CS-TR-3478 December 1995
UMIACS-TR-95-63

OTSO:
A Systematic Process for Reusable Software Component Selection

Jyrki Kontio[*]
Institute for Advanced Computer Studies and
Department of Computer Science
University of Maryland
College Park, MD 20742, U.S.A.
Email: jkontio@cs.umd.edu

Version 1.00
Status: Final

Abstract:

This paper presents a method for evaluating and selecting off-the-self software components to be reused in software development. The paper describes the main motivation and principles of the method and provides a provides a detailed description of it. The method has been tried out in two case studies and initial results of these studies are reported.

[*] This work has been supported by the Hughes Information Technology Corporation and the EOS Program.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation

Reuse has been considered an important solution to many of the problems in software development. It has been claimed to be important in improving productivity and quality of software development [1,9,24,35,41,45] and significant benefits have been reported by many organizations [21,34]. A large volume of research has produced several useful tools to support reuse [24,41,44] but it is widely believed that successful reuse is not only dependent on technical issues, it also requires the solving of organizational, motivational and legal issues [4,5,45-47]. It has been argued that an important characteristic of the infrastructure supporting reuse is the existence of a "marketplace" that both provides access to reuse producers and consumers as well as provides a mechanism to transfer benefits between the parties [11,33,34,48].

Many organizations have implemented systematic reuse programs [21] which have resulted in in-house libraries of reusable components. The increased commercial offering of embeddable software components, standardization of basic software environments (e.g., MS-Windows, UNIX), and popularization of Internet have resulted in a new situation for reusable software consumers: there are many more accessible, potential reuse candidates. Given the high interest in reuse and motivation to the use of commercially available software, many software development projects include the evaluation and selection of reusable components as an important activity in the project, with a high potential impact on the product and project objectives. According to our observations in many organizations, the selection process typically is not defined, each project finds its own approach to it, often under schedule pressure, and there are no mechanisms to learn from previous selection cases. Yet the selection of the right reusable component is often a non-trivial task and requires careful consideration of multiple criteria and careful balancing between application requirements, technical characteristics and financial issues. It seems that there is a lot of potential for non-optimal or inconsistent software reuse decisions.

However, the issues and problems associated with the selection of suitable reusable components have rarely addressed in the reuse community. Poulin et al. present an overall selection process [34] and include some general criteria for assessing the suitability of reuse candidate [43]. Some general criteria have been proposed to help in the search of potential reusable components [35,36]. Boloix and Robillard recently presented a general framework for assessing the software product, process and their impact on the organization [12]. However, little of this work is specific to externally developed, off-the-shelf (OTS[1]) software selection and the issues of how to define the evaluation criteria are not addressed. Furthermore, most of the reusable component literature does not seem to emphasize the sensitivity of such criteria to each situation.

[1] OTS stands for "off-the-shelf". This term is frequently used to refer to software packages that have been developed or are suitable for reuse. The term originates from the term COTS software, commercial off-the-shelf software, but as our approach is not limited to commercial products, we have dropped the letter "C" from the term.

1.2 Executive Summary

We have developed a method that addresses the selection process of packaged, reusable off-the-shelf software. The method, called OTSO[2], supports the search, evaluation and selection of reusable software and provides specific techniques for defining the evaluation criteria, comparing the costs and benefits of alternatives, and consolidating the evaluation results for decision making.

The OTSO method contains a definition of the activities involved in reuse component selection. These activities have been described in a form of a generic process model that can be customized for each organization and selection case as necessary. The OTSO process definition will reduce the planning cost of OTS software selection and will result in systematic and consistent OTS software selections. Furthermore, by formalizing the OTS selection process it is easier to build on past experiences to improve the selection process and methods themselves.

The OTSO method emphasizes a systematic and detailed evaluation criteria definition from reuse goals. We have identified a set of factors that influence these goals. These include application requirements, application architecture, project objectives, organization reuse infrastructure, and availability of OTS alternatives. Evaluation criteria are defined gradually based on these goals.

The cost estimation of reusable software is often problematic as the traditional cost models are not very applicable. Also, the benefits of different OTS alternatives may be different which further complicate the cost benefit estimation problem. The OTSO method partially solves these problems by using a common reference point, called the baseline, to make cost and benefit estimations comparable. The baseline represents the minimum level of features and characteristics that the OTS software will need to meet. The cost estimation question is to estimate the cost of meeting the baseline, the benefit estimation is based on comparing alternatives assuming that they meet the baseline.

The OTSO method uses a multiple criteria decision making technique for consolidating the different criteria that have been used in the evaluation. The commonly used simple techniques, such as weighted scoring tables, make it difficult to define balanced criteria weights, are difficult to understand and interpret intuitively, and boost false confidence in the results. We have integrated an existing, well validated decision support method, AHP, in the OTSO method. Our own experience with the method indicated that AHP produces more reliable results than a simple scoring method.

In summary, the main characteristics of the OTSO method can be characterized with the following points:

- a well-defined, systematic process that covers the whole reusable component selection process,

[2] OTSO stands for Off-The-Shelf Option. The OTSO method represents a systematic approach to evaluate such an option. Otso also means bear in Finnish. In Finnish folklore this word is often used to refer to the bear as the king of the forest.

- a systematic method for deriving detailed OTS evaluation criteria from reuse goals
- a method for estimating the relative effort or cost-benefits of different alternatives,
- a method for comparing the “non-financial” aspects of alternatives, including situations involving multiple criteria, and

We have applied the OTSO method in two case studies that are referred to in this paper. These case studies indicate that the method is feasible and has a low overhead. It also seems that the method results in efficient and consistent evaluations and increases decision makers’ confidence in evaluation results.

1.3 Structure of This Paper

This paper presents the OTSO method and its underlying principles. We have reported about usage experiences of the method separately [28-30].

The structure of this paper is as follows. In the next section we will present the main principles of the OTSO method and discuss potential problems in reusable software selection. In section 3 we present the OTSO method in the context of the Experience Factory and give a more detailed description of the OTSO process. Section 4 will present some highlights from the case studies we have conducted with the method: how they were carried out and what was learnt from them. Finally, the conclusion section discusses the relevance of our results.

1.4 Acknowledgments

Several people have supported the work described in this report. This work has been conducted with the support of Hughes Information Technology Corporation and I am grateful for their sponsorship. Especially, I would like to thank Mike Deutsch, Show-Fune Chen and Kevin Limperos for their contributions. I am also grateful for the discussions with Gianluigi Caldiera, Victor Basili and Roseanne Tesoriero during the course of this research.

2. Main Principles of the OTSO Method

Software reuse decisions need to be made more often and these decisions have a higher potential impact on the projects and product. The software reuse decisions also may need to be done early in the project, when the requirements and overall system design may not have been fixed. Any organization that frequently uses OTS components in their projects should be concerned with the quality and effectiveness of its OTS selection process – correct decisions should be made as efficiently as possible.

This section the underlying main principles of the OTSO method – the OTSO process definition, the evaluation criteria definition process and cost and value estimation approach.

2.1 Well-defined Selection Process

The reusable component selection process is rarely well defined in organizations. Based on our experience, few organizations have detailed guidelines for the selection and evaluation of reuse alternatives and project personnel in charge of this task often need to define their own approach to the problem. If, as often is the case, the selection is done under schedule pressure, there is potential for "quick and dirty" solutions that may not result in a systematic and balanced decision.

The OTSO method includes a general process definition for the reusable component selection process. The purpose of this process definition is to act as a basis for each OTS evaluation project, adopting it to their purposes as they see fit. Each organization should also consider customizing the OTSO process definition for their own use. With a well-defined process model, we believe, the planning time for each project is reduced, an effective and sound approach is used, and different selection cases are consistent.

The OTSO selection process has been divided into six main phases, as presented in Figure 1. The horizontal axis in Figure 1 represents the progress of the evaluation (i.e.,

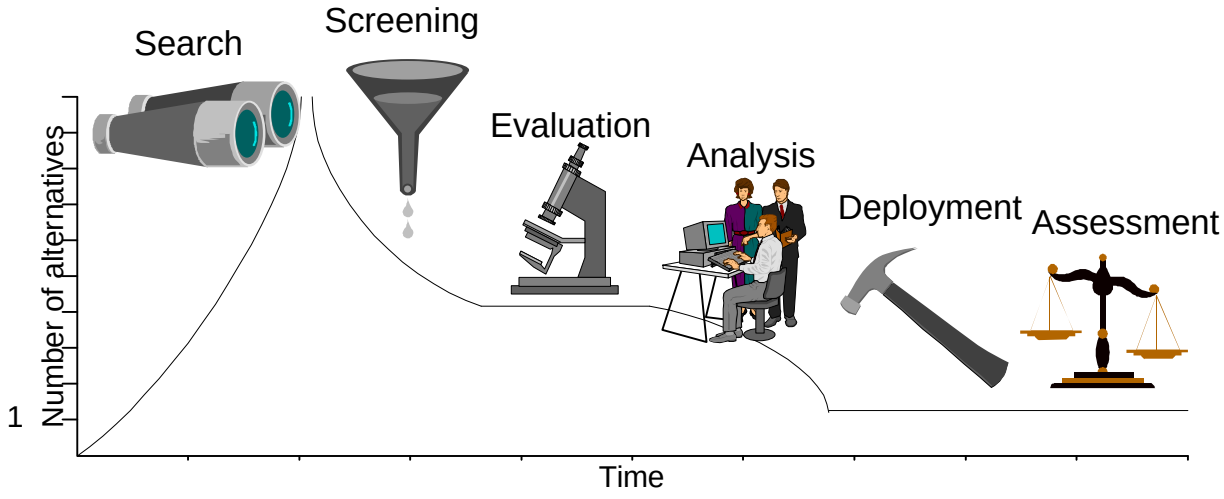

Figure 1: Main phases in the reusable component selection process

time) and vertical axis the number of alternatives considered at each phase. Starting by the *search* phase, the number of possible alternatives may grow quite rapidly. The most potential candidates will need to be sorted out (*screening*) to pick the ones that can be evaluated in more detail with the resources available. Detailed *evaluation* of a limited number of alternatives determines how well each of the alternatives meet the evaluation criteria. These results are systematically documented. We have separated out the *analysis* phase in order to emphasize the importance of interpreting evaluation data. Sometimes it may be possible to make straight-forward conclusions if one of the alternatives is clearly superior to others. However, in most cases it is necessary to use systematic multiple criteria decision making techniques to arrive at a decision. Based on the decisions made, typically one of the alternatives is selected and *deployed.* Finally, in order to improve the selection process and to provide feedback on potential further reuse of the component, it is necessary to *assess* the success of the reuse component used in a project.

The evaluation criteria definition process takes place concurrently during the search, screening and evaluation phases, although it is not explicitly represented in Figure 1. The evaluation criteria play a central role in the selection process, as we will discuss later in this document.

We will provide a more detailed view of the OTSO process in section 3 of this document (section *3: The OTSO Process in Detail*). In the following sections we will first introduce some key principles and techniques that are necessary to understand the process description in section 3.

2.2 Evaluation Criteria Definition

Selecting a suitable OTS alternative requires consideration of several factors. It is often a multiple criteria decision making problem. A potential danger in the process is to emphasize technical characteristics of alternatives and ignore some relevant user or system requirements. We believe that this is likely to happen when there may be little time for the decision, requirements are fuzzy and the selection process has not been defined. In such situations it is easier to use the information that is concrete and easily available, i.e., detailed technical characteristics of the alternatives in question. The potential problem with this approach is that the easily available technical criteria do not necessarily correspond to the issues that should determine the alternative to be used.

2.2.1 Reuse Goals

The overall relationships between influencing factors, reuse goals and evaluation criteria are presented in Figure 2. The first primary task in reusable software evaluation is to define the reuse goals. This must be based on a careful analysis of the influencing factors. The reuse goal definition and the analysis of factors will need to be iterated several times. In many cases it may be logical to start by defining tentative reuse goals, based on intuitive understanding of the factors. The goals can then be evaluated against the factors and revised if necessary.

