

# Not All CBS Are Created Equally: COTS-Intensive Project Types

Barry W. Boehm, Dan Port, Ye Yang, and Jesal Bhuta

Center for Software Engineering  
University of Southern California  
{boehm, dport, yey, jesal}@cse.usc.edu

**Abstract.** COTS products affect development strategies and tactics, but not all CBS development efforts are equal. Based on our experiences with 20 large government and industry CBS projects assessed during our development of the COCOTS estimation model, and our hands-on experience with 52 small e-services CBS projects within USC's graduate level software engineering course, we have identified four distinct CBS activity areas: assessment intensive, tailoring intensive, glue-code intensive, and non-COTS intensive. The CBS activity type fundamentally affects the COTS related activity effort and project risks. In this work we define the three COTS activity intensive CBS types and discuss their strategic comparisons based on an empirical study of the spectrum of large and small CBS projects.

## 1 Introduction

Much has been written regarding the various ways in which commercial off-the-shelf (COTS) products affect the way we develop systems. In particular, how we define requirements, evaluate products, manage risks, and relate the activities of product evaluation and system design (for example see [1] and [14]). We have seen a number of patterns among these, ranging across large government and industry projects to smaller e-service projects. In our experiences since 1996 in both areas, we have observed three main strategies for developing COTS based systems (CBS): find and use COTS products without modification to cover all desired capabilities, find and adapt COTS as needed, find and integrate multiple COTS as components in a custom built application. The decision as to which approach to use is driven primarily by the systems shared vision, economic constraints, and desired capabilities & priorities (DC&P's). In projects where a significant amount of COTS related development effort is expended relative to the overall development effort (which we will later refer to as COTS Based Applications or CBA), we have observed within our USC e-services CBS projects that these approaches will differ considerably in regards to the development focus, critical activities, and project risks.

Although there are some significant differences between our large industry CBS projects and our smaller USC e-services CBS projects, we have found that

they share many common factors such as client demand for sophisticated functionality, fixed time schedules, limited developer resources and skills, lack of client maintenance resources, and many others. As a result, we believe our USC e-services COTS experiences are particularly representative of CBS development in general [10]. In particular, we have observed that, the degree that our comprehensive software development guidelines (MBASE [24]) used by the developers of the e-services projects, have to be adapted is proportional to the intensity of effort within performing COTS related activities such as product assessment, tailoring, and glue-code development. We have also observed that there are considerable risks in these areas and strategic guidance on the management of these is critical to a successful project [13]. Analogous observations have been made for our larger, industry based CBS projects studied for calibration of the COCOTS [23] CBS cost estimation model.

While developing COCOTS, it was found that there is no “one size fits all” CBS effort estimation model. Instead, effort estimates are made by composing individual assessment, tailoring, and glue-code effort sub-models. Following this lead, we surmised that there are four major types of CBS projects: those that are assessment intensive, tailoring intensive, glue-code intensive, or non-COTS activity intensive. To elaborate this:

1. An *assessment intensive* project will have the DC&P’s covered without much modification (or tailoring) by a single COTS package or through the simple integration of several. The primary effort is focused on identifying a collection of candidate COTS products and evaluating (i.e. assessing) a feasible set of products whose existing capabilities directly address a desired operational concept.
2. A *tailoring intensive* project is where the DC&P’s are covered by a single (or very few) COTS package(s). The primary effort is on adapting a COTS framework whose existing general capabilities can be customized (i.e. tailored) in a feasible way to satisfy the majority of a systems capabilities or operational scenarios.
3. The *glue-code intensive* project will have non-trivial DC&P’s covered by multiple COTS packages and components, The primary effort is to identify COTS packages that can feasibly used and integrated as components to address subsets of the required system capabilities and integrate them to develop the system. Typically there is a significant amount of glue code design and development in order to integrate such packages.
4. A *non-COTS intensive* project is any CBS in which all the non-trivial DC&P’s are not covered with COTS package(s), and many DC&P’s must be covered by custom development. The primary effort in such systems is in the custom design and development. Typically there are significant “Buy versus Build” decisions. Critical factors are: available COTS products, schedule limitations, budget limitations, and desired current and future capabilities.

Although non-COTS intensive project development encompasses a large number of CBS projects, it is not the primary focus of this research. Instead, this paper

will discuss the empirical motivation for distinguishing between CBS project types, their development characteristics and considerations, basic development guidelines, and use in strategic project planning and risk management.

