STACE: Social Technical Approach to COTS Software Evaluation

Douglas Kunda

Integrated Financial Management Information Systems (IBIS) Project
Ministry of Finance and National Planning, Lusaka, Zambia
dkunda@zamnet.zm

Abstract. COTS-Based Systems (CBS) development is a process of building systems from pre-fabricated Commercial-Off-The Shelf (COTS) software components. CBS success depends on successful evaluation and selection of software components to fit customer requirements. Selecting COTS software components to fit requirements is still a problem because of a number of problems including lack of a well-defined process, the "black box" nature of COTS components and the rapid changes in marketplace. This chapter reviews some existing frameworks that support COTS software selection and demonstrate that these frameworks do not adequately address the non-technical issues. The chapter then presents a social-technical approach for COTS software selection, how it deals with non-technical issues, its application in an organization and lessons learnt

1 Introduction

Modern software systems are becoming difficult and expensive to develop and organizations are turning to Commercial-Off-Shelf (COTS) software packaged solutions. COTS software package approach can potentially be used to reduce software development and maintenance costs, as well as reducing software development time by bring the system to the markets as early as possible. For example, most organizations spend too much effort defining to the lowest level of detail of the desired characteristics of the systems and how the contractor are to build the system when a COTS products already exist with nearly the same capabilities.

According to Oberndorf [27] the term "COTS" is meant to refer to things that one can buy, ready-made, from some manufacturer's virtual store shelf (e.g., through a catalogue or from a price list). It carries with it a sense of getting, at a reasonable cost, something that already does the job. The scenario of developing unique system components is replaced by the promise of fast, efficient acquisition of cheap (or at least cheaper) component implementations. Examples of COTS products include Geographic Information Systems (GIS), Graphical User Interface (GUI) builders, office automation, email and messaging systems, databases and operating systems.

There are two distinct ways to use COTS software. In one, a single complete working COTS software system that satisfies most of the user requirements is purchased and used as platform upon which to build a system [10]. For example a database management system can be purchased and used to build a payroll system. The second model is one which involves purchasing a number of COTS software components (usually without the source code) each satisfying some part of the requirements of the system and integrating these components into the required system [35]. The second model is important because many systems need functions in multiple orthogonal sub-domains, each of which tends to be addressed by a different COTS software package [10].

However, successful selection of COTS software to fit requirements is still problematic for a number of reasons. These include the lack of a well-defined process, the "black box" nature of COTS components, misuse of data consolidation method and rapid changes in market place. Kontio [19] points out that most organizations are under pressure to perform and therefore do not use a well-defined repeatable process. This makes planning difficult, appropriate evaluation methods and tools are not used, lessons from previous cases are not learnt and the evaluation process efficiency reduced. Another problem with COTS software selection is lack of inclusion of the "soft" issues or non-technical factors such as costs, organizational issues, vendor capability and reputation [29]. In order to address these problems a social technical framework for COTS evaluation (STACE) has been developed [20] and will be discussed in chapter.

2 Background

COTS software selection, also known as component qualification, is a process of determining "fitness for use" of previously-developed components that are being applied in a new system context [13]. Component qualification is also a process for selecting components when a marketplace of competing products exists. Qualification of a component can also extend to include qualification of the development process used to create and maintain it (for example, ensuring algorithms have been validated, and that rigorous code inspection has taken place) [7]. This is most obvious in safety-critical applications, but can also reduce some of the attraction of using pre-existing components.

There are three major strategies to COTS evaluation: progressive filtering, keystone identification and puzzle assembly [28]. Progressive filtering is a strategy whereby a component is selected from a larger set of potential components. This strategy starts with a large number of candidates set of components, progressively more discriminating evaluation mechanisms are applied in order to eliminate less "fit" components [22]. In keystone selection strategy, a keystone characteristic such as vendor or type of technology is selected first before selecting the COTS products [39]. Often, interoperability with the keystone becomes an overriding concern, effectively eliminating a large number of other products from consideration. The puzzle assembly model begins with the premise that a valid COTS solution will require fitting the various components of the system together as a puzzle and applies an evolutionary prototyping technique to build versions that are progressively closer to the final system [28].

2.1 COTS Software Evaluation Process

A number of researchers and organizations have proposed process models for evaluating COTS software (for example, [17,30]). However, most authors partition it into the following phases: requirements engineering; evaluation criteria definition, identification of candidate COTS products and assessment [8,19]. These phases are briefly discussed below (see Fig.1).

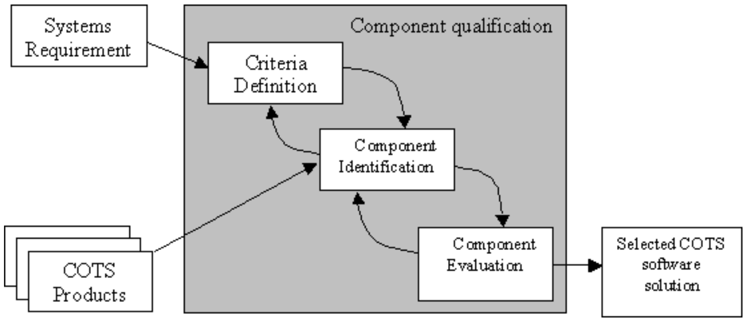

Fig. 1. COTS Software Evaluation Process

Requirements engineering covers all of the activities involved in determining requirements which will assist in establishing a basis for evaluating and selecting appropriate COTS software candidates. Tran, Liu and Hummel [35] argue that the requirements should be broken down and organized into collections of domain-specific requirements. This is important to support the early identification of candidate COTS products for evaluation as well as early identification of subsystems that cannot be supported by COTS products. In order to realize the benefits of COTS software, Vigder, Gentleman and Dean [37] suggest that a procurement process must be in place that defines requirements according to what is available in the marketplace. This is contrarily to the traditionally procurement process, which identifies strict requirements which either excludes the use of COTS components, or requires large modifications to COTS packages in order to satisfy the requirements. However, it is important that requirements are not defined so specifically that only one particular COTS product is suitable.

