A Taxonomy of Processes for Component Selection

Wes J. Lloyd, Sudipto Ghosh, James Bieman
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
{wlloyd, ghosh, bieman}@cs.colostate.edu

Abstract

Components provide the building blocks for developing and delivering software systems in less time and with richer functionality than systems built using traditional software development practices. One challenge associated with component-based software development is the process of selecting the best component to realize required functionality. Component selection processes help software developers evaluate and make component selection decisions. This paper presents a taxonomy of existing component selection processes. Our evaluation of existing component selection processes finds that the processes are very similar in that they try to reduce evaluation effort, improve decision-making and provide order and repeatability to component selection through similar process activities. With many processes making similar contributions analysis suggests the need to focus future process improvement efforts on component evaluation. Improving the accuracy and efficiency of component measurement should help to improve the effectiveness of component selection beyond the improvements realized by existing processes.

1. Introduction

Component Based Software Development (CBSD) involves the use of preexisting software components to realize the functional requirements of a software system. By using components, software development organizations hope to reduce the overall cost of development and the total development time. Component based software development processes typically begin with a requirements phase where system requirements are identified. Next a search is conducted to identify potential components available that meet system requirements. In order to realize benefits from using CBSD practices, developers need to rapidly identify potential components, and then evaluate and select the best component(s) among the candidates for use in the software system. The problem of identifying available components and selecting the most appropriate one is known as the *component selection problem.*

The component selection problem involves selecting the most appropriate component from available alternatives to implement specific software requirements. Component selection aims to select the best component that will help to reduce the cost, and time-to-market for the software project [17]. Errors in component selection can reduce the benefits of using component based software development.

Component selection decisions are often made in an ad-hoc manner [6,9]. Component selection processes are proposed to improve upon the efficiency and effectiveness of ad-hoc methods. We evaluate several existing component selection processes that aim to make component selection decisions repeatable. This paper proceeds to identify the common steps of component selection processes, and then defines common attributes that affect component selection decisions. An evaluation of component selection processes is presented which focuses on identifying similarities and differences among decision-making techniques, methods to reduce component evaluation efforts, and process activities. The paper concludes by identifying future work towards improving component selection by considering the shortcomings of processes evaluated.

2. Common Steps in Component Selection

Many processes have been presented in software engineering literature to help with making component selection decisions [1, 6-19]. Component selection processes typically define four primary steps as shown in Figure 1. The first step is to identify the basic functional requirements of the component. This step is often accomplished in conjunction with the requirements analysis of the software system being built. System requirements are identified and then grouped with the expectation that a single software component will provide implementation. For example a graphical scheduling application identifies

the need to display a calendar with a month-based view. In addition to simply displaying a month, the application will require that the user be able to interact with the month in order to select dates, ranges of dates, navigate forward, backward, and so on. It is desirable that a single calendar component be used to implement all requirements associated with the display and manipulation of the calendar month.

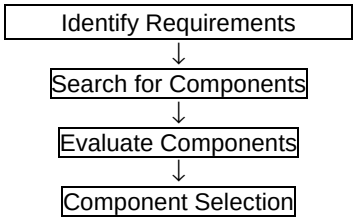

Figure 1. Generalized Component Selection Process

The second step of the generalized component selection processes is to search for components that provide implementation of the functional requirements identified in step one. This step represents the first stage of filtering, where some potential component candidates are eliminated from consideration. For the graphical scheduling application the requirement of multiple selection of dates is defined. Any calendar component that does not provide this feature is eliminated from consideration during step two.

The third step in the generalized component selection process is to perform evaluation and testing of the component candidates that were identified in step 2. Various criteria that are considered important for the component's use in the software system are evaluated using software metrics. A score is then computed based on how well the component performed through the application of various measurements. For the graphical scheduling application one criterion that may be considered for the selection of a calendar component is the complexity of the integration of the component. To predict the effort required for integration the related attribute of component understandability might be measured. Whether measures of understandability accurately describe the complexity of integration is a research question that is beyond the scope of this paper.

The fourth step of the generalized component selection process involves the selection of a component through the application of a decision making model. For the graphical scheduling application example the best component is identified after applying various tests to component candidates. The component that performed the best throughout the evaluation phase is usually selected.