## 1.1 Previous Work

A number of researchers [1,2,7,20] have identified the existence of different types of CBS, and have affirmed that the process and activities involved to be followed in developing each type of system largely differs. There are several proposed CBA development processes such as described in [1,2,3,4,6,7,20].

The authors of [20] have classified COTS based systems based on the manner in which the COTS product is used in the system. While this work includes similar terminology and indicates analogous CBA project types, the present work refines and justifies these types via empirical data gathered from COCOTS calibrations and our hands-on project experience with the various e-services projects.

## 1.2 COTS, COTS Based Systems, and COTS Based Applications

There are a multitude of definitions for COTS [14,25,26,27], but for the purposes of this work we adopt the SEI COTS-Based System Initiative's definition [7] of a *COTS product*: "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 identical copies and used without source code modification."

We also will adopt the definition of a COTS based system (CBS) as any system that makes use of a COTS package. COTS based systems however, may range from simple 'turnkey' systems such as Microsoft Office, Common Desktop Environment or Netscape Communicator, where a single system meets most of the DC&P's to systems that rely on a specific COTS product such as Oracle, but involve a large amount of custom development specific to the application [1]. Making clear distinctions between such systems is difficult, particularly when mainly COTS assessment is involved and no COTS customization. We are primarily concerned with the COTS related development effort required to satisfy the DC&P's relative to the overall development effort, not percentage of COTS used in the system or other such measures. For our purposes, we define a *COTS-Based Application* (CBA) as a system for which at least 30% of the end-user functionality (in terms of functional elements: inputs, outputs, queries, external interfaces, internal files) is provided by COTS products, and at least 10 % of the development effort is devoted to COTS considerations. The numbers 30% and 10% are not sacred quantities, but approximate behavioral CBA boundaries observed in our CBS projects. Here we observed a significant gap observed in COTS-related effort reporting where projects either reported less than 2% or over 10% COTS-related effort, but never between 2-10%.

The above empirical definition is based on our observations of CBS projects that required a great deal of COTS "special handling" with respect to overall

development effort in order to succeed. The percentages are only approximate and are likely to change (perhaps in relation to increasing COTS availability and maturity levels). However, we have noted that the majority of special handling for CBA projects lies in managing assessment, tailoring and glue-code efforts along with their associated project risks and value tradeoffs and other project lifecycle activities such as requirements modeling and system design. A critical element of this is to evaluate the COTS effort and risks within the feasibility analysis and project business case [18]. For example the “USC Collaborative Services” project from USC e-service projects was asked to identify a variety of COTS packages to support distributed project collaboration (e.g. message board, chat). The team quickly identified a number of potentially useful COTS packages and after a cursory assessment settled on an open source product called DotProject. At first DotProject seemed to satisfy nearly all of the desired operational capabilities and better still, it was free and easily customized. However, since the package was still in its beta version and had a lot of bugs that needed to be fixed, and most importantly, most of its collaborative features only supported one type of user, there would have to be a lot of customization for each feature to satisfy the projects needs. As a result, a round of business case analysis concluded that DotProject would be unfeasible due to the high demand of glue code effort needed to address its shortcomings. Though the team had already spent tremendous time on adapting this product into system requirements and architecture design, they chose to switch to another COTS package after a re-negotiation of the requirements with the client.

## 2 Motivation for CBA Project Typing

Many researchers have identified the necessity to use a special process to develop and implement CBSs [1,8] and some have proposed a generic process for CBS development [2]. However within the CBS domain there is a large variation in development approaches (e.g. turnkey, adaptation, integration) for which a single generic CBS process is unable to provide adequate development guidance. We have tried to incorporate such generic CBS processes with limited success. This was evidenced by observing numerous teams succumbing to effort allocation pitfalls. For example performing too much COTS assessment and neglecting to allocate enough time for training, tailoring and integration, resulting in project delays, inability to implement desired functional capabilities, and so forth. In some cases some developers did not perform enough assessment, which resulted in problems and delays during construction phases. For example in case of a project titled ‘Quality Management through Bore’ where the developers were mandated the use of the BORE product, the developers did not perform enough assessment which resulted in delays due to lack of product stability, promised features that were unavailable etc. resulting in serious losses due to schedule delays and unsatisfied capability requirements.