Defining the evaluation criteria. The criteria definition process essentially decomposes the high-level requirements for the COTS software into a hierarchical criteria set and each branch in this hierarchy ends in an evaluation attribute [19]. The criteria is specific to each COTS evaluation case but should include component functionality (what services are provided), other aspects of a component's interface, business concerns such as cost and quality aspects (e.g., reliability, portability, and usability) [8,17,35].

Identification of candidate components (alternatives). The identification of candidate components also known as alternative identification involves the search and screening for COTS candidate components that should be included for assessment in the evaluation phase [8,30]. Many authors highlight a number of techniques for identifying candidate COTS software including Internet search, market surveys, attending computer fairs and shows, invitation to tender (ITT) or request for proposals (RFP), vendor promotions and publications [19,31,35].

Assigning measure of merit to alternatives (evaluation phase). In the evaluation phase, the properties of the candidate components are identified and assessed according to the evaluation criteria [19,31]. Evaluation includes the acquisition of the products to be evaluated, developing evaluation plans, installing them, learning to use them, studying their features and assessing them against the criteria [35]. The methods and techniques for evaluation are discussed in the next section.

2.2 Methods and Techniques for Evaluation

Once the criteria are defined, the screened candidate products can be examined to observe to what extent they exhibit these or other useful attributes. The following are some of the techniques used to evaluate COTS software component:

– *Paper evaluation.* This is the process of evaluating the COTS products based on supplier data in sales brochure, technical documents, telephone conversations, web site information [22]. However, Beus-Dukic and Wellings [3] suggests that vendors claims must be viewed sceptically, therefore this technique must be used in combination other evaluation techniques.
– *Market survey.* A market survey can be made using questionnaires and interviews with vendors, trade shows, user community to compile quantitative and qualitative data about the product and vendors. Finkelstein, Spanoudakis and Ryan [12] point out that in certain circumstances, especially if the package to be bought is expensive, a request for proposal (RFP) can be issued, which enable the vendors to describe their packages in a uniform manner.
– *Experimentation.* This is a rigorous test of the product to assess its compliance with the defined criteria. The experimentation process includes the acquisition and installation of the product, design of the appropriate prototype and test plan, evaluation of product and generation of report [35]. Carney and Wallnau [8] stress the importance of conducting experimentation within the operating environment (context) in which the product will be used. Maiden and Ncube [22] recommend the use of software prototypes to assist in generating test cases for product evaluation. This especially important where the evaluator do not have prior knowledge about the candidate products or prior extensive experience generating test cases.
– *Pilot study.* A pilot study is an extended version of experimentation in which “real” data from the organization is used in the evaluation. Brown and Wallnau [6] argue that it is important to demonstrate the product or technology’s feasibility with a pilot project. Sledge and Carney [34] points out that

because the potential for misinterpretation and misunderstanding when presenting or discussing a commercial product is great, hands-on evaluation of COTS products is mandatory and pilot programs are a useful way to do this.
- *Vendor analysis.* Hokey [15] points out that the vendor must be evaluated in terms of user services (installation assistance, training services and warranty) and vendor characteristics (vendor reputation and vendor stability). Checking vendor discontinuities, such as focus shifts and change of auditor, would help in this process. Haines et al. [13] and McDermid [23] argue that for safety-critical systems it is important to audit the development process that was used to develop the software including the tests carried out, conformance to standards, etc.

3 Problems with COTS Software Selection

The success of COTS-based software systems depends on successful evaluation and selection of COTS software components to fit customer requirements [22]. Successful selection of COTS software to fit requirements is still a problem because of a number of reasons. These include the following:

- *Lack of well-defined process.* Most organizations are under pressure to perform and therefore do not use a well-defined repeatable process [19]. The evaluators may not have the time or experience to plan the selection process in detail and therefore, they may not use the most appropriate methods in the selection process [19]. The resulting urgency means that evaluation decisions become pressured and a difficult decision becomes even more risky [29]. Furthermore, when the selection process is not defined, it is reinvented each time, it is performed inconsistently and learning from previous cases is difficult [19].
- *"Black box" nature of COTS components.* Lack of access to the COTS internals makes it difficult to understand COTS components and therefore evaluation is harder [37]. Sometimes even the supporting documentation for these components is incomplete or wrong. The design assumptions of the component are unknown; there is no source code when it needs debugging; and testing will be necessarily incomplete, since testing is only done for those functional capabilities that the customer care about [8].
- *Rapid changes in the market place.* The component user has little or no control over COTS product evolution [36]. Frequent releases of COTS components and rapid changes in the market place makes evaluation difficult [8]. For example, a new release of the COTS component may have a feature that is not available in the component that is currently being evaluated.
- *Misuse of data consolidation method.* A common approach to consolidating evaluation results is to use some kind of weighted sum method (WSM) [25]. However, the WSM has been criticized because assigning weights for the criteria sometimes can be inconsistent and lead to confusion about which is the most essential customer requirements [22].

However, the major problem with COTS software evaluation is that evaluators tend to focus on technical capabilities at the expense of the non-technical or “soft” factors such as the human and business issues [9,29]. Oberndorf et al. [28] highlight the usefulness defining the criteria to include such issues as vendor’s time in business, responsiveness to customers and willingness to support their product. Therefore, the evaluation criteria must incorporate both technical attributes and non-technical issues such as business issues and vendor capability variables.

4 COTS Software Evaluation and Multi-attribute Decision Making

Carney and Wallnau [8] argue the COTS software selection is a form of decision making. Kontio [19], Maiden and Ncube [22] support this view and further point out that it is a Multiple Attribute Decision-Making (MADM) process. MADM refers to making preference decisions (for example evaluation, prioritization, selection) over the available alternatives that are characterized by multiple, usually conflicting attributes [41]. The goal of MADM is (a) to help the decision maker choose the best action or alternative of those studied (a choice or selection procedure), (b) to help sort out alternatives that seem “good” among a set of alternatives studied (a sorting or segmentation procedure), and/or (c) to help rank the alternatives in decreasing order of preference (an ordering or ranking procedure) [24]. According to Yoon [41], MADM share the following characteristics:

– *Alternatives*: A finite number of alternatives, from several to thousands, are screened, prioritized, selected and/ or ranked.
– *Multiple attributes*: Each problem has multiple attributes or goals or criteria. For each problem setting relevant attributes are generated, for example, to purchase a car you may have price, gas mileage, safety and warranty period.
– *Incommensurable Units*: Each attribute has different units of measurement.
– *Attribute Weights*: Almost all MADM methods require information regarding the relative importance of each attribute, which is usually supplied in an ordinal or cardinal scale.
– *Decision matrix*: A MADM problem can be concisely expressed in a matrix format, where columns indicate attributes considered in a given problem and rows list competing alternatives.

