Systems and Software Design Principles for Large-Scale Mission-Critical Embedded Products from Aerospace and Financial Problem Domains

October 2008

Rick Selby
Head of Software Products
Northrop Grumman Space Technology
310-813-5570, Rick.Selby@NGC.com

Adjunct Professor of Computer Science
University of Southern California
Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

Overview

- Systems and software engineering strategies, principles, benefits, and tradeoffs
  - Example large-scale mission-critical embedded software system

- Investigations of synthesis, analysis, and modeling principles
  - Synthesis: Lifecycle models
  - Synthesis: System architectures
  - Analysis: Reuse analysis
  - Analysis: Structure analysis
  - Modeling: Defect detection techniques
  - Modeling: Measurement and prediction

- Conclusions and future work
### Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

<table>
<thead>
<tr>
<th>PRINCIPLES</th>
<th>BENEFITS and TRADEOFFS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lifecyle models:</strong> Frequent synchronized design cycles and system releases</td>
<td>Organization of and parallelization within large-scale projects</td>
</tr>
<tr>
<td><strong>System architectures:</strong> Layered system architectures containing embedded meta-language programs and interpreters</td>
<td>Rapid feedback and innovation</td>
</tr>
<tr>
<td><strong>Reuse analysis:</strong> Reconfigurable component-driven development</td>
<td>Visibility into stabilization and handoffs</td>
</tr>
<tr>
<td><strong>Structure analysis:</strong> Inter-component connectivity analysis</td>
<td>User-customizability</td>
</tr>
<tr>
<td><strong>Defect detection techniques:</strong> Disciplined team-based peer reviews</td>
<td>Multi-platform portability</td>
</tr>
<tr>
<td><strong>Measurement and prediction:</strong> Automated measurement-driven analysis infrastructure using predictive models</td>
<td>Automated testing</td>
</tr>
<tr>
<td><strong>Organization of and parallelization within large-scale projects</strong></td>
<td>Sustainable multi-project reuse</td>
</tr>
<tr>
<td><strong>Rapid feedback and innovation</strong></td>
<td>Lower component defect rates</td>
</tr>
<tr>
<td><strong>Visibility into stabilization and handoffs</strong></td>
<td>Lower component development effort</td>
</tr>
<tr>
<td><strong>User-customizability</strong></td>
<td>Lower component defect correction effort</td>
</tr>
<tr>
<td><strong>Multi-platform portability</strong></td>
<td>Lower component development effort</td>
</tr>
<tr>
<td><strong>Automated testing</strong></td>
<td>Early lifecycle defect detection</td>
</tr>
<tr>
<td><strong>Sustainable multi-project reuse</strong></td>
<td>Low out-of-phase defect rates</td>
</tr>
<tr>
<td><strong>Lower component defect rates</strong></td>
<td>High return-on-investment for prevention</td>
</tr>
<tr>
<td><strong>Lower component development effort</strong></td>
<td>Early identification of high defect or high effort components</td>
</tr>
<tr>
<td><strong>Lower component defect correction effort</strong></td>
<td>Statistical process control</td>
</tr>
<tr>
<td><strong>Lower component development effort</strong></td>
<td>Pro-active process guidance</td>
</tr>
</tbody>
</table>
Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

Overview

- Systems and software engineering strategies, principles, benefits, and tradeoffs
- Example large-scale mission-critical embedded software system

- Investigations of synthesis, analysis, and modeling principles
  - Synthesis: Lifecycle models
  - Synthesis: System architectures
  - Analysis: Reuse analysis
  - Analysis: Structure analysis
  - Modeling: Defect detection techniques
  - Modeling: Measurement and prediction

- Conclusions and future work
Organizational Charter Focuses on Embedded Software Products

- Embedded software for
  - Advanced robotic spacecraft platforms
  - High-bandwidth satellite payloads
  - High-power laser systems
- Emphasis on both system management and payload software
- Reusable, reconfigurable software architectures and components
- Languages: O-O to C to asm
- CMMI Level 5 for Software in February 2004; ISO/AS9100; Six Sigma
- High-reliability, long-life, real-time embedded software systems
Prometheus Spacecraft Supports Jupiter JIMO Mission over 9 to 14 Year Duration
Prometheus Spacecraft for JIMO and Related Missions Enables Data-Intensive Science

- Spacecraft configuration PB1
  - 58m length
  - 36,375kg launch mass
  - 5 processors, excluding redundancy
  - 250mbps transfer, 500gbit storage, 10mbps downlink
  - Gas cooled power with 200kW Brayton output
  - Stows in 5m diameter fairing
Embedded software implements functions for commands & telemetry, subsystem algorithms, instrument support, data management, and fault protection.