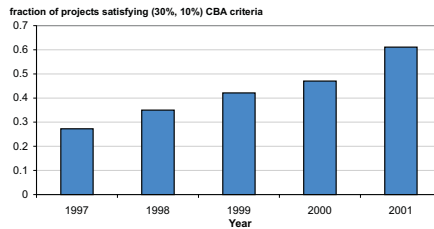
Such losses may be minor when it comes to implementing small e-service applications, however in case of large scale systems and systems required to meet

a ‘specific window of opportunity,’ such losses may be extremely significant. Our experience in USC e-service projects has led us to conclude that a generic CBS process is insufficient for CBA projects. To this end we have introduced guidelines for the early identification and continuous monitoring of a CBA project along with development guidance *based on the CBA project type*. A CBA project can be classified as *assessment intensive, tailoring intensive, or glue-code intensive*.

These classifications provide an overview of the potential risks, characteristics, and development effort priorities and are useful for strategic and tactical development planning. Early identification of the CBA type can help a project team identify a clear development process and mitigate risks especially risks due to COTS effort allocation.

## 2.1 Increase in CBA Projects

Our keen interest in CBAs is due to an observed dramatic increase in the number of CBA projects from 28% in 1997 to 60% in 2001 over a 5-year period as evidenced in Fig. 1.



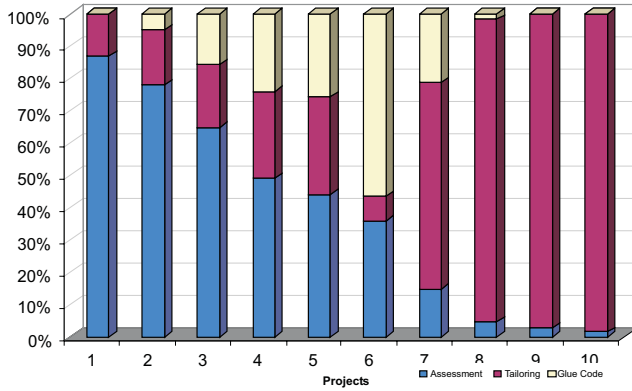
**Fig. 1.** CBA Growth in USC E-services Projects

Some of our USC-CSE affiliates have reported similar qualitative trends, but this is the first quantitative data they and we have seen on the rate of increase of CBA projects under any consistent definition and in any application sector (we note that e-services applications probably have higher rates of increase in recent times than many other sectors). We have experienced many notable effects of this increase: for example, programming skills are necessary but not sufficient for developing CBA's [8,9,10,11].

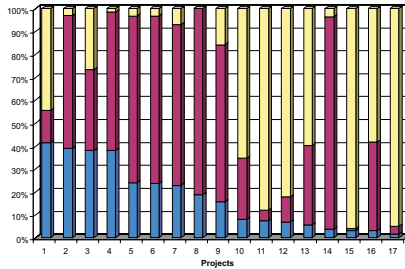
## 2.2 CBA Effort Distributions

Based on 2000-2002 USC-CSE e-services data and 1996-2001 COCOTS calibration data, we observed a large variation of COTS-related effort (assessment, tailoring, and glue code) distributions. This is clearly illustrated in the e-services and COCOTS COTS effort distributions of Fig 2. and Fig. 3 respectively.

Some CBA approaches, including our initial approach to COCOTS, just focus on one CBA activity such as glue code or assessment. As shown in Fig. 2, some



**Fig. 2.** CBA Effort Distribution of USC e-Service Projects



**Fig. 3.** CBAs Effort Distribution of COCOTS Calibration Data

projects (projects 7, 8, 9, 10) are almost purely tailoring efforts, while others (projects 1, 2, 3) spent most of the time on COTS assessment, and still others focused on glue code (project 6). The industry projects in Fig. 3 show similar variation. While some projects did not have glue code (projects 1, 9, 10), all projects had some degree of assessment and tailoring, but never tailoring or glue code only efforts, or a mix of just tailoring and glue code.

The industry projects in Fig. 3 were a mix of small-to-large business management, engineering analysis, and command control applications. Assessment effort ranged from 1.25 to 147.5 person-months (PM). Tailoring effort ranged from 3 to 648 PM; glue code effort ranged from 1 to 1411 PM.

This data indicates that the CBA types described in the introduction do indeed manifest themselves as usually one of the effort areas dominates and there is clearly are patterns in how COTS effort is distributed.

### 3 CBA Project Types

The three CBA types were defined briefly in Sect. 1. In this section we elaborate on the assumptions, some of the primary characteristics, critical activities, and risks for each type. To see the overall similarities and differences between the CBA types, the section closes with an overall summary comparison of the CBA types.