A number of MADM techniques have been applied in software selection, the most common are weighted sum or scoring method [40], analytical hierarchy method [15,19,22] and outranking method [1,25].

4.1 Weighted Sum Method

The Weighed Sum Method (WSM) or scoring method is one of the simplest and probably the most popular technique for solving multi-attribute decision problems [24]. The WSM is based on the multiple attribute utility theory with the following axiom: any decision-maker attempts unconsciously (or implicitly) to maximize some function by aggregating all the different points of view which are taken into account [38]. A score in this method is obtained by adding contributions from each alternative and since two items with different measurement units cannot be added, a common numerical scaling system such as normalization is required to permit addition among attributes values [41]. The total score for each alternative then can be computed by multiplying the comparable rating for each attribute by the importance weight assigned to the attribute and then summing these products over all the attributes.

The main advantage of the WSM is its ease of use and helping the decision-maker to structure and analyze the decision problem [24]. However, Mollaghasemi and Pet-Edwards [24] criticize the WSM arguing that this method tends to involve ad hoc procedures with little theoretical foundation to support it. This can lead to confusion about the most essential customer requirements [22] and make worst products on important attributes have the highest aggregated scores [25]. Another weakness is that it is difficult to define a set of criteria and their weights as advocated in the WSM so that they are either independent of each other or if they overlap, their weights are adjusted to compensate for overlapping areas [19]. This suggests that WSM might not be suitable for aggregating COTS software evaluation attribute data because most COTS software attributes are not independent of each other.

4.2 Outranking Method

Outranking methods are a class of multi-criteria decision-making techniques that provide an ordinal ranking (and sometimes partial ordering) of the alternatives [24]. It has been successfully applied to COTS software evaluation and selection [1,25]. Roy [32] developed the outranking approach and a family of evaluation methods collectively known as ELECTRE methods that are founded on the outranking relations. Yoon [41] points out that ELECTRE methods dichotomizes preferred alternatives and non-preferred ones by establishing outranking relationships. An outranking relationship (A outranks B) states that even though two alternatives A and B do not dominate each other, it is realistic to accept the risk of regarding A as almost surely better than B [41].

The advantage of this approach is the ability to consider both objective and subjective criteria and the least amount of information required from the decision maker [24]. Morisio and Tsoukias [25] suggest that outranking methods are appropriate when the measurement scales of criteria are of an ordinal and when it is not possible to establish trade-offs between criteria. Mollaghasemi and Pet-Edwards [24] point out that, although it can be expressed that alternative A is preferred to alternative B in the outranking method, it does not indicate by how much, for example with ELECTRE I a complete ranking of the alternatives may not be achieved. Therefore, this method is not appropriate for COTS software selection involving tenders that require explaining to the unsuccessful bidders why their bid was unsuccessful and how they were ranked.

4.3 Analytical Hierarchy Process (AHP)

AHP was developed by [33] for multiple criteria decision making and has three basic functions: (1) structuring complexity, (2) measuring on a ratio scale, and (3) synthesizing. AHP has been successfully applied in software and computer selection [42,19,22]. AHP enables decision-makers to structure a multi-criteria decision making problem into a hierarchy [41]. A hierarchy has at least three levels; the overall goal of the problem at the top, multiple criteria that define alternatives in the middle and competing alternatives at the bottom.

AHP technique is based on pair-wise comparison between the alternatives. The result of this pair-wise comparison is converted to a normalized ranking by calculating the eigenvector from the comparison matrix's largest eigenvalue. Section 1.5.3 provides a worked example of the use of AHP. The advantage of the AHP technique is that it provides a systematic approach for consolidating information about alternatives using multiple-criteria [19]. The availability of several software packages to support the AHP has made it a popular technique [24]. AHP also provides a means for measuring the consistency of the decision-maker's judgements, that is, to check the quality of the results in the comparison matrix [24,42].

AHP has been criticized regarding the rank reversal: the reversal of the preference order of alternatives when new options are introduced in the problem [11,24]. Furthermore, that the use of a 1 to 9 measurement scale is inappropriate because of the ambiguity in the meaning of the relative importance of one factor when compared to another. However, Harker and Vargas [14] argue that rank reversal occurs in AHP because ranking of alternatives depends on the alternatives considered, hence, adding or deleting alternatives can lead to changes in the final rank and this is consistent with rational behavior. Furthermore, since AHP facilitates group decision-making it would be suitable for COTS software selection process that emphasizes participation. In addition, AHP would be appropriate for aggregating COTS software evaluation attribute data comprising technical and non-technical issues because it incorporates both quantitative and qualitative data into the decision making process.

5 Social Technical Approach to COTS Software Evaluation (STACE)

A number of frameworks for evaluating and selecting COTS software components have been proposed in literature. Useful work includes Delta technology framework that help evaluate new software technology [7] and PORE, a template based method to support requirements acquisition for COTS product selection [22]. Although the Delta technology framework is useful for evaluating new technology it does not address the political and economic factors that often separate a winning technology from other contenders. The weakness of PORE method is that it is labor-intensive and vulnerable to the neglect of social issues. Another technique the OTSO [19] addresses the complexity of component selection and provides a decision framework that supports multi-variable component selection analysis but neglects the non-technical issues or “soft” factors. Other approaches, such as the Software System Evaluation Framework (SSEF) [4] focuses on assessing the software product, process and their impact on the organization but it does not provide guidance on how to define the evaluation criteria. The following presents a summary of characteristics, strengths and weaknesses of each of these frameworks.

