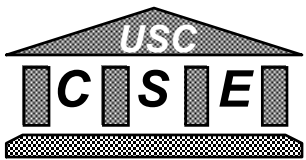


# **COTS Integration and COCOTS**

**Ray Madachy**

**CS 577b**

**February 3, 2003**



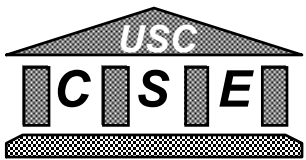
# Background

- **COTS trends in 577b and industry**
- **Research acknowledgements:**
  - **Chris Abts, USC**
  - **Betsy Bailey, Software Metrics Inc.**



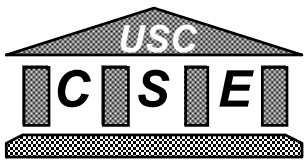
# COTS Definition

- **“Commercial Off the Shelf” Software**
- **Commercial Software Products**
  - sold, leased, licensed at advertised prices
- **Source Code Unavailable**
  - generally an application program interface (API)
  - frequently tailoring options
- **Usually periodic releases with feature growth, obsolescence**



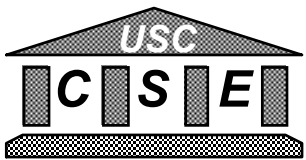
# Related Terms

- **COTS “Commercial Off the Shelf”**
  - **Black Box** (internal mods not allowed)
  - **White Box** (internal mods permitted - we treat as NDI)
- **GFE “Government Furnished Equipment”**
- **GOTS “Government Off the Shelf”**
- **NDI “Non-developmental Item/Not Developed In-house”**
- **REUSE Code**
  - source code originally written for some other project



# Rationale for Using COTS Products

- **Significant change in s/w development practice over past 20 years:**
  - building systems with pre-existing software to keep development & maintenance costs as low as possible
  - One such source: COTS
- **Rationale for COTS based systems:**
  - involve less development time by taking advantage of existing, market proven, vendor supported products, thus lowering overall development costs



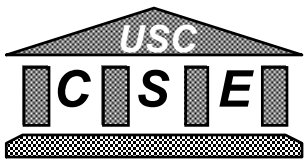
# COTS Advantages and Disadvantages

## Advantages

- Available now; earlier payback
- Avoids expensive development & maintenance
- Predictable license costs & performance
- Rich in functionality
- Broadly used, mature technology
- Frequent upgrades often anticipate organization's needs
- Dedicated support organization
- Hardware/software independence
- Tracks technology trends

## Disadvantages

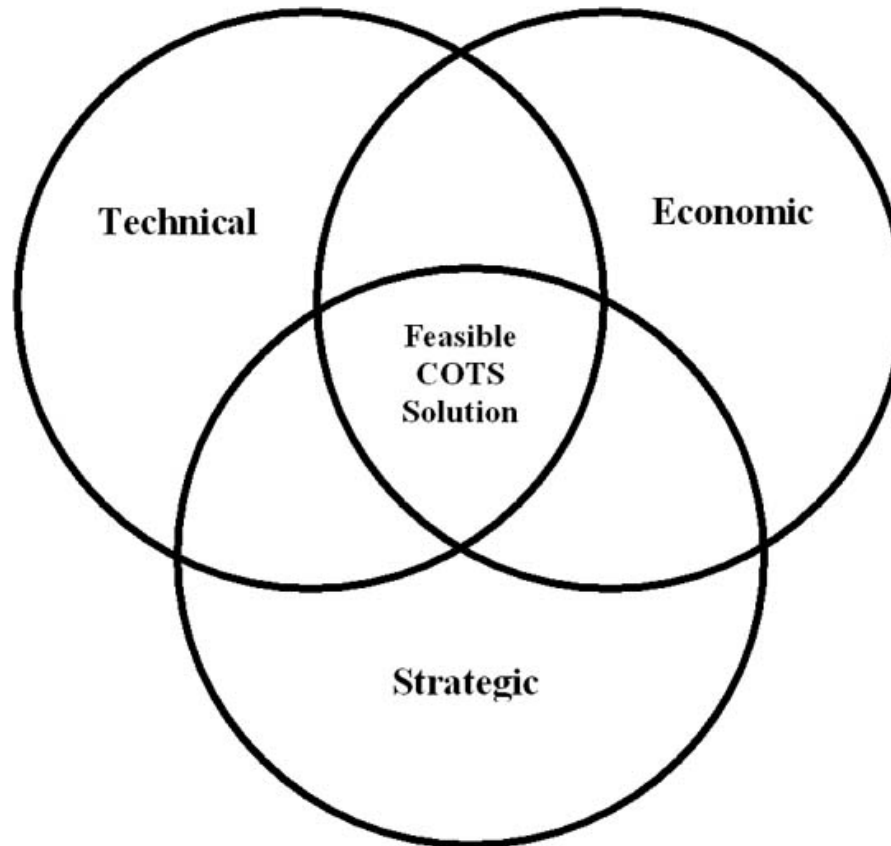
- Licensing and intellectual property procurement delays
- Up front license fees
- Recurring maintenance fees
- Reliability often unknown/ inadequate; scale often difficult to change
- Unnecessary features compromise usability, performance
- Functionality, efficiency constraints
- No control over upgrades/maintenance
- Dependency on vendor
- Efficiency sacrifices
- Integration not always trivial; incompatibilities among vendors
- Synchronizing multiple-vendor upgrades



# Caveat to Using COTS Products

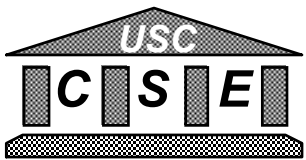
- **Two main characteristics of COTS:**
  - source code not available to developer
  - evolution not under control of developer
- **Results in trade-off:**
  - development time can be reduced, but often at cost of increased s/w component integration work
- **Unique risks associated with COTS:**
  - cost of licensing and redistribution rights, royalties, effort needed to understand the COTS software, pre-integration assessment and evaluation, post-integration certification of compliance with mission critical or safety critical requirements, indemnification against faults or damage caused by vendor supplied components, and costs incurred due to incompatibilities with other needed software and/or hardware

# When are COTS Products the “Right” Solution?



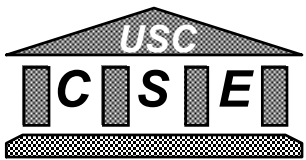
# When are COTS Products the “Right” Solution?

