Improved Software Project Certainty

Automated analysis of user stories for improved quality and measurement

December 13, 2018

Colin Hammond M.Eng MBCS CFPS

Project Manager
Portfolio Manager
Developer
Analyst
Solution Designer
The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

Fred Brooks, 1986

$78Bn wasted on project failures USA p.a.* Gartner 2018

Failed software projects are a bigger problem now than ever!
Executives hope to be delighted but are used to disappointment

Time to delivery is typically the most important factor for them
Early to Market with Strategic IT innovation

Profit from being first to market

Breakeven point

Develop

Time

$
Impact of delay is compound

The graph illustrates the impact of delay on market opportunity. Delaying development can result in missing the breakeven point and losing market opportunity due to being late. The cost of delay accumulates over time, as indicated by the compound effect shown on the graph.
Impact of delay is compound

- Market opportunity missed by being late
- Breakeven point
- Delayed
- Develop

\[ $ \]
Software Project Managers

To avoid failure on large software projects...

“Do all these well and you’ll be fine!”
Let us focus on two aspects

Size & Quality
Size and failure - diseconomy of scale

Size is important
Team size - communication overhead

One of the reasons why size matters

\[ \frac{n^2 - n}{2} \]

Lines of communication

Keep your team small!
How many people?

Limited by the mental capacity of the developer

Functionality   Testing
Configuration   Libraries   Data
Integrations   Architecture

This is an example of functional sizing in action.

Rule of thumb:
Approx 150 - 250 FP per developer
Other “Engineering” disciplines rigorously adopt universal, reliable measurements:

**Size for Software Engineering**

- **Story Points**: Ok for sprint planning but not project metrics
- **Lines of Code**
- **Number of Interfaces / modules**
- **Use case points**
- **Function points (IFPUG)**
- **COSMIC Function points**

14 reasons why CFP are better than SP
Software Size Measurement - history

ISO Standards (principle based)
- Function Points - IFPUG
- COSMIC FP
- Others

ISO Standards (rules based)
- Use Case Points
- Others

Consistent not universally applicable

Inconsistent
- SLOC
- Story Points

Suitability to Engineering Practice

Time

1980’s

2000’s
COSMIC Functional Sizing - Oversimplified

Open Source
Principle based technique
ISO standard
Easy to learn
Suited to modern s/w architectures
Technology Agnostic
Methodology Agnostic
Mature
Better effort correlation than SP
Ideal for benchmarking

**Technique:** Cosmic functional sizing
**Metric:** Cosmic Function Points
**COSMIC:** Common Software Measurement International Consortium
COSMIC Functional Sizing - Oversimplified

1. Define what you are measuring

2. Identify users

3. Add up the unique data movements

\[ \sum \text{Inputs} + \text{Outputs} + \text{Reads} + \text{Writes} \]
Using Size as the Core Metric

The Recommended Metric for Software Project Management

Size
  e.g. Scope in CFP

Quality
  e.g. Defects found per CFP

Resources
  e.g. CFP can a tester test

Schedule
  e.g. CFP per month delivered

Risk
  Doesn’t help directly
Now let’s look at quality
On most large software projects

1. Bug fixing is the single biggest activity

2. Delays nearly always caused by extended bug fixing
Focus on Quality

Root source of defects

- Requirements: 17%
- Code: 27%
- Design
- Security
- Documents
- Bad fixes

Most Activity
- Unit testing
- Systems testing
- Functional testing
- End to end testing
- Acceptance testing

1,000 FP Application
Source: Capers Jones
Achieving Quality

Testing

80%  1,000 left
✔ Test

Quality

95%+  250 left
✔ Prevention
✔ Pre-test removal
✔ Test

Quadruple defect removal

Quality cannot be achieved through testing alone
Focus on Quality

Goal: Defect removal efficiency is >95%

Problem: Defect Potential is FP^{1.2} *
1. Most organisations are **not mature in their Agile** software journey.
2. **User stories** are the main articulation of requirements
3. Outsourced agile developments are typically **T&M** based.
4. **Story points** are the main size measurement
5. It is very hard to establish a learning organisation based on a User Story metric.
Common Misconceptions

These are not true:

1. You have to compromise quality if you want it cheaper or sooner. Generally speaking this is wrong. By doing the right things early in the project to achieve high quality you will usually deliver faster and at lower cost.

2. Measuring software size is impossible.

3. Estimation requires the whole team to understand the epics/stories.
My Aspiration

My goal was to:

Automate (Functional) Sizing

C O S M I C
Value of finding defects early

Phase in which the defect is introduced

- Requirements
- Architecture
- Detailed Design
- Construction

Phase in which the defect is corrected

Cost to Correct
Example User Story

Card, Conversation, Collaboration

Opinions:
Requirements = User stories
Requirements != User stories

Acceptance Criteria ...
I can click pencil to enter edit mode then I can enter my bank details and name and address and click save.
9 months of experimentation

Natural Language Processing (a branch of AI)
User Stories and Requirements Specifications
QA was an accident
Fishing

Expert endorsements:
What is it?