#### 3.1 Assessment Cots Based Application

Implied by the definition of assessment CBA (ACBA) in Sect. 1, the intent is to avoid doing custom development by finding COTS to handle the DC&P's. In this, an ACBA is based on two assumptions. First, that COTS packages are available to cover most of desired system capabilities identified according to system shared vision among stakeholders. Second, a considerable number of COTS packages need to be assessed against established evaluation criteria, or a few must be assessed in great depth. In another words, an ACBA will have the DC&P's covered without much modification (or tailoring) by a single COTS framework or through the simple integration of several.

**Characteristics.** The primary effort is focused on identifying collection of candidate COTS products and evaluating a feasible set of products whose existing capabilities directly address a desired operational concept. About 60-80% of COTS related effort is assessment effort, and accounts for 24% of overall effort.

Custom development, glue-code, and tailoring effort are undesirable. There must be high degree of flexibility in the requirements and design due to COTS capability uncertainties. COTS impose requirements based on their existing capabilities that typically do not match precisely with a systems needs. As a result, even after final COTS selection the system requirements may change to be kept compatible with the chosen COTS package and its evolutionary path.

A business case analysis is critical for a detailed COTS assessment. Early assessment effort may greatly reduce overall system cost, risk, and schedule by helping to establish a sound business case for a particular feasible set of COTS packages.

**Critical Activities.** The following are some critical activities that an ACBA must perform.

1. Plan: set the evaluation procedures and provide a baseline and guide for evaluation process.
2. COTS Identification: search for COTS candidates based on the DC&P's and filter out the risky COTS packages based on observation or other initial filtering criteria.
3. Establish evaluation criteria: Produce the evaluation criteria necessary for detailed evaluation. Identification of evaluation criteria is based on functional, performance, architectural, and financial characteristics of DC&P's.

4. Detailed Evaluation: Perform multiple cycle of evaluation to collect evaluation data. One type of detailed evaluation is scenario-based evaluation testing, focusing on verifying vendor's claim and test the feasibility and performance of COTS package(s) in the proposed system context.
5. COTS Recommendation: Analyze the evaluation data captured from each candidate and make trade-offs among the results to choose the final COTS package.

### 3.2 Tailoring COTS Based Applications

Following the definition of a tailoring CBA (TCBA) from Sect. 1, there is an implied desire to avoid complex custom development by identifying COTS for the DC&P's, but there is expectation that some customization must be done to adapt the selected COTS to the particular needs of the organization. This makes the following two obvious assumptions. There are COTS package(s) that can satisfy most of the requirements or DC&P's of the system. Owing to this, a TCBA must always include some amount of assessment effort—even if the COTS packages are pre-selected or are mandated for use. In this latter circumstance, assessment must be performed in order to identify the extent the COTS packages satisfy the DC&P's and the degree of tailoring required. Another trivial, yet vital assumption is that the COTS packages have some sort of a customization interface or capability for which the product can be tailored to meet the requirements of the system.

**Characteristics.** The primary effort is on tailoring the COTS package to meet the system requirements. Approximately 60-80% COTS related effort is spent on tailoring the system, which accounts for approximately 10% of overall development effort.

Custom development is undesirable and as such there is a need for moderate flexibility in requirements and design due to COTS uncertainties for similar reasons for an ACBA. The primary difference is that some of the requirements will not be totally negotiable. In such cases, the COTS package and the requirements will have to be adapted until they are mutually compatible. Typically this involves GUI and hardware requirements. For example a function selection menu may always need to include the company logo, a selected COTS package may require a certain amount of free disk space to operate.

The focus of system architecture depends upon the type of tailoring features available in the COTS package and the amount of tailoring required in order to satisfy the DC&P's. The different types of tailoring interfaces available include GUI based tailoring interface (e.g. Spearmint [28]) where no design involved, parameter based tailoring interface (e.g. Windows Media Player [29]) where little design may be involved in terms of passing of parameters or programmable tailoring interface (e.g. Hyperwave [30]) where detailed procedural or object oriented design may be required. Some COTS packages may have a combination of one, two or all three types of tailoring interfaces (e.g. Microsoft Office [31]).

