Riding Herd on Total Ownership Costs: Herding Sheep vs. Herding Cats

Barry Boehm, USC
STC 2015 Keynote Address
October 13, 2015
Cowboys Herding Cats
Software O&M Often like Herding Cats

- Some Root Causes of Cat-Like Behavior

- Many opportunities to reduce total ownership costs (TOC)
  - By emphasizing software Changeability and Dependability
  - Both rely on Maintainability via SERC System Qualities Ontology

- Opportunities organized via Maintainability Opportunity Tree
  - Anticipate Modifiability Needs
  - Design, Develop for Modifiability
  - Anticipate Repairability Needs
  - Design, Develop for Repairability
  - Expedite Diagnosis
  - Improve Modification and Repair Verifiability; Skills

- Conclusions
Problem and Opportunity (%O&M costs)

- **US Government IT**: >75%; $62 Billion [GAO 2015]
- **Hardware** [Redman 2008]
  - 12% -- Missiles (average)
  - 60% -- Ships (average)
  - 78% -- Aircraft (F-16)
  - 84% -- Ground vehicles (Bradley)
- **Software** [Koskinen 2010]
  - 75-90% -- Business, Command-Control
  - 50-80% -- Complex platforms as above
  - 10-30% -- Simple embedded software
- **Primary current emphasis**: minimize acquisition costs
Software O&M Often like Herding Cats

- Software and Users evolve in Incompatible directions
  - Non-Developmental Items (COTS, Clouds, Open Source)
  - Independently evolving co-dependent external systems
  - Multi-mission sources of change
  - Breakage of brittle point-solution architectures
  - Priority changes: competition, technology, organizations

- Herders are often ill-prepared
  - Minimal voice in acquisition
  - Missing deliverables: diagnostics, test support, architecture documentation, CM support
  - Diversity of deliverables from multiple sources
  - Unfamiliar domains, infrastructure
  - Missing capabilities: Rainy-day use cases
Some Root Causes of Cat-Like Behavior

- **Stovepipe acquisition of interoperating systems**
  - Incompatible infrastructure, NDIs, user interfaces, ...

- **Acquisitions based on lowest-cost, technically-acceptable implementation of fixed requirements, resulting in**
  - Brittle, point-solution architectures
  - CAIV-driven loss of information on post-delivery needs
  - Minimal interpretation of “technically acceptable”
    - Just implementing sunny-day requirements

- **Minimal maintainer participation, planning, preparation**
  - Missing maintainer deliverables: diagnostics, test support, architecture documentation, CM support
  - Incompatibilities among post-deployment evolution parties

- **Inadequate SysE resources, leading to severe technical debt**
  - First to be impacted by optimistic budgets and schedules
The Conspiracy of Optimism
Take the lower branch of the Cone of Uncertainty
Added Cost of Minimal Software SysE
Based on COCOMO II calibration data

% Added Cost, Very Low vs. Very High RESL Rating

Software Product Size (KSLOC)

10 100 1,000 10,000

18 38 63 92
How Much Architecting is Enough?

Sweet Spot Drivers:
Rapid Change: leftward
High Assurance: rightward
Outline

• Software O&M Often like Herding Cats
  – Some Root Causes of Cat-Like Behavior

Many opportunities to reduce total ownership cost (TOC)
  – By emphasizing software Changeability and Dependability
  – Both rely on Maintainability via SERC System Qualities Ontology

• Opportunities organized via Maintainability Opportunity Tree
  – Anticipate Modifiability Needs
  – Design, Develop for Modifiability
  – Anticipate Repairability Needs
  – Design, Develop for Repairability
  – Expedite Diagnosis
  – Improve Modification and Repair Verifiability; Skills

• Conclusions
Maintainability Challenges and Responses

- Maintainability supports both Dependability and Changeability
  - Distinguish between Repairability and Modifiability
  - Elaborate each via means-ends relationships
- Multiple definitions of Changeability
  - Distinguish between Product Quality and Quality in Use (ISO/IEC 25010)
  - Provide mapping between Product Quality and Quality in Use viewpoints
- Changeability can be both external and internal
  - Distinguish between Maintainability and Adaptability
- Many definitions of Resilience
  - Define Resilience as a combination of Dependability and Changeability
- Variability of Dependability and Changeability values
  - Ontology addresses sources of variation
  - Referents (stakeholder values), States, Processes, Relations with other SQs
- Need to help stakeholders choose options, avoid pitfalls
  - Qualipedia, Synergies and Conflicts, Opportunity Trees
Product Quality View of Changeability