Size of on-board software growing to accelerate data processing and increase science yield.

Software “adds value” to mission by enabling post-delivery changes to expand capabilities and overcome hardware failures.
Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

Overview

- Systems and software engineering strategies, principles, benefits, and tradeoffs

- Example large-scale mission-critical embedded software system

Investigations of synthesis, analysis, and modeling principles

- Synthesis: Lifecycle models
- Synthesis: System architectures
- Analysis: Reuse analysis
- Analysis: Structure analysis
- Modeling: Defect detection techniques
- Modeling: Measurement and prediction

Conclusions and future work
Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

**PRINCIPLES**

- Lifecycle models: Frequent synchronized design cycles and system releases
- System architectures: Layered system architectures containing embedded meta-language programs and interpreters
- Reuse analysis: Reconfigurable component-driven development
- Structure analysis: Inter-component connectivity analysis
- Defect detection techniques: Disciplined team-based peer reviews
- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models

**BENEFITS and TRADEOFFS**

- Organization of and parallelization within large-scale projects
- Rapid feedback and innovation
- Visibility into stabilization and handoffs

**SCALING DIMENSIONS**

- Teams
- Projects
- Domains
### Incremental Software Builds Deliver Early Capabilities and Accelerate Integration and Test

#### CY 2004-2013

<table>
<thead>
<tr>
<th>Year</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>ATP/PMSR</td>
<td>SM PDR</td>
<td>SM CDR</td>
<td>BUS I&amp;T</td>
</tr>
<tr>
<td>11/04</td>
<td>108f_154</td>
<td>108f_154</td>
<td>108f_154</td>
<td>108f_154</td>
</tr>
</tbody>
</table>

#### Flight Computer Unit (FCU) Builds

<table>
<thead>
<tr>
<th>FCU1</th>
<th>Prelim Exec and C&amp;DH Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCU2</td>
<td>Final Exec and C&amp;DH Software</td>
</tr>
<tr>
<td>FCU3</td>
<td>Science Computer Interface</td>
</tr>
<tr>
<td>FCU4</td>
<td>Power Controller Interface</td>
</tr>
<tr>
<td>FCU5</td>
<td>AACS (includes autonomous navigation)</td>
</tr>
<tr>
<td>FCU6</td>
<td>Thermal and Power Control</td>
</tr>
<tr>
<td>FCU7</td>
<td>Configuration and Fault Protection</td>
</tr>
</tbody>
</table>

#### Science Computer Unit (SCU) Builds

<table>
<thead>
<tr>
<th>SCU1</th>
<th>Prelim Exec and C&amp;DH Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCU2</td>
<td>Final Exec and C&amp;DH Software</td>
</tr>
</tbody>
</table>

#### Data Server Unit (DSU) Builds

<table>
<thead>
<tr>
<th>DSU1</th>
<th>Prelim Exec and C&amp;DH Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSU2</td>
<td>Final Exec and C&amp;DH Software</td>
</tr>
<tr>
<td>DSU3</td>
<td>Data Server Unique Software</td>
</tr>
</tbody>
</table>

#### Ground Analysis Software (GAS) Computer Builds

<table>
<thead>
<tr>
<th>GAS1</th>
<th>Preliminary Ground Analysis Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAS2</td>
<td>Final Ground Analysis Software</td>
</tr>
</tbody>
</table>

#### Legend:

- **Design Agent**
- **Performer of Activity N**
- **JPL**
- **NGC**
- **Role/activity shared by JPL and NGC**
- **Prototype Activity**

N is defined as follows:
1. Requirements
2. Preliminary Design
3. Detailed Design
4. Code and Unit Test/Software Integration
5. Verification and Validation