The OTS software reuse goals need to be defined for each situation. The OTS software reuse goal statement essentially should address the following issues:

- Where and how OTS software is to be used in the application; e.g., database component, graphical user interface or communication routines.
- What are the expected benefits of OTS software reuse, e.g., in terms of functionality, quality, schedule impact, or effort savings; e.g., six month schedule savings, one person year effort savings.
- Possible constraints for OTS reuse; e.g., implemented in C++, software must run on both Unix and Windows 95, vendor must be commercially viable.
- Cost budget for the use of OTS software; e.g., acquisition cost must be less than $50 K.

The reuse goal statement should be documented explicitly, although initially the goal statement may seem abstract and simple. Our experience indicates that it will be revised and detailed as the OTS evaluation progresses.

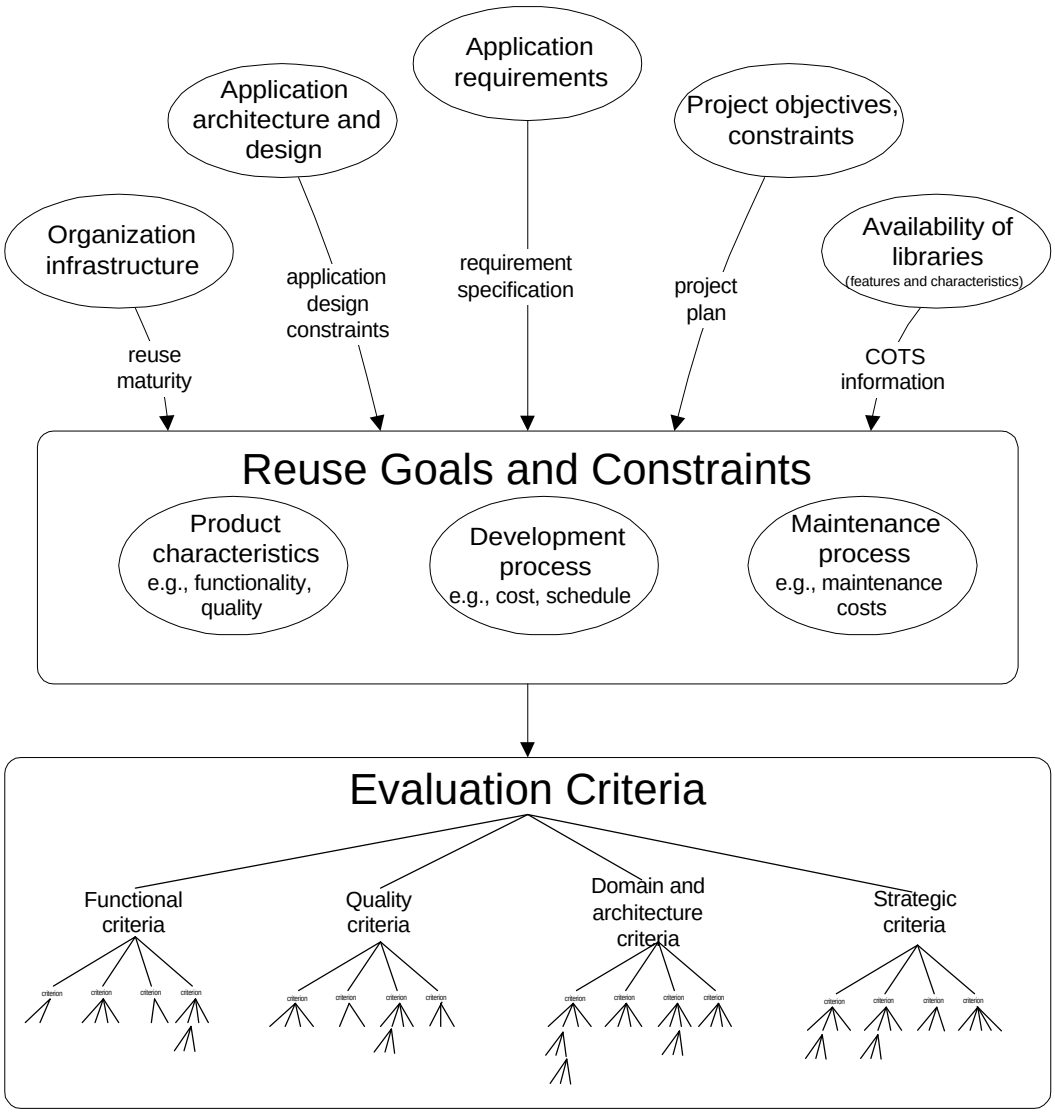

Figure 2: Factors, reuse goals and evaluation criteria

Reuse goals can divided into development process goals, maintenance process goals and product characteristics goals. Development process goals relate to cost, effort and schedule of the development project. The maintenance process goals deal with issues that affect the maintenance of the system, such as the ease or cost of maintenance and who will be responsible for maintenance. Product characteristics refer to product functionality and product quality.

We have identified five groups of factors that primarily influence the OTS software selection. In the following sections we will discuss each of these groups. Note that our list of factors is not meant to be a conclusive list, additional factors may be relevant in different situations.

Application requirements are likely to be the most important factor in reusable software evaluation cases. Such requirements can include functional requirements, such as ability to manage and display graphical geographical data, and non-functional requirements, such as available memory or speed of operations. The requirement specification, if available, should be used as a basis for interpreting such requirements.

The requirement specification, however, can only give partial support for the interpretation of software reuse goals. First, the requirement specification typically does not define how the system should be implemented and what components could be implemented through OTS software. Considering the use of OTS software in a system means making some assumptions about the architecture and design of the system and requires some decisions on what system features should be covered by the OTS software. Second, the requirement specification may not be detailed enough for evaluating OTS software alternatives. The formulation of software reuse goals requires interpretation and further refinement of requirements and some design concepts.

Application architecture and application domain may pose additional considerations for the evaluation. The software environment may rule out some incompatible alternatives (e.g., Mac software does not run on MS-Windows), software architecture may or design may make integration of some alternatives impractical (e.g., software that is not compatible with OLE in MS-Windows) and application domain may have some specific characteristics that are not addressed by OTS software developed for other domains (e.g., real-time applications vs. batch processing). Sometimes the architecture is given and acts as a constraint in the OTS software selection, sometimes the selection of the right OTS software may determine or influence the system architecture.

Project objectives and constraints may influence the library selection through the schedule or the budget of the project. For instance, early deadlines or low personnel budget may require the use of externally produced software. Project objectives can also imply the use of external software if, e.g., these external libraries provide a best way to comply to standards or are proven reliable in implementing some aspects of the library. There may also be some organizational constraints that are set for the project, such as availability of personnel with specific skills or personnel availability constraints at different points in time.

The availability of features in potential software reuse candidates also affects the reuse goal definition. This works in two ways. On one hand it is important to check that the

reuse goals are based on realistic expectations, i.e., the goals should not assume characteristics that are not provided by any OTS software alternative. At the same time it may be useful to learn to know about the possibilities that OTS software alternatives offer but which may not have been included in the requirement specification. For example, during our first case study one of the OTS alternatives had a feature that was not initially specified as a requirement but as the application designers considered in a valuable feature they proposed it to be included in the requirements specification.

Finally, an organization's reuse infrastructure and reuse maturity is also a factor that should be considered in reuse goal definition. The reuse maturity means the experience in reuse, organization's commitment and interest in continuing systematic reuse, knowledge and skills of personnel with respect to reuse, specific tool available for supporting reuse (configuration management tools, information databases, etc.), and current software development environment [15,27]. The reuse maturity is particularly important for the in-house production of reusable components. Also, if an organization has no experience in OTS software reuse, it may have a limited ability to integrate OTS software and to estimate OTS software integration effort.

The main point of this discussion is that the reuse goal formulation process should take all the factors into account. Consequently, *reuse goals should be formulated with full awareness of these factors*. In most cases this requires that each factor is explicitly analyzed and, ideally, the results of this analysis are documented and used as input in the final definition of the evaluation criteria.

2.2.2 Classes of evaluation criteria

Once the reuse goals have been defined it is necessary to define the evaluation criteria for OTS alternatives. The content and priorities of the reuse goals influence what are the important characteristics to be considered in the of the OTS software selection process.

The evaluation criteria themselves can be categorized into four main areas: functional requirements, product quality characteristics, strategic concerns, and domain and architecture compatibility. We will give examples and discuss each in the following.

Functional requirements are identifiable, functional features or characteristics required for the application. These criteria are derived from the requirement of design specification and are expressed in form of requirements. Examples from an application dealing with geographical data:

- "Display ocean bathymetry data"
- "Show political boundaries"

Functional requirements vary between each OTS reuse situation and they may need to be developed from scratch each time.

Product quality characteristics[3] refer to "non-functional" product characteristics. High level examples of such characteristics include maintainability, reliability, portability, performance, etc. On a concrete level, the following examples could be given:

- "Defect rate"
- "Compliance to the project UI guidelines"
- "Clarity of documentation"

Product quality characteristics do not necessarily vary between OTS reuse evaluation cases as much as product functionality. In fact, some general product quality characteristics have been proposed over the years [10,17,26] and these can be used a template for defining this criteria. Typically the structure and relationships of product quality characteristics remain relatively unchanged between OTS evaluation cases but their acceptable values may vary more. An example of a product quality characteristics hierarchy from our first case study is presented in Figure 8 (page 33).

Strategic concerns refer to issues that are important factors in considering both short-term and long-term effects of the reuse candidate, cost-benefit issues and organizational issues beyond the scope of the project in question. Many of the criteria under strategic concerns are derivatives of the other criteria as they, e.g., consolidate information for decision making. Examples:

- "Acquisition costs"
- "Effort saved"
- "Vendor's future plans"

Domain and architecture compatibility refers to characteristics that are defined by the application domain or the architecture and design of the system. An application domain may set specific requirements require specific characteristics from reuse candidates. For instance, all flight control software need to be very reliable and they must have been developed for time-sensitive and reactive issues in mind. A reuse candidate originally developed for an accounting system may have fundamental design and performance characteristics that make it unsuitable for a real-time application.

Domain compatibility refers to how well the reuse candidate and its features map into domain terminology and concepts. In the case of object oriented reuse candidates, this can refer to match between domain objects and object definitions in the reuse candidate. Architecture compatibility refers to software or hardware architecture requirements that are common to the application area. Examples of both kinds are listed below:

- Domain compatibility:
    - "system states can be modeled and represented "
    - "geographical data manipulation capability"
- Architecture compatibility:
    - "supports or is compatible with CORBA"
    - "compatible with Microsoft Windows OLE"

[3] Note that we use the term "quality" in a limited meaning, as we have excluded the product functionality from it. A common, contemporary definition of quality does not make this limitation [14,16,25]. In some cases we have used the term "non-functional requirements" but as it seemed to have a negative connotation to some audiences we decided to use the term quality characteristics instead.

It is important to emphasize that the evaluation criteria must be customized for each selection situation. Especially the functional requirements, which often or central in the selection process, are often unique in each application. For the product quality characteristics and strategic concerns it is possible to define some templates that have stable elements between applications.

2.2.3 Hierarchical decomposition of evaluation criteria

An important aspect of the evaluation criteria definition is the hierarchical decomposition of them to a level that can be used for evaluation purposes. Initially, people can typically identify high level goals, such as "maintainability", "reliability" or "provide graphical user interface". These high level goals need to be refined in order for them to be useful. The OTSO method uses a decomposition approach that is based on Basili's GQM method [3,7] and Saaty's criteria decomposition approach [38]. We have adopted GQM's explicit goal definition template to express the relevant reuse goals explicitly. The template is presented in Table 1.

Attribute of GQM	Explanation/examples
Object/entity	The entity being analyzed, e.g., OTS product, OTS vendor
Purpose	Evaluate: evaluate the characteristics of the entity w.r.t. a relevant benchmark. This attribute is typically the same in all reuse evaluation cases.
Focus (issue)	The attributes that are of interest, e.g: cost, reliability, or efficiency.
Point of view (perspective)	Who's interest is being expressed, e.g., project manager, corporation, customer, developer, etc.