3. Factors affecting Component Selection

Component attributes are the properties of the component that are considered in making component selection decisions. Attributes can be divided into two categories as shown in Table 1: internal attributes, and external attributes. Internal attributes deal with aspects of the component's implementation. Attributes include the component architecture, functional, and non-functional requirements. In general components are architecture specific. For example they may be built using COM, CORBA, or Javabeans. Component selection typically restricts choices to those of the same architectural type. Functional requirements describe what the component does. Non-functional requirements describe aspects of the component's implementation other than the mere functions performed [7,9,10]. Non-functional requirements include complexity, performance, usability, understandability, reliability, maintainability, testability, and documentation quality. Kunda and Brooks refer to non-functional requirements as product quality factors [9]. Their case study measures aspects of seven ad-hoc component selection processes in use at different organizations. The most common non-functional requirements identified in the case study included: interoperability (5 of 7), performance (4 of 7), reliability (4 of 7), and usability (4 of 7). Kunda and Brooks refer to external attributes as non-technical factors [9,10]. Aves and Finkelstein call them vendor issues [2]. External attributes include properties such as component cost, business viability of component vendor, licensing requirements, and quality of developer support.

Table 1. Component Attributes Affecting Selection

Internal Attributes	*External Attributes*
-Functional Requirements -Component Architecture -Complexity -Performance -Usability -Understandability -Reliability -Maintainability -Testability -Documentation Quality	-Component Price -Business Viability of Component Vendor -Licensing Requirements -Source Code Availability -Quality of Technical Support

Attributes presented in Table 1 are not an exclusive set of all attributes that could affect component selection. Many other internal and external attributes can exist that play a role in component selection.

Component evaluation should consider both short-term and long-term factors. Short-term factors include those that immediately affect the cost of making a selection decision. Some short-term factors

include component cost, component functionality, understandability, and testability. Some questions that arrive in relation to short term factors affecting component selection include:

Component Cost
- Is the component's price competitive?

Component Functionality
- Does the component meet the current set of functional requirements?

Understandability
- Is the component sufficiently easy to understand from an integrator's point of view?
- Does component documentation provide sufficient information to allow for easy integration into the system?
- Does the component provide unnecessary features that complicate the component making it more difficult to understand from an integrator's point of view?

Testability
- How testable is the component?
- Is source code available to enable white-box testing techniques?
- Is documentation sufficient to enable black-box and integration testing?

Long-term factors affecting component selection are primarily concerned with maintainability. In making a component selection it is desirable to predict future maintenance costs. Maintainability questions to consider regarding long-term factors in component selection include:

Vendor Viability
- What is the business viability of the component vendor?
- Will the vendor still exist in a few years to support the component?
- Are bug fixes available for the component?
- Is source code available for the component?
- In the event of a failure of the component vendor is there a risk mitigation strategy?

Understandability
- Is the component integration sufficiently easy to understand from a maintainer's point of view?
- Does the component have a complex interface that may complicate future software maintenance?
- Is the component's documentation sufficient to enable system maintainers to understand how the component is used?

Component Updates
- Does the component provide many unnecessary features that make the component overly complex and more difficult to understand from a maintainer's point of view?

Extensibility
- Does the component provide support for features extending beyond the basic functional requirements that may be of interest in the future?

Component Cost
- Is there a cost associated with ongoing vendor support of the component?
- Are updates available free or charge?

4. Component Selection Processes

Most component selection processes usually include methodologies for component evaluation, decision-making, and component measurement. This paper evaluates seven component processes according to these criteria.

Kunda and Brooks define three types of component evaluation methods [9,10]. Progressive Filtering involves iterating through component evaluation cycles adding more discriminating criteria for each cycle's iteration to eliminate less appropriate candidates. The Keystone selection strategy initially identifies a key component requirement. During the component search phase the lack of support of a keystone characteristic quickly eliminates components that do not implement the characteristic. Puzzle Assembly assumes that a valid COTS system requires fitting the various components of the system together as pieces of a puzzle.

Decision-making considers how to select the best component available given the choices. Typically a form of averaged summation of overall factors affecting the component selection decision is used. Two decision-making strategies commonly used are: Analytic Hierarchy Process (AHP) and the Weighted Sum Method (WSM) [20,21].

Component evaluation relies on the use of both qualitative and quantitative measures to assess component attributes. Qualitative measures are based on opinions from observations and perceptions about the function and quality of the component. For example a developer might rank the usability of a set of components based on their perceptions of usability after ad hoc testing. Quantitative measures generate numeric data based on concrete observations of the component. For example consider a metric that measures component interface size. The size of the component's application program interface (API) can be precisely determined by counting the number of methods, parameters, etc.