**Critical Activities.** The critical activities for tailoring intensive systems include:

1. **Assessment:** Some assessment may be required in identifying the right COTS product to be used for the system. The amount of assessment shall be dependent upon the number of available COTS products, the number of constraints, available budget, type of tailoring interface etc.
2. **Tailoring Interface Identification:** The tailoring interface utilized in order to tailor the COTS product to satisfy the desired capabilities and priorities shall determine the future activities to be implemented.
3. **For a GUI Based Tailoring Interface:** Plan and Tailor the COTS product and document the actions and activities performed during the GUI tailoring of the system.
4. **For a parameter based tailoring system:** Design the interactions between the components within the COTS and implement the design.
5. **For a programmable tailoring system:** Design the procedures and objects required to implement the desired capabilities and priorities and implement the system.

### 3.3 The Glue-Code COTS Based Application

As specified in the definition in Sect. 1, the intent of a glue-code CBA (GCBA) is to use COTS as basic system components. The GCBA assumes that COTS packages are available to satisfy significantly valuable subsets of system capabilities or project limitations dictate use (e.g. schedule, skill, or complexity factors). This implies there is a reasonable buy versus build cost-benefit return on investment. In particular, the integration overhead is minimal with respect to custom development cost and that there is low assessment and tailoring effort needed. As with the TCBA, some amount of assessment will always be required.

**Characteristics.** The primary effort is focused on identifying system components and requirements that may be risky to custom implement and mapping them to a collection of candidate COTS products. Further, the COTS packages are expected to implement core system requirements; there is little flexibility in these requirements.

It is expected that the packages will require a significant amount of custom integration effort (glue-code) as the COTS packages used are not specialized to a particular application domain and their generic capabilities will need to be adapted to the particular system needs. Custom code development is typically necessary to satisfy requirements not covered by the selected COTS or where the business case indicates that ‘build’ considerations outweigh investing in COTS. In general, the COTS components utilized are not designed to execute independently and must be built application to function. Typically many possible COTS packages may apply and some cursory assessments must be performed to identify which best match the requirements. There may be many evolutionary requirements with respect to the custom components and anticipated COTS features.

## Critical Activities

1. Identify Implementation Risks: Consider risky development areas due to schedule limitations, skill limitations, high complexity, etc.
2. COTS Identification: search for COTS candidates based on the DC&P's and implementation risk areas.
3. Assessment: Evaluate the COTS candidates with respects to DC&P's, determine COTS risks due to requirements mismatch, faulty vendor claims, vendor support, familiarity with COTS packages, etc.
4. Buy Versus Build: Compare expected cost of utilizing individual COTS packages with building a component with similar capabilities, choose best mix of COTS, custom components.
5. Implement: Tailor COTS or develop glue code or integrate COTS or develop custom components as needed.
6. Integrate: Integrate COTS and custom components and tests.

**Table 1.** CBA Type Comparison

CBA type Project Type	ACBA	TCBA	GCBA	Non-CBA
<b>Number of COTSS</b>	Various single COTS or COTS mix solutions	Usually one leading COTS product much covers the system functions.	A mix of many COTS products.	Varies, but not any one or a combination of many intensively drives the system.
<b>Requirements Covered by COTSS</b>	Should be covered as much as possible based on assessment and business case analysis.	Mostly covered by adapting existing capabilities	Covers effort intensive or risky subsets	Few
<b>Requirements Flexibility</b>	High, requirements depend on COTS selection	Moderate (Change non-critical requirements if needed)	Low to moderate	Varies, but usually low
<b>COTS Assessment</b>	Significant (75%)	Little to moderate (1%)	Little to moderate (19%)	Varies
<b>COTS Tailoring</b>	None or little (22%)	Significant (89%)	Little to moderate (42%)	Varies
<b>COTS Glue Code</b>	None to little (3%)	None or little (10%)	Significant (39%)	Varies
<b>Domain Specificity</b>	Generic to domain or COTS is domain specific; organization will adapt to COTS	General to domain and can be refined to specific organization or mission	Independent of domain	Usually highly domain specific
<b>Custom Development</b>	None or very little	None or very little	Little	Significant development to implement custom capabilities
<b>Evolution Requirements Degree</b>	Generally Very Low, but included within assessed products	Limited to what is tailorable	High with respect to custom components and anticipated COTS features	Generally Very High
<b>Top-10 Risks (see Table 2)</b>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	7, 10, 14, 4, 8, 6, 13, 12, 15	4, 13, 11, 12, 16, 5, 13, 1, 7, 10	7, 13, 16, 2, 3, 4, 5, 14, 10, 12, 15