Table 1: GQM-based evaluation criteria definition template

In the OTSO evaluation criteria definition approach the goals can be decomposed through several sub-goals and, correspondingly, each question by several sub-question. As the difference between a sub-goal and a question is fuzzy, we have used the term criterion (criteria) to refer to refer to all goals, sub-goals, questions or sub-questions in the hierarchy. The principle of decomposition remains similar to Basili's and Saaty's approaches but we do not differentiate between goals and questions. Also, we have used the term *evaluation attribute* instead of "metric" to refer to observable or measurable units in the criteria hierarchy. In other words, an evaluation attribute can be an observation, a measurement, or a piece of information to be obtained. Table 2 lists examples of possible types of evaluation attributes.

Type of Evaluation Attribute	Examples
Measurement	• memory used when loaded • time to initialize • number of bugs found during evaluation • support of incremental image loading (yes/no)
Description	• list of reported bugs • description of the “undo” function
Subjective assessment	• estimate of vendors growth potential • look and feel of GUI

Table 2: Examples of Evaluation Attributes

The basic steps of criteria decomposition are the following:

1. Identify and formulate reuse goals.
2. For each goal, define a set of high level criteria or questions that characterize them.
3. For each criterion, write down an unambiguous definition of it.
4. If the value for the criterion can be determined with a objective measurement, observation or judgment, call it a evaluation attribute and continue to decompose and define other criteria. If the criterion is too abstract to be measured with a single metric, has too many aspects to be assessed through observation or it cannot judged objectively, continue decomposing it.

The number of items at each level should be less than ten, preferably around 3-5. The decomposition of evaluation criteria into concrete evaluation attributes can be supported by the template we present later in this document (Table 5).

2.3 Cost and Value Estimation of OTS Software

The decision of using OTS software is based on the estimated costs of an OTS alternative and the estimated value it will bring to the project. The cost and value estimation has two challenges. First, as with all estimation problems, it is difficult to estimate characteristics that are based on events and processes that have not taken place. The second problem is that each OTS alternative may have distinct characteristics that make their comparison rather difficult, even if reliable cost and feature estimates were available. One alternative may have features that others lack so it is difficult to normalize the costs associated with each alternative.

The OTSO method uses an approach that allows a balanced comparison of alternatives with respect to their costs and value they provide. This section presents the general principle of the OTSO cost and value assessment, as well as the individual approach possible for evaluating cost and value separately.

2.3.1 Defining the Baseline

The OTSO cost and value assessment is based on the idea of making all alternatives through a common reference point. This reference point is called the *baseline*. Baseline should be defined as a set of characteristics that each OTS alternative must meet, or

exceed, after they have been modified and developed further for the project's purposes. Baseline should reflect realistically the true situation in the project. It should represent the characteristics and features that must be satisfied, no more, no less. The baseline should be derived from the requirement specification and good understanding of the possible implied requirements.

Note that an OTS alternative does not need to meet the baseline requirements as it is at the time of evaluation. In many cases additional effort would need to be spent in order to meet the baseline requirements. If the cost of meeting the baseline becomes prohibitive, the alternative must be dropped as not being realistic[4].

The cost estimation for each alternative is based on how much it costs to obtain, develop them further and integrate them to meet the baseline.

The value estimation of each alternative is based on their characteristics *assuming that they have been developed further and integrated to meet the baseline.* This way, the cost estimation problem is dealt with separately and results in cost estimates that are comparable with respect to the baseline requirements. The value estimation is based on the baseline reference point and each alternative is rewarded for characteristics that exceed the baseline.

The idea behind the baseline as a basis for cost and value estimation can be illustrated with the help of Figure 3. The different characteristics that are relevant for the baseline are presented on *x* axis. The Figure 3 assumes that each characteristic can be expressed as a vector, in terms of being able to refer to how well it meets the baseline. Each vector represents how well each alternative (A_i) satisfies the characteristics (c_j) defined in the baseline. A vector can be referenced by defining the alternative and the characteristic in question (A_ic_j). Examples of possible characteristics include:

- stated functionality (e.g., "zoom in capability" or "supports background printing")
- quality characteristic (e.g., "reliability", "level of documentation")
- consumption of resources (e.g., "disk space required")
- standards compliance (e.g., "Win95 compatible", "matches our coding standards")

Figure 3 shows an example situation where an alternative's situation is presented as a set of vectors. Baseline is defined as a set of vectors B_j and represented by a horizontal, zagged bold line in Figure 3. Alternative A_1's current vector set is represented by vectors A_1c_j, where j=1,…10. The cost estimation problem for A_1 is to estimate the cost of "upgrading" A_1's characteristics to meet the baseline. These improvements in characteristics are represented by thick vertical lines, where applicable.

As Figure 3 shows, some characteristics can exceed the baseline (A_1c_3 and A_1c_9) or they can be the same as the baseline requirement (A_1c_5) and there is no need to estimate the cost. In some cases the characteristic does not exist in an alternative at all (A_1c_7) or exists but does not meet the baseline (e.g., A_1c_1 and A_1c_6) and the required features will need to

[4] Note that if an alternative is included even though the cost to meet the baseline is prohibitive, this implies that the baseline is not representative of true minimum requirements. In such a case one should redefine the baseline.

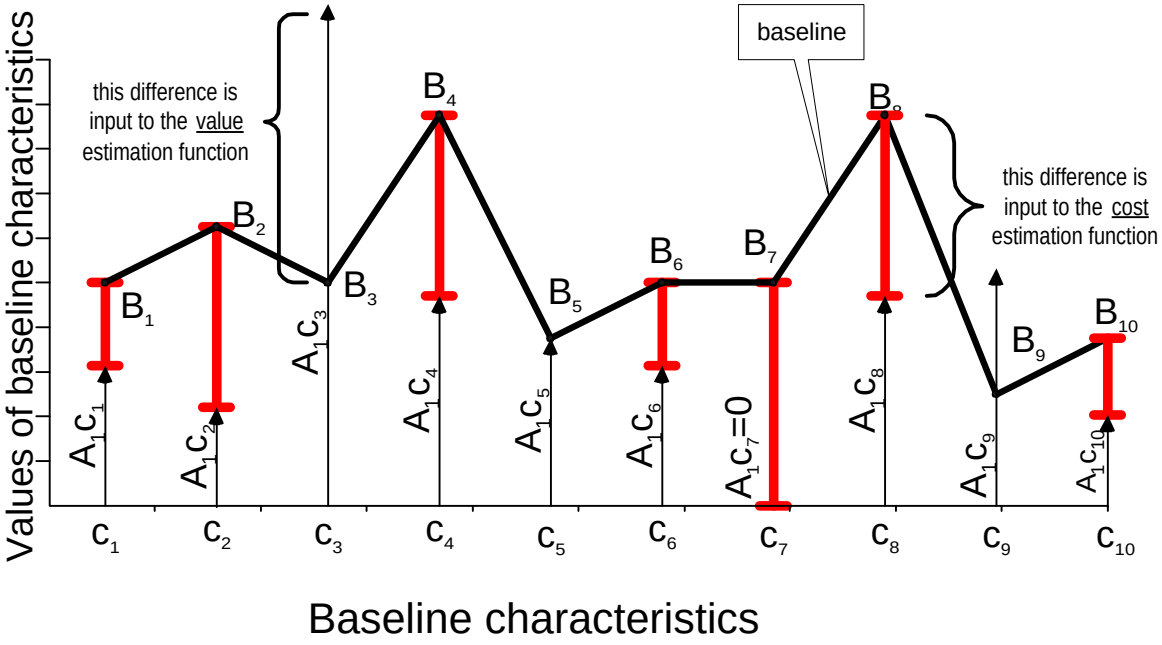

Figure 3: The "cost to baseline" estimation principle.

be implemented. Assuming that we have a reliable cost estimation function *Costf* the cost estimation problem can be expressed as follows[5]:

$$\text{cost of } A_1 = Costf\left(\mathrm{B_i} - A_1 c_i\right), \text{ for all } \mathrm{B_i} - A_1 c_i > 0, \text{ where } i = 1,..,n$$

The relative value comparison of alternatives is based on characteristics that exceed the baseline. The OTSO method uses a multiple criteria decision making technique that allows the estimation of how valuable each the instance of exceeding requirements is. In our example the relative values of A_1c_3 and A_1c_9 would need to be estimated. Assuming that we have a reliable value estimation function *Valuef*, the value estimation problem can be expressed as follows:

$$\text{value of } A_1 = Valuef\left(A_1 c_i - \mathrm{B_i}\right), \text{ for all } A_1 c_i - \mathrm{B_i} > 0, \text{ where } i = 1,..,n$$

Note that even though we used vectors (and, in effect, ratio scale metrics) for expressing values of characteristics, the ability to measure the vector is not required for the use of this method. All that is required is

- to be able to express the state of characteristics for each alternative,
- to be able to express the baseline requirement,
- the ability to estimate the cost of making the characteristic meet the baseline, and
- the ability to estimate the value for characteristics that exceed the baseline.

[5] Note that we assume that the cost function can deal with situations where the alternative characteristics exceed the baseline, i.e., if $B_i < A_j c_i$ the corresponding cost impact is zero.

The details of both cost and value estimation principles are discussed in the following two sections.

2.3.2 Cost Estimation

Two main approaches for cost estimation are available: use of cost models or work breakdown structure analysis. Cost models, in theory, may provide a way to obtain unbiased estimates but their problem is that traditional cost models are not very applicable for OTS software cost estimation and it is difficult to capture all relevant factors into a single cost model. A COTS specific cost model has been recently developed and this may provide a partial solution to such cost estimation [18]. However, this kind of model may require customization and calibration for each organization to be effective.

The work breakdown structure analysis may be, for many organizations, the feasible method for OTS software cost estimation: the development and integration tasks for a OTS software are listed and decomposed, effort for each task estimated and total effort summed up. The disadvantage of this approach is that it can be very sensitive to bias or the experience of the personnel.

The OTSO method does not address what method or model is used for OTS reuse cost estimation. Whatever approach is used, the OTSO method extends the financial OTS software evaluation by allowing the consideration of other factors that may influence the decision. Examples of such factors include the consideration of features that exceed the requirement specification, quality characteristics that are not included in the cost estimation model (e.g., reliability, maintainability, portability, efficiency, etc.), and business or strategic issues that may influence the decision. These issues can sometimes be decisive in OTS software selection and cost estimation alone cannot effectively cover these aspects.

The costs of acquiring OTS software can be broken down to three main classes: acquisition costs, further development costs and integration costs. The OTSO method contains a template for breaking these down further to support OTS software cost estimation (Table 3). While the acquisition costs are relatively straightforward to estimate, the further development costs and integration costs present much more challenging problems.

The further development costs of OTS products are based on developing them to meet the baseline. However, as the baseline may be difficult to define accurately and as few organizations have accurate OTS software cost estimation models or expertise, this cost estimate may have a large margin of error.

Sometimes OTS software includes features that were not originally required for the application. We refer to such functionality or characteristics features as *unrequired features*. Dealing with these unrequired features may complicate the cost estimation process. Although some unrequired features may be marginally useful for users, they may make the system too complex for some users. Added functionality may also increase integration and development costs.

Cost item	Explanation
Acquisition costs	
Development version acquisition costs	The cost involved in obtaining an adequate number of licenses for software development
Delivery (run-time) version costs	The possible costs of obtaining the right to deliver the OTS software as a part of the software to users.
Maintenance costs	The possible maintenance fees for the OTS software, e.g., for obtaining the rights for version upgrades or bug patches.
Learning costs	Cost of providing training, independent learning and the impact of learning curve effect on productivity.
Infrastructure upgrade costs	The possible costs involved in having to upgrade some parts of the computer or software environment, e.g., obtaining additional memory capacity, upgrading operating system versions or obtaining necessary support software.
Further development costs	
Cost to develop the OTS software to meet the baseline requirements	The cost of making changes to the OTS software so that it meets the requirements set for the system.
Integration costs	
Modification costs	The cost of making modifications to OTS software.
Costs involved in building additional interface modules	The cost of making interface modules that facilitate the integration of OTS software to the system.
The impact of testing external components	The effort to test OTS software components and correct errors associated with them may be different from the effort to in-house components. However, the OTS software impact can be both positive and negative, depending on the situation.
Increased testing costs for unrequired features	The increased cost of testing the unrequired features that OTS software bring in to the system.