4.1. CAP

The COTS Acquisition Process (CAP) assumes a fixed set of system requirements before searching for components [15,16]. A 3-step evaluation scheme is

applies component measurements in a cyclic approach. The first phase uses metrics and measurements that are the least expensive and time-consuming to calculate. Some components are eliminated by the initial phase, but remaining components are evaluated using more stringent metrics in each of the two remaining phases. Each phase applies additional measurement leading to the elimination of inferior component candidates. This approach can be considered a progressive filter evaluation method. During the evaluation phases CAP uses the Analytic Hierarchy Process to assign rankings to the candidate components. CAP identifies (120) software metrics from several sources including about (60) from the ISO9126 standard on product quality [4]. Remaining metrics are from interviews, literature reviews, and applied research activities. Noted that the ISO9126 metrics predates the era of component-based development. The metrics, although applicable to components were not developed with component assessment in mind. Before using the ISO9126 metrics to measure component attributes it may be worthwhile to consider how applicable they are for making component measurements. CAP does not consider the efficiency of the metrics at evaluating component attributes, although progressive filtering reduces the number of measurements that need to be made.

4.2. STACE

The Social-Technical Approach to COTS Evaluation (STACE) employs the keystone identification strategy with the underlying technology, the component framework, as the keystone issue [9,10]. Components that do not operate within a particular component framework (CORBA, COM, etc.) are quickly eliminated from consideration. A variety of evaluation techniques are employed by STACE. The selection of appropriate techniques is based on resources and experience available. AHP is used to consolidate evaluation data in order to generate the best component selection decisions. STACE does not specify metrics for the evaluation of components. Any measurements that are convenient and easy to use are suggested. Contacting the vendor for technical support, performance testing, functionality testing and comparison of costs associated with different vendors are suggested activities for acquiring data about components.

4.3. CEP

The Component (Comparative) Evaluation Process (CEP) first identifies component candidates by only considering those that provide the minimal required system functionality [17]. Next minimum thresholds are applied to filter remaining candidates. This incremental screening of components allows the complete evaluation of all components to be avoided. This application of thresholds for filtering is a form of progressive filtering. Component measurement then centers on: functional, architectural, management, strategic, and performance attribute measurements. Specific metrics are not mentioned. CEP suggests several observational techniques to gather component data including: hands-on experience, witnessing vendor demonstrations, observing a user, and reading product documentation. To compile data, simple weighted averages are used. For weighting purposes evaluation criteria are divided into two levels. The first hierarchical level represents the category of evaluation attribute such as non-functional requirements or vendor issues. The second level represents individual attributes within the category. Examples of non-functional attributes are usability and maintainability. Both the individual attributes and the categories are weighted. For each attribute a global weighted score is then computed by multiplying the category weight with the attribute weight. The total global weight is 1.00, or 100% for all attributes. Weighting an entire category (vendor issues, functional requirements, etc.) reduces the granularity possible for individual attributes. It is highly likely that a non-functional requirement such as maintainability would play an important role, whereas another non-functional requirement may be insignificant. This appears to be a disadvantage with CEP's approach to weighting.

4.4. PORE

The Procurement-Oriented Requirements Engineering process (PORE) is more than just a component selection process [12,13,14]. It encompasses other activities of the component-based software development process including selection of component vendors, developing contracts with vendors, and acceptance testing of vendor products. PORE proposes using models of candidate components and system requirements for analysis to enable requirements acquisition and product selection. PORE uses an iterative process for the elicitation of component requirements. Through process iterations of PORE, requirements are elicited, and candidate components are eliminated from consideration. This iterative approach is a progressive filter. When all candidate products meet customer requirements, PORE states "Requirements that enable discrimination between the COTS candidate products must be elicited [13]." Rather than weighting which component better meets non-functional requirements such as performance, understandability, and maintainability, PORE seeks to elicit additional requirements through process

iteration to eliminate candidate components. PORE suggests the use of AHP in the event that only a small number of candidate components exist. PORE defines (3) template processes for acquiring component information based on available sources. PORE templates are defined for evaluating components using supplier-given information, supplier-led demonstrations, and customer-led product evaluation. Component measurement is conducted by PORE, but specific measurement techniques are only briefly mentioned. PORE suggests the use of Kitchenham and Jones' feature analysis techniques [5].