- **When they lie at the intersection of the three determinants of feasibility, *and do so demonstrably better than could original code:***
  - technical, economic, and strategic constraints
- **Technical**
  - ability to supply the desired functionality at the required level of reliability
- **Economic**
  - ability to be incorporated and maintained in the new system within the available budget and schedule
- **Strategic**
  - ability to meet needs of the system operating environment--including technical, political, and legal considerations--now, and as environment is expected to evolve in the future



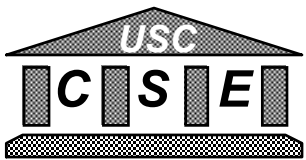
# **COTS Phenomena, Pitfalls and Practices**

- **You have no control over a COTS product's functionality or performance**
- **Most COTS products are not designed to interoperate with each other**
- **You have no control over a COTS product's evolution**
- **COTS vendor behavior varies widely**



## ***1. You have no control over a COTS product's functionality or performance.***

<b>Pitfalls to Avoid</b>	<b>Recommended Practices to Adopt</b>
<ul style="list-style-type: none"><li>• Using the waterfall model on a COTS integration project.</li><li>• Using evolutionary development with the assumption that every undesired feature can be changed to fit your needs.</li><li>• Believing that advertised COTS capabilities are real.</li></ul>	<ul style="list-style-type: none"><li>• Use risk management and risk-driven spiral-type process models.</li><li>• Perform the equivalent of a “receiving inspection” upon initial COTS receipt.</li><li>• Keep requirements negotiable until the system’s architecture and COTS choices stabilize.</li><li>• Involve all key stakeholders in critical COTS decisions.</li></ul>



## ***2. Most COTS products are not assigned to interoperate with each other.***

<b>Pitfalls to Avoid</b>	<b>Recommended Practices to Adopt</b>
<ul style="list-style-type: none"><li>• Premature commitment to incompatible combinations of COTS products.</li><li>• Trying to integrate too many incompatible COTS products.</li><li>• Deferring COTS integration till the end of the development cycle.</li><li>• Committing to a tightly coupled subset of COTS products with closed, proprietary interfaces.</li></ul>	<ul style="list-style-type: none"><li>• Use the Life Cycle Architecture milestone as a process anchor point.</li><li>• Use the Architecture Review Board (ARB) best commercial practice at the Life Cycle Architecture milestone.</li><li>• Go for open architectures and COTS substitutability.</li></ul>

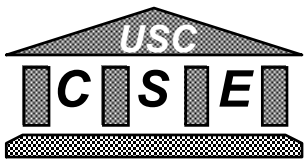
### ***3. You have no control over a COTS product's evolution.***

#### **Pitfalls to Avoid**

- “Snapshot” requirements specs and corresponding point-solution architectures.
- Understaffing for software maintenance,
- Tightly coupled, independently evolving COTS products.
- Assuming that uncontrollable COTS evolution is just a maintenance problem.

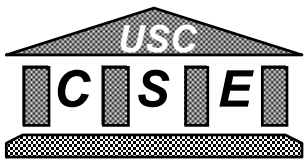
#### **Recommended Practices to Adopt**

- Stick with dominant commercial standards.
- Use likely future system and product line needs as well as current needs as COTS selection criteria.
- Use flexible architectures facilitating adaptation to change.
- Carefully evaluate COTS vendors’ track records with respect to predictability of product evolution.
- Establish a pro-active system release strategy, synchronizing COTS upgrades with system releases.



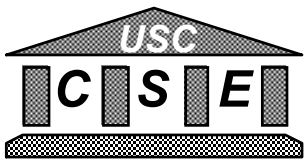
#### ***4. COTS vendor behavior varies widely.***

<b>Pitfalls to Avoid</b>	<b>Recommended Practices to Adopt</b>
<ul style="list-style-type: none"><li>• Uncritically accepting COTS vendors' statements about product capabilities and support.</li><li>• Lack of fallbacks or contingency plans.</li><li>• Assuming that an initial vendor support honeymoon will last forever.</li></ul>	<ul style="list-style-type: none"><li>• Perform extensive evaluation and reference checking of a COTS vendor's advertised capabilities and support track record.</li><li>• Establish strategic partnerships or other incentives for COTS vendors to provide support.</li><li>• Negotiate and document critical vendor support agreements.</li></ul>

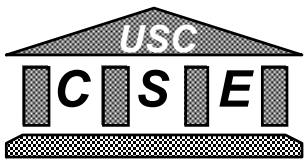


# **General Lessons Learned When Using COTS Products**

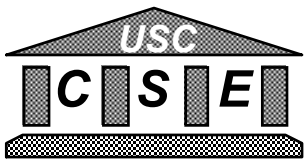
- **Problems with vendors (6)**
- **Need for flexibility in defining requirements (5)**
- **Importance of operational demos (5)**
- **Assessment of specific attributes (5)**
- **Life-cycle issues (5)**
- **COTS integrator experience (3)**
- **Product maturity (3)**
- **Training on COTS packages (2)**
- **Need for technology watch to keep up with vendors (2)**
- **Impacts of volatility during development (1)**



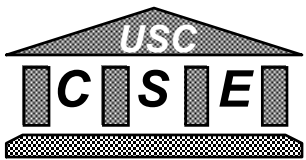
- **Vendors promise and don't deliver**
- **Products don't work as advertised**
  - one contractor is currently in litigation with an OS vendor
- **Don't assume a quantity discount**
  - Negotiate entire price up front



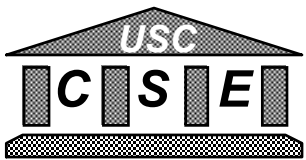
- **Distinguish between essential and negotiable requirements. Be flexible where you can.**
- **What we did right - spent 14 out of a total of 22 months iterating between requirements, business processes and the marketplace**
- **If you can bend your requirements, COTS is cheaper. Otherwise you're better off with custom developed.**



