COTS: The Future is Here

COTS-Based Systems

A Winsor Brown & Jesal Bhuta ({AWBrown, Jesal}@usc.edu)
USC Center for Systems and Software Engineering
http://csse.usc.edu/
Presentation for: CSCI 510 Fall 2006
Outline

Introduction to COTS, CBS and CBA

- COTS Trends in Industry
- Using COTS
- Building and Estimating COTS-Based Applications
- CBA/COTS Risks and Lessons Learned
- Other COTS Cost Estimation Techniques
What is COTS

- **COTS definition by SEI**
  - A product that is
    - Sold, leased or licensed to the general public
    - Offered by a vendor trying to profit from it
    - Supported and evolved by the vendor, who retains the intellectual property rights
    - Available in multiple copies
    - Used without internal modification by a consumer
Related Terms

- COTS (Commercial off the shelf)
  - Black box (internal modification not possible)
  - White box (internal modification possible)
- GOTS (Government Off The shelf)
- ROTS (Research Off The Shelf)
- NDI (Non Developmental Item/Not Developed In-house)
- Reuse Code
  - Source code originally written for some other project
COTS Systems Definitions

- **COTS Based Systems (CBS)**
  - Any system that uses COTS

- **COTS Based Applications (CBA)**

- A system for which
  - at least 30% of the end-user functionality is provided by COTS products and
  - at least 10% of the development effort is devoted to COTS considerations
COTS Implications

- Source code may or may not be available

- No longer a COTS if the source code is modified internally
  - No vendor support for modified COTS

- Can be tailored or extended using
  - Tailoring options

- Can be extended using
  - An application programming interface (API)

- Usually periodic releases with feature growth

- Older versions eventually become obsolete
  - No vendor support (e.g. Windows 95, 98, 2000)
CBA – Major Activities

- **Assessment**
  - Activity of determining the feasibility of using specific COTS to fulfill required system functions

- **Tailoring**
  - Activity associated with setting or defining shell parameters or configuration options for a COTS, but which do not require modification of source code, including defining I/O report formats, screens, etc.

- **Glue-code**
  - Custom development needed to integrate COTS packages within an application external to the packages themselves
Outline

- Introduction to COTS, CBS and CBA

COTS Trends in Industry

- Using COTS

- Building and Estimating COTS-Based Applications

- CBA/COTS Risks and Lessons Learned

- Other COTS Cost Estimation Techniques
1980 - Zon.com - Architecture

- User-Interface
  - Web-browser
- Inventory Mgt
- Shopping Cart
- Credit Card Auth
- Application Server
  - Submit orders
  - Display inventory
  - ... HTTP service
  - ... Data storage
  - Date retrieval
  - Data Indexing
  - Data Backup
  - ...

Custom components (developed in-house)
Third-Party components (developed else-where)
1990 – Zon.com - Architecture

User-Interface
Web-browser

Inventory Mgt

Application Server

Shopping Cart

Credit Card Auth

Submit orders
Display inventory

HTTP service

Database System
(Sybase)

Custom components
(developed in-house)

Third-Party components
(developed else-where)

Data storage
Date retrieval
Data Indexing
Data Backup

Submit orders
Display inventory

…

HTTP service

…

Data storage
Date retrieval
Data Indexing
Data Backup

…
2000 – Zon.com - Architecture

Custom components (developed in-house)

Third-Party components (developed elsewhere)

User-Interface
Web-browser

Application Server

Inventory Mgt

Shopping Cart

Credit Card Auth

Submit orders
Display inventory
...

HTTP service
...

Database System
(Sybase)

Data storage
Date retrieval
Data Indexing
Data Backup
...

Center for Systems and Software Engineering
2004 – Zon.com - Architecture

- User-Interface
  - Web-browser
- Application Server
  - Submit orders
  - Display inventory
  - …
  - HTTP service
  - …
- Database System
  - (Sybase)
  - Data storage
  - Date retrieval
  - Data Indexing
  - Data Backup
  - …
- Custom components
  - (developed in-house)
- Third-Party components
  - (developed else-where)

- Inventory Mgt
- Shopping Cart
- Credit Card Auth

Center for Systems and Software Engineering
CBA Growth Trend

- USC e-services project data shows: increase in # of projects from 28% in 1997 to 70% in 2002
Implications for Software Engineers

- New skills required for system development
  - COTS assessment
  - COTS tailoring
  - COTS integration
  - COTS system feasibility analysis …
# Software Development Phase Activities: Custom Vs COTS