Table 3: Cost components

We recommend that organizations involved in COTS reuse initiate procedures to develop and customize their cost models for this purpose. As it takes time to develop these models, most organizations may have to rely on expert judgments in these cost estimates.

2.3.3 Estimating the Value of OTS Alternatives

As the OTS alternatives have been evaluated the evaluation data needs to be used for making a decision. A common approach for this in many organizations is an approach that can be called the *weighted scoring method* (WSM). The WSM method is typically applied in the following fashion: criteria are defined and each criterion is assigned a weight or a score. In the case of using weights, they may be normalized so that their total is one. If "scoring" is used, this is done, e.g., by assigning a "weight score" between one and five for each criterion. Then, each alternative is given a score on each criterion. The score for each alternative is counted by the following formula:

$$\text{score}_a = \sum_{j=1}^{n} (weight_j * \text{score}_{aj})$$

where subscript *a* represents an alternative and *n* represents the number of criteria.

There are several shortcomings in this approach. Scoring in this manner essentially is the same as taking a weighted average of individual scores. This can be mathematically

justified as a correct representation of the utility of the alternatives only when the following assumptions can be made about the criteria and the data:

1. Individual scores are based on distance scale or better, i.e., distance, ratio or absolute scales are allowed. Strictly speaking, taking an average of values expressed in ordinal or nominal scales is not meaningful and mathematically not an allowed operation.
2. Scores have been normalized with each other, i.e., compensating the effect of high or low absolute numerical values on each criterion.
3. There is a linear, monotonic relationship between each criteria and the value it represents to the stakeholder. In other words, the value increments on any criterion should be directly proportional to the "goodness", or utility, the criterion represents.
4. The number of criteria is relatively small. If there are many criteria, it becomes difficult to define weights so that they represent the real preferences and ratios of importance.
5. All criteria are independent, i.e., they do not measure overlapping areas and they do not conflict with each other. This assumption can be relaxed if overlapping and conflicts are taken into account when defining weights for criteria.

Clearly, the above assumptions are seldom true in practice and would require a great deal of effort and discipline to reach.

The application is the WSM is straight-forward and presents results that intuitively make sense. However, the WSM as presented above has several serious limitations that are often ignored when it is applied. First, as the WSM produces real numbers as results, these are easily interpreted as if they represented truly the differences between alternatives in ratio or distance scales. However, this would be true only if all the criteria scores have been given using distance or ratio scale. Our experience strongly suggests that this is rarely the case. If even one of the criteria was rated on an ordinal scale score, the resulting scores only represent relative ranking of the alternatives and the differences in their value do not give indication of their relative superiority.

Second, assigning of weights for the criteria is very difficult when the number of criteria is high. It seems that people are able to deal with less than ten alternatives at a time [37]. With large number of issues to keep in mind people have difficulties in mentally coping with the dependencies of individual factors. The assigning of scores instead of weights is even more limiting. If this method is used, this effectively sets predetermined lower and upper limits to the weights one can assign to criteria. For example, if a scale of 1 to 5 is used and we have 20 criteria, the minimum and maximum weights given to a single criterion can be counted by

$$\min = \frac{1}{\sum_{j=1}^{20} weight_j}, \qquad \max = \frac{5}{\sum_{j=1}^{20} weight_j}$$

In other words, the weight range possible for each criterion is within 1.04% and 4.17%. In practice the range is even smaller as weight scores are likely be more distributed than in the extreme cases used in the above calculations. For instance, in our second case study the maximum weight happened to be 3.4% with the WSM. The problem with this method is that in practice some of the criteria may be more important than the maximum would allow.

Third, it is rather difficult to define a set of criteria and their weights so that they are either independent from each other or, if they overlap, their weights are adjusted to compensate for the overlapping areas.

Based on the discussion above and the criticism towards the WSM method presented in the DSS literature [42] we do not recommend the use of WSM for decision making, except in most simple cases.

The OTSO method relies on the use of the Analytic Hierarchy Process (AHP) for consolidating the evaluation data for decision making purposes. The AHP technique was developed by Thomas Saaty for multiple criteria decision making situations [37,38]. The technique has been widely and successfully used in several fields [39], including software engineering [19] and software selection [23,32]. It has been reported to be an effective technique in multiple criteria decision making situations in several case studies and experiments [13,20,39,42]. Due to the hierarchical treatment of our criteria, AHP fits well into our evaluation process as well. AHP is supported by a commercial tool that supports the entering of judgments and performs all the necessary calculations [40].

The AHP is based on the idea of decomposing a multiple criteria decision making problem into a criteria hierarchy. At each level in the hierarchy the relative importance of factors is assessed by comparing them in pairs. Finally, the alternatives are compared in pairs with respect to the criteria. The following are the main steps in applying AHP [19,38]:

1. Define a hierarchy of factors that influence the decision, resulting in a hierarchical structure of factors that have alternatives as the leaf nodes in the hierarchy
2. Define the importance of factors on each level
3. Define the preferences of alternatives
4. Check the consistency of rankings and revise the ranking if rankings are too inconsistent
5. Present the results of the evaluation, the alternative with the highest priority being the one that is recommended as the best alternative.

The rankings obtained through paired comparisons between the alternatives are converted to normalized rankings using the eigenvalue method [38], i.e., the relative rankings of alternatives are presented in ratio scale values which total one.

Saaty argues that hierarchies are a natural way for humans to organize their view of the world and they represent real world phenomena well [38]. Criteria and alternatives are compared in pairs, which results in more reliable comparison results. This way we are able to avoid the problem of having to assign absolute values to alternatives, only their relative preference or values are compared.

From our perspective the main advantage of AHP is that it provides a systematic, validated approach for consolidating information about alternatives using multiple criteria. AHP can be used to “add up” the characteristics of each alternative. Furthermore, an additional benefit of AHP is that we can choose the level of consolidation. We recommend that consolidation is only carried out to the level that is possible without sacrificing important information. On the other hand, some consolidation may be necessary in order not to overflow the decision makers with too much detailed, unstructured information.

In the OTSO method the weighting of alternatives is done using the AHP method, preferably using a supporting tool [40]. Preferences are collected and consolidated to the level stakeholders prefer. The AHP allows the consolidation of all qualitative information and the financial information into a single ranking of alternatives. However, we believe that this would condense valuable information too much. Instead, we recommend that information about the evaluation is consolidated to a level where a few main items remain and stakeholders can discuss their impact and preferences verbally. The full consolidation can be done at the end as a sanity check, if desired. The selection of what are the right main items depends on the reuse goals and the criteria hierarchy used.

3. The OTSO Process in Detail

In this section we present the OTSO method in the context of the Experience Factory and describe the activities involved in OTS software selection.

3.1 OTSO in the Experience Factory

The OTSO method can be seen as an extension to the Experience Factory concept [2,5]. The OTSO method, provides a repeatable process that evaluates, selects and implements reusable software components in an organization. OTSO supports systematic reuse and allows accumulation of OTS software selection experience, supporting continuous improvement of a component factory [4,8].

Figure 4 presents the OTSO method in the context of the Experience Factory. The OTSO components are highlighted in darker color. A customized, enactable model of the OTSO method is captured in a form of a process definition in the Experience Base. This process definition is reused, and modified if necessary, each time a project needs to consider OTS software. This can take place in the planning phase of the project or during a projects enactment. As a project is initialized, the Experience Factory can assist the project in refining and formulating the reuse goals for the project and in the enactment of the OTS selection process. The Experience Factory is also involved in the analysis of the experiences gained during the OTS software selection process. These experiences can be used to improve the OTSO process definition and in-house reuse library contents and access.

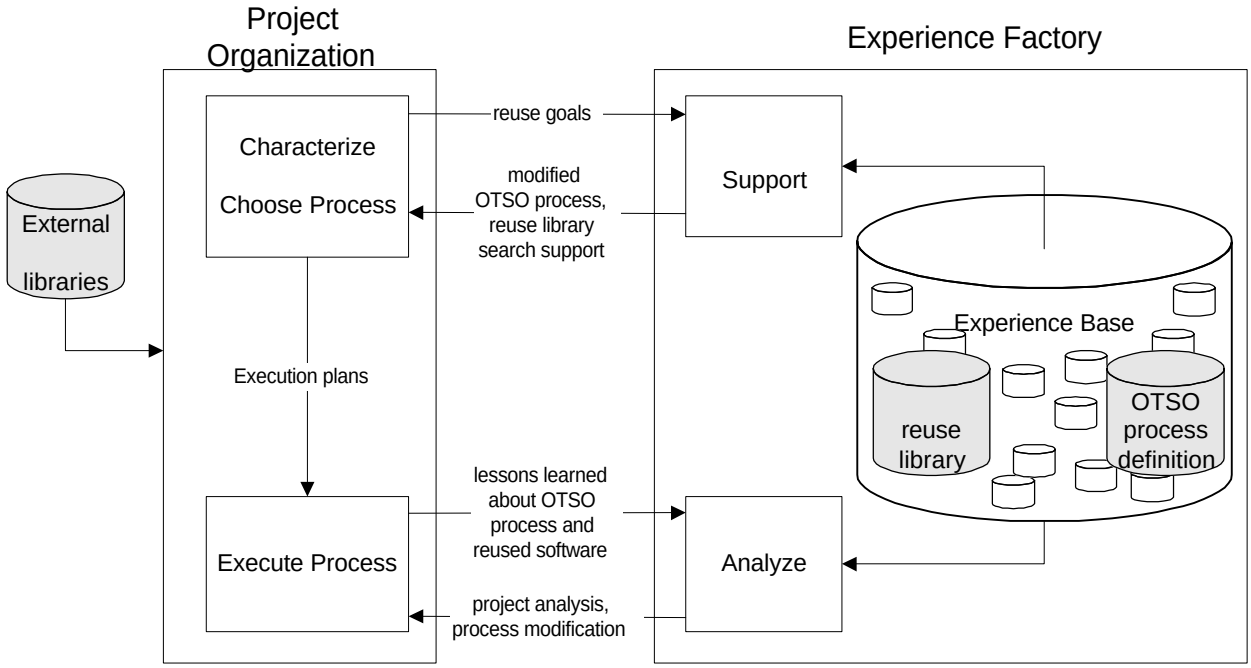

Figure 4: OTSO method in the Experience Factory context

Note that in-house developed OTS alternatives are also stored in the Experience Base but the OTS evaluation process can also take advantage of the external OTS sources.

As the OTSO process is a part of the Experience Base, experiences from its usage are also collected and analyzed by the Experience Factory. Also, experiences from the reused OTS components can be collected and analyzed and used in future OTS selection cases.

3.2 OTSO Process Overview

The main aspects of library selection are presented in Figure 5 in a data flow diagram. The process model in Figure 5 represents a complete view of the OTSO reusable component selection process. Each process (represented by a circle) and their work products (represented by a data storage symbol) in Figure 5 are explained and discussed in the following sub sections.

described in more detail in Figure 6 and Figure 7.

In the following sections we describe each of the processes presented in Figure 5. The description for each process includes the goals of the process, definition of the entry and exit criteria, definition of the output of the process and some guidelines about the process in question.

A summary of the main characteristics of each process is presented in Table 4. For each main process in the OTSO method we have defined its objectives and what is the main output of the process. We have also described the entry and exit criteria which essentially determine when a process can be initiated and when it can be concluded. Entry and exit criteria definitions help managing concurrency in the process.

Process	Objective	Output	Entry criteria	Exit criteria
Define evaluation criteria	Define evaluation criteria incrementally.	Search criteria Screening thresholds Detailed Evaluation criteria definitions	Reuse objectives defined. Application requirements defined. Project plan defined.	Evaluation criteria defined.
Search	Find all relevant alternatives.	List of alternatives with search criteria data.	The search criteria have been defined.	Frequency of discovering new candidates becomes very low.
Screening	Decide which alternatives are selected for detailed evaluation.	List of selected alternatives Rationale of selections.	At least one alternative has been identified and all search criteria data for it is available.	Evaluation alternatives selected and evaluation tasks assigned.
Evaluation	Evaluate the selected alternatives by the evaluation criteria and document evaluation results.	Evaluation data.	At least one alternative has been selected for detailed evaluation. Evaluation criteria has been defined in detail.	All alternatives have been evaluated by the defined criteria or required data has been determined not to be available.
Analysis of results	Decide which alternative is the best for the application.	Decision and decision rationale.	All alternatives have been evaluated by the criteria defined	Decision on the alternatives is made.