SAAS analyser of Software Requirements or User Stories

Free form requirements or user stories (import CSV)

Analyses the text

"Within and across the requirements"

Reword and refine

Size CFP estimate

Quality Finds defects & suggests test

Estimates Project level

"Static analyser for user requirements"
Helps

Not a cure, but it helps, think of it as:

“Ultimate in shift-left testing”

or

“Static analyser of user stories”
Early days

Experience so far  May - November 2018
> 100 projects
> 11,000 User stories
> 12,000 potential defects found
  Finds more than one defect per story

Performance
< 2 seconds to analyse a story (typical)
< 2 seconds to find a potential defect

“(Using ScopeMaster) is … out and out the most productive quality work you can do on a software project”
Features

- English only
- Does not need training
- Detects parts of speech
- Detects phrase dependencies
- Detects singulars/plurals
- Fixed list of verbs
- Handles multiple steps per user story
- Looks across requirements for reference to similar objects
- Looks for a full set of CRUD operations
- Uses a template of data movements for each of C,R,U and D
- Very fast. 1-5 seconds per user story.
Intelligent Interpretation of intended data movements

Shows:
Story quality, size, including functional steps.
Intelligent Interpretation of intended data movements

1. The tool analyses this

2. Detects independent steps

3. Determines data movements

4. Estimates size in COSMIC FP

### Estimated Functions and Data movements

<table>
<thead>
<tr>
<th>Function</th>
<th>Object</th>
<th>Action</th>
<th>Data movements</th>
</tr>
</thead>
<tbody>
<tr>
<td>validate device</td>
<td>device</td>
<td>Read</td>
<td>object id E read from storage E display data X</td>
</tr>
<tr>
<td>validate permission</td>
<td>permission</td>
<td>Read</td>
<td>object id E read from storage E display data X</td>
</tr>
<tr>
<td>insert order</td>
<td>order</td>
<td>Create</td>
<td>new object data E check if id exists R insert W</td>
</tr>
</tbody>
</table>
### Automated Requirements Quality

Finds approx 50% of all requirements defects

<table>
<thead>
<tr>
<th>Independent</th>
<th>Clear (unambiguous)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negotiable (Concise)</td>
<td>Complete</td>
</tr>
<tr>
<td>Valuable</td>
<td>Concise</td>
</tr>
<tr>
<td>Estimable</td>
<td>Consistent</td>
</tr>
<tr>
<td>Sized</td>
<td>Correct</td>
</tr>
<tr>
<td>Testable (partly)</td>
<td>Current</td>
</tr>
</tbody>
</table>

Does not replace any agile ceremonies, it just makes them more efficient
Case Study - Gaming Application

1 Person
16 hours
No training
90 Stories
150 defects found & fixed before coding even started!

Value $35k - $100k, in 2 days

https://www.scopemaster.com/case-study/scopemaster-gaming-application/
Find and fix a requirements problems

User Story Refinement Meeting

2-5 hours effort

15 minutes effort

8-20X Faster
Minimun typical effort cost to find and fix is $125, with ScopeMaster you can do that in less than 15 minutes.

But this is often QA work that is so boring it doesn’t happen, so you end up carrying the defect into coding then causing $1000’s of rework.
Poor quality is the cause of most delays

Root source of defects

- Requirements: 16.9%
- Code: 26.8%
- Design
- Security
- Documents
- Bad fixes

Most Activity
- Unit testing
- Systems testing
- Functional testing
- End to end testing
- Acceptance testing

1,000 FP Application
Source Capers Jones
Size and failure

Our vision

Likelihood to Fail vs. Size

ScopeMaster
## Portfolio Overview

### SOFTWARE

- **8,129** Requirements in 90 apps
- **29,271** Cosmic Function Points
- **5,091** Pot. Defects Found
- **$3,690,750** Indicative value delivered by ScopeMaster.

<table>
<thead>
<tr>
<th>Software</th>
<th>Measurable</th>
<th>Estimated Size</th>
<th>Defects per Story</th>
<th>ScopeMaster Quality Score</th>
<th>Defects</th>
<th>Value of FINDING problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-REG MM IMPORT TEST&lt;br&gt;reimport of C-REG MM IMPORT TEST 1&lt;br&gt;OWNER access</td>
<td>100%&lt;br&gt;22 of 22</td>
<td>165 CFP</td>
<td>4.4</td>
<td>3.5</td>
<td>Ambiguous : 0&lt;br&gt;Pot. Missing : 88&lt;br&gt;Pot. Duplicated : 9</td>
<td>$14,550</td>
</tr>
<tr>
<td>Library 2 (as user stories)&lt;br&gt;Same as library 2 but written as user stories&lt;br&gt;OWNER access</td>
<td>68%&lt;br&gt;7 of 8</td>
<td>38 CFP</td>
<td>2.5</td>
<td>0.9</td>
<td>Ambiguous : 1&lt;br&gt;Pot. Missing : 19</td>
<td>$3,000</td>
</tr>
<tr>
<td>A big one&lt;br&gt;test of auto generated requirements&lt;br&gt;OWNER access</td>
<td>64%&lt;br&gt;102 of 160</td>
<td>833 CFP</td>
<td>2.6</td>
<td>2.6</td>
<td>Ambiguous : 58&lt;br&gt;Pot. Missing : 333&lt;br&gt;Pot. Duplicated : 23</td>
<td>$62,100</td>
</tr>
</tbody>
</table>
Detailed Quality Report