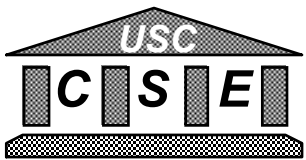
- **We were successful because we designed requirements around the capabilities of the system. It's much easier to find COTS products that fit together than it is to find product that fit specific requirements**
- **Not all projects have flexibility**
  - **ATC system: Controllers wanted new system to do exactly what old one did. Began with system that looked exactly like old one. Gradually evolved additional functional with controller input into new screens.**



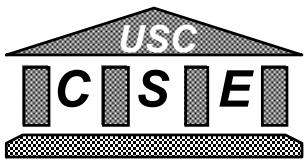
- **We spent a lot of time in detailed performance demonstrations with real users. This is something we did right.**
- **Up-front time is critical. That's when you have leverage with vendors. Once you buy their product, they are a lot less willing to help out.**
- **We should have done operational demonstrations.**



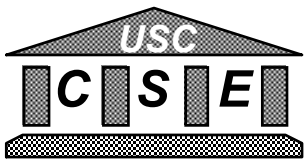
- **A number of projects expressed regret that they did not spend more time assessing:**
  - **portability (2)**
  - **inter-component compatibility (1)**
  - **flexibility (of user interface) (1)**
  - **installation ease (1)**



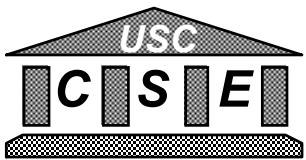
- **Volatility of COTS products was not a problem during development.**
  - Only one project pointed to volatility as a problem at all
- **Life cycle concerns were commonly expressed**



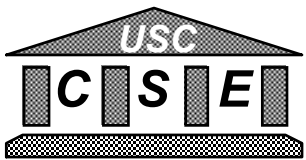
- **Supportability of COTS viewed as a major issue for safety-critical systems**
- **Out of service is a critical problem**
  - contractor purchased source code and will maintain COTS software
- **Operational disruption and expense to upgrade**
  - system configuration is frozen
  - complete replacement in ten years
- **On-line software maintenance**
  - How do you upgrade systems once they are in place and operating?



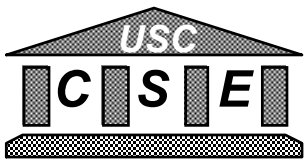
- **What is an effective strategy for upgrading? Products reach end of life in two years.**
  - Freeze and redo the system in 10 years?
  - Incorporate all versions from all vendors whenever they come out?
  - Refresh every 2 years?
  - Refresh a selected set of components every 2 years?
- **Should have an environment set up so you can load new versions onto the existing configuration and decide whether or not to upgrade.**



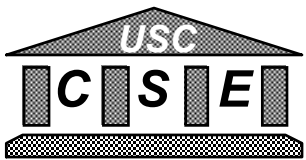
- **Several projects expressed the view that COTS saved money during development but shifted costs to operational side of the life cycle**
- **People have to look at the entire life cycle realistically - not just development**
- **We need to know how to estimate costs for the entire lifecycle**



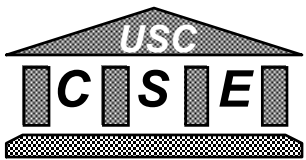
- **Important that they have experience integrating COTS**
- **Look carefully at their credentials. They will oversell themselves**
- **Our integrator used this contract to get smart rather than manage vendors**



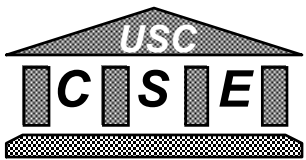
- **Never use an untried OS**
- **Maturity of the software was very important in COTS selection**
- **If you have a safety-critical system, you don't want state-of-the-art COTS**



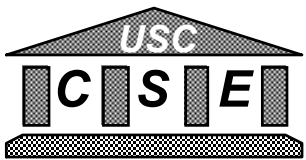
- **Need for technology watch to keep up with vendors**
  - these two projects had 120 to 150 products
  - SERC function?
  - One contractor had a vendor go out of business. They were caught by surprise.



- **We would have planned 6 months of training for those doing the tailoring**
- **We spent a month just becoming familiar enough with the COTS packages to use them**



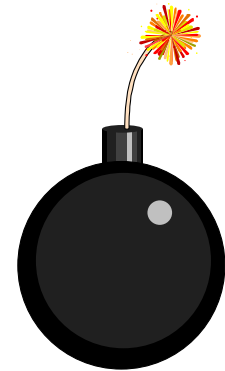
- **With COTS, you are buying into a different process**
  - operational demos are critical
  - requirements must be flexible
  - costs shift to the right
- **COTS is high risk because we are dependent on someone else. There needs to be a process to help people evaluate their risks.**

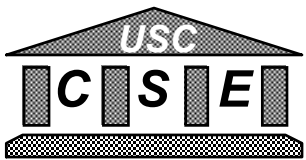


- **Volatility during development was not viewed as a problem.**
- **Only mentioned by one project**
  - **We had to redo the tailoring with new releases (e.g., new scripts to accommodate new features)**



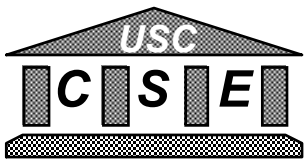
If you assume,  
your COTS are  
doomed!





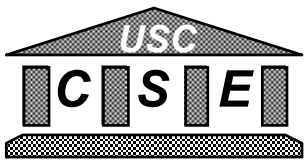
# Selection of COTS Software Components

- **Functional Requirements**
  - capability offered
- **Performance Requirements**
  - timing & sizing constraints
- **Nonfunctional Requirements**
  - cost/training/installation/maintenance/reliability