Table 4: Sub-process objectives, output, and entry and exit criteria

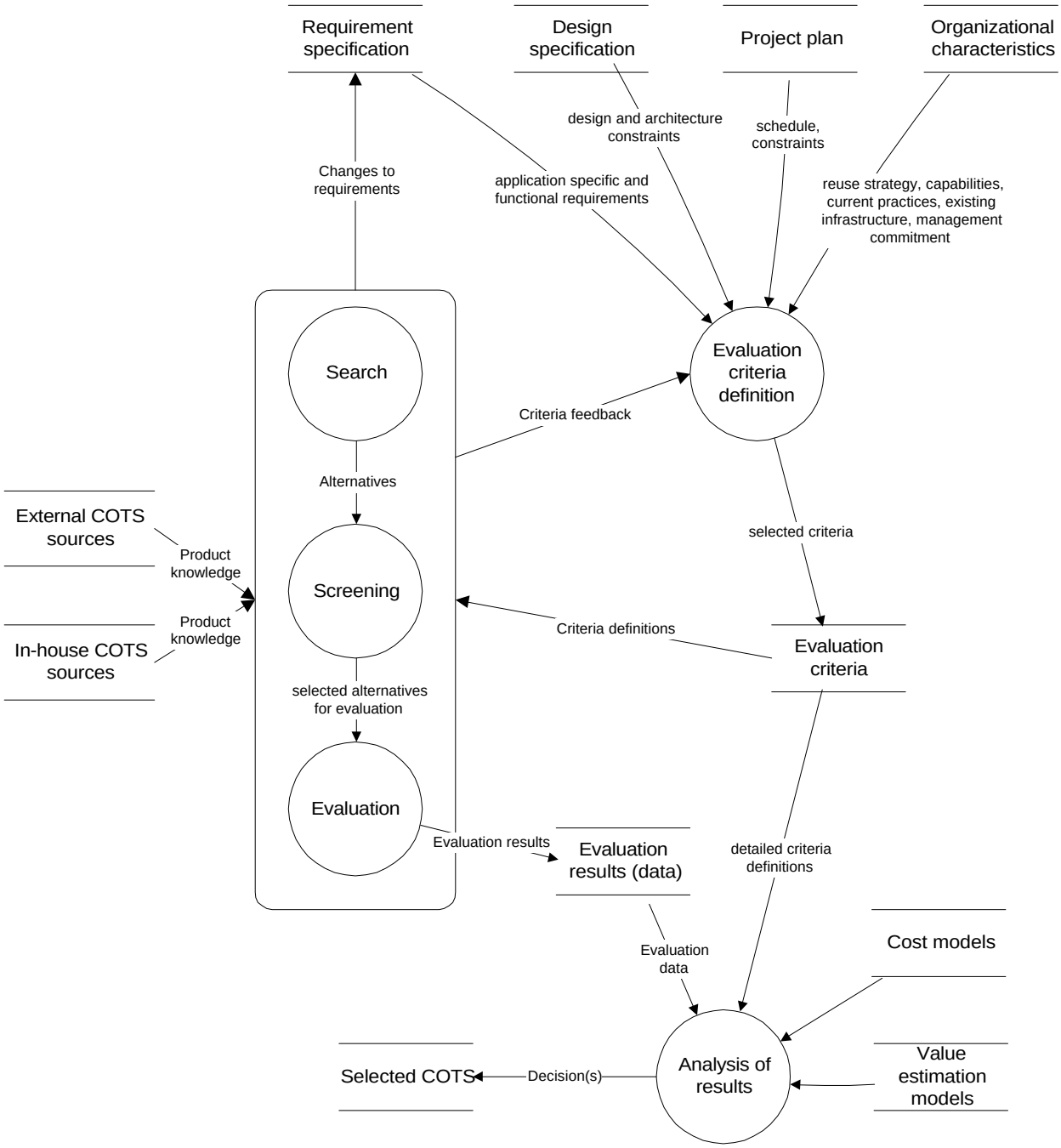

Figure 5: The reuse library selection process[6]

[6] Our process descriptions are presented using data flow diagrams (DFD). Processes are represented as circles and artifacts as a data storage symbol (a rectangle without sides: artifact). The only extension to the standard DFD notation is the use of higraph contours [22] to group items together hierarchically. The contours are used to indicate data flows that affect all of higraph contour members, without having to draw lines to each one of them. These contours are also used in some other figures to group items together.

3.3 Evaluation Criteria Definition

The evaluation criteria definition process essentially decomposes the reuse objectives into a hierarchical set of criteria. Each branch in this hierarchy ends in a "test": well defined measurement, observation or test that is carried out to determine how an alternative fulfills the criteria the test is associated with. The tests should have clear operational definitions [16] so that consistency can be maintained during evaluation.

It is possible to identify four different subprocesses in the definition of evaluation criteria search criteria definition, definition of the baseline, detailed evaluation criteria definition, and weighting of criteria. Figure 6 presents a graphical representation of these processes.

First, when the available alternatives are searched and surveyed it is necessary to define the main search criteria and the information that needs to be collected for each alternative. The search criteria is typically based on the required main functionality (e.g., "visualization of earth's surface" or "hypertext browser") and some key constraints (e.g., "must run on Unix and MS-Windows" or "cost must be less than $X"). As far as the main functionality is concerned, an effective way to communicate such requirements is to use an existing product as a reference point, i.e., defining the functionality search criteria as "look for OTS products that are similar to our prototype".

It is enough to define the search criteria broadly so that the search is not unnecessarily limited by too many constraints. The reuse strategy and application requirements are used as the main input in the definition of this criteria. In Figure 6 the search criteria definition and actual search are presented as separate processes.

The screening process uses the criteria and determines the "qualifying thresholds", which are in deciding which alternatives are selected for closer evaluation. These threshold values will be documented together with the criteria definitions.

The definition of the baseline criteria set is essential for cost estimates and for conducting qualitative ranking of alternatives. This can be done in parallel with the detailed evaluation criteria definition.

The search criteria, however, often is not detailed enough to act as a basis for detailed technical evaluation. Therefore, the criteria will need to be refined and formalized before initiating the technical evaluation. The evaluation criteria for the search of alternatives do not need to be very detailed or formally defined. However, as we discussed earlier, there must be detailed and unambiguous definitions for the criteria before detailed technical evaluation can be carried out. Without such definitions it is difficult to conduct a consistent and systematic evaluation, let alone consolidate the evaluation results for decision making. We have defined a template for criteria test definition that helps in defining criteria and tests in adequate detail. The template is presented in Table 5.

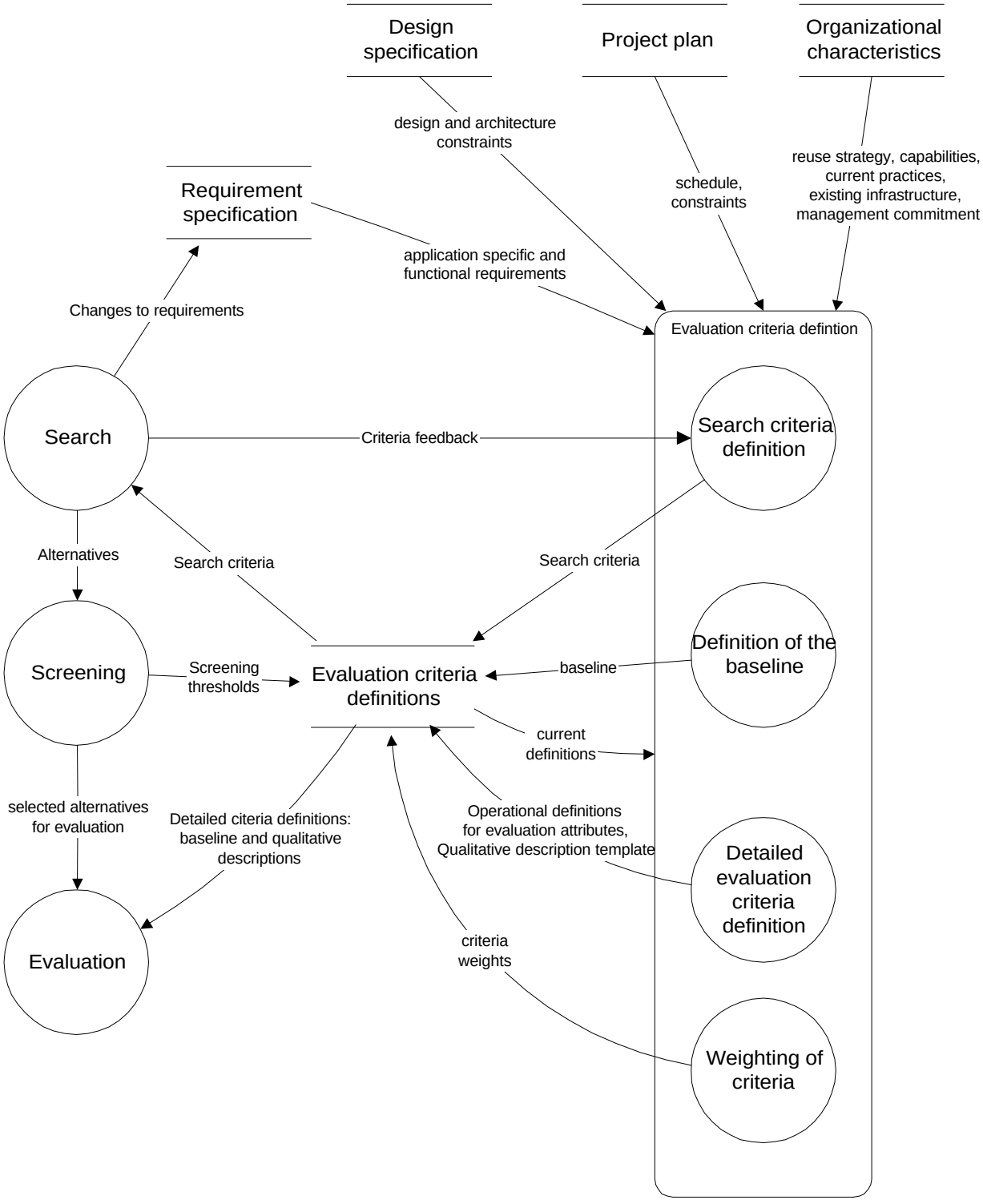

Figure 6: Evaluation criteria definition process[7]

[7] Note that the Figure 6 is a refinement of Figure 5.

As the evaluation criteria evolves in this process, it is gradually detailed and decomposed, using the principles we presented in section 2.2.3 Hierarchical decomposition of evaluation criteria. The criteria used for screening can be more abstract (e.g., "runs under Windows") but it will need to be more formally defined for the evaluation phase (e.g., "Windows95 compatible", "supports embedded objects").

Item name	Description
Heading	Heading for each evaluation attribute acts as a unique identifier.
Definition	A definition of the evaluation attribute.
Rationale	Description of the rationale for the evaluation attribute and how it relates to the evaluation criteria.
Scale	The scale or type of description used. • *Free format description:* The evaluation attribute will be documented with a free format description. • *List:* A list of features, characteristics, functions etc. Is produced. • *Structured description:* There is a template or a checklist that defines what should be described for each alternative. • *Nominal:* Classes are identified but they are not ordered. • *Ordered:* Classes are identified and they are ordered. • *Interval:* The scale has meaningful interpretation of distance between entities, but their ratios cannot be calculated, i.e., "there is no meaningful zero point". • *Ratio:* Entities can have ratios, "zero is a meaningful concept". • *Absolute:* The number of entities is counted.
Unit/classes	Definition of the unit of measure or the classes used, which ever is applicable.
Screening rule	Definition of a possible level that is required for an alternative to be selected for detailed evaluation. This field is used for documenting which criteria were used in the screening phase.
Baseline	Baseline is the minimum required level of functionality and features that the application must satisfy when it is delivered.
Qualitative description	Guidelines how additional information about the evaluation attribute should be documented.
Source	How the value for the evaluation attribute can be determined for each alternative.
Priority	Description of how important the particular evaluation attribute is. The priority classes are as follows: • *Required:* The value for the evaluation attribute is essential for the evaluation and must be obtained. • *Recommended:* It is recommended that the value for the evaluation attribute is obtained, if time available for the evaluation allows it. • *Optional:* The result of the evaluation attribute could be useful in the evaluation. The value should be obtained only if all other criteria have been covered and there is time available.