MIT Quality in Use View also valuable

- **Changeability (PQ):** Ability to become different product
  - Swiss Army Knife, Brick Useful but not Changeable
- **Changeability (Q in Use):** Ability to accommodate changes in use
  - Swiss Army Knife does not change as a product but is Changeable
  - Brick is Changeable in Use (Can help build a wall or break a window?)

```
Adaptability
  • Self-Diagnosability
  • Self-Modifiability
  • Self-Testability

Modifiability
  • Understandability
  • Modularity
  • Scalability
  • Portability

Maintainability
  • Defects
  • Repairability

Testability
  • Diagnosability
  • Accessibility
  • Restorability
```

Internal

External

Subclass of

Means to End
Dependability, Changeability, and Resilience

- Reliability
  - Defect Freedom
  - Survivability
  - Fault Tolerance
    - Complete
      - Robustness
      - Self-Repairability
    - Partial
      - Graceful Degradation
      - Choices of Security, Safety ...
  - Testability, Diagnosability, etc.

- Dependability, Availability
  - Resilience
  - Changeability
  - Maintainability
    - Repairability
    - Testability
      - Test Plans, Coverage
      - Test Scenarios, Data
      - Test Drivers, Oracles
        - Test Software Qualities
      - A
        - B
      - Means to End
      - Subclass of
Outline

• Software O&M Often like Herding Cats
  – Some Root Causes of Cat-Like Behavior

• Many opportunities to reduce total ownership costa (TOC)
  – By emphasizing software Changeability and Dependability
  – Both rely on Maintainability via SERC System Qualities Ontology

Opportunities organized via Maintainability Opportunity Tree
  – Anticipate Modifiability Needs
  – Design, Develop for Modifiability
  – Improve Modification V&V
  – Anticipate Repairability Needs
  – Design, Develop for Repairability
  – Improve Repair Diagnosis
  – Improve Repair, V&V Processes

• Conclusions
Maintainability Opportunity Tree: Modifiability

- Evolution requirements
  - Trend analysis
  - Hotspot (change source) analysis
  - Modifier involvement
  - Address Potential Conflicts

- Design/Develop for Modifiability
  - Modularize around hotspots
  - Service-orientation; loose coupling
  - Spare capacity; product line engineering
  - Domain-specific architecture in domain
  - Enable user programmability
  - Address Potential Conflicts

- Improve Modification V&V
  - Prioritize, Schedule Modifications, V&V
  - Modification compatibility analysis
  - Regression test capabilities
  - Address Potential Conflicts
Maintainability Opportunity Tree: Repairability

Anticipate Repairability Needs
- Repair facility requirements
- Repair-type trend analysis
- Repair skills analysis
- Repair personnel involvement
- Address Potential Conflicts

Design/Develop for Repairability
- Repair facility design, development
- Replacement parts trend analysis
- Replacement parts logistics development
- Replacement accessibility design
- Address Potential Conflicts

Improve Repair Diagnosis
- Smart system anomaly, trend analysis
- Informative error messages
- Multimedia diagnosis guides
- Fault localization
- Switchable spare components
- Address Potential Conflicts

Improve Repair, V&V Processes
- Prioritize, Schedule Repairs, V&V
- Repair compatibility analysis
- Regression test capabilities
- Address Potential Conflicts
Elaborating Modifiability Benefits - I

- **Evolution Requirements**
  - Keep, prioritize below-the-line IOC requirements
  - Use to determine modularization around sources of change, reduce ripple effects of changes

- **Trend Analysis**
  - Identify, prioritize responses to sources of change
  - Marketplace, competition, usage trends, mobility trends
  - Use to refine, evolve architecture

- **Agile Methods, User Programmability**
  - Enable rapid response to rapid change

- **Hotspot Analysis**
  - Gather data on most common sources of change
  - Use to modularize architecture, reduce ripple effects of changes
Use of Empirical Data in TOC Models:
Pareto 80-20 Cost-to-fix Distribution
Contracts: Fixed cost and nominal-case requirements; 90 days to PDR

Major Rework Sources:
Off-Nominal Architecture-Breakers
A - Network Failover
B - Extra-Long Messages
Rework Sources Analysis: Projects A and B