**Table 2.** Top CBA Project Risks from USC e-service projects 2000-2002

Risk	Common Mitigation Plan
1. Requirements changes and mismatches	Prototyping and business case analysis can help to estimate the effect of change and corresponding team effort and schedule needed. Win Win negotiation among all stakeholders must be maintained in each development phase.
2. Many new non technical activities are introduced in the Software Engineering Process	Stakeholders with Domain Expertise, Evaluation expertise must be included in the evaluation process
3. Miss possible COTS candidates within the COTS process	Stay as broad as possible when doing the initial searching for candidates
4. Too much time spent in assessment due to too many requirements and too many COTS candidates	Identify the core 'showstopper' requirements and filter all the COTS candidates that do not meet these during the initial assessment and then proceed for a more detailed assessment with the remaining COTS candidates
5. Might not include all key aspects for establishing evaluation criteria set. (Inadequate COTS assessment)	Involve experienced, knowledgeable stakeholders for reviewing evaluation criteria and weight distribution judgments
6. Introducing new COTS candidates is likely and requires re-planning	Develop a contingency plan in cases of addition of a new COTS product. Identify the limits on schedule and budget while making the introduction
7. Faulty Vendor Claims may result in feature loss and/or significant delays	Detailed analysis provides greater assurance of COTS characteristics with respect to vendor documentation (although at significant effort). Detailed assessment beyond literature review or vendor provided documentation should be performed in the form of hands-on experiments and prototyping (especially of core capabilities to be utilized).
8. Ability or willingness of the organization to accept the impact of COTS requirements	The project operational concept must identify such risks and they must be conveyed to the higher management
9. Difficulty in coordinating meetings with key personnel may result in significant delays	The key decision making personnel must be well accounted for the project life cycles. The project manager must make them aware of the approximate time required to be spent by them during the process of assessment etc. The decision making personnel must be kept as minimal as possible
10. Inadequate vendor support may result in significant project delays	The licensing of COTS products must account for vendor support details. In case of contracting labor the developers with experience in using the COTS must be selected
11. COTS package incompatibilities may result in feature loss and significant project delays (Integration Clash)	COTS integration issues must be considered during assessment. The number of COTS products must be kept as minimal as possible.
12. Added complexity of unused COTS features	The number of unused features could be identified and the added complexity because of the presence of such features must be calculated during COTS assessment
13. Overly optimistic expectations of COTS quality attributes	Significant quality features must be tested before selecting COTS products. Special testing packages may be used. Evaluations could be carried out at sites where the COTS is actually being used
14. Overly optimistic COTS package learning curve	An most likely COTS package learning curve must be accounted for during planning the schedule
15. A version upgrade may result in re-tailoring of COTS package	Ensure that the features used to implement the capabilities still exist in the new version before the version upgrade.

Risk	Common Mitigation Plan
16. Imposed black box testing of COTS components	A risk based testing plan must be developed and accounted for in the project life cycle schedule. The test cases must be designed to satisfy at least the high priority capabilities that the COTS package is responsible for implementing. In case of mission critical systems 'all' the capabilities being satisfied by the COTS package must be tested to the appropriate level of service. Additionally, the developers should ensure that the capabilities that were not tested are NOT implemented or used in the system (one way to do this is to build wrappers around the COTS components to ensure that the system can access only the capabilities that have been tested).

In Table 1, *Number of COTS* indicates the number of COTS used to develop the application. *COTS Requirements Coverage* indicates the number of requirements covered by the COTS application (s). *Requirements Flexibility* indicates the willingness of the organization (or project team) to modify the requirements based on the available capability of the COTS application. *Glue Code* indicates the custom development needed to integrate COTS packages within an application external to the packages themselves. *Tailoring* indicates the activities relating to adapting COTS packages internally for use within an application. *Assessment* indicates the suitability evaluation and analysis of the desired attributes of the COTS packages for an application. *Domain specificity* indicates how specific the application is within a given domain (e.g. A library management system is highly specific to the library domains). *Custom Development* indicates the custom development required to satisfy requirements for the system. *Evolution Requirements Degree* indicates the degree to which the system can evolve once it has been developed.

Based on the developers weekly top risk reports, our architecture review board comments and observations (which are performed twice each semester), the risks in Table 2 have been identified for CBA's.

