Managing COTS Integration for High Integrity Systems: Observations from the COCOTS Database

Betsy Clark
March 5, 2003

GSAW 2003

This work is sponsored by the FAA’s Software Engineering Resource Center
Topics Covered

• Background
• Some empirical observations about the use of COTS in high integrity systems
  – Types of products
  – Attributes considered in evaluating COTS
  – Strategies observed
• Conclusions
Background

• Empirical observations come from COCOTS database

• **Constructive COTS Model**
  – Estimation tool
  – Analysis tool

• Part of the COCOMO suite of tools
  – Open model
  – In the public domain
COCOMO 81 (1981)

COCOMO II (2000)
COCOTS

- Sponsored by FAA’s Software Engineering Resource Center (SERC)

- Calibrated with data from twenty projects
  - 11 can be classified as “high integrity”
  - 9 are other (business, support systems)
“High Integrity” COTS-Based Systems in COCOTS Database

• Classification based on whether or not system is safety-critical
• Most of these systems operate 24/7
• Eleven total
  – 5 are Air Traffic Management (FAA)
  – 3 are Air-to-Ground Communication (FAA)
  – 1 is Radar Processing (FAA)
  – 1 is Missile Tracking (Air Force)
  – 1 is Satellite Control (NASA)
Questions Asked about COTS and High Integrity Systems

• What types of COTS products do they use?
• What attributes do they consider when selecting a product?
• What strategies do they use to ensure system integrity?
Defining COTS

• Commercial-Off-The-Shelf Software
  – Sold, leased, licensed at advertised prices
  – Source code unavailable
  – Periodic releases with feature growth and fixes
  – Eventual obsolescence, end of life

• Each part of this definition has implications
Implications

Sold, leased, licensed at advertised prices

- Market forces play an important role
- Success or failure is no longer simply a technical issue
- Is it in the vendor’s interest to be cooperative?
- Will the vendor be in business in a few years?
Implications -2

Source code unavailable

- If the source code is available for modification, *from an estimating perspective*, this is a case of reuse
  - Effort is a function of lines of code to be understood, added, modified, deleted
- Without source code, activities change
  - Assessing/evaluating
  - Tailoring (using vendor-provided mechanisms)
  - Writing glue code
- These are the activities that are modeled by COCOTS
Implications - 3

Periodic releases with feature growth and fixes
Eventual obsolescence, end of life

- Requires continual upgrades to avoid end of life
- You have no control over product evolution
- Maintenance complexity grows very quickly with the number of COTS products
Topics Covered

• Background

• Some empirical observations about the use of COTS in high integrity systems
  – Types of products
  – Attributes considered in evaluating COTS
  – Strategies observed
  – Lessons learned

• Conclusions
## Types of COTS Products

<table>
<thead>
<tr>
<th>Type of Product</th>
<th>Projects Using</th>
</tr>
</thead>
<tbody>
<tr>
<td>operating system(s)</td>
<td>91%</td>
</tr>
<tr>
<td>GUI generator</td>
<td>73%</td>
</tr>
<tr>
<td>DBMS</td>
<td>55%</td>
</tr>
<tr>
<td>network management</td>
<td>27%</td>
</tr>
<tr>
<td>communications protocol</td>
<td>18%</td>
</tr>
<tr>
<td>disk array</td>
<td>18%</td>
</tr>
<tr>
<td>data warehouse, device drivers, telemetry processing, off-line analysis tools,</td>
<td>9%</td>
</tr>
<tr>
<td>C++ class library</td>
<td></td>
</tr>
</tbody>
</table>

*Source: [Software Metrics, Inc.](http://example.com)*
Something interesting is going on…

• The 9 projects that were NOT classified as “high integrity” almost never mentioned operating systems and other infrastructure as COTS
  – Why not?
• What they did mention were higher-level applications
  – e.g., Oracle Financials
• Which leads to another interesting part of the definition of COTS
Revisiting the Definition of COTS

• People view “COTS” as products that are associated with some risk
  – Point made by Vic Basili during keynote address at ICCBSS 2003

• For our set of high integrity systems, operating systems are viewed as a source of risk and are subject to risk-mitigation activities, e.g.
  – Assessment before buying
  – Purchasing source code

• Not the case for the other (non high-integrity) systems
Topics Covered

• Background
• Some empirical observations about the use of COTS in high integrity systems
  – Types of products
  – Attributes considered in evaluating COTS
  – Strategies observed
• Conclusions
Attributes Considered in Selecting COTS Products: “High Integrity” CBS

- Rank Order by Frequency

<table>
<thead>
<tr>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Product Performance (throughput, response time)</td>
</tr>
<tr>
<td>- Inter-component compatibility</td>
</tr>
<tr>
<td>- Availability/Robustness (fault tolerance, input error tolerance, reliability)</td>
</tr>
<tr>
<td>- Functionality</td>
</tr>
<tr>
<td>- Price</td>
</tr>
<tr>
<td>- Vendor support (response time for critical problems)</td>
</tr>
<tr>
<td>- Product and Vendor Maturity</td>
</tr>
<tr>
<td>- Understandability (documentation quality, testability)</td>
</tr>
<tr>
<td>- Version compatibility (upward, downward)</td>
</tr>
<tr>
<td>- Ease of use</td>
</tr>
<tr>
<td>- Correctness</td>
</tr>
<tr>
<td>- Flexibility, extendibility</td>
</tr>
<tr>
<td>- Vendor concessions (access to source code)</td>
</tr>
<tr>
<td>- Installation ease</td>
</tr>
<tr>
<td>- Portability</td>
</tr>
<tr>
<td>- Security</td>
</tr>
<tr>
<td>- User training</td>
</tr>
</tbody>
</table>
Topics Covered

• Background
• Some empirical observations about the use of COTS in high integrity systems
  – Types of products
  – Attributes considered in evaluating COTS
  – Strategies observed
• Conclusions
Strategies Observed to Ensure System Availability/Reliability

- Fault-tolerant architectures
- Detailed evaluations before purchasing COTS
- Use of mature components
- Support agreements requiring 24-hour response time for critical problems
- Purchase of source code
Maintenance Challenges

- Managing COTS volatility (new versions over time)
  - Lots of time spent analyzing impact of upgrading to new versions
- Initial observations suggest a non-linear impact of the sheer number of products on maintenance complexity
  - Multiple configurations make this much worse
Strategies to Address Maintenance

• Glue code wrappers
  – Used to hide functionality to allow upgrades without impacting rest of system

  “We wanted to be able to replace a product without damage. As an example, we have a wrapper around the data base. It could be a flat file, relational…the custom application doesn’t care.”

• Freezing configuration (not upgrading any COTS products) while purchasing source code for critical components

• Distinguishing between critical and non-critical components with focus on the former to avoid end-of-life
Conclusions: COTS and High Integrity Systems

- Observations from the COCOTS Database
  - Types of Products
    - Infrastructure
    - GUI generators
    - DBMS
  - Attributes Evaluated
    - Product performance
    - Interoperability
    - Availability/fault tolerance
  - Challenges faced
    - Ensuring reliability and availability in the initial system
    - Maintenance
  - Strategies
    - Variety of strategies including detailed evaluations, purchase of source code, use of mature components
    - Maintenance strategies ranged from freezing the configuration to use of wrappers to minimize negative impact of upgrades
Plea for more data!

Betsy Clark  
(703) 754-0115  
Betsy@software-metrics.com

Chris Abts  
(979) 862-8055  
cabts@cgssb.tamu.edu  
cabts@sunet.usc.edu
For more information about COCOTS
http://sunset.usc.edu
Questions?