COSYSMO 3.0: Lessons Learned from Collecting Systems Engineering Data

Jim Alstad  USC Center for Systems and Software Engineering
Barry W Boehm

Marilee Wheaton  The Aerospace Corporation

GSAW Evening Session
Interactive Session on Cost Estimation for Next-Generation Ground Systems
March 2, 2016

© 2016 by USC CSSE.
Published by The Aerospace Corporation with permission.
Outline

• Introduction & motivation
• Systems Engineering (SE) sizing with COSYSMO
• Lessons learned
• Conclusions
Introduction & Motivation

• Constructive Systems Engineering Cost Model (COSYSMO)
  – Development began in 2001 on COSYSMO 1.0
  – COSYSMO 1.0 published in 2005, COSYSMO 2.0 in 2009
  – Now working on COSYSMO 3.0
• Extensive practitioner support
  – PSSM, ISPA, INCOSE, GSAW, CSSE Corporate Affiliates
• Historical project data & industry calibration enables
  – understanding the model’s robustness
  – establishment of initial relationships between parameters and outcomes
  – validation of drivers
• Challenge is that SE measurement is still not standardized
Counting Guides for Sizing Systems Engineering

<table>
<thead>
<tr>
<th>Driver Name</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td># of System Requirements</td>
<td>Counted from system specification or requirements in system level model</td>
</tr>
<tr>
<td># of Interfaces</td>
<td>Counted from interface control document(s) or from model</td>
</tr>
<tr>
<td># of Operational Scenarios</td>
<td>Counted from test cases or use cases</td>
</tr>
<tr>
<td># of Critical Algorithms</td>
<td>Counted from system spec or mode description docs or from model</td>
</tr>
</tbody>
</table>

Lessons Learned:
Detailed counting rules can ensure that size drivers, specifically requirements, are counted consistently across the diverse set of systems engineering projects.

Detailed examples need to be provided to prevent double dipping across multiple size drivers.
Harmonized COSYSMO 3.0
Effort Multiplier Model

- Here are the 15 effort multipliers:

<table>
<thead>
<tr>
<th>Driver Name</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONOPS &amp; requirements understanding</td>
<td>Subjective assessment of the CONOPS &amp; the system requirements</td>
</tr>
<tr>
<td>Architecture understanding</td>
<td>Subjective assessment of the system architecture</td>
</tr>
<tr>
<td>Level of service requirements</td>
<td>Subjective difficulty of satisfying the key performance parameters</td>
</tr>
<tr>
<td>Migration complexity</td>
<td>Influence of legacy system (if applicable)</td>
</tr>
<tr>
<td>Technology risk</td>
<td>Maturity, readiness, and obsolescence of technology</td>
</tr>
<tr>
<td>Interoperability</td>
<td>Degree to which this system has to interoperate with others</td>
</tr>
<tr>
<td># and Diversity of installations/platforms</td>
<td>Sites, installations, operating environment, and diverse platforms</td>
</tr>
<tr>
<td># of Recursive levels in the design</td>
<td>Number of applicable levels of the Work Breakdown Structure</td>
</tr>
<tr>
<td>Stakeholder team cohesion</td>
<td>Subjective assessment of all stakeholders</td>
</tr>
<tr>
<td>Personnel/team capability</td>
<td>Subjective assessment of the team’s intellectual capability</td>
</tr>
<tr>
<td>Personnel experience/continuity</td>
<td>Subjective assessment of staff consistency</td>
</tr>
<tr>
<td>Process capability</td>
<td>CMMI level or equivalent rating</td>
</tr>
<tr>
<td>Multisite coordination</td>
<td>Location of stakeholders and coordination barriers</td>
</tr>
<tr>
<td>Tool support</td>
<td>Subjective assessment of SE tools</td>
</tr>
<tr>
<td>Development for reuse</td>
<td>Is this project developing artifacts for later reuse?</td>
</tr>
</tbody>
</table>
Lessons Learned

Lesson #1: Scope of the model
A standardized WBS and dictionary provides the foundation for decisions on what is within the scope of the model for both data collection and for estimating.

Lesson #2: Types of projects needed for data collection effort
Careful examination of potential projects is necessary to ensure completeness, consistency and accuracy across all required data collection items for the project.

Lesson #3: Size drivers
The collection of the size driver parameters requires access to project technical documentation as well as project systems engineering staff that can help interpret the content.
Lessons Learned

Lesson #4: Effort Multiplier
The rating of effort multiplier parameters for a completed project requires an assessment from the total project perspective.

Lesson #5: Systems Engineering hours across life cycle stages
Agree on a standardized set of life cycle stages for the model despite the different processes used by Affiliate companies.

Lesson #6: Data collection form
The data collection form must be easy to understand and flexible enough to accommodate organizations with different levels of detail so that they can contribute data and use the model.
Lessons Learned

Lesson #7: Definition
Spending more time on improving the driver definitions has ensured consistent interpretation and improved the model’s validity.

Lesson #8: Significance vs. data availability
If no data can be collected for a particular driver then that driver cannot be used because its influence on systems engineering effort cannot be validated.