SSEF
- *Characteristics*
  - It proposes a top-down approach that identifies the important elements that a software system must include to foster high-level understanding.
  - Uses knowledge world’s concepts (i.e., usage world, development world and system world).
  - Multiple viewpoints approach to evaluation (user satisfaction and economic returns).
  - Defines the elements (dimensions, factors, and categories) clearly to facilitate evaluation and reduce the evaluators’ conflicting viewpoints.
  - It is organized along three dimensions corresponding to the software’s producers, operators, and users.
- *Strengths*
  - It provides a baseline for establishing metrics programs in organization [4].
  - It offers a broad system snapshot by considering a number of different perspectives (end users, developers, and operators) [7].
  - A top-down approach has the advantage of flexibility, permitting extensions by following a predefined pattern [4].
- *Weaknesses*
  - It is not specific to COTS selection and the issues of how to define the evaluation criteria are not addressed [19].
  - It gives little detailed insight into the strengths and weaknesses of a technology in comparison with its peers [7].

OTSO
- *Characteristics*
  - Provides explicit definitions of tasks in the selection process, including entry and exit criteria [19];
  - Advocates incremental, hierarchical and detailed definition of evaluation criteria;
  - Provides a model for comparing the costs and value associated with each alternative, making them comparable with each other;
  - Uses appropriate decision-making methods to analyze and summarize evaluation results.
- *Strengths*
  - It addresses the complexity of COTS software evaluation [7].
  - The systematic repeatable process can promote learning through experience and improve the COTS selection process [19].
  - The use of the AHP provides evaluation consistency and provides structured information.

- *Weaknesses*
  - AHP is only appropriate when there are few comparisons and when all criteria are independent [22]
  - Neglect of non-technical issues or “soft” factors [29].

Delta
- *Characteristics*
  - Evaluate a new software technology by examining its features in relation to its peers and competitors
  - It is a systematic approach that includes modelling and experiments.
  - That technology evaluation depends on understanding technology “delta” descriptions of how a new technology’s features differ from other technologies.
  - Evaluates how these “delta” differences address the needs of specific usage contexts.
- *Strengths*
  - It provides techniques for evaluating the product underlying technology.
  - It can also facilitates individual product evaluations that concentrate on their distinguishing characteristics in relation to their technology precursors and product peers [6].
- *Weaknesses*
  - It focuses on technology evaluation and neglect product and vendor evaluation
  - It does not address the political and economic factors that often separate a winning technology from other contenders.

PORE
- *Characteristics*
  - It integrates existing requirements engineering methods and other techniques such as feature analysis and multi-criteria decision-making.
  - It is template-based (templates provide guidelines for conducting evaluation).
  - It advocates for a parallel and an iterative requirements acquisition and product selection/rejection.
- *Strengths*
  - It provides guidance to model requirements for COTS software selection
  - The parallel requirements acquisition and COTS software selection means requirements acquisition informs COTS software selection and vice versa.
- *Weaknesses*
  - Use of traditional approaches make it vulnerable to neglect of social issues
  - It is labor-intensive.

STACE – *Characteristics*
  - It supports a systematic approach to COTS evaluation and selection
  - It proposes a keystone evaluation strategy in which the underlying technology is selected before selecting the COTS products.
  - It uses social-technical techniques (i.e., social-technical criteria and participation) to improve the COTS software selection process.
  - It uses multi-criteria decision-making techniques (i.e. AHP) to consolidate evaluation attribute data.

– *Strengths*
  - It addresses the non-technical issues through use of social-technical techniques.
  - It supports evaluation of both COTS products and the underlying technology.
  - It provides for reuse of lessons learnt from previous evaluation cases.
  - Use of the AHP promotes consensus, transparency and consistency checking.

– *Weaknesses*
  - It increases the cost of the evaluation process because of inclusion of non-technical issues and user participation.
  - Some aspects of AHP having subjective bias.
  - Some users/ stakeholders may not make an effective contribution.

In general what is missing in these frameworks is how to address the "soft" issues or the non-technical factors, such as costs, organizational issues, vendor capability and reputation. Therefore, STACE was developed to facilitate a systematic requirements-driven COTS software selection and address this problem using social-technical techniques. Furthermore, STACE supports the evaluation of both COTS products and the underlying technology while the other frameworks emphasize product or technology evaluation. Another advantage of STACE is that it provides for reuse of lessons learnt from previous evaluation cases by maintaining a database of evaluation results.

5.1 Objective and Principles of STACE

The STACE framework has been developed through literature survey and case studies [20]. STACE is based on a number of important principles:

– Support for a *systematic approach* to COTS evaluation and selection. Most organizations select their COTS components in an ad-hoc manner. There is a need, for example, to reuse lessons learnt from previous evaluation cases by maintaining a database of evaluation results.
– Support for *evaluation of both COTS products and the underlying technology*. Most COTS evaluation frameworks emphasize either COTS products evaluation or technology evaluation. This method proposes using keystone evaluation strategy in which the underlying technology is selected before selecting the COTS products.