4.5. OTSO

The Off-The-Shelf option (OTSO) is a systematic approach to evaluating component candidates [6,7,8]. OTSO applies an incremental evolutionary definition of evaluation criteria. OTSO uses a progressive filtering approach that is limited to three steps in which stronger requirements are applied to eliminate candidate components. During the search phase potential candidates that meet basic criteria are identified. The screening phase filters the candidates through the use of stronger thresholds than the search phase. The evaluation phase evaluates remaining candidates. The importance of each selection criteria is considered before conducting measurement [8]. This is a time and cost saving measure similar to that of the CAP process that is required for component evaluation to be completed in a reasonable amount of time. OTSO uses AHP to evaluate results of component evaluation and make component selection decisions. OTSO does not state specific techniques for measurement of component attributes. Ancillary assessment activities are suggested including: obtaining the components, installing them, learning how to use them, and studying their features.

4.6. BAREMO

The Balanced Reuse Model (BAREMO) is not a selection process that enumerates the typical lifecycle steps for component selection [11]. Activities such as component search and requirements identification are not defined. BAREMO instead presents an adaptation of the analytic hierarchy process for decision-making. AHP allows people to gather knowledge about a particular problem, quantify subjective information and enable the comparison of alternatives in relation to established criteria. [11]. BAREMO defines AHP steps specific for component selection. The BAREMO process defines these steps: (1) specification of project objectives, (2) construction of a decision tree, (3) generation of comparison matrices with criterion from the decision tree, (4) assessment of characteristics of each component being evaluated, (4.1) establishment of scales for ranking criterion, (4.2) evaluation of criteria using established scales, (5) Calculation of a final value for each component using weighted addition scales. BAREMO does not define a component evaluation mechanism such as other processes to minimize the number of measurements for assessing components. In addition specific metrics and techniques to measure component attributes are not specified.

4.7. CRE

COTS Based Requirements Engineering (CRE) uses a progressive filtering of components through a three-stage process for component evaluation [1,18]. In the first stage core requirements are identified and candidate components are identified. In the second stage further component requirements are identified to refine component requirements. The Non-Functional Requirements (NFR) Framework is used to decompose nonfunctional requirements into sub-requirements that can be more easily evaluated. Candidate components are eliminated as they fail to meet the requirements identified in stage two. In the third stage the remaining components are evaluated. The COCOTS (COnstructive COTS) cost model is suggested for cost vs. benefits analysis of candidate components. Once data is available the weighted scoring method is suggested for simple decisions, or for more complex decisions AHP can be used. Overall CRE is similar to PORE in that both processes focus on defining additional requirements to eliminate components from selection. Both models approach component selection through rejection of candidates.

5. Discussion

Methods to reduce the component evaluation task are summarized in table 2. Progressive Filtering was the most common method for reducing the component evaluation task. Five of seven processes specified the use of filtering to reduce the set of candidate components before proceeding with a full analysis using all measurements. PORE and CRE both elicited additional requirements iteratively to effectively reject candidate components. STACE identified keystone requirements such as the component architecture and eliminated candidates based on non-compliance. BAREMO did not specify any method to reduce the component evaluation task. Puzzle Assembly was not used by any of the component selection processes.

The processes evaluated used two decision-making methods. Table 3 summarizes the decision-

making methods used by the processes evaluated. Nearly all processes suggested or formally describe the use of the Analytic Hierarchy Process (AHP) for multi-variable decision-making. CEP specified the use of the weighted sum method and CRE specified the use of both methods.

Table 2. Component Evaluation Methods

	Progressive Filtering	Keystone Selection	Puzzle Assembly
CAP	X*		
STACE		X	
CEP	X*		
PORE	X^R		
OTSO	X*		
BAREMO			
CRE	X^R		

* - Indicates that filtering has a predefined number of steps
R - Indicates that filtering is accomplished through addition of requirements

Table 3. Decision Making Method

	WSM	AHP
CAP		X
STACE		X
CEP	X	
PORE		Weak
OTSO		X
BAREMO		X
CRE	X	X

Table 4 presents the generalized steps of the seven component selection processes. Some processes define many small steps that are not listed in the table. By comparing steps among the processes similarities can be seen. CAP, STACE, CEP, OTSO and CRE each define the process steps of Requirements identification, Component search, Component evaluation, and Component selection, which are identified in the generalized component selection process shown in section 2. These processes are fairly similar except for the additional activities they define.