# **COTS Software Integration Lifecycle**

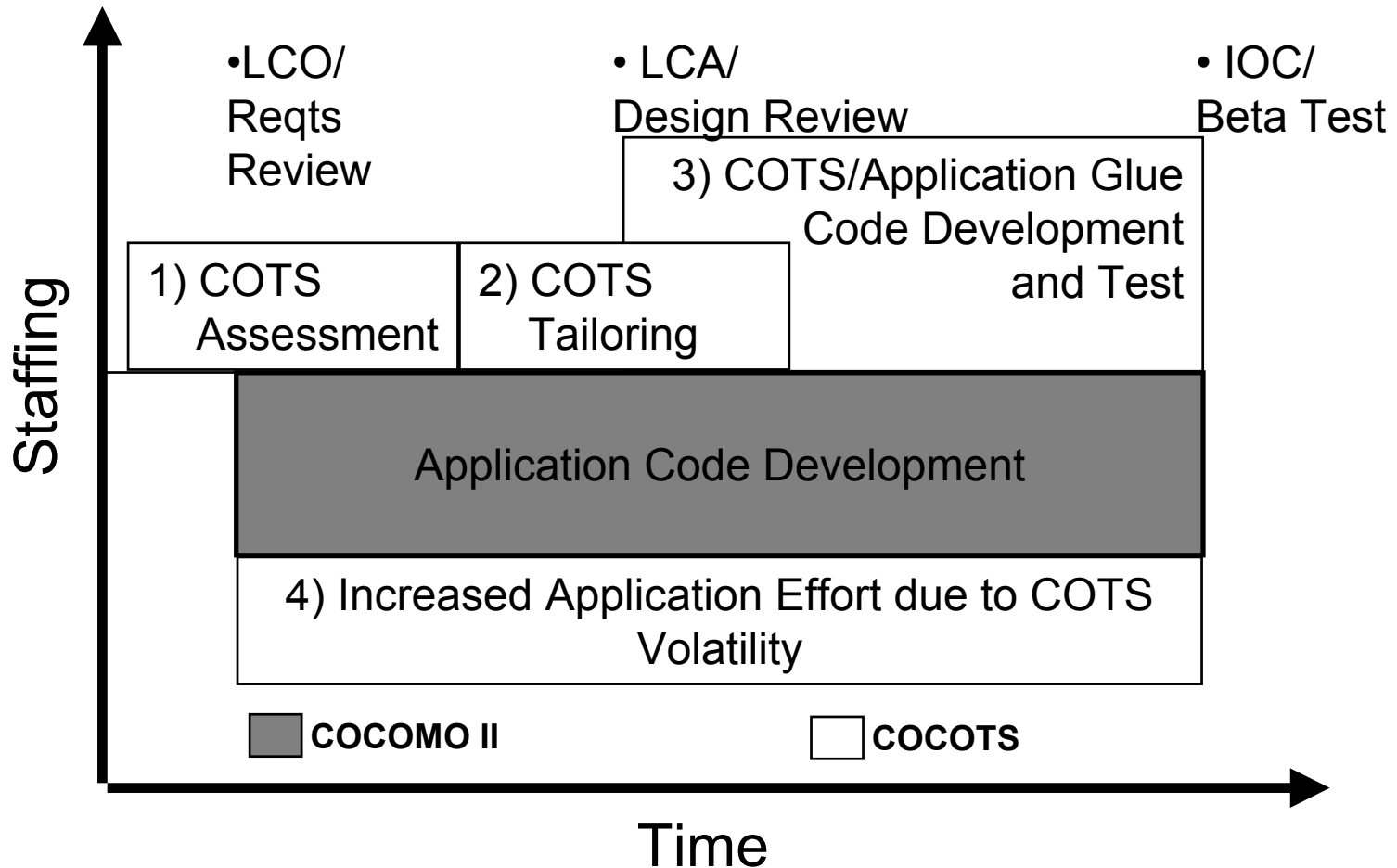
- 1) Qualify COTS product**
- 2) Perform system requirements**
- 3) Administer COTS software acquisition**
- 4) Prototype the system including COTS software**
- 5) Fully integrate COTS s/w and interface code**
- 6) Test completed prototype**

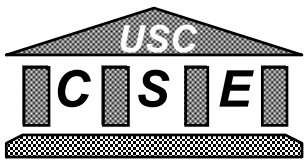


# **COTS Integration Sources of Effort**

- **COTS Assessment (pre- and post- commitment)**
  - Of functionality, performance, interoperability, etc.
- **COTS Tailoring and Tuning**
  - Effects of platform, other COTS products
- **Glue Code Development**
  - Similar to other COCOMO II estimation
- **Application Volatility Due to COTS**
  - COTS volatility, shortfalls, learning curve
- **Added Application V&V Effort**
  - COTS option and stress testing
  - Debugging complications, incorrect fixes