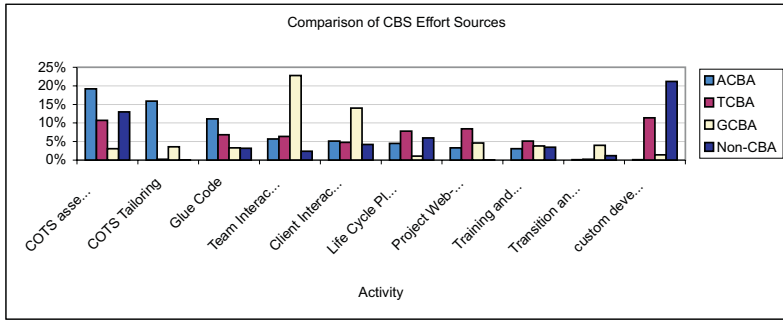
## 4 CBA Project Type Lifecycle Effort Comparisons

Up to this point, we have discussed three types of CBA's. Though the typing is based on the intensity of certain COTS related activities, most of other activities as well as the entire development process will also be significantly affected by the CBA type [10]. This section will examine such differences among CBA types.

The following chart illustrates the effort sources comparison among four CBA categories based on effort data reported by 9 CBA projects from year 2001 to 2002.

From Fig. 4, the following observations show the impact of different CBA types on development effort:

1. Both CBA projects and non-CBA projects have a lot (around 15%) of total effort spent on team interactions (activity 1).
2. ACBA has more than 15% of total effort on COTS assessment (activity 2), while TCBA and GCBA have less, but larger than 1% assessment effort.



**Fig. 4.** Comparison of USC e-service CBA Projects Effort

- CBA projects have spent about twice as much effort on client interaction as non-CBA projects as shown by activity 3. It is known that at least 5% of client interaction is unreported assessment or tailoring effort.
- Life Cycle Planning (activity 4) also accounts for about 5% of both Assessment and TCBA's total effort, which corresponds to life cycle re-planning caused by COTS uncertainty and volatility.
- CBA projects require equal or more effort on training and preparation (activity 6) to thoroughly research an application. This effort is included as assessment effort (note that non-CBA projects did not report COTS assessment directly).
- The reported effort data shows there is about 8% COTS tailoring effort (activity 7) for TCBA, however, we believe the number should be certainly much higher since according to our investigation on those teams, at least one team has reported most of their tailoring effort as critical component development (activity 10).
- GCBA has about 3% of glue code effort (activity 9) compared to that it only has 1% of critical component development effort (activity 10) as well.

## 5 Conclusions

### 5.1 Benefits of CBA Typing

Our empirical data and hands-on experience has confirmed that the three major CBA types have a significant impact on CBS project development. That is, not all CBA projects are created equally. Insight into the characteristics and risks of these types can help a project rapidly converge on a successful development focus. Further, more specific guidance ('special handling') can be provided within a CBA development process. In particular, enabling more effective use of COCOTS to estimate CBA development cost and effort via its three CBA activity sub-models. With this, COCOTS also provides a detailed set of assessment and tailoring checklists, glue-code activities, cost drivers, and risk assessments.

## 5.2 Future Work

We have provided a classification of COTS based systems based on data obtained from COCOTS and USC e-service CBA projects. There are however many questions are yet to be answered. What is the process to be followed for the development of each type of COTS intensive systems? What is to process followed in case of a composite system that contains COTS based activities as well as non-COTS development? Are there more factors that shall affect the business case that would help identify the type of system being developed?

We are currently developing a detailed CBA process model that accounts for the special handling of CBA, which will be used for future USC e-service CBA projects and its effectiveness, tested in terms of risk management, schedule management and other project success factors.