The component selection processes reviewed, in general do not elicit specific methods for measuring component attributes. Most of the processes did make suggestions on ways to evaluate components but they did not explicitly define methods for evaluating each of the attributes identified in section 3. CAP provided the most guidance by identifying the use of software metrics from the ISO9126 standard. In total CAP identifies 120 metrics for component evaluation. PORE proposes the use of Kitchenham and Jones' feature analysis techniques. CRE proposes the use of COCOTS to perform a cost benefits analysis. OTSO suggests qualitative assessment of components through manual study and analysis. CEP suggests qualitative measurements through observations of hands-on use, witnessing vendor demonstrations, and examining product documentation. STACE suggests using cost effective measurements to obtain data, but does not specify specific measures.

Table 4. Component Selection Process Steps

Component Selection Process	Process Activities
CAP*	Identify component selection criteria, Estimate measurement effort, Define measurement plan, Review plan, Search for components, Component evaluation, Component selection, Review selection, Document evaluation information for reuse
STACE*	Requirements definition, Social-technical criteria definition, Search for Components, Evaluate Components, Component selection
CEP*	Estimate measurement effort, Search and screen candidate components, Identify component selection criteria, Evaluate component alternatives, Analyze evaluation results, Component selection
PORE	Requirements definition, Supplier selection, Component selection, Contract production, Component acceptance. Ongoing activity: Manage selection process to control time/costs
OTSO*	Identify component selection criteria, Search for components, Screen components, Evaluate components, Analysis of results, Component selection
BAREMO	Specify project objectives, Construct decision tree, Generate comparison matrices, Assess component characteristics, Establish numeric scales for criteria comparison, Assign values based on scale, Calculate final values for components, Component selection
CRE*	Identify component selection criteria, Elicit component requirements, Evaluate components, Component selection, Component acceptance

* - Indicates process includes steps of the generalized component selection process

A key challenge to component selection is how to acquire the necessary information about candidates in order to compare them. Test and performance data could be vendor supplied through the use of formal specifications [3]. However there is no agreed upon standard for specifying such information and any vendor-supplied information that is available through documentation and product help files is rarely sufficient in providing enough information to make selection decisions. Developers often must perform

their own evaluation of component candidates to acquire information to make selection decisions. Evaluation techniques suggested by several component processes evaluated here rely on qualitative assessment and observation. STACE suggests to "select appropriate (measurement) techniques depending on resources and experience" [9]. Such an approach suggests the use of convenient measures as opposed to accurate ones.

6. Conclusion and Future Work

In general, the component selection processes evaluated in this paper identify ways to reduce the component evaluation effort, improve the analytical decision making process of component selection, and organize the component selection process into an ordered sequence of activities. The processes evaluated appear very similar. Five of the seven processes identify the same core process steps. Five of the seven processes use a form of progressive filtering to reduce the component evaluation task. Six of the seven processes use, or suggest the use of AHP as a decision-making technique. However the component selection processes evaluated do not specify explicit methods to measure and compare components quantitatively. In this regard the processes are vague about which techniques to use for component evaluation. In order for component selection to become a repeatable activity, with opportunities for reuse and process optimization after each iteration of the process, more specific methods for component evaluation should be defined.

Future work on component selection processes should focus on the identification and development of accurate and cost effective metrics to quantify component attributes. The development of a common suite of metrics is desired. A suite of metrics could include both qualitative and quantitative metrics for evaluating component attributes. Metrics in the suite could be evaluated to assess their ability to accurately measure specific component attributes. Given a suite of component metrics, any component selection process that needs to measure a particular component attribute could use metrics from the suite. Existing metrics, such as those in the ISO9126 standard and others could be adapted to the task of evaluating components. Ultimately the development of automated tools for performing component measurement is desired to reduce evaluation time and improve accuracy of measurement. It is generally accepted that manual evaluation of component attributes is an expensive and time-consuming endeavor.

By using component selection processes for component based software development, software developers hope to realize improvements in component selection through cost effective component evaluation and decision-making. Existing component selection processes define repeatable processes with optimizations to reduce evaluation effort and improve decision-making. One of the largest costs of a component selection process is component evaluation and measurement. Improvements and automation of component measurement is desired to further improve the process of component selection for component based software development.