<table>
<thead>
<tr>
<th>Requirements (Inception)</th>
<th>Design (Elaboration)</th>
<th>Implementation (Construction)</th>
<th>Acceptance &amp; Deployment (Transition)</th>
<th>Sustainment (Maintenance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Development Activities</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| - Ops Concept  
- Planning  
- Software Reqs  
- Architecture |
| - Preliminary Design  
- Detailed Design |
| - Code & Unit Test  
- Component Test  
- System Test |
| - Acceptance Test  
- Site Installation  
- Site Activation |
| - Operations  
- Maintenance  
- Enhancements |

<table>
<thead>
<tr>
<th>COTS Specific Activities</th>
</tr>
</thead>
</table>
| - Definition of Objectives Constraints & Priorities  
- COTS Component Identification  
- COTS Assessment/Selection  
- Prototyping |
| - Tailoring  
- Glue code development  
- COTS Integration & Test  
- Component Refresh |
| - Acceptance Test  
- Site Installation  
- Site Activation |
| - Component Refresh  
- Maintenance |
Outline

- Introduction to COTS, CBS and CBA

COTS Trends in Industry

- Using COTS
- Building and Estimating COTS-Based Applications
- CBA/COTS Risks and Lessons Learned
- Other COTS Cost Estimation Techniques
Why use COTS?

- Change in software development practice over the past 20 years
  - Build system with pre-existing software to reduce development and maintenance costs
  - One such source: COTS

- COTS Based Systems
  - Involve less development time and lower development cost by taking advantage of existing, market proven, vendor supported products.
Using COTS - Trade Off’s

- 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 software component integration work
# COTS Pros and Cons

**Pros**
- Available now, earlier payback
- Avoids expensive development & maintenance
- Predictable license costs & performance
- Rich in functionality

**Cons**
- Licensing and procurement delays
- Up front license fees
- Recurring maintenance fees
- Reliability often unknown/inadequate
- Unnecessary features compromise usability, performance
COTS Pros and Cons

Pros
- Broadly used, mature technology
- Frequent upgrades often anticipate organization’s needs
- Dedicated support organization
- Hardware/software independence
- Tracks technology trends

Cons
- Functionality, efficiency constraints
- No control over upgrades/maintenance
- Dependency on vendor
- Integration not always trivial; incompatibilities among different COTS
- Synchronizing multiple-vendor upgrades
When is COTS right for you

- When they lie at the intersection of the three determinants of feasibility, and do so demonstrably better than could original code:
  - Technical
  - Economic
  - Strategic constraints
When is COTS right for you

- **Technical constraint**
  - Ability supply the desired functionality at the required level of reliability

- **Economic constraint**
  - Ability to be incorporated and maintained in the new system within the available budget and schedule

- **Strategic constraint**
  - 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
Outline

- Introduction to COTS, CBS and CBA
- COTS Trends in Industry

Using COTS

- Building and Estimating COTS-Based Applications
- CBA/COTS Risks and Lessons Learned
- Other COTS Cost Estimation Techniques
Principles of CBA Models

1. Process happens where the effort happens

2. Don’t start with **Requirements**

3. Avoid premature commitments -- but have and use a plan

4. Buy information early to reduce risk and rework

5. Prepare for COTS changes
Outline

- Introduction to COTS, CBS and CBA
- COTS Trends in Industry
- Using COTS
- Building and Estimating COTS-Based Applications
- CBA/COTS Risks and Lessons Learned
- Other COTS Cost Estimation Techniques
Building and Estimating COTS-Based Applications

- Elements for Developing CBS/CBA
- COCOTS: Estimating the CBS/CBA Development costs
- MBASE/RUP
- The Problem: COTS capabilities “control” Requirements
- Developing CBS/CBA Mapped onto MBASE/RUP
Elements for Developing CBS/CBA

CBA Process Decision Framework
- Process Elements

COTS Market/Vendors
- COTS price models

COTS integration characteristics
- Cost exposure for each candidate

CBA Activity Sequence Patterns
- Sequences
- Indicated Risks
- Risk Mitigations

CBA Classifications
- 3 types of CBA
- Guidelines
- Top risk lists

Stakeholders’ WinWin Negotiation

Win conditions, system OC&P’s, Agreements, etc.

Information, updates, changes

COCOMO II, COCOTS, etc.
Value-Based CBA Process Framework

1. Start

P1: Identify Scenarios and Criteria

P2: Do Relevant COTS Products Exist?