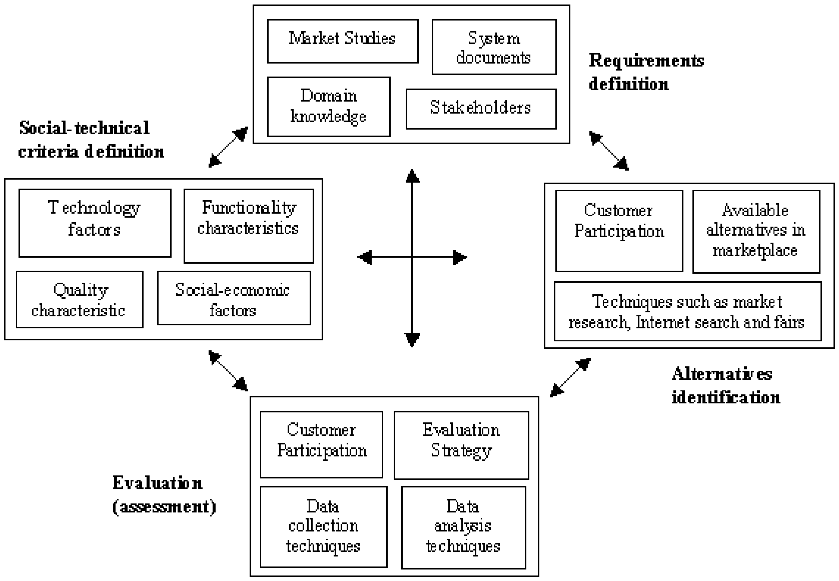

Fig. 2. STACE Framework

- Use of *social-technical techniques* to improve the COTS software selection process. This has been greatly influenced by the social-technical school and work by [26]. STACE recommends the use of a social-technical evaluation criteria and customer participation in the COTS selection process. User participation is regarded as an effective strategy as a means of improving software design outcomes and as a means of incorporating human and organizational aspects such as the design of jobs, work processes and usability [5,2].
- Use of *multi-criteria decision-making techniques* to consolidate evaluation attribute data. The STACE proposes the use of Analytic Hierarchy Process (AHP) as developed by Saaty [33] and successfully used in software selection [19,42].

5.2 STACE Method

The STACE method (see Fig. 2) comprises four interrelated processes: 1) requirements definition; 2) social-technical criteria definition; 3) alternatives identification; and 4) evaluation or assessment.

In the requirements definition process, the high-level customer and systems requirements are discovered through consultation with stakeholders, from system documents, domain knowledge and market studies. The traditional requirements engineering methods emphasize the technical issues while neglecting the equally important social issues [18]. Therefore, the STACE framework recommends the use of the social-technical approach to systems development. Customer participation is one of the strategies used in social-technical approaches to incorporate the social issues in the development of the system. The STACE framework recommends the use of Joint Application Development (JAD) sessions and review meetings with top management to elicit and validate requirements from stakeholders. The use of JAD or stakeholder workshops is an important strategy that operationalizes customer participation.

In the social-technical criteria definition process, the high-level requirements from the requirements definition phase are decomposed into a hierarchical criteria set and each branch in this hierarchy ends in an evaluation attribute [19]. The STACE framework uses a decomposition approach that is based on social technical analysis and the AHP criteria decomposition method. The STACE framework recommends decomposition of the high level requirements into a hierarchy of social-technical criteria comprising functionality characteristics, technology factors, product quality characteristics, and social-economic factors. Socio-economic factors are non-technical factors that should be included in the evaluation and selection of COTS components such as costs, business issues, vendor performance and reliability.

The objective of the alternatives identification process is to identify COTS components that meet the high level requirements, so that they can be considered for a more rigorous evaluation. In the STACE framework, this phase begins with identifying the domains relevant to the problem and understanding the types of packages available in those domains. The STACE framework recommends a number of techniques and tools for identifying candidate COTS products. These include networking, mailing list and user community, Internet search, market surveys, invitation to tender (ITT) or request for proposals (RFP), vendor promotions and publications.

The evaluation or assessment phase involves contacting vendor technical support for evaluation information, reviewing vendor documentation and product testing for quality and functionality. It also includes evaluating COTS performance, interfaces and ease of integration, comparing short-term and long-term licensing costs against integration costs. STACE recommends the keystone selection strategy with the technology as the keystone issue. The separation of COTS underlying technology from COTS products during evaluation allows fair comparisons between products.

The STACE framework also recommends separating the data collection and data analysis of the evaluation. Kontio [19] argues that the advantage of separating the data collection from analysis is to allow the use of appropriate decision making techniques in the data analysis stage. There are a number of data collection techniques such as examining the products and vendor supplied documentation, vendor analysis, viewing demonstration and interviewing demonstrators, executing test cases and applying the products in pilot projects. STACE proposes selecting appropriate techniques depending on resources and experience. STACE framework recommends the use of the AHP to consolidate evaluation data because of a number of advantages discussed in section 1.4.3.

Table 1. Social-technical criteria for GIS software selection

FUNCTIONALITY	**QUALITY ATTRIBUTES**
Data Capture (digitize)	Interoperability
Data Integration	Efficiency/Resource utilization
Projection and registration.	Usability
Data restructuring	**NON-TECHNICAL FACTORS**
Data and topological modelling	Vendor reputation
Information retrieval	User experience
Map overlays	Local support
Data Output	**COSTS ISSUES**
Internet support	Product cost
ODBC support	

5.3 Application of STACE Method

The STACE method was used by a public organization mandated to protect the environment and control pollution. The organization was established in 1992 with an annual budget of about $1million and employs over sixty persons, seven of which are in the IT department. The main application of IT in this organization is Geographic Information Systems (GIS). The organization was using standalone ArcInfo 4.2D and ArcView software while some of the users were trained in Idrisi. The organization installed a Local Area Network (LAN) and STACE method was used to select new GIS software in a multi-user LAN environment.

The workbook to operationalize the STACE framework was developed and used to guide the organization in evaluating and selecting COTS software. The workbook explicitly describes each stage of the STACE framework. The organization was invited to attend a workshop at which the STACE framework and workbook were presented and discussed.

STACE Process in Selecting a GIS Software. The following are the step by step description of the procedures used to evaluate and select the GIS software.

Step 1: Requirements definition. The sponsor and stakeholders from the organization were identified. It was agreed with the sponsor regarding the composition of the evaluation team and the resources required for the evaluation work. The high level user requirements were elicited from system documents, domain knowledge, and interviews with stakeholders.