© Copyright 2001-2008. Richard W. Selby and Northrop Grumman Corporation. All rights reserved.
Projects Define Risk Mitigation “Burn Down” Charts with Specific Tasks and Exit Criteria

- **Risk Score**
- **Risk Score**
- **Risk Score**

**Exit/Success Criteria:**
1. BM1 complete; customer concurs with approach
2. Software requirements scope estimated (preliminary)
3. Software control board established (preliminary); change control process established.
4. SDP released. Spec tree defined.
5. RTOSS lab evaluation completed. Capabilities validated using sim.
6. Software requirements scope estimated (final)
7. System development process flow models implemented.
8. Spacecraft/subsystems/etc. users define use cases (for IFs, functions, nominal ops, off-nominal ops, etc.) completed.
10. Baseline allocation of SW requirements to IFCs with growth/correction/deficiency completed.
11. Software control board (final) established
12. SwRR conducted. NASA customer agrees with software requirements.
13. Finalize IFC2 requirements: Inter-module & inter-subsystem IFs completed. Validated requirements using models/sim.
15. Finalize IFC3 requirements: Subsystems major functions completed. Validated requirements using models/sim.
18. Finalize IFC5 requirements: Subsystems off-nominal operations completed. Validated requirements using models/sim.
19. Finalize IFC5 requirements: Subsystems off-nominal operations completed. Validated requirements using models/sim.
20. Deliver IFC7: No new capabilities; Only system I&T corrections completed. SW complete for 1st mission.

**Events**

- 2005
- 2006
- 2007
- 2008
- 2009
- 2010

**Planned Risk Level**

- **Actual Risk Level**

<table>
<thead>
<tr>
<th>Planned (Solid = Linked, Hollow = Unlinked)</th>
<th>Control Points</th>
<th>Completed</th>
</tr>
</thead>
</table>
Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

**PRINCIPLES**

- **Lifecycle models**: Frequent synchronized design cycles and system releases
- **System architectures**: Layered system architectures containing embedded meta-language programs and interpreters
- **Reuse analysis**: Reconfigurable component-driven development
- **Structure analysis**: Inter-component connectivity analysis
- **Defect detection techniques**: Disciplined team-based peer reviews
- **Measurement and prediction**: Automated measurement-driven analysis infrastructure using predictive models

**SYNTHESIS**

**ANALYSIS**

**MODELING**

**BENEFITS and TRADEOFFS**

- User-customizability
- Multi-platform portability
- Automated testing

**SCALING DIMENSIONS**

© Copyright 2001-2008. Richard W. Selby and Northrop Grumman Corporation. All rights reserved.
Common Requirements Enable Software Product Lines and Layered Architectures Across Projects

Legend
- Mission-specific
- Common across projects

Mission-Specific requirements

Project 1
- Mission-Specific SW
  - Table-driven configurations, commands
- Common SW
  - Functions for
    - Protocols, Fault management, Attitude, Power, Thermal, etc.
    - Real-time operating system
- Mission-Specific HW
  - Processor & I/O Interfaces

Project 2
- Mission-Specific SW
  - Table-driven configurations, commands
- Common SW
  - Functions for
    - Protocols, Fault management, Attitude, Power, Thermal, etc.
    - Real-time operating system
- Mission-Specific HW
  - Processor & I/O Interfaces

Project 3
- Mission-Specific SW
  - Table-driven configurations, commands
- Common SW
  - Functions for
    - Protocols, Fault management, Attitude, Power, Thermal, etc.
    - Real-time operating system
- Mission-Specific HW
  - Processor & I/O Interfaces

Project 4
- Mission-Specific SW
  - Table-driven configurations, commands
- Common SW
  - Functions for
    - Protocols, Fault management, Attitude, Power, Thermal, etc.
    - Real-time operating system
- Mission-Specific HW
  - Processor & I/O Interfaces
Architecture Uses Simple Task Structure, Deterministic Processing, and Predictable Timeline

- Three-task structure: 32ms task (high rate), 128ms (minor cycle), and background task
- Minor cycle serves as the main workhorse task that executes commands, formats telemetry, and handles fault protection
- Minor cycle command processor reads active command sequences and executes individual deterministic commands
- >50% margins at system delivery for processor, memory, storage, and bus