# Traditional vs. COTS Cost Sources

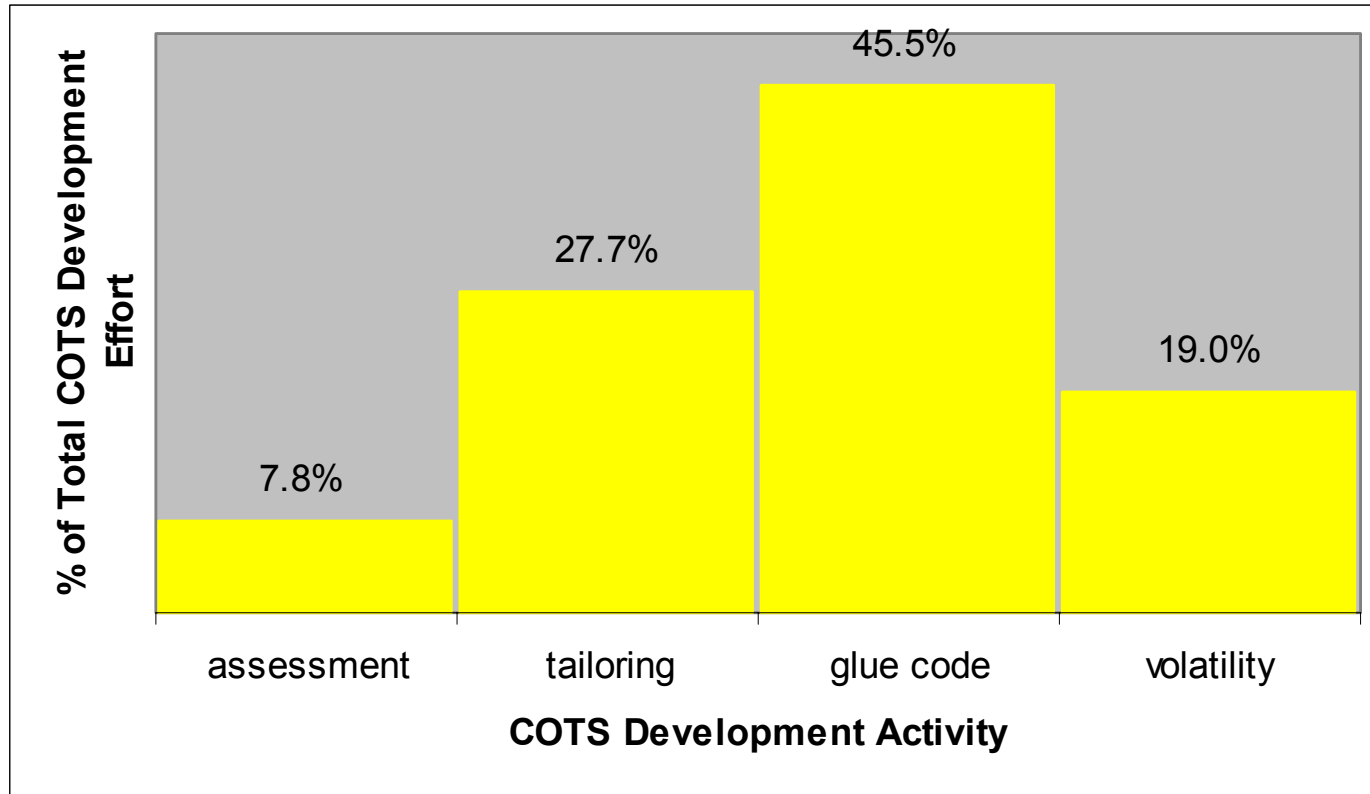




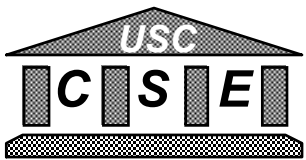
# Current Scope of COCOTS

- **COCOTS covers**
  - assessment
  - tailoring
  - glue code development and integration
  - impact of new releases (volatility)
- **COCOTS does not cover**
  - cost of re-engineering business processes
  - vendor management
  - licenses
  - training (for COTS integrators or end users)
  - COTS platform or tool experience or maturity
    - Covered by COCOMO II PLEX, LTEX, PVOL, TOOL factors

# COTS Activity Percentages



- This data offers for the first time the beginnings of an *empirically*-based, initial COTS development effort distributed by activity



# Inputs Needed to Estimate Assessment Effort

- **Initial Filtering of COTS products**
  - estimate of the total number of candidate COTS components to be filtered
- **More detailed assessment of specific candidates against attributes that are important**
  - class(es) of COTS components to be assessed
  - for each class,
    - number assessed
    - attributes considered

# COTS Integration Cost Sources: Assessment - Assessment Attributes

<b>Correctness</b>		<b>Understandability</b>		<b>Portability</b>
Accuracy		Documentation quality		Portability
Correctness		Simplicity		
		Testability		<b>Functionality</b>
<b>Availability/Robustness</b>				Functionality
Availability		<b>Ease of use</b>		
Fail safe		Usability/Human Factors		<b>Price</b>
Fail soft				Initial purchase/lease
Fault tolerance		<b>Version Compatibility</b>		Recurring costs
Input error tolerance		Downward compatibility		
Redundancy		Upward compatibility		<b>Maturity</b>
Reliability				Product Maturity
Robustness		<b>Inter-component Compatibility</b>		Vendor Maturity
Safety		Compatibility with other components		
		Interoperability		<b>Vendor Support</b>
<b>Security</b>				Response time for critical problems
Security (Access related)		<b>Flexibility</b>		Support
Security (sabotage related)		Extendability		Warranty
		Flexibility		
<b>Product Performance</b>				<b>User Training</b>
Execution performance		<b>Installation/Upgrade Ease</b>		User training
Information/data capacity		Installation Ease		
Precision		Upgrade/Refresh ease		<b>Vendor Concessions</b>
Memory performance				Willingness to escrow source code
Response time				Willingness to make modifications
Throughput				

# Inputs Needed to Estimate Tailoring Effort

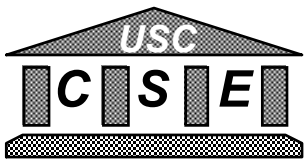
- **COTS tailoring - activities required to prepare or initialize a component for use in a specific system**
- **Tailoring includes**
  - parameter specification
  - script writing
  - GUI screen specification
  - Report specification
  - Security/Access Protocol initialization and set up
- **For each class of COTS component,**
  - rate the complexity of tailoring for each of the above activities



# Tailoring Complexity Table

Tailoring Activities & Aids	Individual Activity & Aid Complexity Ratings					Corresponding Points
	Very Low (point value = 1)	Low (point value = 2)	Nominal (point value = 3)	High (point value = 4)	Very High (point value = 5)	
<b>Parameter Specification</b>	Zero to 50 parms to be initialized.	51 to 100 parms to be initialized.	101 to 500 parms to be initialized.	501 to 1000 parms to be initialized.	1001 or more parms to be initialized.	-----  -----  -----  -----  -----
<b>Script Writing</b>	Menu driven; 1 to 5 line scripts; 1 to 5 scripts needed.	Menu driven; 6 to 10 line scripts; 6 to 15 scripts needed.	Hand written; 11 to 25 line scripts; 16 to 30 scripts needed.	Hand written; 26 to 50 line scripts; 31 to 50 scripts needed.	Hand written; 51 or more line scripts; 51 or more scripts needed.	
<b>I/O Report &amp; GUI Screen Specification &amp; Layout</b>	Automated or standard templates used; 1 to 5 reports/screens needed.	Automated or standard templates used; 6 to 15 reports/screens needed.	Automated or standard templates used; 16 to 25 reports/screens needed.	Hand written or custom designed; 26 to 50 reports/screens needed.	Hand written or custom designed; 51 or more reports/screens needed.	
<b>Security/Access Protocol Initialization &amp; Set-up</b>	1 security level; 1 to 20 user profiles; 1 input screen/user.	2 security levels 21 to 50 user profiles; 2 input screens/user.	3 security levels 51 to 75 user profiles; 3 input screens/user.	4 security levels 76 to 100 user profiles; 4 input screens/user.	5 or more security levels 101 or more user profiles; 5 or more input screens/user.	
<b>Availability of COTS Tailoring Tools</b>	No tools available.	N/A	N/A	N/A	Tools are available.	
<b>Very Low</b>	<b>Low</b>	<b>Nominal</b>	<b>High</b>	<b>Very High</b>		