Step 2: Social-technical criteria definition. The high level requirements were decomposed in social technical criteria. According to the STACE method the social-technical criteria include: 1) technology factors, 2) functionality characteristics, 3) product quality characteristics, and 4) social-economic factors. The technology criteria was not used in the hierarchy priority because technology was adopted as the keystone and therefore all the software to be selected must be compatible with the keystone in this case Windows 2000. The social-economic factors were divided into non-technical issues and cost issues because of the importance

Table 2. Relative importance of criteria

	Functionality attributes	Quality attributes	Non-technical	Costs issues	Relative importance
Functionality attributes	1	3	4	5	0.550
Quality attributes	1/3	1	2	2	0.214
Non-technical	1/4	1/2	1	2	0.142
Costs issues	1/5	1/2	1/2	1	0.094
Total					1.000

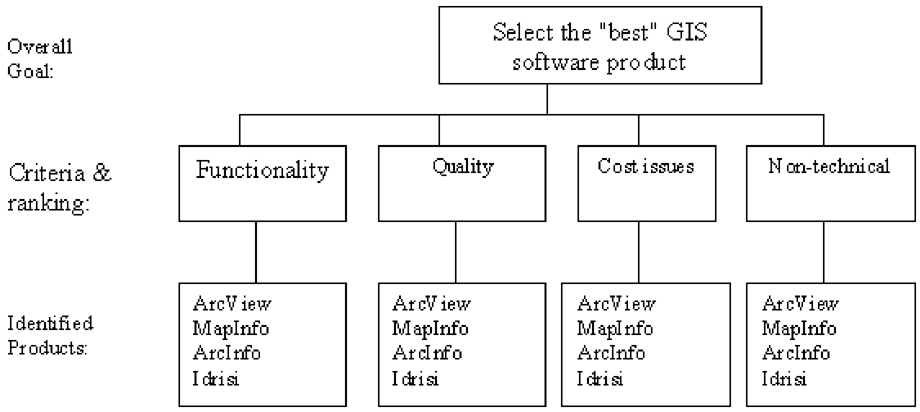

Fig. 3. Hierarchy of Criteria and GIS Products

separating cost issues from other attributes. The outcome of this process is the social-technical criteria presented in table 1.

Step 3: Using AHP to determine the relative importance of the criteria. Using pairwise comparisons, the relative importance of one criterion over another was computed. A total number of six pairwise comparisons were made to calculate the AHP's eigen vector values and these are shown in Table 2. The result in Table 2 shows that the functionality attributes is the most preferred criterion and cost issues is the least preferred criterion. Pairwise comparisons were also computed for the sub criteria to determine the relative importance of the sub criteria relative to the criteria. These are presented later in the chapter.

Step 4: Identify candidate software (vendors). The search for candidate products was conducted using the GIS user community; Internet search; vendor publications and sales promotions. The identified products were screened to reduce them to ArcView, MapInfo, ArcInfo and Idrisi (see Fig. 3). The basis of screening was the technology criteria and user experience with the products. It was required that the product must run on Windows 2000 and at least one user from within the organization must be familiar with the product.

Step 5: Evaluation and priority ranking of the candidate product. Evaluation copies of the candidate software were obtained. The evaluation involved con-

Table 3. Priority ranking of candidate product

	Relative importance	**ArcView**	**MapInfo**	**ArcInfo**	**Idrisi**
Functionality					
Data Capture	0.100	0.038	0.060	0.712	0.190
Integrate	0.100	0.222	0.057	0.681	0.040
Projection and registration.	0.100	0.109	0.042	0.666	0.182
Data restructuring	0.100	0.050	0.081	0.510	0.359
Data and topological modelling	0.100	0.050	0.081	0.510	0.359
Powerful information retrieval	0.100	0.597	0.251	0.061	0.090
Map overlays	0.100	0.034	0.063	0.684	0.219
Data Output	0.100	0.089	0.029	0.607	0.275
Internet	0.100	0.715	0.176	0.046	0.062
ODBC	0.100	0.090	0.295	0.567	0.048
Quality attributes					
Interoperability	0.540	0.169	0.451	0.261	0.119
Efficiency/Resource utilization	0.163	0.123	0.346	0.358	0.173
Usability	0.297	0.487	0.311	0.084	0.118
Non-technical factors					
Vendor reputation	0.250	0.286	0.286	0.286	0.143
User experience	0.500	0.472	0.108	0.256	0.164
Local support	0.250	0.372	0.150	0.372	0.106
Costs issues					
Product Cost	1.000	0.174	0.104	0.098	0.625

tacting vendor technical support for evaluation information, review of vendor documentation and experimenting with the product to assess its quality and functionality. It included evaluating product performance, interfaces and ease of integration, comparing short-term and long-term licensing costs. In addition data collection included interviewing actual users of the products, and examining sample outputs from projects that have used the products.

Having experimented with the software and reviewed documentation the alternatives were assessed against criteria, for example in terms of quality characteristics, pairwise comparisons are made to determine the preference of each alternative over another. The eigenvector were then calculated from these matrices and the result is shown in the Table 3. The table also provides the relative ranking of each sub criteria, for example each functionality sub criteria is of equal importance (0.100).

From table 3 above it can be concluded that regarding data capture ArcInfo is the most preferred product (preference=0.712) while ArcView is the least preferred (preference=0.038). The results also shows that ArcInfo is preferred in almost all the functionality except support for Information retrieval and support for Internet in which ArcView was preferred.

Step 6: Using AHP to synthesize the evaluation results and select the product. The priority ranking were then synthesized with the help of ExpertChoice, a

Table 4. Results of evaluation exercise

	Priority ranking of criteria	**ArcView**	**MapInfo**	**ArcInfo**	**Idrisi**
Functionality	0.550	0.274	0.110	0.480	0.136
Quality attributes	0.214	0.256	0.392	0.224	0.128
Non-technical factors	0.142	0.400	0.293	0.163	0.144
Costs issues	0.094	0.174	0.104	0.098	0.625
Overall of GIS software		**0.279**	**0.196**	**0.344**	**0.181**

software tool that supports AHP process and the results shown in Table 4. The table shows that ArcInfo is the recommended GIS software package for the organization. It can be noted from this table that although Idrisi scored highly regarding costs issues it did not emerge as the winning package because according to the organization cost issues had low ranking compared to functionality issues.

Experience in Using STACE. The organization pointed out that they found STACE framework useful because it addresses the non-technical issues and brought about decision support satisfaction. They argued that the use of AHP brought about confidence in the evaluation results and also promoted consensus in evaluation process. In addition, because the AHP provides an audit trial, it made the whole evaluation process transparent. However, the organization indicated that the AHP involved too many pairwise comparisons when the criteria increased, in their case they made over 100 pairwise comparisons for what they considered to be simple software selection process. Furthermore, they indicated that some aspects of AHP were subjective.