7. References

[1] Alves, C., Castro, J., CRE: A Systematic Method for COTS Components Selection. XV Brazilian Symposium on Software Engineering (SBES) Rio de Janeiro, Brazil, 2001.

[2] Aves, C., Finkelstein, A., Challenges in COTS Decision-Making: A Goal-Driven Requirements Engineering Perspective, in Proceedings of the 14th international conference on Software Engineering and Knowledge Engineering (SEKE) 2002, Ischia, Italy, 2002, pp. 789-794

[3] Edwards, S., Toward Reflective Metadata Wrappers for Formally Specified Software Components, In Proceedings of the Workshop on Specification and Verification of Component Based Systems held in conjunction with OOPSLA, October, 2001

[4] ISO/IEC 9126 Standard, Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their Use, International Organization for Standardization (ISO), Geneva, 1991.

[5] Kitchenham, B., Jones, L., Evaluating Software Engineering Methods and Tools: Part 5, The Influence of Human Factors, Software Engineering Notes, Vol. 22, No.1, 1997.

[6] Kontio, J. A Case Study in Applying a Systematic Method of COTS Selection, in Proceedings of the 18th International Conference on Software Engineering, Berlin, Germany, 1996.

[7] Kontio, J., Caldiera, G., Basili, V., Defining Factors, Goals and Criteria for Reusable Component Evaluation, in Proceedings of the 1996 CASCON Conference, Toronto, Canada, 1996.

[8] Konito, J., Chen, S., Limperos, K., Tesoriero, R., Caldiera, G., Deutsch, M., A COTS Selection Method and Experiences of Its Use, NASA Software Engineering Laboratoty, Greenbelt, MD, 1995.

[9] Kunda, D; Brooks, L. Applying Social-Technical Approach for COTS Selection, Proceedings of 4th UKAIS Conference, University of York, McGraw Hill, 1999.

[10] Kunda, D., Brooks, L. Identifying and Classifying Processes (traditional and soft factors) that Support COTS

Component Selection: A Case Study. Proceedings of the 8th European Conference on Information Systems, Vienna, Austria, 2000

[11] Lozano-Tello, A., Gomez-Perez, A., BAREMO: How to Choose the Appropriate Software Component Using the Analytic Hierarchy Process, in Proceedings of the 14th international conference on Software Engineering and Knowledge Engineering (SEKE) 2002, Ischia, Italy, 2002, pp. 781-788.

[12] Maiden, N., Ncube, C., Acquiring COTS Software Selection Requirements. IEEE Software March/April, 1999, pp. 46-56.

[13] Ncube, C., Maiden, N., Guiding Parallel Requirements Acquisition and COTS Software Selection, in Proceedings of the IEEE Symposium on Requirements Engineering. Limerick, Ireland, 1999, 133-140.

[14] Ncube, C., Maiden, N., PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm, in Proceedings of the International Workshop on Component-Based Software Engineering held in conjunction with ICSE'99, Los Angeles, CA, May 1999.

[15] Ochs, M.A.; Pfahl, D.; Chrobok-Diening, G. ; Nothhelfer-Kolb, B. A COTS Acquisition Process: Definition and Application Experience, in Proceedings of the 11th ESCOM Conference, Shaker, Maastricht, 2000. pp. 335-343.

[16] Ochs, M.A.; Pfahl, D.; Chrobok-Diening, G.; Nothhelfer-Kolb, B. A Method for Efficient Measurement-based COTS Assessment and Selection – Method Description and Evaluation Results, in Proceedings of the 7th International Software Metrics Symposium, London, England, 2001. pp. 285-297..

[17] Phillips, B.C., Polen, S.M. Add Decision Analysis to Your COTS Selection Process, Crosstalk The Journal of Defense Software Engineering, April 2002.

[18] Rosa, N., Alves, C., Cunha, P,. Castro, J., Justo, G. "Using Non-Functional Requirements to Select Components: A Formal Approach." In Fourth Workshop Iberoamerican on Software Engineering and Software Environment (IDEAS'01), San Jose, Costa Rica, April 2001.

[19] Ruhe, G., Intelligent Support for Selection of COTS Products, in Proceedings of the Net.Object Days 2002, Erfurt, Springer, 2003, pp. 34-45.

[20] Saaty, T.L., The Analytic Hierarchy Process, McGraw-Hill, New York, 1990.

[21] Saaty, T.L., Decision Making for Leaders, Lifetime Learning Publications, Belmont, California, 1982.