Very Low	Low	Nominal	High	Very High
Point Total < 10	11 < Point Total < 15	16 < Point Total < 20	21 < Point Total < 25	26 < Point Total < 30

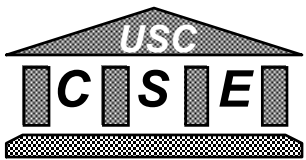


# Inputs Needed to Estimate Glue Code Effort

- **Definition of glue code:**
  - code needed to facilitate data or information exchange between the COTS component and the system into which it is being integrated
  - code needed to provide required functionality missing in the COTS component AND which depends on or must interact with the COTS component
- **Estimate of the total delivered lines of glue code**
- **Estimate of glue code rework due to COTS volatility or requirements volatility**

# Inputs Needed to Estimate Glue Code Effort (continued)

- **Integration Personnel**
  - Integrator experience with product (VL - VH)
  - Integrator personnel capability (VL - VH)
  - Integrator experience with COTS integration process (L - VH)
  - Integrator personnel continuity (VL - VH)
- **COTS Component**
  - COTS product maturity (VL - VH)
  - COTS supplier product extension willingness (L - VH)
  - COTS product interface complexity (L - VH)
  - COTS supplier product support (L - VH)
  - COTS supplier provided training and documentation (VL - VH)



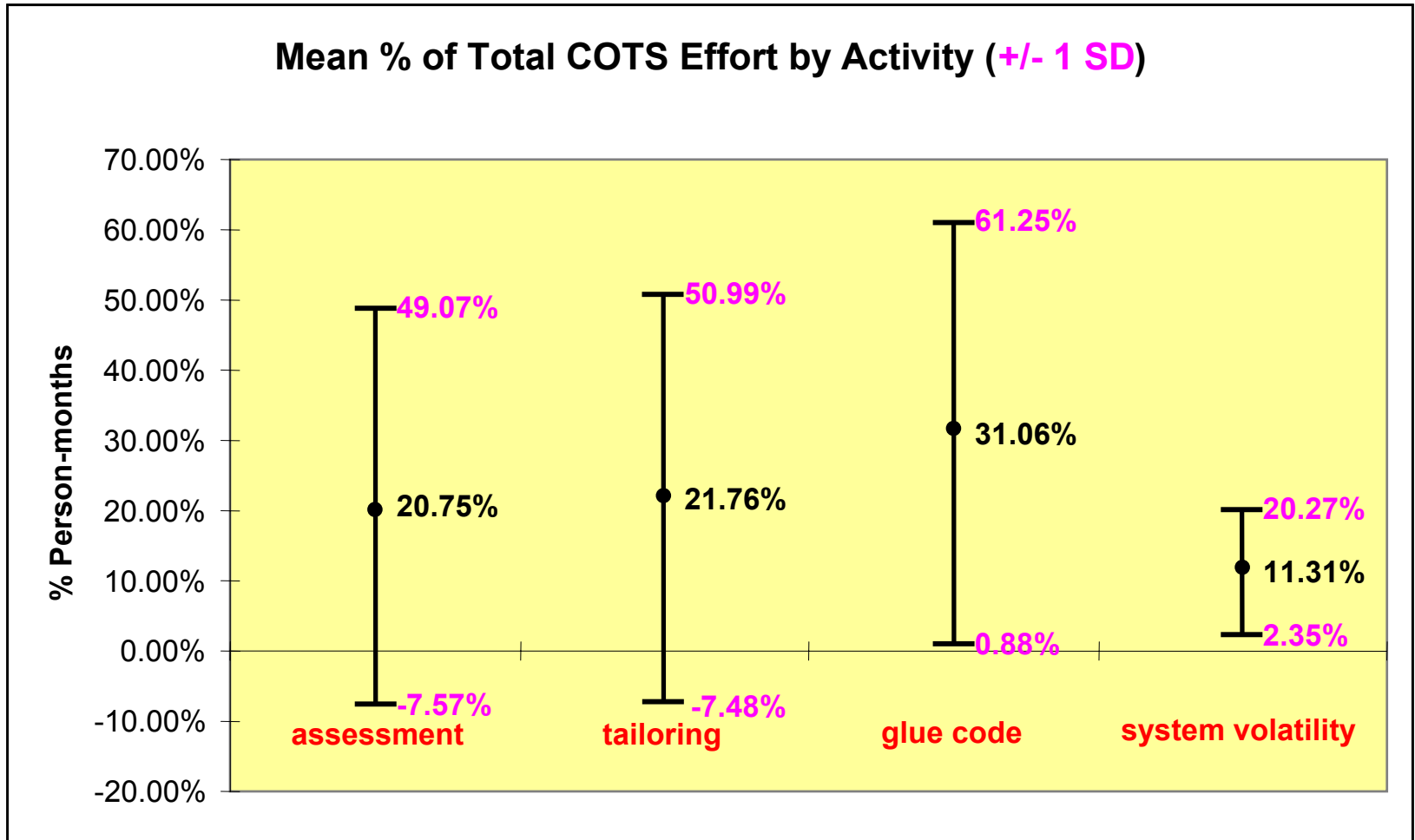
# Inputs Needed to Estimate Glue Code Effort (continued)

- **Application/System**
  - **Constraints on system/subsystem reliability (L - VH)**
  - **Constraints on system/subsystem technical performance (N-VH)**
  - **System portability (N - VH)**
  - **Application architectural engineering (VL - VH)**