The organization indicated that stakeholder participation as advocated in the STACE framework is very important in COTS software evaluation and selection as it facilitates dialogue and consensus building with stakeholders. However, the organization indicated that the inclusion of stakeholder participation increases the cost of the software evaluation process.

6 Future Research

The STACE framework provides a classification of important processes (including traditional and soft factors) that support COTS software selection. The framework also allows the classification of a set of techniques and tools within each process. It highlights relationships between processes (and factors within each process) and thus facilitates the examination of relationships between factors in different processes and their impact on the success of COTS software selection process. The identification of important processes and factors supporting COTS software evaluation and selection has highlighted a number of areas that require further research. For example, future work can focus on the examination of each of the identified factor and its impact on the COTS software selection process. Further work can also investigate the relationships between the factors and the organizations, or draw conclusions about COTS component selection in different organizations.

The evaluation of the STACE framework revealed the problem of additional costs introduced by the inclusion of non-technical issues in the evaluation criteria and customer participation [21]. Furthermore, that the evaluation process takes a longer time because of additional work. A number of templates were provided to speed up the process and reduce the additional work of reinventing for evaluation criteria, each time the evaluation is done. However, the problem was not completely solved. Therefore, future work can focus on the development of a software tool to support all the processes in the STACE framework. The software tool would automate the evaluation process; suggest techniques and criteria according to the type of evaluation problem; management of past evaluation results in order to inform future evaluation cases; and support the use of a multi-criteria decision method. This would initially involve developing the prototype and then testing it in a number of organizations.

In the STACE framework, the AHP was proposed for consolidation of evaluation data because it incorporates both objective and subjective measures into decision making process. However, the AHP has a number of problems, for example that AHP has a potential for bias and subjectivity especially when dealing with non-technical issues. Furthermore, that the AHP is time consuming because of the mathematical calculations and the number of pairwise comparisons that increases as the number of alternatives and criteria increases. Therefore, further work can focus on developing a software tool that supports the AHP and addresses some of these problems. Future work can also investigate the use of other multi-criteria techniques, such as outranking.

This chapter focussed on COTS software evaluation and selection supporting the Component-based software development. However, this has implications for the other stages of the software development cycle. For example, systems built using COTS packages will require maintenance and enhancements, some prompted by vendor updates and changing customer requirements. Therefore, determining procedures or guidelines for deciding when to accept upgrades would be an interesting research area.

7 Conclusion

Component-based development is a process of building software systems by integrating multiple software that are ready “off-the-the shelf” whether from a commercial source (such as COTS) or re-used from another system. Building systems from COTS software components offers the opportunity to lower costs by sharing them with other users and has potential for reduced training and infrastructure costs. Therefore, by employing this strategy, organizations will not spend too much time on developing expensive systems, with only one customer to bear the development and maintenance costs over the life of the system.

Building of systems from COTS software depends on successful evaluation and selection of COTS software to meet customer requirements. COTS software evaluation and selection ensures that a candidate component will perform the functionality required and will exhibit the quality characteristics (e.g. performance, reliability, usability) that are required. A number of problems associated with COTS software evaluation and selection have been identified including rapid changes in market place; lack of well-defined process; "black box" nature of COTS components; and misuse of data consolidation method. COTS software evaluation and selection frameworks have been developed aimed at addressing these problems, for example the OTSO framework; Delta technology framework and PORE framework.

However, what is missing in these frameworks is the "soft" issues or the non-technical issues such as costs, organizational issues, vendor capability and reputation. Furthermore, these frameworks do not provide a means for identifying and classifying important processes and factors supporting COTS software selection for practical use. Therefore, the STACE framework was developed. The STACE framework addresses the weaknesses of the existing evaluation models by attempting to ensure that all issues (including non-technical) are structured and addressed. This is achieved by integrating social-technical criteria as well as customer participation in the COTS software evaluation process. Customer participation provides for consensus building during evaluation by allowing evaluators and stakeholders to discuss and agree on evaluation parameters. The framework also provides a structured evaluation model; thus allowing the designation of a hierarchy of selection criteria based on organizational needs. The evaluation and selection process is clearly defined, in terms of processes, techniques to be used and activities to be performed.

The STACE framework provides a classification of important processes (including traditional and soft factors) that support COTS software selection. The framework also allows the classification of a set of techniques and tools within each process. It highlights relationships between processes (and factors within each process) and thus facilitates the examination of relationships between factors in different processes and their impact on COTS-based systems success. Therefore, the framework could be used for research purposes not only in COTS software evaluation research but also in the wider software engineering research field.

References

1. Anderson E., "A heuristic for software evaluation and selection," Software Practice and Experience, Vol. 19, No. 8, pp. 707–717, August 1989.
2. Axtell C. M., Waterson P. E. and Clegg C. W. (1997), "Problems integrating user participation into software development," International Journal of Human-Computer Studies, Academic Press Limited, Vol. 47, pp 323–345.
3. Beus-Dukic, L. and Wellings A., "Requirements for a COTS software component: a case study," Conference on European Industrial Requirements Engineering (CEIRE '98), Requirements Engineering, Springer-Verlag, Vol.3, No.2, pp. 115–120, October 1998.
4. Boloix, G. and Robillard, P. (1995), "A Software System Evaluation Framework," IEEE Computer, Vol. 28, No. 12, pp. 17–26.