Lesson #9: Influence of data on the drivers and statistical significance
Historical data can help determine which drivers should be kept in the model and which should be discarded.
Lessons Learned

Lesson #10: Data safeguarding procedure
Establishing non-disclosure agreements early on in the process enables the data sharing and collaboration to easily take place.

Lesson #11: Buy-in from constituents
The success of the model hinges on the support from the end-user community.
Conclusions

• Great support from practitioners during the development of previous versions of COSYSMO
  – Industry team resonated with critical need for model; and
  – Facilitated data source identification and collection

• Lessons learned are applicable to
  – parametric model building
  – systems engineering measurement

• More lessons to be learned as we proceed to model calibration with COSYSMO 3.0
References


5. “Lessons Learned from Collecting Systems Engineering Data”, Valerdi, Rieff, Roedler, Wheaton, CSER, 2004

Backup Charts
History of COSYSMO Models

COSYSMO 1.0
Valerdi, 2005
- Identifies form of model
- Identifies basic cost drivers
- Identifies Size measure

With Reuse
Wang et al, 2008
- Adds weights to Size elements, reducing net Size in the presence of reuse

Req’ts Volatile
Pena, 2012
- Adds scale factor based on requirements volatility

For Reuse
Wang et al, 2014
- Adds weights to Size elements, reducing net Size when artifacts are only partially completed

Sys of Sys
Lane et al, 2011
- Adds effort multiplier when in the presence of system-of-systems

COSYSMO 3.0
Alstad, 2016?
- Integrates features of previous models
COSYSMO 3.0
Top-Level Model

\[ PH = A \cdot (\text{AdjSize})^E \cdot \prod_{j=1}^{15} EM_j \]

Elements of the Harmonized COSYSMO 3.0 model:

- **Calibration parameter** \( A \)
- **Interoperability**
- **Size model**
  - eReq submodel, where 4 products contribute to size
  - Reuse submodel
- **Exponent (E) model**
  - Accounts for diseconomy of scale
  - Constant and 3 scale factors
- **Effort multipliers** \( EM \)
  - 15 EMs
Harmonized COSYSMO 3.0
Size Model

\[ AdjSize = \sum_{SizeDrivers} eReq(\text{Type}(SD), \text{Difficulty}(SD)) \times \]
\[ \text{PartialDevFactor}(RML_{\text{Start}}(SD), RML_{\text{End}}(SD), RType(SD)) \]

- **SizeDriver** is one of the system engineering products that determines size in the COSYSMO family (per [2]). Any product of these types is included:
  - System requirement
  - System interface
  - System algorithm
  - Operational scenario

- There are two submodels:
  - Equivalent nominal requirements ("eReq")
    - Raw size
  - Partial development
    - Adjusts size for reuse
Size Model – eReq Submodel

• The eReq submodel is unchanged from [2].
• The submodel computes the size of a SizeDriver, in units of eReq (“equivalent nominal requirements”)
• Each SizeDriver is evaluated as being easy, nominal, or difficult.
• Each SizeDriver is looked up in this size table to get its number of eReq:

<table>
<thead>
<tr>
<th>Size Driver Type</th>
<th>Easy</th>
<th>Nominal</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Requirement</td>
<td>0.5</td>
<td>1.0</td>
<td>5.0</td>
</tr>
<tr>
<td>System Interface</td>
<td>1.1</td>
<td>2.8</td>
<td>6.3</td>
</tr>
<tr>
<td>System Algorithm</td>
<td>2.2</td>
<td>4.1</td>
<td>11.5</td>
</tr>
<tr>
<td>Operational Scenario</td>
<td>6.2</td>
<td>14.4</td>
<td>30.0</td>
</tr>
</tbody>
</table>
Size Model –
Partial Development Submodel

• The basic concept:
  – If a reused SizeDriver is being brought in, that saves effort, and so we adjust the size by multiplying the raw size by a PartialDevFactor less than 1.
  – The value of PartialDevFactor is based on the maturity of the reused SizeDriver, and is looked up in a table [1].
    • How fully developed was the SizeDriver?
  – If there is no reuse for this SizeDriver, then PartialDevFactor = 1 (no adjustment).

<table>
<thead>
<tr>
<th>DWR Reuse Maturity Level:</th>
<th>New</th>
<th>Modified</th>
<th>Adapted</th>
<th>Adopted</th>
<th>Managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWR % of full-project cost (Table 4):</td>
<td>100.00%</td>
<td>66.73%</td>
<td>56.27%</td>
<td>38.80%</td>
<td>21.70%</td>
</tr>
</tbody>
</table>
COSYSMO 3.0

Exponent Model

• Exponent model is expanded from Peña [4, 9]

\[ E = E_{COSYSMO1} + SF_{ROR} + SF_{PC} + SF_{RV} \]

Where:

• \( E_{COSYSMO1} = 1.06 \) [2]
• \( ROR \) = Risk and Opportunity Resolution
• \( PC \) = Process Capability
• \( RV \) = Requirements Volatility

The effect of a large exponent is more pronounced on bigger projects