COTS Products Characterization

Marco Torchiano, Letizia Jaccheri, Carl-Fredrik Sørensen, and Alf Inge Wang
Department of Computer and Information Science (IDI), Norwegian University of Science and Technology (NTNU)
Sem Saelands vei 7-9, N-7491 Trondheim, Norway
Phone: +47 735 94489, Fax: +47 735 94466
*{marco,letizia,carlfrs,alfw}@idi.ntnu.*no

ABSTRACT

A way to learn about Commercial Off-The-Shelf (COTS) products is to define a set of characteristics or attributes and then to collect information about these attributes. In an industrial context, the attributes used to select COTS clearly depend on project specific goals. In our educational context we made an attempt to define general COTS characterization attributes. The resulting framework provides a structure, which facilitate the learning process. Our proposed attributes have several similarities with the generic evaluation attributes defined by ISO 9126. A comparison with such a standard provides a deeper insight into the problem of characterizing COTS products.

Categories and Subject Descriptors

K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software selection.*

General Terms

Measurement, Experimentation.

Keywords

COTS, Selection, Learning, Metrics.

1. Introduction

The topic of COTS based development has become very important in industry and research [8]. At our institution, we recognized the need for a new master level course on COTS software technology. But how do we teach and learn about COTS products?

A way to learn about Commercial Off-The-Shelf (COTS) products is to define a set of characteristics or attributes and then to collect information about these attributes

The motivation for our work is to develop a structured approach for COTS exploration and learning in a pedagogical setting. Our course is supposed to be a support course for master students who will work with technology based software projects. In the Internet era, new technologies and products arise at a high rate; the current COTS knowledge quickly becomes obsolescent. Our pedagogical approach is an attempt to cope with such a contraction of time by providing students with a reusable and structured framework.

Knowledge related to COTS products is subject to continuous evolution, conventional text books and classroom teaching tend to be structured, and thereby not suitable to convey this kind of knowledge.

The ISO/IEC 9126 standard [4] represents an important reference point in software product characterization. This standard defines attributes to measure the quality of software products, but it was designed with a different usage scenario: to characterize large monolithic applications. A lot of insight in the characterization attributes adopted in the experiment can be gained by comparing with the ISO 9126 standard.

In this paper we try to answer the research question: "*Which attributes are useful to characterize COTS products for the purpose to learn about them?*" The problem of learning about COTS products is important in industrial settings too, because the time spent for learning activities is expensive and often scarce.

This paper proposes a set of attributes for COTS product characterization in the context of learning, and shows why ISO 9126 is not sufficient for this purpose.

The context is a process of technology learning that is based upon two main phases: classification and characterization [4]. The key point of this approach is the distinction between classification and characterization attributes. The former can be used to identify homogenous groups of products; the latter are intended to provide a tool for characterizing them.

This paper is structured as follows: section 2 introduces the context of this work, section 3 describes the proposed characterization attributes and the process used to devise them, section 4 presents the related work, section 5 provides a comparison with the ISO 9126 framework, and section 6 draws some conclusions.

2. Context

The benefits of the proposed work can be found at the conjunction of the issues posed by the COTS-based development and the education of people that will meet these issues in their careers.

The use of Commercial Off-The-Shelf (COTS) products, as opposed to the development of components in-house, has the potential to lead to faster development, reduced effort, and higher quality. The products are already available and thereby no resources are required to develop the functionalities they offer. This could also result in a cheaper process where the cost of adopting the COTS product is less than the cost of developing the required functionalities from scratch. The products have been developed by a team, which hopefully has expertise both in the specific domain and in the development techniques.

On the other hand there are several possible scenarios for the adoption of COTS products, which are characterized by different levels of renegotiation of the system requirements and architecture. Each scenario poses different issues.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SEKE '02, July 15-19, 2002, Ischia, Italy.
Copyright 2002 ACM 1-58113-556-4/02/0700…$5.00.

Well-defined and crystallized requirements, and probably architecture, require a thorough COTS selection process and possibly a lot of effort in COTS adaptation.

Renegotiable requirements and architecture allow a more relaxed selection process. There is a balance between adapting the system (requirements and/or architecture) to the available products and vice versa.

Vague and mostly undefined requirements require a technology harvesting process in order to find the most promising COTS products, and then the requirements can be adapted to fit the characteristics of such products.

While learning about new software is important in all kind of scenarios, the latter scenario can draw the most benefits from a structured approach.

The pedagogical context is a fifth year course in the computer science master degree at Norwegian University of Science and Technology (NTNU), Norway [7]. The course was run in parallel to other courses and was intended to provide support to them.

The course is based on student activities, which are supervised by the teacher and teaching assistants. These activities can be summarized as follows: (1) Define classification criteria, (2). