Large Scale Software System Development

The role of creating systems engineering in the Interfront iCBS development program
(11 April 2013)
Overview of Case Study presentation

1. Introductions
2. Program size and context complexity
3. System of systems challenges encountered
4. Program achievements
5. Systems engineering approach
6. Engineering the creating system
   a. Organisational system
   b. Technical enabling system
7. Some notes on the application of agile development methodologies
Introduction

• Interfront
• SARS Customs Modernisation Program
• My involvement
Product size and complexity

- **Functional scope:** Core declaration processing sub-system – 27 Use Cases and 300 requirements
- **Business Rules:**
  - 1,321 Configurable rules
  - 158 Complex rules
  - 146,340 Default conditionalities
- **Sub-systems:** 6
- **Managed xsd interfaces:** 29
- **Queues:** 71
- **Code base:** 2 EAR’s, 32 WAR’s, 325 JAR’s
- **Performance/Load requirements:**
  - end-to-end transaction: Average 50 sec (1-10,000 line declarations)
  - current production load: 20,000/day. Must scale to 60,000/day.
  - Parallel run: 25,000/4 hrs = 6250/hr = 104/minute
- **Availability requirements:** 24/7 (Upgrades midnight)
- **Reliability/Integrity requirements:** 100% transactions accounted for. Accuracy of duties/taxes (scents-level)
System performance

![Graph of System Performance](image-url)
Context complexity

- **Context complexity**
  - Team size: 63
  - External interfacing systems: 10
  - User/Operator interfaces: 4 (different operational functional/support departments)
  - Contracting parties: 6
  - Distributed program team
  - Dynamic, fast pace project and industry in government organisation (bureaucratic establishment)

- **High degree of uncertainty (context complexity)**
  - Full scope of system unclear and continuously changing
  - No specification or complete description of existing system to be replaced
  - Transition strategy being developed as the system matures
  - Daily detail requirements changes (small to large impact) in a response to daily legacy system parallel testing
The SARS context...
Typical system of systems (SoS) Challenges

1. System elements operate independently
2. System elements have different life cycles
3. The initial requirements are likely to be ambiguous: Requirements for an SoS mature as the system elements mature
4. Complexity is a major issue: Added system elements result in non-linear interaction growth
5. Management can overshadow engineering: coordination of requirements, budget constraints, schedules, interfaces, and technology complicates development
6. Fuzzy boundaries cause confusion: No one controls the definition of the external interfaces unless someone defines and controls the system boundaries
7. SoS is never finished
Software Development still seriously challenged: Standish CHAOS report

- 50,000+ Software Projects
- Around the world
- Since 1994
- 2009 results – even worse:
  - 32% succeed
  - 44% challenged
  - 24% failed

Source: Standish Group
Standish CHAOS report summary

1994 to 2006 Standish Chaos Report Summary

<table>
<thead>
<tr>
<th>Year</th>
<th>Succeeded</th>
<th>Failed</th>
<th>Challenged</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>16%</td>
<td>31%</td>
<td>53%</td>
</tr>
<tr>
<td>1996</td>
<td>27%</td>
<td>40%</td>
<td>33%</td>
</tr>
<tr>
<td>1998</td>
<td>26%</td>
<td>28%</td>
<td>46%</td>
</tr>
<tr>
<td>2000</td>
<td>28%</td>
<td>23%</td>
<td>49%</td>
</tr>
<tr>
<td>2002</td>
<td>34%</td>
<td>15%</td>
<td>51%</td>
</tr>
<tr>
<td>2004</td>
<td>29%</td>
<td>18%</td>
<td>53%</td>
</tr>
<tr>
<td>2006</td>
<td>35%</td>
<td>19%</td>
<td>46%</td>
</tr>
</tbody>
</table>
Program Achievements

• **Daily full system releases**: Full build, full regression testing, integration, transition into production environment, data loading, deploy, stabilise, full parallel testing run

• **Running whole country’s declarations through INTF system and comparing to legacy daily – automated** (execution, results and comparison analysis). Currently less than 5% overall differences in INTF and legacy results.

• **Daily production delta data take-on** (of all sub-systems)

• **Coverage of all system activity**, catching all exceptions and faults, with ability to manage and resolve issues with needed security clearance levels (System Management Console)

• **Track any message end-to-end** from any of the systems in the greater SARS system. “Every transaction is precious”
Learning fast...