No

Yes or Unsure

Yes

C

No acceptable COTS-Based Solution

A

P3: Assess COTS

Partial COTS solution identified

University of Southern California
Assessment Process Element

A1: Establish evaluation criteria, A2: Initial Filtering:
Tailoring Process Element

**Tailoring options:**
1. GUI Based
2. Parameter Based
3. Programmable

<table>
<thead>
<tr>
<th>Eval. Parameters</th>
<th>GUI Based</th>
<th>Parameter Based</th>
<th>Programmable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Details</td>
<td>Low - None</td>
<td>Low</td>
<td>Detailed</td>
</tr>
<tr>
<td>Complexity</td>
<td>Low - Moderate</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Adaptability</td>
<td>GUI - High; Other - Low</td>
<td>Low - Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Developer Resources</td>
<td>Low</td>
<td>Low - Moderate</td>
<td>Moderate - High</td>
</tr>
</tbody>
</table>
Glue Code Process Element

Architecture considerations:
- Determine interconnection topology options
- Minimize the complexity of interactions
- Select connectors w.r.t. COTS interfaces
- Identify potential architectural mismatches
COCOTS: COTS Modeling Problem Context

(COTS Components as Application Elements, Infrastructure, or Tools)

- COTS and Custom Applications Components
- New Modeling Problem
- COTS as Infrastructure (COCOMOII parms PVOL, PLEX)
- COTS as Tools (COCOMOII parms LTEX TOOL)

Cost Modeling Currently Addressed Within
COCOMOII: COTS as Infrastructure and Tools
COCOTS: COTS’ Development Cost Sources

1. COTS Assessment
2. COTS Tailoring
3. COTS Glue Code Development

Application Code Development, Integration, and Test Separate from COTS Effects

LCO – Lifecycle Objectives
LCA – Lifecycle Architecture
IOC – Initial Operational Capability

COCOTS Effort Estimate
COCOMO II Effort Estimate
COCOTS - Current Status – (1)

- Three Sub-models
  - Assessment sub-model
  - Tailoring sub-model
  - Glue code sub-model

- Mathematical form of each sub-model is different
  - However, a common feature is estimates based upon classes of COTS components being examined
  - Example COTS classes: GUI builders, operating systems, databases, word processors, etc.
COCOTS - Current Status – (2)

- Calibrated on 20 data points

- Project Domains
  - Air Traffic Management
  - Business (including databases)
  - Communication, Navigation, & Surveillance
  - Logistics
  - Mission Planning
  - Operations
  - Web-based Maps
MBASE/RUP: What’s wrong with this “picture”
Problem: COTS capabilities “control” Requirements

- CBS/CBA
  Needs/Wants/WinCs -> COTS or New or OSSw ->
  -> Architecture -> Requirements -> Design …

- Greenfield
  Needs/Wants/WinCs -> Requirements -> Architecture ->
  -> Design …
Developing CBS/CBA Mapped onto 577 ICM

- Start Of Fall Semester
- Semester Break
- Spring Semester Ends
- Summer Semester Ends

Time Line:
- Teams formed; Projects assigned
- Design and Code Review

- Inc. Cycles: Assess COTS Candidates
- Elab. Cycles: Develop Glue Code
- Const. Cycles
- Trans. Cycles: Enhancement Cycle?
- Tailor COTS

Center for Systems and Software Engineering
Outline

- Introduction to COTS, CBS and CBA
- COTS Trends in Industry
- Using COTS
- Building and Estimating COTS-Based Applications
- CBA/COTS Risks and Lessons Learned
- Other COTS Cost Estimation Techniques
## CBA Top N Risk List