## References

1. David C.: Assembling Large Scale Systems from COTS Components: Opportunities, Cautions, and Complexities. In: SEI Monograph on the Use of Commercial Software in Government Systems
2. Morisio, M., Seaman, C. B., Parra, A. T., Basili, V. R., Kraft, S. E., and Condon S. E.: Investigating and Improving a COTS-based Software Development Process. In: Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, June (2000)
3. Jyrki, K.: A Case Study in Applying a Systematic Method for COTS Selection. In: Proceedings of the 18th International Conference on Software Engineering, Berlin, Germany, May (1996)
4. Braun, C. L.: A Life Cycle Process for Effective Reuse of Commercial Off-the-Shelf (COTS) Software. In: Proceedings of the Fifth Symposium on Software Reusability, Los Angeles, California (1999)
5. Boehm B., and Abts, C.: COTS Integration Plug and Pray. In: IEEE Computer, January (1999)
6. Vigder, M. R., and Dean, J. C.: Building Maintainable COTS Based Systems. National Research Council of Canada
7. Dean, J., Oberndorf, P., Vigder, M., Abts, C., Erdogmus, H., Maiden, N., Looney, M., Heineman, G., and Guntersdorfer, M.: COTS Workshop: Continuing Collaborations for Successful COTS Development
8. Carney, D.: Requirements and COTS-based Systems: A Thorny Question Indeed. Available at:  
[http://interactive.sei.cmu.edu/news@sei/columns/the\\_cots\\_spot/2001/1q01/cots-spot-1q01.htm](http://interactive.sei.cmu.edu/news@sei/columns/the_cots_spot/2001/1q01/cots-spot-1q01.htm)
9. Seacord, R. C.: Building Systems from Commercial Components: Classroom Experiences. Available at:  
[http://interactive.sei.cmu.edu/news@sei/columns/the\\_cots\\_spot/cots-spot.htm](http://interactive.sei.cmu.edu/news@sei/columns/the_cots_spot/cots-spot.htm)
10. Hissam, W. S., and Seacord, R.: Building Systems from Commercial Components. Addison-Wesley (2002)
11. Boehm, B., Port, D., Abi-Antoun, M., and Egyed, A.: Guidelines for the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) Deliverables for Model-Based Architecting and Software Engineering (MBASE). USC Technical Report USC-CSE-98-519, University of Southern California, Los Angeles, CA, February (1999)

12. Boehm, B., and Port, D.: Educating Software Engineering Students to Manage Risk, *Software Engineering*. In: ICSE 2001 (2001)
13. COTS Risk Mitigation Guide.  
Available at: <http://www.faa.gov/aua/resources/COTS/Guide/crmg122101.pdf>
14. Victor R. B., and Barry W. B.: COTS-Based Systems Top 10 List. *IEEE Computer* 34(5) (2001)
15. Vidger, M. R., Gentleman, W. M., and Dean, J.: COTS Software Integration: State of the Art (1998) Available at: <http://wwwsel.iit.nrc.ca/abstracts/NRC39198.abs>
16. Brownsword, L., Oberndorf, P., and Sledge, C.: Developing New Processes for COTS-Based Systems. In: *IEEE Software*, July/August (2000) 48–55
17. <http://www.cebase.org>
18. Donald J. R.: *Making the Software Business Case*. Addison-Wesley, September (2001)
19. Boehm, B., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D., and Steece, B.: *Software Cost Estimation with COCOMO II*. Prentice Hall PTR, July (2000)
20. Wallnau, K. C., Carney, D., and Pollak, B.: *How COTS Software Affects the Design of COTS-Intensive Systems*. Available at:  
[http://interactive.sei.cmu.edu/Features/1998/june/cots\\_software/Cots\\_Software.htm](http://interactive.sei.cmu.edu/Features/1998/june/cots_software/Cots_Software.htm)
21. Maurizio, M., and Marco, T.: *Definition and Classification of COTS: A Proposal*. In: *First International Conference, ICCBSS 2002, Orlando, FL, USA, February (2002)*
22. C. for Software Engineering. *Cocots*, Technical Report, University of Southern California, Los Angeles, CA, June (1999)
23. *COCOTS home page*. Available at: [http://sunset.usc.edu/research/COCOTS/cocots\\_main.html](http://sunset.usc.edu/research/COCOTS/cocots_main.html)
24. *Center for Software Engineering, University of Southern California, Los Angeles, CA MBASE home page*.  
Available at: <http://sunset.usc.edu/research/MBASE/index.html>
25. Oberndorf, T.: *COTS and Open Systems—An Overview (1997)* Available at: <http://wei.sei.cmu.edu/str/descriptions/cots.html#ndi>
26. Vidger, M. R., Gentleman, W. M., and Dean, J.: *COTS Software Integration: State of the Art (1998)* Available at: <http://wwwsel.iit.nrc.ca/abstracts/NRC39198.abs>
27. Brownsword, L., Oberndorf, T., and Sledge, C. A.: *Developing New Processes for COTS-Based Systems*. In: *IEEE Software*, July/August (2000)
28. *Fraunhofer IESE*. Available at: [http://www.iese.fhg.de/Spearmint\\_EPG/](http://www.iese.fhg.de/Spearmint_EPG/)
29. *Windows Media Technologies Web Site*. Available at:  
<http://www.microsoft.com/windows/windowsmedia/download/default.asp>
30. *Hyperwave AG, Humboldtstraße 10, 85609 Munich-Dornach, Germany Web Site*. Available at: <http://www.hyperwave.com>
31. *Microsoft Office Web Site*. Available at: <http://www.microsoft.com/office/>