Table 5: Evaluation criteria definition template

Finally, once the evaluation criteria have been defined, they need to be weighted so that decisions can be made based on the evaluation results. As we discussed in section 2.3 Cost and Value Estimation of OTS Software, there are several methods for deriving weights for the criteria. The OTSO method recommends the use of AHP method for this purpose.

3.4 Search

The search in the selection process attempts to identify and find all potential candidates for reuse. The search is driven by the guidelines and criteria defined in the criteria definition process. The search process should produce a list of alternatives with at least the following information:

- name of the alternative
- reference to the source of the alternative (name and address of the company, FTP or WWW address, etc.)
- main characteristics and features of the alternative, such as operating platforms, price, popularity, main functionality, and vendor background.

By its nature, search is an opportunistic process and it is not meaningful to define the search process in detail. However, some guidelines about the main issues involved in the search can be presented.

It is important to use several sources, or leads, of information in the search process. Relying on a single source limits the search space drastically. If the search for OTS software is repeated often in an organization, it is a good idea to document the possible sources well so that access to these is as easy as possible. Typical sources include the following:

- *In-house reuse libraries*: an organization may have an internal library of components that have been developed for reuse. This internal reuse library should be used to determine whether any suitable components exist.
- *Internet and World Wide Web*: they contain large amounts of up-to-date information on most commercial and shareware software products. Some search facilities may be used, e.g., the following may provide good leads:
  - Yahoo -- http://www.yahoo.com/
  - Lycos -- http://www.lycos.com/
  - InfoSeek -- http://www.infoseek.com/
  - CUI W3 Catalog -- http://cuiwww.unige.ch/w3catalog
  - WWW Virtual Library -- http://www.w3.org/hypertext/DataSources/bySubject/Overview.html
- *Magazines and journals*: there are several magazines that contain reviews of products and large amounts of advertisements. Many of these are dedicated to the type of platform (e.g., Mac or MS-Windows), given technology (e.g., object oriented programming, user interfaces or databases) or application area.
- *Trade shows and conferences*: many conferences include extensive vendor exhibitions where it is possible to see several products at the same time, ask detailed questions and order for more information.
- *Vendors*: once some vendors have been recognized, one of the best ways to identify the most important competitors is to ask the vendors directly (“what are your main competitors and how is your product different from them?”).
- *Colleagues, experts and consultants*: it is important to utilize the network of people that may have been exposed to reuse candidates.
- *Other organizations*: other organizations may have developed software that has the required features and functionality. They may have internal reuse programs that may make it easy to access a large amount of reuse candidates. Even when there is no reuse library and there may not be components that have been developed for reuse, it may be possible to identify similar applications and define joint development efforts. The potential for such sharing of reusable components is particularly promising in

the government domain, as the proprietary and competitive issues are not as big of a problem as they may in industry.

The search process can be initiated as soon as the main features of the required component have been defined. In other words, the entry criterion is: main features for the reuse candidates have been defined.

One main challenge in the search is the difficulty of deciding when to stop the search: how do you know that you have searched enough and found all the relevant alternatives ? A simple strategy for ending the search is to use several sources in the search, conduct the search in small increments (e.g., a few days at a time) and review the frequency of discovering new alternatives at each increment. When the all sources have resulted in more or less the same set of alternatives and new alternatives have not appeared for a while, there is a reason to believe that the marginal benefits of additional searches are low.

Note that the search process can also influence the requirements defined for the whole system and, consequently, the reuse goals set for the project. It is quite possible that when new tools are encountered, they trigger new ideas about the possible functionality in the application. This is an important feedback mechanism that can be used to enhance the development process and user satisfaction.

3.5 Screening

The objective of the screening process is to decide which alternatives should be selected for more detailed evaluation. In most cases the results of the search process are too general to be taken as the basis for the OTS software reuse decision. Evaluating and analyzing all the relevant characteristics of any one alternative takes a non-trivial amount of time, typically more than the organization has available for evaluating all of the alternatives. Therefore, it is both necessary and cost-effective to select the most promising candidates for detailed evaluation.

Screening is based on the same, or extended, criteria that was used in the search process. In screening, the “qualifying thresholds” are defined. In other words, the criteria and rationale for selecting alternatives for detailed evaluation is defined and documented.

The screening process can be initiated as soon as there is at least one relevant alternative to consider. This may be, in fact, a necessary way to shorten the overall duration of the selection process: arrangements for obtaining copies of tools for evaluation can be initiated as soon as decisions are made. While incremental screening decisions may result premature decisions and it may be theoretically biased, in practice it may be a good way to reduce the overall duration of the process.

Screening is considered to be complete when evaluation alternatives selected and evaluation tasks have been assigned. Note that when the screening is done, the process may need to reactivated if new alternatives are discovered.

3.6 Evaluation

The objective of the evaluation process is to evaluate the selected alternatives by the evaluation criteria and document evaluation results. Evaluation produces data on how well each alternative meets the criteria defined.

Evaluation includes the practical arrangements of obtaining copies of the tools to be evaluated, installing them, learning to use them, studying their features and assessing them against the criteria. It is important to point out that there can be considerable time delays in the evaluation process. Procurement for obtaining legitimate copies of tools may take time, as well as shipping and handling, there may be significant installation problems due to compatibility problems, and it may take a considerable amount of time to learn to use the tool. Given the potential for delays, it is recommended that evaluation process is initiated as early as possible.

The evaluation criteria typically is so comprehensive that all of it may not be covered within the time available for evaluation. Therefore, the ranking of importance of evaluating each criteria should be used as a guideline in evaluation. Nevertheless, it is still quite likely that not all data for the criteria is available. Missing data will need to be handled in the analysis process.

The results of the evaluation phase will be documented using the evaluation attribute template defined by the evaluation criteria definition process. The results of the evaluation are documented, together with a qualitative description that elaborates any issues that could be relevant in interpreting the outcome of the test.

The evaluation is completed when all alternatives have been evaluated by the defined criteria or required data has been determined not to be available.

3.7 Analysis of Results

The evaluation of alternatives in the OTSO method concentrates in producing consistent data about the alternatives. We deliberately want to separate the analysis of this data from producing the data. This allows the use of appropriate techniques in evaluation data analysis for decision making.

The analysis process of graphically presented in Figure 7. Note that the hierarchical decomposition of criteria and their weighting has already been done in the criteria definition process. The OTSO method also assumes the use of AHP as a multiple criteria decision making technique.

As Figure 7 shows, there are three particular tasks in the evaluation. Cost and value estimation can be done independently when the baseline has been established. Cost estimation can be done using the method that is most appropriate for the organization and the qualitative characteristics of each alternative are evaluated using the AHP method as we described in section 2.3.3 Estimating the Value of OTS Alternatives.

Cost estimation process can produce more than one cost related figure. The most common cost figures are direct financial costs (relating to the out-of-pocket costs of acquiring the OTS product), but sometimes it may be beneficial to consider effort (i.e., the resource load) and cycle time (i.e., expected elapsed calendar time) as well. They may give further insights to the indirect costs that may be involved in OTS software usage.

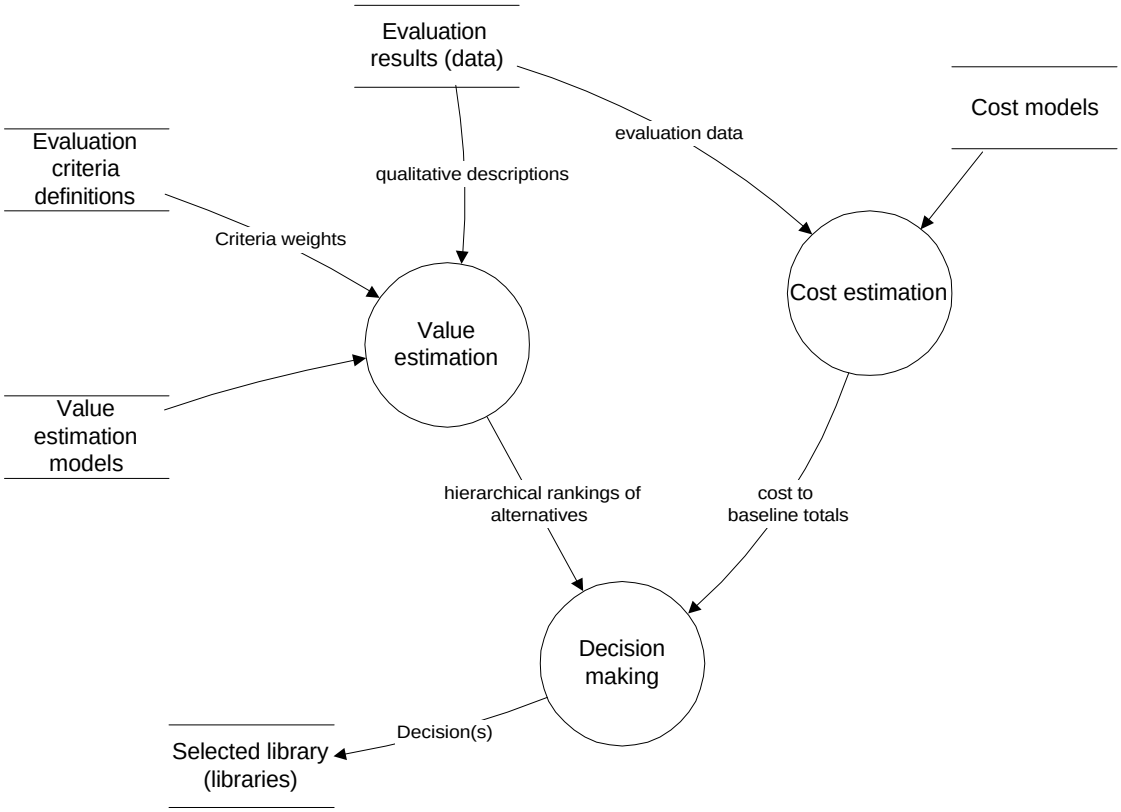

Figure 7: Analysis of results process

The weighting of alternatives is done using the AHP method, preferably using a supporting tool, as we described in section 2.3.3 Estimating the Value of OTS Alternatives.

Preferences are collected and consolidated to the level stakeholders prefer. The AHP allows the consolidation of all qualitative information and the financial information into a single ranking of alternatives. However, we believe that this would condense valuable information too much. Instead, we recommend that information about the evaluation is consolidated to a level where a few main items remain and stakeholders can discuss their impact and preferences verbally. The full consolidation can be done at the end as a sanity check, if desired. The selection of what are the right main items depends on the reuse goals and the criteria hierarchy used.

Decision making consists of consolidating the evaluation data into suitable level and format for the decision makers.

4. Experiences from Case Studies

We initially developed the OTSO method based on experiences gained in an exploratory case study project. This project allowed us to test some of the concepts and ideas in the OTSO method and provided valuable feedback on the practical use of the method. We conducted a slightly more formal case study after the ReMap project and we were able to obtain some initial, qualitative indications of the usefulness of the method.

Both case studies were carried out with Hughes corporation in the EOS program being developed for NASA. The EOS program is developing systems that integrate Earth environmental data from satellites and makes this data available to scientists all over the world. The project required a hypertext browser to provide easy-to-use access to the system.

In the following we present the main results and observations from both case studies. Note, however, that we do not yet have any strong proof of the usefulness of the method as both case studies have been limited in scope and they have not been designed as formal experiments.