5. Bravo E., "The Hazards of leaving out users," In Participatory Design: Principles and Practices, Schuler D. and Namioka A. (eds.), Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 3–12, 1993.
6. Brown A. W. and Wallnau K. C., "A Framework for Systematic Evaluation of Software Technologies," IEEE Software, Vol. 13, No. 5, pp. 39–49, 1996a.
7. Brown A. W. and Wallnau K. C., "Engineering of Component-Based Systems," in Brown A. W. (ed.), "Component-based Software Engineering: Selected papers from Software Engineering Institute," IEEE Computer Society Press, Los Alamitos, California, pp. 7–15, 1996b.
8. Carney D. J. and Wallnau K. C., "A basis for evaluation of commercial software," Information and Software Technology, Vol. 40, pp. 851–860, 1998.
9. Clements P. C. (1995) From Subroutines to Subsystems: Component-Based Software Development, American Programmer 8(11), Cutter Information Corp.
10. Coppit D. and Sullivan K. J., "Multiple mass-market applications as components," Proceedings of the International conference of Software Engineering (ICSE), IEEE computer society, Los Alamitos, California, pp. 273–282, 2000.
11. Dyer J. S., "Remarks on the Analytical Hierarchy Process," Management Science, Vol. 36, No. 3, pp. 249–259, 1990.
12. Finkelstein A., Spanoudakis G. and Ryan M. (1996), "Software Package Requirements and Procurements," Proceedings 8th International Workshop on Software Specification and Design, IEEE Computer Society Press, pp. 141–145.
13. Haines G., Carney D. and Foreman J., "Component-Based Software Development/ COTS Integration," Software Technology Review, Software Engineering Institute, `http://www.sei.cmu.edu/str/descriptions/cbsd_body.html`, 1997.
14. Harker P. T. and Vargas L. G., "Reply to – Remarks on the Analytical Hierarchy Process – by J. S. Dyer," Management Science, Vol. 36, No. 3, pp. 269–273, 1990.
15. Hokey M., "Selection of Software: The Analytic Hierarchy Process," International Journal of Physical Distribution and Logistics Management, Vol. 22, No. 1, pp. 42–52, 1992.
16. IEEE std 1209–1992 (1993), IEEE Recommended Practice for the Evaluation and Selection of Case Tools, IEEE, New York.
17. ISO/IEC 9126: 1991, "Information technology-Software product evaluation-Quality characteristics and guidelines for their use," ISO/IEC, Geneva, 1991.
18. Jirotka M. and Goguen J. A. (eds.) (1994), "Requirements Engineering social and technical issues," Academic Press Limited, London.
19. Kontio, J. (1996), "A Case Study in Applying a Systematic Method for COTS Selection," Proceedings of the 18th International Conference on Software Engineering (ICSE), IEEE Computer Society.
20. Kunda, D. and Brooks L. (2000), "Identifying and Classifying Processes (traditional and soft factors) that Support COTS Component Selection: A Case Study," European Journal of Information Systems, Vol. 9, No. 4, pp. 226–234, 2000.
21. Kunda, D. (2002), "A social-technical approach to selecting software supporting COTS-Based Systems," PhD dissertation, University of York.
22. Maiden N. A. and Ncube C. (1998), "Acquiring COTS Software Selection Requirements," IEEE Software, pp. 46–56.
23. McDermid J. A., "The Cost of COTS", IEEE Computer, Vol. 31, No. 6, pp. 46–52, 1998
24. Mollaghasemi M. and Pet-Edwards J., "Technical briefing: making multiple-objective decisions", IEEE computer society press, Los Alamitos, California, 1997.

25. Morisio M., and Tsoukis A., “IusWare: a methodology for the evaluation and selection of software products,” IEEE Proceedings of Software Engineering, Vol. 144, No. 3, pp. 162–174, June 1997.
26. Mumford E. (1995), Effective Systems Design and Requirements Analysis: The ETHICS Approach, Macmillan Press Ltd, Hampshire.
27. Oberndorf P. (1997) Facilitating Component-Based software Engineering: COTS and Open Systems, Proceedings of the Fifth International Symposium on Assessment of Software Tools, IEEE Computer Society, Los Alamitos, California.
28. Oberndorf P. A., Brownsword L. and Morris E., “Workshop on COTS-Based Systems,” Software Engineering Institute, Carnegie Mellon University, Special Report CMU/SEI-97-SR-019, November 1997.
29. Powell, A., Vickers, A. and Lam, W. (1997), “Evaluating Tools to support Component Based Software Engineering,” Proceedings of the Fifth International Symposium on Assessment of Software Tools, IEEE Computer Society, Los Alamitos, pp. 80–89.
30. Puma Systems, Inc., “Commercial-Off-The-Shelf System Evaluation Technique (COSSET),” http://www.pumasys.com/cosset.htm, March 1999.
31. Rowley J. E. “Selection and Evaluation of Software,” Aslib Proceedings, Vol. 45, pp. 77–81, 1993.
32. Roy, B., “The Outranking Approach and the Foundations of ELECTRE Methods,” Theory and Decision, Vol. 31, pp. 49–73, Kluwer Academic Publishers, Netherlands, 1991.
33. Saaty, T. L. (1990), The Analytic Hierarchy Process, McGraw-Hill, New York.
34. Sledge C. and Carney D., “Case Study: Evaluating COTS Products for DoD Information Systems,” SEI Monographs, http:www.sei.cmu.edu/cbs/ monographs.html, July 1998.
35. Tran, V., Liu D. and Hummel B., “Component-based systems development: challenges and lessons learned,” Proceedings of the Eighth IEEE International Workshop on Software Technology and Engineering Practice incorporating Computer Aided Software Engineering, IEEE Computer Society, Los Alamitos, California, pp. 452–462, 1997.
36. Vigder M. R. and Dean J. (1997), Architectural Approach to Building Systems from COTS Software, Proceedings of the 1997 Center for Advanced Studies Conference (CASCON 97), Toronto, Ontario.
37. Vigder M. R., Gentleman W. M. and Dean J. (1996) COTS Software Integration: State of the art, National Research Council, Canada, NRC Report Number 39198.
38. Vincke P., “Multicriteria decision-aid,” Wiley publishing, Chichester, 1992.
39. Walters N., “Systems Architecture and COTS Integration,” Proceedings of SEI/MCC Symposium on the use of COTS in systems Integration, Software Engineering Institute Special Report CMU/SEI-95-SR-007, June 1995.
40. Williams F., “Appraisal and evaluation of software products,” Journal of Information Science, Principles and Practice, Vol. 18, pp. 121–125, 1992.
41. Yoon, K. and Hwang C. (1995), “Multiple Attribute Decision-Making: an Introduction,” Sage Publisher.
42. Zviran, M. (1993), “A Comprehensive Methodology for Computer Family Selection,” Journal of Systems Software, Vol. 22, pp 17–26.