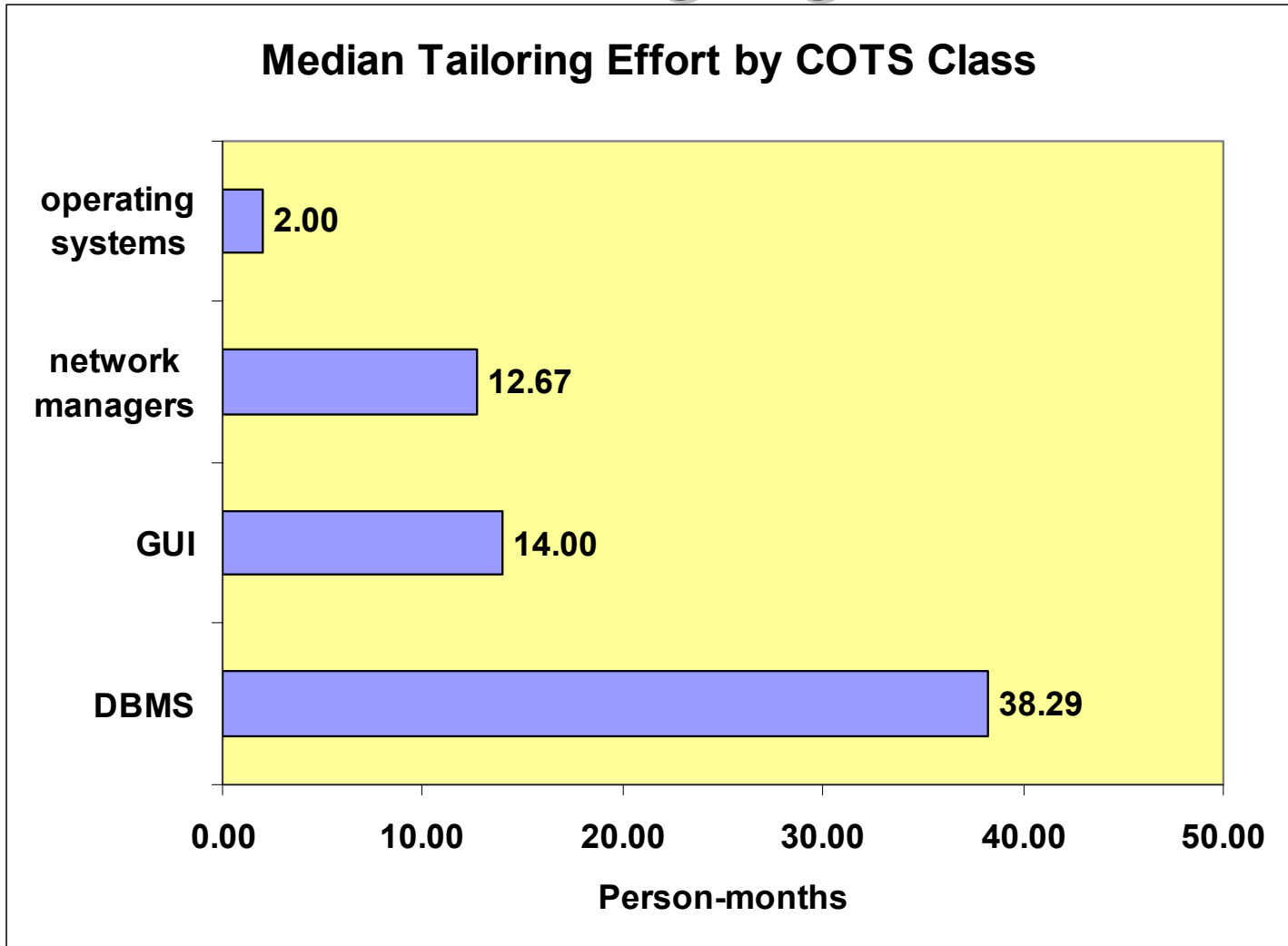
# Inputs Needed to Estimate Volatility Impacts

- **Captures impact of new COTS releases on the custom/new application effort**
- **Inputs:**
  - **Estimate of new development effort (derived via COCOMO II or other cost models)**
  - **Percentage of new development rework due to**
    - **requirements changes**
    - **COTS volatility**
- **Note: This submodel is being revised**

# COCOTS Effort Distribution: 20 Projects



# Data Highlights



# New Glue Code Submodel Results

- **Current calibration looking reasonably good**
  - **Excluding projects with very large, very small amounts of glue code (Effort Pred):**
    - **[0.5 - 100 KLOC]: Pred (.30) = 9/17 = 53%**
    - **[2 - 100 KLOC]: Pred (.30) = 8/13 = 62%**
  - **For comparison, calibration results shown at ARR 2000:**
    - **[0.1 - 390 KLOC]: Pred (.30) = 4/13 = 31%**
- **Propose to revisit large, small, anomalous projects**
  - **A few follow-up questions on categories of code & effort**
    - **Glue code vs. application code**
    - **Glue code effort vs. other sources**

## Assessment Submodel

$$\text{Initial Filtering Effort (IFE)} = \sum_{\text{all classes}} \left( \# \text{ COTS Candidates in class filtered} \right) \left( \text{Average Filtering Effort for product class} \right)$$

$$\text{Detailed Assessment Effort (DAE)} = \sum_{\substack{\text{all classes,} \\ \text{by project} \\ \text{domain}}} \left( \# \text{ COTS Candidates in class detailed assessed} \right) \left( \text{Average Assessment Effort for product class}^* \right)$$

*\* Qualified by assessment attributes most associated with that class*

$$\text{Final Project Assessment Effort (FPAE)} = \text{IFE} + \text{DAE}$$

# Assessment Attributes

<b>Correctness</b>		<b>Understandability</b>		<b>Portability</b>
Accuracy		Documentation quality		Portability
Correctness		Simplicity		
		Testability		<b>Functionality</b>
<b>Availability/Robustness</b>				Functionality
Availability		<b>Ease of use</b>		
Fail safe		Usability/Human Factors		<b>Price</b>
Fail soft				Initial purchase/lease
Fault tolerance		<b>Version Compatibility</b>		Recurring costs
Input error tolerance		Downward compatibility		
Redundancy		Upward compatibility		<b>Maturity</b>
Reliability				Product Maturity
Robustness		<b>Inter-component Compatibility</b>		Vendor Maturity
Safety		Compatibility with other components		
		Interoperability		<b>Vendor Support</b>
<b>Security</b>				Response time for critical problems
Security (Access related)		<b>Flexibility</b>		Support
Security (sabotage related)		Extendability		Warranty
		Flexibility		
<b>Product Performance</b>				<b>User Training</b>
Execution performance		<b>Installation/Upgrade Ease</b>		User training
Information/data capacity		Installation Ease		
Precision		Upgrade/Refresh ease		<b>Vendor Concessions</b>
Memory performance				Willingness to escrow source code
Response time				Willingness to make modifications
Throughput				