- Change processing over 1 person-month = 152 person-hours

<table>
<thead>
<tr>
<th>Category</th>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra long messages</td>
<td>3404+626+443+328+244= 5045</td>
<td></td>
</tr>
<tr>
<td>Network failover</td>
<td>2050+470+360+160= 3040</td>
<td></td>
</tr>
<tr>
<td>Hardware-software interface</td>
<td>620+200= 820</td>
<td>1629+513+289+232+166= 2832</td>
</tr>
<tr>
<td>Encryption algorithms</td>
<td></td>
<td>1247+368= 1615</td>
</tr>
<tr>
<td>Subcontractor interface</td>
<td>1100+760+200= 2060</td>
<td></td>
</tr>
<tr>
<td>GUI revision</td>
<td>980+730+420+240+180 =2550</td>
<td></td>
</tr>
<tr>
<td>Data compression algorithm</td>
<td></td>
<td>910</td>
</tr>
<tr>
<td>External applications interface</td>
<td>770+330+200+160= 1460</td>
<td></td>
</tr>
<tr>
<td>COTS upgrades</td>
<td>540+380+190= 1110</td>
<td>741+302+221+197= 1461</td>
</tr>
<tr>
<td>Database restructure</td>
<td>690+480+310+210+170= 1860</td>
<td></td>
</tr>
<tr>
<td>Routing algorithms</td>
<td>494+198= 692</td>
<td></td>
</tr>
<tr>
<td>Diagnostic aids</td>
<td>360</td>
<td>477+318+184= 979</td>
</tr>
<tr>
<td><strong>TOTAL:</strong></td>
<td><strong>13620</strong></td>
<td><strong>13531</strong></td>
</tr>
</tbody>
</table>
When investments made in architecture, average time for change order becomes relatively stable over time…

Relative* Total Ownership Cost (TOC)
For single system life cycle (TOC-SS)

* Cumulative architecting and rework effort relative to initial development effort
Elaborating Modifiability Benefits – II and Repairability Benefits

• Service-Oriented Architecture improves Interoperability
• Product-Line Engineering improves Total Ownership Cost (TOC)
  – Identify, modularize around product line Commonalities
  – Develop domain architecture, interfaces to Variabilities
  – Fewer components to modify, repair
• Improved Repairability improves Availability, TOC
  – Availability = MTBF / (MTBF + MTTR)
• Stakeholder Value-Based V&V improves Cost, Mission Effectiveness
  – Prioritizing inspection, test activities
  – Balancing level of inspection, test activities vs. rapid fielding
Systems Product Line Flexibility Value Model

For Set of Products:
- Average Product Cost
- Annual Change Cost
- Ownership Time
- Percent Mission-Unique, Adapted, Reused
- Relative Cost of Developing for PL Flexibility via Reuse
- Relative Costs of Reuse

As Functions of # Products, # Years in Life Cycle:
- PL Total Ownership Costs
- PL Flexibility Investment
- PL Savings (ROI)

2/21/2011
Current Systems Product Line Model

Systems Product Line Flexibility Value Model

Welcome SERC Collaborator

System Costs
Average Product Development Cost (Burdened $M) 5
Annual Change Cost (% of Development Cost) 10
Ownership Time (Years) 3
Interest Rate (Annual %) 7

Product Line Percentages
Unique % 40
Adapted % 30
Reused % 30

Relative Costs of Reuse (%)
Relative Cost of Reuse for Adapted 40
Relative Cost of Reuse for Reused 5

Investment Cost
Relative Cost of Developing for PL Flexibility via Reuse 1.7

Results

<table>
<thead>
<tr>
<th># of Products</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Cost ($M)</td>
<td>$7.1</td>
<td>$2.7</td>
<td>$2.7</td>
<td>$2.7</td>
<td>$2.7</td>
<td>$2.7</td>
<td>$2.7</td>
</tr>
<tr>
<td>Ownership Cost ($M)</td>
<td>$2.1</td>
<td>$0.8</td>
<td>$0.8</td>
<td>$0.8</td>
<td>$0.8</td>
<td>$0.8</td>
<td>$0.8</td>
</tr>
<tr>
<td>Cum. PL Cost ($M)</td>
<td>$8.2</td>
<td>$12.7</td>
<td>$16.2</td>
<td>$19.7</td>
<td>$23.1</td>
<td>$26.6</td>
<td>$30.1</td>
</tr>
<tr>
<td>PL Flexibility Investment ($M)</td>
<td>$2.1</td>
<td>$0.0</td>
<td>$0.0</td>
<td>$0.0</td>
<td>$0.0</td>
<td>$0.0</td>
<td>$0.0</td>
</tr>
<tr>
<td>PL Effort Savings</td>
<td>($2.7)</td>
<td>$0.3</td>
<td>$3.3</td>
<td>$6.3</td>
<td>$9.4</td>
<td>$12.4</td>
<td>$15.4</td>
</tr>
<tr>
<td>Return on Investment</td>
<td>-1.30</td>
<td>0.14</td>
<td>1.58</td>
<td>3.02</td>
<td>4.46</td>
<td>5.90</td>
<td>7.34</td>
</tr>
</tbody>
</table>