<table>
<thead>
<tr>
<th>No.</th>
<th>Risk Items</th>
<th>No.</th>
<th>Risk Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirements Changes and Mismatches</td>
<td>9</td>
<td>Difficulty in coordinating meetings with key personnel may result in significant delays</td>
</tr>
<tr>
<td>2</td>
<td>Many new non technical activities are introduced in the Software Engineering Process</td>
<td>10</td>
<td>Inadequate vendor support may result in significant project delays</td>
</tr>
<tr>
<td>3</td>
<td>Miss possible COTS candidates within the COTS process</td>
<td>11</td>
<td>COTS package incompatibilities may result in feature loss and significant project delays (Integration Clash)</td>
</tr>
<tr>
<td>4</td>
<td>Too much time spent in assessment due to too many requirements and too many COTS candidates</td>
<td>12</td>
<td>Added complexity of unused COTS features</td>
</tr>
<tr>
<td>5</td>
<td>Might not include all key aspects for establishing evaluation criteria set. (Inadequate COTS assessment)</td>
<td>13</td>
<td>Overly optimistic expectations of COTS quality attributes</td>
</tr>
<tr>
<td>6</td>
<td>Introducing new COTS candidates is likely and requires re-planning</td>
<td>14</td>
<td>Overly optimistic COTS package learning curve</td>
</tr>
<tr>
<td>7</td>
<td>Faulty Vendor Claims may result in feature loss and/or significant delays</td>
<td>15</td>
<td>A version upgrade may result in re-tailoring of COTS package</td>
</tr>
<tr>
<td>8</td>
<td>Ability or willingness of the organization to accept the impact of COTS requirements</td>
<td>16</td>
<td>Imposed black box testing of COTS components</td>
</tr>
</tbody>
</table>

**Source:** USC e-service projects 2000-2002
Lessons Learned Using COTS I

- Problems with vendors
  - Vendors promise and don’t deliver
  - Products don’t work as advertised
  - Don’t assume a quantity discount, negotiate price upfront

- Need for flexibility in defining requirements
  - 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. (Not all projects may be flexible)
Lessons Learned Using COTS II

- Importance of operational demos
  - Spend a lot of time in detailed performance demonstrations with real users.
  - 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.

- Assessment of specific attributes
  - Projects (COCOTS), in the past have expressed regret that they did not spend more time assessing portability, inter-component compatibility, flexibility (of user interface), and installation ease.
Lessons Learned Using COTS III

- Life-cycle issues
  - 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
  - Projects, in past have expressed the view that COTS saved money during development but shifted costs to operational side of the life cycle
  - On-line software maintenance
    - How do you upgrade systems once they are in place and operating?
Lessons Learned Using COTS IV

- Life Cycle Issues (Upgrading)
  - 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.
  - Look at the entire life cycle realistically - not just development
Lessons Learned Using COTS V

- COTS integrator experience
  - Important that they have experience integrating COTS.
  - Look carefully at their credentials. They will oversell themselves

- Product maturity
  - 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
Lessons Learned Using COTS VI

- Training on COTS packages
  - Significant learning curve

- Need for technology and market watch to keep up with vendors and technologies

- Impacts of volatility during development
  - redo the tailoring with new releases
Outline

- Introduction to COTS, CBS and CBA
- COTS Trends in Industry
- Using COTS
- Building and Estimating COTS-Based Applications
- CBA/COTS Risks and Lessons Learned

Other COTS Cost Estimation Techniques
Other COTS Cost Estimation Techniques

- COCOTS (already discussed)
- Early COCOTS
- Price-S (Uses COCOMO & COCOTS)
- Seer-SEM (Galorath Corp)
- SLIM
Early COCOTS Motivation

- Information known early in the life-cycle is limited

- Require a rough order-of-magnitude estimates for basic investment decisions
  - Build Vs. Buy decision
Early COCOTS - Highlights

- Handle’s COTS, NDI, and new Code
- Cost drivers can be estimated or are known early-on
- Costs will be estimated at system level, not at the level of components
- Model addresses the total cost of ownership
  - COTS licensing
  - Effort to build/integrate
  - Business process re-engineering, training, consulting
  - Maintenance costs
Model Drivers – Size inputs

- Number of independent COTS products
- Number of COTS-provided user functions
- Degree of uncertainty about product choices
- Amount of newly developed software (equivalent SLOC)
Model Drivers – Cost Drivers

- Complexity of Integration
- Required tailoring, BPR, training, data conversion
- Integrator difficulties with COTS products and integration
- Degree of mismatch between COTS capabilities and user needs; maturity, requirements flexibility
- Requirements Volatility
- COTS Volatility
Questions
Additional Slides
CBA Project Types

- **Assessment Intensive CBA**
  - Assessment is the dominant activity
    - Projects 1-7 in USC e-services effort distribution chart

- **Tailoring Intensive CBA**
  - Tailoring is the dominant activity
    - Projects 8-10 in USC e-services effort distribution chart

- **Glue Code Intensive CBA**
  - Glue code is the dominant activity
    - Projects 14-17 in USC e-services effort distribution chart
CBA Activities Effort Distribution

- USC e-services project data
  - 5 person teams
  - 24 week projects

- COCOTS calibration data
  - Small to large business mgmt., analysis, and control applications
Effort Sources in CBA Project Types