- Clear (unambiguous)
- Complete
- Concise
- Consistent

**Potential Defects: Ambiguity and Duplication**

- Jump to Missing and Duplication Defects
- Congratulations all requirements were interpreted and sized. See additional suggestions of potential ambiguities.

**Potential Defects: requirements CRUD Omissions and Duplication**

- Missing: 63
- Duplicate: 2

The value of finding ($25–100) and fixing ($150) these defects early is at least $11,375

<table>
<thead>
<tr>
<th>Object</th>
<th>Create</th>
<th>Read</th>
<th>Update</th>
<th>Delete</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>account</td>
<td>missing</td>
<td>2 proceed to th</td>
<td>missing</td>
<td>missing</td>
<td>missing</td>
</tr>
<tr>
<td>behaviour</td>
<td>missing</td>
<td>missing</td>
<td>2 proceed to th</td>
<td>missing</td>
<td>missing</td>
</tr>
<tr>
<td>category</td>
<td>missing</td>
<td>duplicate</td>
<td>missing</td>
<td>missing</td>
<td>missing</td>
</tr>
</tbody>
</table>
Proposes Functional Tests

Suggested positive and negative functional tests
Software Analysis Tools that help you write better software

Code the Right Thing

Work as a team

Code it Well
“…it’s the only static analysis tool for requirements that I have ever seen”

“(should) improve development team productivity by at least 10%”

“…would have saved 3 – 6 months in the requirement’s gathering process”

“to me it’s a no brainer…”

“…Its amazing that you have come up with the right set of functional requirements.”

“Using ScopeMaster, not only I did I find and fix 150 defects in 2 days, it was actually fun to use!”

“doesn’t just help find the requirements defects but helps educate authors to prevent them in future!”
Adding Stories

There are four ways to add requirements:

1. **One at a time**
2. **CSV**
3. **Jira plugin**
4. **Story Builder**
Jira Cloud Plugin

Avoids double entry
Recap

• Large Projects **Need Help** - focus on size and quality will help reduce failure.

• Finding defects **early is VERY efficient**

• The COSMIC measurement process helps improve quality and is the **engineering metric** of software functionality

• Natural Language Processing of user stories, Can help both quality and measurement. **Fast and highly effective.**
Thanks!

References

https://cosmic-sizing.org

Economics of Software Quality by Capers Jones

https://www.amazon.co.uk/Applied-Software-Measurement-Analysis-Productivity/dp/0071502440

Story points and CFP
https://www.scopemaster.com/blog/story-points-compared-with-cfp/

Appendix
Suited to:
- English text
- Agile or Waterfall
- New applications
- System changes
- Business Applications (inc ERP)
- Mobile Apps
- Embedded systems
- Before, During or After Coding
- Benchmarking development work

Expect:
- Better Quality
- Faster Delivery
- Reduce rework
- Fewer Bugs
- Reduced Project Risk
- Lower Costs
- Teaches requirements authors

Not Suited to:
- Computational intense systems
- AI applications
Ask yourself…

1. Do we have a **software project with quality problems**?

2. How many of those problems could be attributed to the **requirements quality or volatility**?

3. How much could we have saved in **time and effort** if we’d used ScopeMaster.
And for outsourced development...

1. Are we **paying the right amount** for our development? How do we know? (resources)
2. Do we have **reliable estimates** of duration and cost?
3. Has scope change been **costly**?
<table>
<thead>
<tr>
<th>Scope and Quality</th>
<th>Project</th>
<th>Commercial</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Higher quality achieved</td>
<td>Improve Team Productivity</td>
<td>Lower Outsourced dev costs</td>
<td>Organisational Learning</td>
</tr>
<tr>
<td>Better, faster, cheaper deliveries</td>
<td>Better Project Control</td>
<td>Vendor Management</td>
<td>Portfolio Management</td>
</tr>
<tr>
<td>More Stable</td>
<td>Project risk reduction</td>
<td>Project Assurance</td>
<td>Benchmarking</td>
</tr>
<tr>
<td></td>
<td>Avoid project Disasters</td>
<td></td>
<td>Learn to write better stories</td>
</tr>
<tr>
<td></td>
<td>Recover failing project</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Effective Functional Sizing - Speed

125 – 500 FP / Day

500 – 2500 FP / day
Allows the user to review the tools’ interpretation