Return on investment

-1.3 0.1 1.6 3.0 4.5 5.9 7.3
1 2 3 4 5 6 7

Product #
Reuse at HP’s Queensferry Telecommunication Division

Time to Market (months)

Year

Non-reuse Project
Reuse project
Tradeoffs Among Cost, Schedule, and Reliability:
COCOMO II

- For 100-KSLOC set of features
- Can “pick all three” with 77-KSLOC set of features
Cost of Downtime Survey

- **Industry Sector Revenue/Hour**
  - Energy $2.8 million
  - Telecommunications $2.0 million
  - Manufacturing $1.6 million
  - Financial Institutions $1.4 million
  - Information Technology $1.3 million
  - Insurance $1.2 million
  - Retail $1.1 million
  - Pharmaceuticals $1.0 million
  - Banking $996,000

• Multi-mission sources of change
• Many independently-evolving co-dependent systems
  – Over 155 on Future Combat Systems
• Multiple-contract-change complexities
  – Average 141 vs. 48 workdays to process on 2 large SoSs
• Limited ability to apply interoperability improvements

• These also affect systems that become parts of SoSs
Examples of Utility Functions: Cost of Delay

- **Real-Time Control; Event Support**
  - **Value** vs. **Time**

- **Mission Planning, Competitive Time-to-Market**
  - **Value** vs. **Time**
    - **Critical Region**

- **Event Prediction - Weather; Software Size**
  - **Value** vs. **Time**

- **Data Archiving**
  - **Value** vs. **Time**
Cost of Delay vs. Dependability Assurance

- Early Startup: Risk due to low dependability
- Commercial: Risk due to low dependability
- High Finance: Risk due to low dependability
- Risk due to market share erosion

Combined Risk Exposure

\[ RE = P(L) \times S(L) \]

Sweet Spot

COCOMO II: 0 12 22 34 54
COQUALMO: 1.0 .475 .24 .125 .06
Early Startup: .33 .19 .11 .06 .03
Commercial: 1.0 .56 .32 .18 .10
High Finance: 3.0 1.68 .96 .54 .30
Market Risk: .008 .027 .09 .30 1.0

Added % test time

P(L) S(L) RE

RE = Market Share Erosion
Early Startup
Commercial
High Finance
Addressing Potential Conflicts

- With Performance: Loose vs. tight coupling (supercomputing)
- With Development Cost and Schedule: More to design, develop, V&V (rapid fielding)
- With Usability: Too many options (Office 2010)
- With Security: Too many entry points (Windows)
- With Scalability, Safety, Security: Agile methods
- With Dependability: User Programming, Self-Adaptiveness
- With Interoperability: Multi-Domain Architectures
- With Cost, Resource Consumption: Spare Capacity

These are not always conflicts, but candidates to consider. Need to balance risk of too little Modifiability with risk of too much.
• Alternatives analyzed shall include at least one architecture organized around:
  – Common sources of life-cycle change
    • For RPVs, these usually include user interface displays and controls, new payloads, self-defense and electronic warfare, data fusion, NCSoS protocols, and hardware maintainability
  – Risk analysis and prototyping of critical off-nominal scenarios
    • For RPVs, these usually include communications outages, anti-RPV threats, noisy and intermittent data sources, redirected missions, and cross-RPV coordination of responses
• Analyses of alternatives shall include total ownership cost comparisons
  – Based on relevantly-calibrated life cycle cost models
Outline

• Software O&M Often like Herding Cats
  – Some Root Causes of Cat-Like Behavior

• Many opportunities to reduce total ownership costa (TOC)
  – By emphasizing software Changeability and Dependability
  – Both rely on Maintainability via SERC System Qualities Ontology

• Opportunities organized via Maintainability Opportunity Tree
  – Anticipate Modifiability Needs
  – Design, Develop for Modifiability
  – Improve Modification V&V
  – Anticipate Repairability Needs
  – Design, Develop for Repairability
  – Improve Repair Diagnosis
  – Improve Repair, V&V Processes

Conclusions
Conclusions

• Major opportunities to reduce Total Ownership Costs
  – US Government IT: $62 Billion O&M (over 75% of TOC)

• Current investments overfocused on initial acquisition costs
  – Resulting in numerous herding-cats phenomena

• Maintainability Opportunity Trees offer attractive means to reduce Total Ownership Costs
  – But need to balance risks of too little Maintainability with risks of too much