* Not to scale
Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

**PRINCIPLES**

- Lifecycle models: Frequent synchronized design cycles and system releases
- System architectures: Layered system architectures containing embedded meta-language programs and interpreters
- Reuse analysis: Reconfigurable component-driven development
- Structure analysis: Inter-component connectivity analysis
- Defect detection techniques: Disciplined team-based peer reviews
- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models

**BENEFITS and TRADEOFFS**

- Sustainable multi-project reuse
- Lower component defect rates
- Lower component development effort

**SCALING DIMENSIONS**

- Projects: 1
- Teams: >1
- Domains: >1
32% of Software Components are Either Reused or Modified from Previous Systems

- Data from 25 NASA systems
- Component origins: 68.0% new development, 4.6% major revision, 10.3% slight revision, and 17.1% complete reuse without revision
Analyses of Component-Based Software Reuse Shows Favorable Trends for Decreasing Faults

- Data from 25 NASA systems
- Overall difference is statistically significant ($\alpha < .0001$). Number of components (or modules) in each category is: 1629, 205, 300, 820, and 2954, respectively.
Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

**PRINCIPLES**

- Lifecycle models: Frequent synchronized design cycles and system releases
- System architectures: Layered system architectures containing embedded meta-language programs and interpreters
- Reuse analysis: Reconfigurable component-driven development
- Structure analysis: Inter-component connectivity analysis
- Defect detection techniques: Disciplined team-based peer reviews
- Measurement and prediction: Automated measurement-driven analysis infrastructure using predictive models

**SCALING DIMENSIONS**

1  

>1

1  

>1

**BENEFITS and TRADEOFFS**

- Lower component defect rates
- Lower component defect correction effort
- Lower component development effort

Enables?
Analyses of Software Architectures Shows Fault Trends for Component Interactions

- Data from 23 NASA systems
- 5469 components analyzed and categorized by quintiles
Analyses of Software Architectures Shows Fault Trends for Component Interaction Relative Factors

- Data from 23 NASA systems
- 5469 components analyzed and categorized by quintiles

**Relative Factors for Component Interactions**

- Faults (ave.)
- FaultCorrectionEffort (ave.)
- DevelopmentEffort (ave.)

<table>
<thead>
<tr>
<th>Quintile</th>
<th>Interactions per KSLOC per Component (relative factor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>0.85 0.74 0.53</td>
</tr>
<tr>
<td>2nd</td>
<td>0.92 0.93 0.66</td>
</tr>
<tr>
<td>3rd</td>
<td>1.00 1.00 1.00</td>
</tr>
<tr>
<td>4th</td>
<td>1.25 1.26 1.09</td>
</tr>
<tr>
<td>5th</td>
<td>1.44 1.28 2.11</td>
</tr>
</tbody>
</table>
Analyses of Software Requirements Shows Leading Indicators for Implementation Scope and Priorities

- Data from 14 NASA systems
- Ratio of implementation size to software requirements has 81:1 average and 35:1 median; Excluding system #14, the ratio has 46:1 average and 33:1 median
- Ratio of software requirements to system requirements has 6:1 average
Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

**PRINCIPLES**

- **Lifecycle models**: Frequent synchronized design cycles and system releases
- **System architectures**: Layered system architectures containing embedded meta-language programs and interpreters
- **Reuse analysis**: Reconfigurable component-driven development
- **Structure analysis**: Inter-component connectivity analysis
- **Defect detection techniques**: Disciplined team-based peer reviews
- **Measurement and prediction**: Automated measurement-driven analysis infrastructure using predictive models

**SYNTHESIS**

**ANALYSIS**

**MODELING**

**SCALING DIMENSIONS**

- Early lifecycle defect detection
- Low out-of-phase defect rates
- High return-on-investment for prevention

**ENABLES**

- Enables
Analyses of Software Defect Injection Phases Reveals Distributions

- Distribution of software defect injection phases based on using peer reviews across 12 system development phases
- 3418 defects, 731 peer reviews, 14 systems, 2.67 years
- 49% of defects injected during requirements phase
Analyses of Software Defect Injection and Detection Phases Reveal Distributions and Gaps

