IT Complexity Crisis; Danger and Opportunity

Agenda

- The Failure of Failure
- Another Approach to Failure Analysis
- Measuring Complexity
- A Practical Example
- SIP: A Methodology for Reducing Complexity
- Summary
- Call to Action
Two Studies

- 2009 Chaos report published by the Standish Group

- The Impact of Size and Volatility on IT Project Performance by Chris Sauer, Andrew Gemino, and Blaize Horner Reich, Comms of the ACM Nov 07

2002 Data on IT Success

- Standish:
  - Successful: 28%
  - Challenged: 51%
  - Failed: 20%

- Sauer
  - Good or Star: 68%
  - Challenged: 23%
  - Failed: 9%

Who is right?
Four Problems with Studies

- Poor definitions
- Self selected respondents
- No reporting of statistical significance
- Wrong analysis

The Wrong Analysis

<table>
<thead>
<tr>
<th>Project</th>
<th>Budget</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>$100K</td>
<td>Succeed</td>
</tr>
<tr>
<td>P2</td>
<td>$100K</td>
<td>Succeed</td>
</tr>
<tr>
<td>P3</td>
<td>$250K</td>
<td>Succeed</td>
</tr>
<tr>
<td>P4</td>
<td>$300K</td>
<td>Succeed</td>
</tr>
<tr>
<td>P5</td>
<td>$5M</td>
<td>Fail</td>
</tr>
</tbody>
</table>

According to Standish/Sauer, this is a 75% success rate. (4 out of 5 projects successful)

But based on IT budget, this is a 13% success rate. ($750K out of 5,750K delivered value.

Which is right?
BUT

What if all projects are equally likely to fail?

Then the Standish/Sauer numbers make more sense.

Comparing Both Studies

<table>
<thead>
<tr>
<th>Labor Cost (USD)</th>
<th>Sauer</th>
<th>Standish (2009)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;750K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$1.9M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10M+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effort (Person Months)</th>
<th>Sauer</th>
<th>Standish (2009)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 24PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>750PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1700PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2400+PM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Four Conclusions

Failure rates go up exponentially with size.

If your project is more than $2M, it will probably not be successful.

If your project is more than $10M, it will definitely not be successful.

It makes no difference what methodology you use.

Agenda

- The Failure of Failure
- Another Approach to Failure Analysis
- Measuring Complexity
- A Practical Example
- SIP: A Methodology for Reducing Complexity
- Summary
- Call to Action
The Cost of Failure

- 2.75% GDP spent on hardware, software, services
- 66% Federal IT dollars in “at risk” projects
- 65% “at risk” dollars will be lost
- 7.5 indirect costs for every dollar direct costs.

\[0.0275 \times 0.66 \times 0.66 \times 7.5 = 0.089 = \text{Failure factor}\]

Example (UK)

GDP of UK = $2,260 B

\[0.089 \times 2,260 \text{ B} = 200 \text{ B/year}\]

= $548 M/day
Who is the enemy?

Low Complexity
- well managed
- well aligned
- well understood.

High Complexity
- poorly managed
- poorly aligned
- poorly understood

Dependent variables

Independent variable

These are not the problem

This is the problem

Agenda

- The Failure of Failure
- Another Approach to Failure Analysis
- Measuring Complexity
- A Practical Example
- SIP: A Methodology for Reducing Complexity
- Summary
- Call to Action
Glass’s Law

For every 25 percent increase in problem complexity, there is a 100 percent increase in solution complexity.

- Facts and Fallacies of Software Engineering by Robert Glass

Glass’s Law Applied to SOAs

For every 25 percent increase in the business functionality in a service, there is a 100 percent increase in the complexity of that service.

For every 25 percent increase in the number of connections in a service, there is a 100 percent increase in the complexity of that service.
SCU = Standard Complexity Unit

The amount of complexity in an average atomic business function implemented in an idealized green field environment with no internal or external interactions.

Examples of atomic business functions: process-check, hire-employee, remove-from-inventory

Bird’s Formula

Glass’s Law

1.25 \text{ functionality} = \frac{2}{F_x} \times C_x

Bird’s Formula

A system with \(bf\) business functions is

\[ 10 \log(C_x) \times \log(bf) / \log(F_x) \]

\text{times more complex than a system of 1 business function.}
Simplifying Bird’s Formula

10^{\log(Cx) \times \log(bf) / \log(Fx)}

is with constants replaced

10^{\log(2) \times \log(bf) / \log(1.25)}

Equivalent to

10^{0.30103 \times \log(bf) / 0.09691}

Equivalent to Glass’s Constant

10^{3.10 \log(bf)}

Sessions’s Formula for Service Complexity

A system with bf business functions and cn connections has a complexity value (in SCUs) of

10^{\log(Cx) \times \log(bf) / \log(Fx)} + 10^{\log(Cx) \times \log(cn) / \log(Fx)}

Equivalent to

10^{3.10 \log(bf)} + 10^{3.10 \log(cn)}
Sessions’s Formula for SOA Complexity

An SOA with m services has a total complexity (in SCUs) of

\[ \sum_{i=1}^{m} 10^{3.10 \log(bfi)} + 10^{3.10 \log(cni)} \]

Agenda

- The Failure of Failure
- Another Approach to Failure Analysis
- Measuring Complexity
- A Practical Example
- SIP: A Methodology for Reducing Complexity
- Summary
- Call to Action
Practical Problem

- Say architecture A and B both solve a business problem
- Say A is twice as complex as B
- Then A will cost twice as much, work half as well, and be twice as likely to fail as B
- BUT ... how do we know which is more complex?

