td>
<td>1.0</td>
</tr>
<tr>
<td>T11</td>
<td>Security features</td>
<td>1.0</td>
</tr>
<tr>
<td>T12</td>
<td>Access for third parties</td>
<td>1.0</td>
</tr>
<tr>
<td>T13</td>
<td>End user training</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Familiarity with development process used</td>
<td>1.5</td>
</tr>
<tr>
<td>E2</td>
<td>Application experience</td>
<td>0.5</td>
</tr>
<tr>
<td>E3</td>
<td>Object-oriented experience of team</td>
<td>1.0</td>
</tr>
<tr>
<td>E4</td>
<td>Lead analyst capability</td>
<td>0.5</td>
</tr>
<tr>
<td>E5</td>
<td>Motivation of the team</td>
<td>1.0</td>
</tr>
<tr>
<td>E6</td>
<td>Stability of requirements</td>
<td>2.0</td>
</tr>
<tr>
<td>E7</td>
<td>Part-time staff</td>
<td>-1.0</td>
</tr>
<tr>
<td>E8</td>
<td>Difficult programming language</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

Assign value 0 to 5 depending on relevance
No standard for Usecase documentation;
Usecase Points

- No standard for Usecase documentation;
- No historical data available;
- Not supported by most parametric tools.
  - Some organizations convert UCP to FP... 
    theoretically impossible!

- UCP is not useful for project estimation, unless...
  - There are strict definitions on how to model usecases implemented;
  - The process of assessing the TCF and EF is formalized somehow.
Fast Function Points

- Gartner method, part FPA, part Configuration;
- ‘Function Points’ is misleading statement;
- Pushed on management level, no specialist support;
- Many **unsupported (and false)** claims, like
  - Faster counting than Nesma / IFPUG;
  - Faster training than Nesma / IFPUG;
  - Accuracy higher than Nesma / IFPUG;

- Vendor lock-in;
- No historical data available outside Gartner;
- No support from parametric tools;
- Fast Function Points is **not suitable** for effort estimation.
Story Points

- Often used in agile projects;
- Planning Poker → assign SP to backlog items;
- SP is not a size measure, but an effort measure;
  - Very Subjective;
  - No standard;
  - Only valid in 1 team;
- No historical data available outside team (useless);
- No support from parametric tools.

- Story Points are very useful for sprint planning, but not useful for project estimation.
Story Points

- SP are useful to plan the product backlog items that can be realized per sprint;
- Hard to do on Product Backlog that is not yet detailed;
- Early Project Lifecycle methods are better suitable.
Estimating maturity model

Level 0: Informal or no estimating
Level 1: Direct Task Estimation, Spreadsheets, Ad Hoc Process
Level 2: Formal Sizing (e.g., function points), Simple model (Size * Productivity), Some measurement & analysis, Informal Process
Level 3: Formal Sizing, Estimate vs. actual capture, Rigorous measurement & analysis, Parametric planning & Control, Repeatability processes
Level 4: Formal sizing, Robust parametric estimating, Rigorous measurement & analysis, Process improvement via lessons learned
Level 5: Formal sizing, Robust parametric estimating, Continuous process improvement

* Developed by Dan Galorath, www.galorath.com
Conclusions

- There are many sizing techniques available, but few are useful for software project estimation;

- Project Estimation based on Sloc, Usecase points, Story points or Fast function points may work in certain situations, but is not recommended in a mature ‘Estimating & Performance Measurement’ process.

- Best Practice: select one of the functional size measurement methods (IFPUG, Nesma or COSMIC) and implement this method in the E&PM process.
Thanks for your attention!