Cumulative distribution of software defect injection and detection phases based on using peer reviews across 12 system development phases

- 3418 defects, 731 peer reviews, 14 systems, 2.67 years
- 50% defects injected by requirements, 70% by detailed design; Gap shows leakage
Web-Based Workflow Tools and Infrastructure Support Software Process Flow

**Process flow**

**Peer reviews**

**Exit checklist**

**Action items**
Research Investigates Systems and Software Engineering Principles, Benefits, and Tradeoffs

**PRINCIPLES**

- Lifecycle models: Frequent synchronized design cycles and system releases
- System architectures: Layered system architectures containing embedded meta-language programs and interpreters
- Reuse analysis: Reconfigurable component-driven development
- Structure analysis: Inter-component connectivity analysis
- Defect detection techniques: Disciplined team-based peer reviews

**SYNTHESIS**

**ANALYSIS**

**MODELING**

**SCALING DIMENSIONS**

- Early identification of high defect or high effort components
- Statistical process control
- Pro-active process guidance

**BENEFITS and TRADEOFFS**

**Enables?**

- Automated measurement-driven analysis infrastructure using predictive models
Data-Driven Statistical Analyses Identify Trends, Outliers, and Process Improvements for Defects

Defect Density for Code / Unit Test Peer Reviews

- Six Sigma Project Introduced New Peer Review Process
- Provided Training on Process

- New Web-Based Peer Review Tool
- Provided Training on Tool

These defects are action items resulting from peer reviews of software code and unit testing plans and results.

Data from 10 systems

- Control chart of metric data from example Six Sigma projects focusing on fault (or defect) density in peer reviews of software components
- Process improvements decreased variances and decreased means
Measurement-Driven Decision Models (Trees, Networks) Predict High-Risk Software Components

- Focus on high-payoff areas: the 80:20 rule
- Generate decision trees or networks automatically
  - Scalable to large systems
  - Leverage previous experience and calibrate to new environments
- Integrate measurements from processes, products, projects, teams, and organizations

Example:

```
+ : Classified as likely to have property P (e.g., high integration faults)
- : Classified as unlikely to have property P
```

```
+ : Classified as likely to have property P (e.g., high integration faults)
- : Classified as unlikely to have property P
```

```
+ : Classified as likely to have property P (e.g., high integration faults)
- : Classified as unlikely to have property P
```

```
+ : Classified as likely to have property P (e.g., high integration faults)
- : Classified as unlikely to have property P
```
Predictive Models Identify Leading Indicators of High-Fault and High-Effort Components

- Data from 16 NASA systems. 1920 model variations.
- Consistency is 100% minus percent false positives. Completeness is 100% minus percent false negatives.
Interactive metric dashboards incorporate a variety of information and features to help developers and managers characterize progress, identify outliers, compare alternatives, evaluate risks, and predict outcomes.
Research Investigates Systems and Software Synthesis, Analysis, and Modeling Principles

Overview

- Systems and software engineering strategies, principles, benefits, and tradeoffs
- Example large-scale mission-critical embedded software system
- Investigations of synthesis, analysis, and modeling principles
  - Synthesis: Lifecycle models
  - Synthesis: System architectures
  - Analysis: Reuse analysis
  - Analysis: Structure analysis
  - Modeling: Defect detection techniques
  - Modeling: Measurement and prediction

- Conclusions and future work
Define “Grand Challenges” Problems for Systems and Software Engineering

Example from Computing (2004)

Contents

Introduction
Tony Hoare and Robin Milner

GC1 In Vivo–in Silico (ViS): the virtual worm, weed and bug
Ronan Sleep

GC2 Science for global ubiquitous computing
Marta Kwiatkowska and Vladimiro Sassone

GC3 Memories for life: managing information over a human lifetime
Andrew Fitzgibbon and Ehud Reiter

GC4 Scalable ubiquitous computing systems
Jon Crowcroft

GC5 The architecture of brain and mind
Aaron Sloman

GC6 Dependable systems evolution
Jim Woodcock

GC7 Journeys in non-classical computation
Susan Stepney

Source: http://www.ukcrc.org.uk/gcresearch.pdf


- Rick.Selby@NGC.com