5 Second MBA: New Product Launch

- Ready
- Aim
- Fire

X = 1980s

- Ready
- Fire

X = 1990s

- Fire

X = 2000s
Systems Engineering: System of systems in context

Source: INCOSE Systems Engineering Handbook, 2010
Key SE-related concepts

• Change management
• Configuration Management
• Non-functionals ("ilities" / RAMS)
• Error/Failure Handling (FMEA/FMECA)
• Created vs creating system (Enabling Systems Engineering)
• Management of internal and external dependencies
• Information Management
• Quality Management
• Decision Management
• Risk Management

Reference: SE Body of Knowledge (http://www.sebokwiki.org)
Creating/Enabling Systems Engineering

• Estimated 30-50% of program effort went into development and maintenance of the ‘creating system’ (i.e. 20-30 out of every 60 man hours)

• Includes:
  – Enterprise systems engineering (incl. Process)
  – Tooling and supporting system engineering
Structuring teams and organisations to develop SoS’s

- **Conway’s law:** “…organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations”

- **Balance in system & organisational hierarchies:** 7 ± 2 rule (manage organisational and communication complexity)

Source: SE Body of Knowledge (http://www.sebokwiki.org)
SARS & Interfront SDLC’s overlay

SARS Bus Req

SARS SubSys Req

SARS QA Test

SARS Int Test

INFT Sys Req

SARS subSys Dev & Test

INFT Sys Test

INFT Subsys Req

INFT subSys Test

INFT subSys Dev & Test
Relevant ‘Creating system’ concepts

- **Enterprise Systems Engineering** (ESE): The application of systems engineering principles, concepts, and methods to the planning, design, improvement, and operation of an enterprise (SE Body of Knowledge [http://www.sebokwiki.org](http://www.sebokwiki.org))

- **Self-organising teams**: (i.e., not organised by mandate) in that the sentient beings in the enterprise will find for themselves some way in which they can interact to produce greater results than can be done by the individuals alone. Self-organizing enterprises are often more flexible and agile than if they were organized from above (Dyer and Ericksen 2009; Stacey 2006)

- **DevOps**: DevOps is frequently described as a more collaborative and productive relationship between development teams and operations teams. This improved relationship and collaboration increases efficiency and reduces the production risk associated with frequent changes. (Wikipedia)

- **Release Management**: The process of managing software releases from development stage to software release. (Wikipedia)

- **Continuous Integration**: the practice, in software engineering, of merging all developer workspaces with a shared mainline several times a day
Capabilities in the Enterprise

Individual Competence Leads to Organizational, System & Operational Capability
Key ‘creating system’ elements (Process & Organisation)

• **Team structuring**
  – Systems engineering capability in Interfront and SARS
  – Independent data team controlling interface specifications and data baselines
  – Instilling self-contained functions for **new** software development disciplines:
    • Release Management
    • DevOps
  – Self-organising teams
  – Interdependence between teams (reflected in KPIs)

• **Requirements and specification practices**
  – Blue Print and specification base lining (Enterprise Architect)
  – Formal Change management (ECPs, issue logs etc)
  – Documentation management (drafts, releases, distribution control)

• **Interfront release approach**
  – Incremental system releases
  – Release planning and documentation (continuous)
  – Disciplined issue tracking and reporting
  – Internal Release Management incl. full release content audits
  – Baseline control register (system of systems level)
Key ‘creating system’ elements (Process & Organisation) cont....

- Continuous integration
- **Systems Engineering & Design Authority (SE&DA)** (Cross-discipline, cross-team design function facilitated by SE)
- **Test and Qualification approach**
  - Verification and coverage measure **against specification** (RVTM’s)
  - **Levels of qualification** testing with **quality gates**
  - **Regression testing documentation** and **configuration control** for repeatable execution, and change control of test case procedures
  - **Qualification environments** management
- **Establishment of rhythms**
  - **Daily rhythms** enabled through self organising teams: Program level, team level, client interactions
    Daily dev cycle: reporting, analysis, planning, implementation, build, test and release
  - **Weekly rhythms**: Program steerco, Project reporting, SEDA sessions, ECPs scheduling and building
Business Agility – Enterprise Architecture

One type of enterprise architecture that supports agility is a non-hierarchical organization without a single point of control. Individuals function autonomously, constantly interacting with each other to define the vision and aims, maintain a common understanding of requirements and monitor the work that needs to be done. Roles and responsibilities are not predetermined but rather emerge from individuals’ self-organizing activities and are constantly in flux. Key decisions are made collaboratively, on the spot, and on the fly. Because of this, knowledge, power, and intelligence are spread through the enterprise, making it uniquely capable of quickly recovering and adapting to the loss of any key enterprise component.

Source: (http://en.wikipedia.org/wiki/Business_agility)
Processes...

Last year we recognized that our processes were far too complex.

So we put them into the cloud.

Let the clouds make your life easier.
ECP Process

• ECP: Engineering Change Proposal
• Changes in requirements are managed by ECP’s
• It provides a structured and controlled approach to product changes and project planning (incl. costing)
• ECP’s includes:
  – Detail descriptions/specification of the required change
  – An impact assessment of the change i.t.o. :
    • Work required to implement
    • Cost to implement
    • Timescale impact on other items already in development
• ECP’s are formally approved by the client before development is triggered, which serves as the delivery contract
• Approved ECP’s are added to the product’s functional baseline
Continuous Integration

Continuous Integration (CI) is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily – leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

(Martin Fowler)

- Early identification of faulty software
- Automated
- 100% pass rate of all tests required
- Immediate roll-back on failure (strict baseline and stability control)
- Down-stream qualification efforts maximised for system level functional testing
System Engineering & Design Authority (SE&DA)