Example: Interlibrary Loan

Library 1

Library 2

Inter Library Loan Agency
Two Architectural Solutions

- Solution 1: 892 SCUs
- Solution 2: 1170 SCUs
- Solution 2 will cost 30% more than Solution 1 and be 30% more likely to fail.

The Solution Space

Problem P has n solutions: \{s_1, s_2, s_3, \ldots s_n\}

Every solution si has a complexity measure C(si)

We want to find the solution sx that has the lowest complexity measure C(sx)

If P contains 20 business functions, n > 50 trillion.

We need a process to drive us to sx.
Agenda

- The Failure of Failure
- Another Approach to Failure Analysis
- Measuring Complexity
- A Practical Example
- SIP: A Methodology for Reducing Complexity
- Summary
- Call to Action

SIP

Simple Iterative Partitions
SIP Process

The Process - Decomposition

Interlibrary Loan

Borrower

Lender

Ask To Borrow
Process Positive Response
Process Negative Response

Process Loan Request
Agree To Loan
Reject to Loan
Process New Book
Update-Catalog
### Partitioning and Synergistic Analysis

<table>
<thead>
<tr>
<th>Task</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ask to borrow</td>
<td>A</td>
</tr>
<tr>
<td>Process loan request</td>
<td>B</td>
</tr>
<tr>
<td>Agree to loan</td>
<td>B</td>
</tr>
<tr>
<td>Process positive response</td>
<td>A</td>
</tr>
<tr>
<td>Refuse to loan</td>
<td>B</td>
</tr>
<tr>
<td>Process negative response</td>
<td>A</td>
</tr>
<tr>
<td>Process new book</td>
<td>B</td>
</tr>
<tr>
<td>Add book to catalog</td>
<td>C</td>
</tr>
</tbody>
</table>

### The Process - Subsetting

<table>
<thead>
<tr>
<th>Task</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ask to borrow</td>
<td>A</td>
</tr>
<tr>
<td>Process positive response</td>
<td>A</td>
</tr>
<tr>
<td>Process negative response</td>
<td>A</td>
</tr>
<tr>
<td>Receive catalog</td>
<td>A</td>
</tr>
<tr>
<td>Process loan request</td>
<td>B</td>
</tr>
<tr>
<td>Agree to loan</td>
<td>B</td>
</tr>
<tr>
<td>Refuse to loan</td>
<td>B</td>
</tr>
<tr>
<td>Process new book</td>
<td>B</td>
</tr>
<tr>
<td>Add book to catalog</td>
<td>C</td>
</tr>
</tbody>
</table>
The Process - Connecting

- Ask to borrow
  - Process positive response
  - Process negative response
  - Receive catalog

- Process loan request
  - Agree to loan
  - Refuse to loan
  - Process new book

- Add book to catalog
  - Send out catalog

The SIP Architecture

- Borrower
  - Borrow Book
  - Process Positive Response
  - Process Negative Response
  - Receive Catalog

- ILLA
  - Receive New Book Item
  - Update Catalog

- Lender
  - Process Borrow Request
  - Agree to Loan Request
  - Reject Loan Request
  - Process New Book

306 SCUs
Stepping Back

- Standard Architecture 1: 892 SCUs
- Standard Architecture 2: 1170 SCUs
- SIP Architecture: 314 SCUs

The SIP Generated Solution is 35% of the cost of the least expensive non-SIP solution.

The best non-SIP Solution is almost 3 times more likely to fail as the SIP solution.

Side by Side

- Architecture 1 (892)
- Architecture 2 (1170)
- SIP Architecture (306)
Goals of SIP

- A process to drive the simplest possible system design that solves a given business problem
- Based on mathematical models of complexity, partitioning, and set theory.
- Reproducible and verifiable.

Agenda

- The Failure of Failure
- Another Approach to Failure Analysis
- Measuring Complexity
- A Practical Example
- SIP: A Methodology for Reducing Complexity
- Summary
- Call to Action
Presentation Summary

- Project success rates is meaningless.
- Budget success rates are important.
- There appears a strong correlation between complexity and success.
- We can measure complexity.
- We can minimize it.
- We can save a lot of money.

It’s simple to make things complex.
It’s complex to make things simple.
But simple things are the only things that work.
Agenda

- The Failure of Failure
- Another Approach to Failure Analysis
- Measuring Complexity
- A Practical Example
- SIP: A Methodology for Reducing Complexity
- Summary
- Call to Action

Call To Action

- A major opportunity to improve the world.
  - Financial Benefits
  - Efficiency Benefits
  - Social Benefits
- We need to work together to investigate this problem.
  - Research
  - Better analytic tools
  - Standardized complexity management processes
- We need a consortium to make this happen.