4.1 ReMap

The objective of our first case study, the ReMap project, was to select an OTS package for the “map application” in the EOS system. The map application is an application that allows users define areas graphically on earth’s surface. These areas are used to select data in the EOS database.

The main alternatives for the map application were the rudimentary demonstration prototype developed earlier in the project, a product (UIT) from European Space Agency, and two commercial products, Delphi and STK. However, it became apparent early in the evaluation process that UIT was superior to other alternatives in terms of matching the required functionality. Therefore, an official decision to select UIT was done before the OTSO method was thoroughly carried out. As we wanted to test our method in any case, we conducted a “simulation” of the process to the end, i.e., we conducted a limited analysis of all alternatives using most of the criteria and carried out financial and qualitative comparisons. While the reliability of the results in such a limited case study is low, this allowed us to try out the OTSO process. Nevertheless, the outcome of our analysis seemed to support the early decision made in the project.

The ReMap project used two primary sets of criteria in the selection: functional requirements and product quality characteristics. Figure 8 presents the product quality characteristics that were used in the project. Note that the Figure 8 does not contain all the evaluation criteria used for the ReMap project. Full definitions of the criteria and evaluation results are documented separately [31].

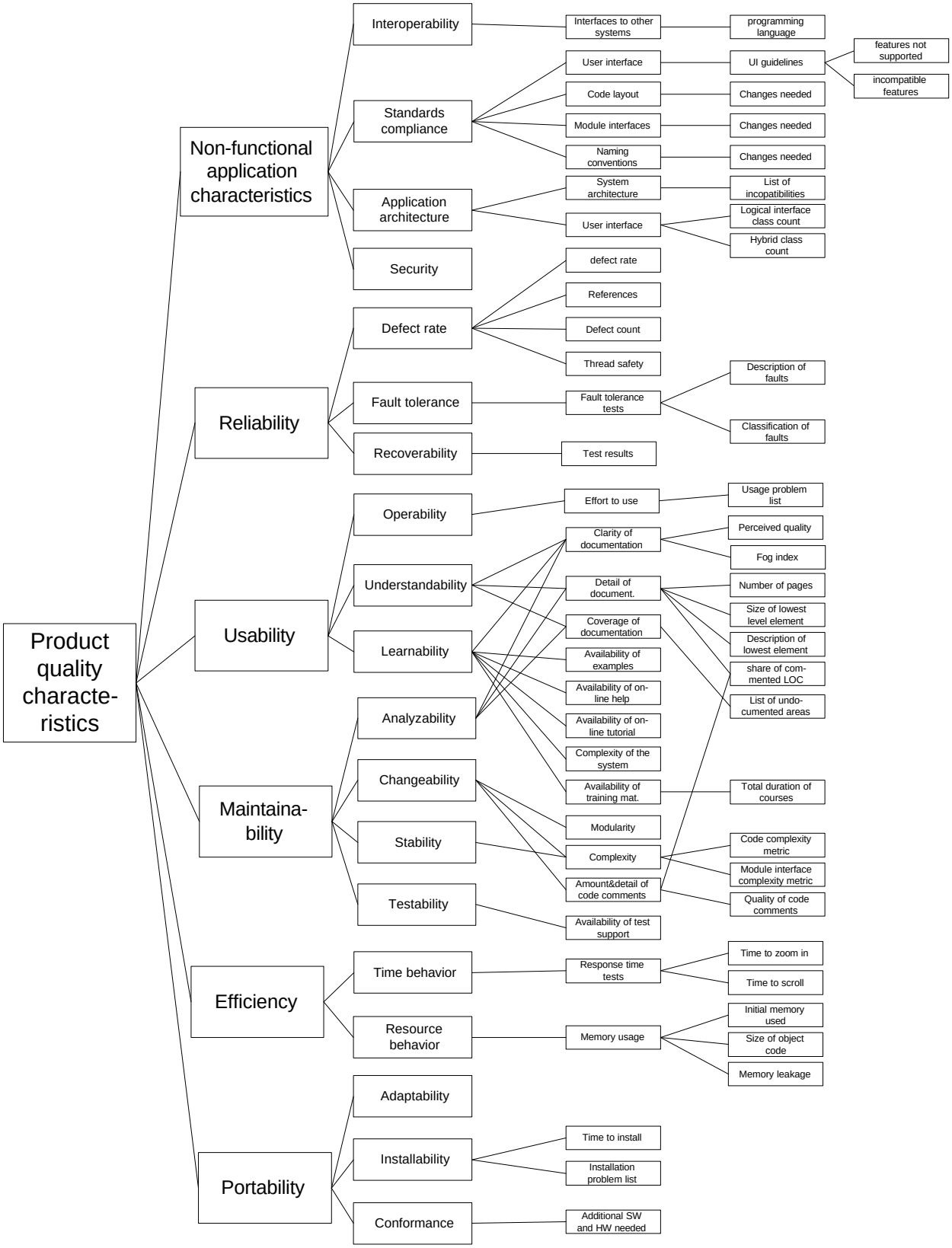

Figure 8: Product quality characteristics in the ReMap project

In addition to providing feedback to the development of the OTSO method, the ReMap project lead us to make some other observations that are useful in similar selection processes. First, it was necessary to refine the stated requirements significantly in order to develop a meaningful evaluation criteria set. We believe that this is a common phenomenon. When the OTS software selection process takes place, requirements are typically not defined in much detail. Yet detailed requirement definitions are necessary for evaluating different products. The interaction of the OTS software selection process and requirements definition process is essential. We also witnessed that evaluating OTS alternatives not only helped in *refining* the requirements, it also lead to *extending* of requirements in some situations. This is an additional challenge in requirements management. In our case the extended requirements were limited to the area of the COTS but it is also quite conceivable that the evaluation process influences the whole system requirements.

Second, a considerable amount of calendar time may need to be spent on installation and logistics before the evaluation can commence. In our case, this limited the time available for detailed evaluation. If the OTS software selection is in the critical path in the project, some attention needs to be placed on the logistics and procurement so that unnecessary delays are avoided. Overall, it seems that the actual effort spent on evaluating each alternative was actually not very high but the calendar time elapsed was.

The third observation in the project was that project personnel were unsure about where to look for alternatives and when to stop the search. We already alluded to this earlier in this document by emphasizing the need for search support and defining a possible strategy for ending the search in section *3.4 Search*.

Finally, despite all the efforts in criteria definition and evaluation, there will inevitably some data that is missing. This may be because the data is simply not available, because it would be too costly to obtain the data or the data is not available in time.

4.2 Hypertext Browser Selection

The objectives of the second case study [28] were to (i) validate the feasibility of the evaluation criteria definition approach in the OTSO method and (ii) compare the AHP and WSM methods for analyzing the data. Our hypotheses were that the more detailed evaluation criteria definition will result in more effective, consistent and reliable evaluation process. We also expected that the AHP method would give decision makers more confidence in the decisions they made.

A total of over 48 tools were found during the search for possible tools. Based on the screening criteria, four of them were selected for hands on evaluation: Mosaic for X, Netscape, Webworks for Mosaic, and HotJava. These tools were each evaluated by two independent evaluators, most of whom evaluated two tools. This arrangement allowed each evaluator to have more than one reference point and enabled discussion and comparison of tools. All evaluators were Hughes project personnel and two of them acted as “key evaluators”, i.e., they had previous experience in OTS software selection and they coordinated much of the evaluation and analysis in the case study.

Criteria/tests	weight score	weight%	Mosaic for X	Netscape	Webworks	HotJava
Test: Level 2 compatibility	5	3.4%	3	3	3	3
Test: HTML Level 3 compatibility	5	3.4%	0	3	0	0
.....	...	...	...	...	...	...
Test: Required disk space	2	1.4%	1	2	2	5
Test: Ease of installation	3	2.1%	5	5	5	5
Test: Popularity of the tool	4	2.8%	3	5	0	0
Total of weight scores	145	1				
Score			470	591	467	427

Table 6: The results of the weighted scoring method (WSM)

The evaluation criteria were derived from existing, broad requirements. However, it was soon discovered that the level of detail in the documented requirements was clearly insufficient for detailed technical evaluation of the OTS software selected. The requirements had to be elaborated and detailed substantially during this process.

The hands-on evaluation was based on the evaluation criteria defined earlier and evaluators wrote reports addressing all the evaluation criteria for each tool. The results were discussed in a joint meeting with all evaluators, and all open issues or conflicting evaluation results were logged and assigned as action items.

In the joint evaluation meeting the differences between tools were first discussed qualitatively. The scores for the WSM were also assigned for each tool were assigned after this discussion. In most cases only the “high” and “low” values were explicitly defined for the scores verbally before assigning a score. Note that the scores were, therefore, mostly based on ordinal scales.

The qualitative differences between tools were documented separately [29]. This report represents the “raw” differences between the tools without any judgments of their importance or ranking to each other.

After the evaluation meeting we waited for two weeks before continuing with the analysis, partially because of the logistics of completing the action items on missing information, partially to allow the two key evaluators to “forget” the numerical scores that were assigned to alternatives.

The final step in the analysis was to complete the WSM by defining the criteria weights

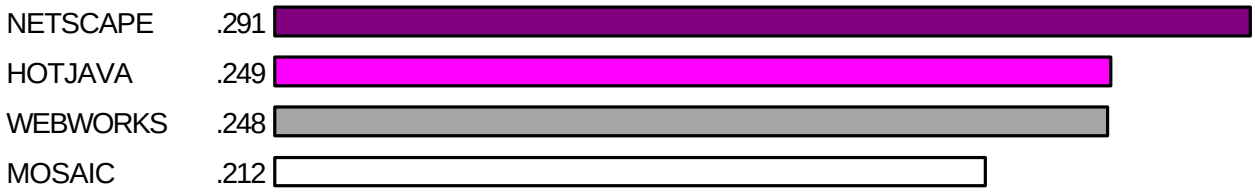

Figure 9: Results of the AHP method

and to apply the AHP method. The WSM criteria weights were assigned using a scoring method, i.e., assigning a value between one and five to each criterion. For this purpose, a “flat” table of criteria was produced from our originally hierarchical criteria definition. The two key evaluators did this independently and they discussed the scores until they reached a consensus on each. The moderator in the session was responsible for making sure that both evaluators' opinions had an equal weight in the discussion. The results of the WSM are presented in Table 6, which also includes some of the criteria, their weight score, weight in percentages and tool scores for some criteria. Note that we have only included five examples of the total of 38 criteria actually used.

The AHP method was applied using the Expert Choice tool [40] in an on-line session. We first ranked the criteria hierarchically and then proceeded to rank the alternatives against each leaf-level criterion. The results of the AHP analysis method are presented in Figure 9.

The total effort distribution for the evaluation process is presented in Table 7. After the case study was completed, we interviewed the key evaluators for their experiences, observations and perception of the process. Specifically, they were asked to state how useful they found the criteria definition process and the two analysis methods used.

As far as the OTSO criteria definition process is concerned, we noticed that the effort spent in criteria definition, 44 hours (i.e., 28% of the total effort) was within the range we expected it to be. According to our interviews with the evaluators, it clearly had a positive impact in the efficiency, consistency and quality of evaluations. Evaluators were given a well-defined template that allowed them to focus on important characteristics of the tools, they were able to discuss the features with each other and evaluation results were relatively consistent. However, despite our efforts, there were still some vague definitions for some of the criteria: a couple of criteria definitions were misunderstood and one was found to be irrelevant during evaluation.

The comparison of WSM and AHP methods supported our hypothesis. The results of the WSM seemed reasonable for the evaluators but the WSM table did not provide any insights to the sensitivity of the results. Also, with 38 criteria, it was rather difficult to see the big picture in the data.

The AHP method was initially perceived as difficult as its calculation model is more complex and it involves a high number of paired comparisons. In fact, during the evaluation we made 322 such comparisons, not including revisions made to some comparisons. This would not have been practical without a graphical, easy to use tool that allowed this to take place within a single session.