• Multi-disciplinary
• Typical topics:
  – Archiving strategy and implications
  – System Upgrade and data migration strategy
  – Logging approach and levels
  – Error handling across the system
  – Transaction auditing across the system
  – Queue monitoring and reporting for support
  – Operationalisation and external system Integration
General Model for Interfront Levels of testing & Environments

Development Test environments

Dev Component Testing → Dev Integration Testing

QA Test environments

Sub-system QA Testing → System Integration Testing

Integration environments

Performance, availability & endurance Testing → External Integration Testing

Support environments

Client QA Issues

Client QA support & reproduction

Production issues

Client Prod support & reproduction

Client environments

Prod. Bug Fixes QA Testing

Prod. Bug Fixes Dev Testing

Client QA Fixes QA Testing

Client QA Fixes Dev Testing

Production
Key ‘creating system’ elements (Technical and Tooling)

- **Code generation**: Spec change results in new interface model which gets directly translated into code (reduced time from 6 wks average to 10mins)
- **Integrated System Life Cycle Management infrastructure (ALM)**:
  - Bugzilla (end-to-end item management, target milestone deliveries, item statuses)
  - SVN
  - Continuous Integration Infrastructure (Jenkins, Maven, Master-POMs, SVN, Green Screens)
  - Release content auditing: $(I^2T^2)$ and health check of builds and data
  - Internal release notes
- **Integrated Qualification System**:
  - Requirements definition (Enterprise Architect)
  - Change impact analysis
  - Test Design
  - Test Procedures (TestLink)
  - Test execution automation tool
  - Test scripts
  - Test Results
  - Requirements Verification and Traceability Matrix (RVTM)
  - Comprehensive Test Reporting (Dashboard)
Key ‘creating system’ elements (Technical and Tooling) cont...

- **System non-functional qualification infrastructure:**
  - System probes and monitoring framework (Tivoly, Hyperic)
  - Performance and load test scripting and execution framework (J-Meter, Squirrel, LoadUI)
- **System Processing Visualisation** (custom tool)
- **System Error reporting and transaction logging** (Issue diagnosis and resolution efficiency) (SMC)
- **Data Management infrastructure:**
  - Data Management Tool (DMT)
  - Extraction, Translation, Validation, Enrichment, Loading (ETL+) scripts and processing (to achieve alignment with WCO data model)
- **Infrastructure to support transitioning of product** into Production:
  - Compare tool (Automated Production parallel run comparison processing and reporting)
  - Adaptable Comparison Analysis reporting framework (Excel pivot table system)
  - Parallel Assistent Tool (PA Tool): Interrogation of parallel systems of systems for end-to-end transaction information
  - Production Replay Tool: Simulating full production environment on individual system-system interaction level, i.e. external system behaviour per production transaction, to reproduce exactly a production scenario
The realities of working on the iCBS system
Integrated Qualification system

1. **RVTM & Test Report**
   - Results & coverage
   - Requirements References
   - Import Test Results into TL
   - Alternative manual Export/Import results transfer to TL

2. **TestLink**
   - Test Link Plan (Manual)
   - Write test results into TL
   - Export Test Results (Manual)

3. **SVN**
   - Incorporate Test Plan in system build and execute
   -Send test results

4. **Automation Tool Plug-in**
   - Build of target system
   - Execute tests
   - Test scripts and test data
   - Test results

5. **Results Listener**
   - Automated results transfer to TL
   - Test results

6. **Automation Tool**
   - Test Scripts, Data & Results
   - Export Test Results

7. **Test Results**
   - Incorporate Test Plan in system build and execute

8. **Integrated Qualification system**
   - Automated in CI environment. Manual trigger mechanism in qualification environment

---

**TestLink Adapter**

---

**Smart Systems. Better Borders.**
Twelve Principles of Agile SW Development

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software: **Daily full system deploy**

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage: **Daily prioritisation of new development items**

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. **Daily full system deploy**

4. Business people and developers must work together daily throughout the project. **Daily through systems engineering teams and program management**

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. **Self-organising teams**

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. **Daily communication rhythms, open plan, Skype groups**
Twelve Principles of Agile SW Development cont...

7. Working software is the primary measure of progress. **Continuous integration – full regression testing with results**

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. **Multiple releases/week with stable baselines over months (Major system changes included)**

9. Continuous attention to technical excellence and good design enhances agility. **Collaborative, continuous design function – SE&DA**

10. Simplicity—the art of maximizing the amount of work not done—is essential. **As simple as needed, and not simpler. Making things simple in a large development is not easy.**

11. The best architectures, requirements, and designs emerge from self-organizing teams. **Yes**

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. **Retrospectives, Roles dedicated to improving efficiencies and ways of work (DevOps, Org systems engineer). Continuous pressure to up performance.**

[Source: Agile Alliance: http://www.agilealliance.org/the-alliance/the-agile-manifesto/the-twelve-principles-of-agile-software/]
Key ingredient #1: Pressure creates diamonds...

YOUR DAD'S BEEN UNDER A LOT OF PRESSURE LATELY.
Key ingredient #2: If nothing makes sense, make fun...

“This Venn diagram tells us nothing, but it’s so cute!”
Thank You!

Questions / Comments?