Comparison of CBA Effort Sources

Activity

Team Interaction
COTS assessment
Client Interaction
Life Cycle Planning
Project Website
Training and Preparation
COTS Tailoring
Transition and Support
Glue Code
custom development

ACBA
TCBA
GCBA
Non-CBA
Principles of CBA Model

1. Process happens where the effort happens
2. Don’t start with Requirements
3. Avoid premature commitments -- but have and use a plan
4. Buy information early to reduce risk and rework
5. Prepare for COTS changes
1. Process Happens where Effort Happens

- Effort sources:
  - COTS Assessment, Tailoring, Glue Code/Integration
Activity Sequence Patterns

- COTS Activity Sequences

<table>
<thead>
<tr>
<th>No.</th>
<th>Inception</th>
<th>Elabration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>AC</td>
<td>ATG</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>AT</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>(TG)A</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>A(TG)</td>
<td>A(TG)</td>
<td>G</td>
</tr>
<tr>
<td>5</td>
<td>AT</td>
<td>AT</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>T</td>
<td>TG</td>
<td>G</td>
</tr>
<tr>
<td>7</td>
<td>AT</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>8</td>
<td>AT</td>
<td>(AA) TG</td>
<td>(TGC)</td>
<td>G</td>
</tr>
<tr>
<td>9</td>
<td>A</td>
<td>AT</td>
<td>TG</td>
<td>G</td>
</tr>
</tbody>
</table>

- Time-ordered sequence of A, T, G, and C
- Parentheses: parallel activities

Data Source: Weekly Progress Reports
2. Don’t Start With Requirements

- Definition of Requirement: Webster
  - Something required: claimed or asked for by right and authority

- Semantics ingrained into customer and user cultures

- Pitfalls: unmanageable expectations, severe COTS constraints

- Process examples: Waterfall variants

- Counterexample: large TRW-Government project

- Hazard: Many corporate, Government policies enforce Waterfall
Waterfall Variant: UMD CBA Process

Diagram showing the CBA Process:
- Vendor
  - Requirements:
    - Package Identification/Evaluation/Selection
    - Requirements Review
    - Requirements Analysis
  - Design:
    - Identify Glueware and Integration Requirements
    - Design Review
    - Non-COTS Design
  - Coding:
    - Write Glueware and Interfaces
    - Non-COTS Coding
  - Integration:
    - Integration and Test
    - Target System Installation and Acceptance Test

Customer:

Key:
- Information flow - bidirectional
- Process check or review
- Process phase
- Process activity
- External role

Center for Systems and Software Engineering
Waterfall Processes Over-constrain COTS Options

Arch. A: Custom many cache processors

Arch. B: Modified Client-Server

Original Spec

After Prototyping

Response Time (sec)

$100M

$50M
Hazard Avoidance: Waterfall Policies

- Interpret as “risk-driven waterfall”

- Defer Requirement Review until risks are resolved or covered by risk management plans

- Concurrently engineer requirements, architecture, plans, COTS

- Use anchor point milestone definitions
  - With Pass/Fail Feasibility Rationales
3. Avoid Premature Commitments- But have and use a plan

- Pitfalls: study-wait-study cycle; lack of progress metrics
- Process examples: convergent spheres of concern
- Counterexample: software environment S-W-S cycle
- Hazard: Avoiding premature commitments by avoiding commitments
- Hazard avoidance
- Use a tailorable planning framework
- Use goal-oriented adaptive control process to monitor and adjust plans
Converging Spheres: SEI EPIC Process
4. Buy Information Early to Reduce Risk and Rework

- Pitfall: COTS interoperability, scalability, culture match

- Hazard: pressure to be decisive, commit early

- Hazard avoidance: Evaluate COTS risk mitigation options; use CBA-specialized spiral process framework
5. Prepare for COTS Changes

- New releases every 10 months (GSAW 2000-03)
- Releases unsupported after 3 newer releases
- Releases likely to diverge
  - Vendor need product differentiation
- Change interactions scale as high as square of # COTS products
- Large outsourced applications may have unsupported COTS
  - Example: 3-year project; 120 COTS; 55 unsupported (46%)
Coping with COTS Changes

- Win-win relationships with COTS vendors
  - Pro-active market watch

- Reduce number of COTS products and vendors

- Reduce inter-COTS coupling
  - Wrappers, standards, mediators

- Develop, evolve COTS refresh strategy

- Contract for delivery of refreshed COTS