The large volume of individual assessments in the AHP method is perhaps its main weakness. Even though the overall duration of the assessment session was not too long, the repetitive assessments may cause fatigue in evaluators. Fortunately, the paired assessment method automatically produces information on how consistent the evaluations are, which would be the likely result of fatigue. The moderator can monitor the consistency, as we did in our case study, and call a break if consistency rates become alarming.

Activity	Effort (hrs)		%
Search		20	14%
Screening		8	6%
Evaluation		79	55%
Criteria definition	40		28%
Mosaic for X	10		7%
Netscape	9		6%
Webworks	9.5		7%
HotJava	10.5		7%
Analysis/WSM		5	3%
Analysis/AHP		7	5%
Management/administration (planning meeti		20	14%
Learning about the methods and techniques		1	1%
other (vendor contacts, installations)		4	3%
Total		**144**	

Table 7: Effort distribution in the OTSO case study

The AHP method allowed the key evaluators to analyze the results from various perspectives and play what-if scenarios by changing weights for different criteria groups. Also, the redundancy built into the paired ranking method and possibility to get feedback on the consistency of comparisons further increased the confidence in results. As the AHP method produced ratio scale rankings, the method produced more information for decision making. The key evaluators agreed that the AHP method produced more information and more reliable information for decision making.

The surprising result in our case study was that the relative ranking of browsers was different using the two analysis methods. Both methods ranked Netscape as the best alternative, but the remaining order of tools was different as Table 6 and Figure 9 show. WSP ranked HotJava the worst but AHP ranked it the second, although very close to Webworks. The WSP could not differentiate between Webworks and Mosaic for X but AHP found clear differences between them.

Given the serious limitations of the WSP approach, we believe that the AHP results are more reliable and they better represent the real rankings between the tools. We draw this conclusion primarily based on the confidence the evaluators had with the analysis results. Unfortunately, it is practically impossible to verify this conclusions with certainty. As in all multiple criteria decision making situations involving future behavior of a system, preferences change over time and between initial information may have contained errors, individuals, situations and information available may change and objectives may also change.

The fact that the analysis method can have such a big impact on the evaluation results is striking. The last couple of hours in OTS software selection can be decisive. Therefore, it seems that the selection of the analysis method should be done with care in COTS evaluation.

5. Conclusions

The OTSO method was developed to consolidate some of the best practices we have been able to identify for COTS selection. We believe that the primary benefit of formalizing the COTS selection process is that it allows further improvement of it. A repeatable process supports learning through experience.

Our limited case study was intended to provide practical experience in applying the method and to provide some indication of its feasibility in practice. The case study seemed to suggest that the method is practical and it may improve the COTS selection process if it is currently conducted in an ad hoc manner.

In particular, the requirements driven, detailed evaluation criteria definition seemed to have a positive impact on the evaluation process. Furthermore, the marginal cost of more formal criteria definition seems to be within acceptable range as it can be accomplished within person days of effort. It also can have a positive effect on the definition of the application requirements.

Second, our case study showed that the AHP method can produce more relevant information for COTS selection and this information is perceived as more reliable by decision makers. At the same time, the additional cost of applying AHP is small, compared to the WSM approach. However, when the number of alternatives and criteria are small, WSP may still be a reasonable method to use, provided that its limitations are taken into account and compensated.

Third, our case study also showed that the choice of the evaluation data analysis method can have more than a minor impact on evaluation results. If this is true in the general case as well, this has strong implications on the way evaluation data should be handled in COTS selection cases.

The case study reported in this paper provided initial results and practical feedback on main aspects of the OTSO method. It seems that the OTSO method addresses important and often ignored problems in COTS usage. However, due to the limited number of data points, i.e., evaluators and cases, the results are not conclusive. We plan to carry out additional case studies and experiments to validate our method further.

6. References

[1] B. Barnes, T. Durek, J. Gaffney, and A. Pyster. A Framework and Economic Foundation for Software Reuse. In: *Tutorial: Software Reuse: Emerging Technology*, ed. W. Tracz. Washington: IEEE Computer Society, 1988.pp. 77-88.

[2] V. R. Basili. Software Development: A Paradigm for the Future (keynote address). In: *Proceedings of the 13th Annual International Computer Software and Applications Conference (COMPSAC)*, Anonymous1989.

[3] V. R. Basili, Software Modeling and Measurement: The Goal/Question/Metric Paradigm CS-TR-2956, 1992. Computer Science Technical Report Series. University of Maryland. College Park, MD.

[4] V. R. Basili, G. Caldiera, and G. Cantone, A Reference Architecture for the Component Factory, *ACM Transactions on Software Engineering and Methodology*, vol. 1, 1. pp. 53-80, 1992.

[5] V. R. Basili, G. Caldiera, and H. D. Rombach. The Experience Factory. In: *Encyclopedia of Software Engineering*, Anonymous New York: John Wiley & Sons, 1994.pp. 470-476.

[6] V. R. Basili, G. Caldiera, and H. D. Rombach. Goal Question Metric Paradigm. In: *Ensyclopedia of Software Engineering*, ed. J. J. Marciniak. New York: John Wiley & Sons, 1994.pp. 528-532.

[7] V. R. Basili and H. D. Rombach, Tailoring the Software Process to Project Goals and Environments, pp. 345-357, 1987. Proceedings of the 9th International Conference on Software Engineering. IEEE Computer Society Press.

[8] V. R. Basili and H. D. Rombach, Support for comprehensive reuse, *Software Engineering Journal*, vol. September. pp. 303-316, 1991.

[9] T. Birgerstaff and C. Richter, Reusability Framework, Assessment, and Directions, *IEEE Software*, vol. March. pp. 41-49, 1987.

[10] B. W. Boehm, J. R. Brown, and M. Lipow, Quantitative Evaluation of Software Quality, pp. 592-605, 1976. Pcodeedings of the Second International Conference on Software Engineering. IEEE.

[11] T. B. Bollinger and S. L. Pfleeger, Economics of Reuse: issues and alternatives, *Information and Software Technology*, vol. 32, 10. pp. 643-652, 1991.

[12] G. Boloix and P. N. Robillard, A Software System Evaluation Framework, *IEEE Computer*, vol. 28, 12. pp. 17-26, 1995.

[13] A. T. W. Chu and R. E. Kalaba, A Comparison of Two Methods for Determining the Weights Belonging to Fuzzy Sets, *Journal of Optimization Theory and Applications*, vol. 27, 4. pp. 531-538, 1979.

[14] P. B. Crosby. *Quality is Free*, New York: McGraw-Hill Book Company, 1979.

[15] T. Davis, Toward a reuse maturity model, eds. M. L. Griss and L. Latour. pp. Davis_t-1-7, 1992. Proceedings of the 5th Annual Workshop on Software Reuse. University of Maine.

[16] W. E. Deming. *Out of the Crisis*, Cambridge: Massachusetts Institute of Technology, 1986. 507 pages.

[17] M. S. Deutsch and R. R. Willis. *Software Quality Engineering - A total Technical and Management Approach*, Englewood Cliffs: Prentice-Hall, 1988. 317 pages.

[18] T. Ellis, COTS Integration in Software Solutions - A Cost Model, 1995. Proceedings of the NCOSE International Symposium "Systems Engineering in the Global Marketplace".

[19] G. R. Finnie, G. E. Wittig, and D. I. Petkov, Prioritizing Software Development Productivity Factors Using the Analytic Hierarchy Process, *Journal of Systems and Software*, vol. 22, pp. 129-139, 1995.

[20] E. H. Forman, Facts and Fictions about the Analytic Hierarchy Process, *Mathematical and Computer Modelling*, vol. 17, 4-5. pp. 19-26, 1993.

[21] M. L. Griss, Software reuse: From library to factory, *IBM Systems Journal*, vol. 32, 4. pp. 548-566, 1993.

[22] D. Harel, On Visual Formalisms, *Communications of the ACM*, vol. 31, 5. pp. 514-530, 1988.

[23] S. Hong and R. Nigam. Analytic Hierarchy Process Applied to Evaluation of Financial Modeling Software. In: *Proceedings of the 1st International Conference on Decision Support Systems, Atlanta, GA*, Anonymous1981.

[24] J. W. Hooper and R. O. Chester. *Software Reuse: Guidelines and Methods*, R.A. Demillo (Ed). New York: Plenum Press, 1991.

[25] ISO. *ISO 9001, Quality systems -- Model for quality assurance in design/development, production, installation and servicing*, International Standards Organization, 1987.

[26] ISO. *Information technology - Software product evaluation - Quality characteristics and quidelines for their use, ISO/IEC 9126:1991(E)*, Geneve, Switzerland: International Standards Organization, 1991.

[27] P. Koltun and A. Hudson, A Reuse Maturity Model, ed. W. B. Frakes. pp. 1-4, 1991. Proceedings of the 4th Annual Workshop on Software Reuse. University of Maine. Department of Computer Science.

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

[29] J. Kontio and S. Chen, Hypertext Document Viewing Tool Trade Study: Summary of Evaluation Results 441-TP-002-001, 1995. EOS project Technical Paper. Hughes Corporation, EOS project.

[30] J. Kontio, S. Chen, K. Limperos, R. Tesoriero, G. Caldiera, and M. S. Deutsch, A COTS Selection Method and Experiences of Its Use, 1995. Proceedings of the 20th Annual Software Engineering Workshop. NASA. Greenbelt, Maryland.

[31] J. Kontio, K. Limperos, R. Tesoriero, and G. Caldiera. *ReMap: Evaluation Criteria Definition*, 1995. (UnPub)

[32] H. Min, Selection of Software: The Analytic Hierarchy Process, *International Journal of Physical Distribution & Logistics Management*, vol. 22, 1. pp. 42-52, 1992.

[33] S. L. Pfleeger and T. B. Bollinger, The Economics of Reuse: New Approaches to Modeling and Assessing Cost, *Information and Software Technology*, vol. 1994.

[34] J. S. Poulin, J. M. Caruso, and D. R. Hancock, The business case for software reuse, *IBM Systems Journal*, vol. 32, 4. pp. 567-594, 1993.

[35] R. Prieto-Díaz, Implementing faceted classification for software reuse, *Communications of the ACM*, vol. 34, 5.1991.

[36] C. V. Ramamoorthy, V. Garg, and A. Prakash, Support for Reusability in Genesis, pp. 299-305, 1986. Proceedings of Compsac 86. Chicago.

[37] T. L. Saaty. *Decision Making for Leaders*, Belmont, California: Lifetime Learning Publications, 1982. 291 pages.

[38] T. L. Saaty. *The Analytic Hierarchy Process*, New York: McGraw-Hill, 1990. 287 pages.

[39] T. L. Saaty. Analytic Hierarchy. In: *Encyclopedia of Science & Technology*, Anonymous McGraw-Hill, 1992.pp. 559-563.

[40] T. L. Saaty, Expert Choice software 1995, ver. 9, rel. 1995. Expert Choice Inc. IBM. DOS.

[41] W. Schäfer, R. Prieto-Díaz, and M. Matsumoto. *Software Reusability*, W. Schäfer, R. Prieto-Díaz, and M. Matsumoto (Eds). Hemel Hempstead: Ellis Horwood, 1994.

[42] P. J. Schoemaker and C. C. Waid, An Experimental Comparison of Different Approaches to Determining Weights in Additive Utility Models, *Management Science*, vol. 28, 2. pp. 182-196, 1982.

[43] W. Tracz, Reusability Comes of Age, *IEEE Software*, vol. July. pp. 6-8, 1987.

[44] W. Tracz. *Tutorial: Software Reuse: Emerging Technology*, W. Tracz (Ed). Washington: IEEE Computer Society, 1988.

[45] W. Tracz. Software Reuse: Motivators and Inhibitors. In: *Tutorial: Software Reuse: Emerging Technology*, ed. W. Tracz. Washington: IEEE Computer Society, 1988.pp. 62-67.

[46] W. Tracz, Legal obligations for software reuse, *American Programmer*, vol. March.1991.

[47] M. Wasmund, Implementing Critical Success Factors in software reuse, *IBM Systems Journal*, vol. 32, 4. pp. 595-611, 1993.

[48] F. Wolff, Long-term controlling of software reuse, *Information and Software Technology*, vol. 34, 3. pp. 178-184, 1992.