# Tailoring Submodel

**Project Tailoring Effort (PTE) =**

$$\sum \left[ \left( \# \text{ COTS Tailored in class} \right) \left( \text{Average Tailoring Effort for product class} \right) \cdot \text{TCQ}_{r, \text{ class}} \right]$$

**Over  
all classes,  
by project  
domain**

**where  $\text{TCQ}_{r, \text{ class}}$  = Tailoring Complexity Qualifier, calibrated within a class,  
for each of five possible ratings from Very Low to Very High,  
and with the  $\text{TCQ}_{\text{NOMINAL}} = 1.0$**



# Tailoring Complexity Table

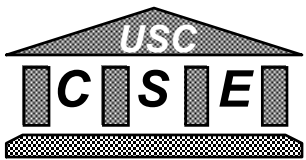
Tailoring Activities & Aids	Individual Activity & Aid Complexity Ratings					Corresponding Points
	Very Low (point value = 1)	Low (point value = 2)	Nominal (point value = 3)	High (point value = 4)	Very High (point value = 5)	
<b>Parameter Specification</b>	Zero to 50 parms to be initialized.	51 to 100 parms to be initialized.	101 to 500 parms to be initialized.	501 to 1000 parms to be initialized.	1001 or more parms to be initialized.	-----  -----  -----  -----  -----
<b>Script Writing</b>	Menu driven; 1 to 5 line scripts; 1 to 5 scripts needed.	Menu driven; 6 to 10 line scripts; 6 to 15 scripts needed.	Hand written; 11 to 25 line scripts; 16 to 30 scripts needed.	Hand written; 26 to 50 line scripts; 31 to 50 scripts needed.	Hand written; 51 or more line scripts; 51 or more scripts needed.	
<b>I/O Report &amp; GUI Screen Specification &amp; Layout</b>	Automated or standard templates used; 1 to 5 reports/screens needed.	Automated or standard templates used; 6 to 15 reports/screens needed.	Automated or standard templates used; 16 to 25 reports/screens needed.	Hand written or custom designed; 26 to 50 reports/screens needed.	Hand written or custom designed; 51 or more reports/screens needed.	
<b>Security/Access Protocol Initialization &amp; Set-up</b>	1 security level; 1 to 20 user profiles; 1 input screen/user.	2 security levels 21 to 50 user profiles; 2 input screens/user.	3 security levels 51 to 75 user profiles; 3 input screens/user.	4 security levels 76 to 100 user profiles; 4 input screens/user.	5 or more security levels 101 or more user profiles; 5 or more input screens/user.	
<b>Availability of COTS Tailoring Tools</b>	No tools available.	N/A	N/A	N/A	Tools are available.	

Very Low	Low	Nominal	High	Total Point Score	Very High
Point Total < 10	11 < Point Total < 15	16 < Point Total < 20	21 < Point Total < 25		26 < Point Total < 30

## Glue Code Submodel

$$\text{Total Effort} = A \cdot [(\text{size})(1 + \text{breakage})]^B \cdot \Pi (\text{effort multipliers})$$

- **A** - a linear scaling constant
- **Size** - of the glue code in SLOC or FP
- **Breakage** - of the glue code due to change in requirements and/or COTS volatility
- **Effort Multipliers** - **13** parameters, each with settings ranging **VL** to **VH**
- **B** - an architectural scale factor with settings **VL** to **VH**



# Glue Code Cost Drivers

## Personnel Drivers

- 1) ACIEP - COTS Integrator Experience with Product
- 2) ACIPC - COTS Integrator Personnel Capability
- 3) AXCIP - Integrator Experience with COTS Integration Processes
- 4) APCON - Integrator Personnel Continuity

## COTS Component Drivers

- 5) ACPMT - COTS Product Maturity
- 6) ACSEW - COTS Supplier Product Extension Willingness
- 7) APCPX - COTS Product Interface Complexity
- 8) ACPPS - COTS Supplier Product Support
- 9) ACPTD - COTS Supplier Provided Training and Documentation

## Application/System Drivers

- 10) ACREL - Constraints on Application System/Subsystem Reliability
- 11) AACPX - Application Interface Complexity
- 12) ACPER - Constraints on COTS Technical Performance
- 13) ASPRT - Application System Portability

## Nonlinear Scale Factor

- 1) AAREN - Application Architectural Engineering

# Volatility Submodel

## Approximate Model:

$$\text{Total Effort} = (\text{Application Effort}) \cdot \left[ \frac{\text{BRAK COTS}}{100} \right] \cdot (\text{EAF})_{\text{COTS}}$$

## Detailed Model with COCOMO II Parameters:

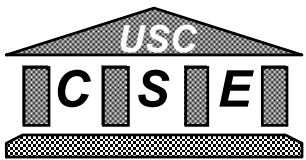
$$\text{Total Effort} = (\text{Application Effort}) \cdot \left[ \left( 1 + \frac{\text{BRAK COTS}}{1 + \text{BRAK}} \right)^{1.01 + \Sigma} - 1 \right] \cdot (\text{EAF})_{\text{COTS}}$$

**BRAK COTS:** % application code breakage due to COTS volatility

**BRAK** : % application code breakage otherwise

**$\Sigma$**  : COCOMO II scale factor

**EAF** : Effort Adjustment Factor (product of effort multipliers)



# Total COTS Integration Cost Estimate

**Total Integration Effort (in Person-Months) =  
Assessment Effort + Tailoring Effort + Glue Code Effort + Volatility Effort**

*where*

**Assessment Effort = Filtering Effort + Final Selection Effort**

**Total integration Cost =  
(Total Integration Effort) • (\$\$/Person-Month)**



# COCOTS' Contribution

- **COCOTS is completely open. Regardless of whatever estimates it provides, the descriptions of the elements that have gone into the model help highlight the most important factors that should be of concern to managers and developers of software systems using COTS software components.**
- **It's the very essence of a "constructive" cost model:**
  - one that helps an estimator better *understand* the complexities of a given software job to be done
  - by being open permits the estimator to know exactly *why* a model gives the estimate it does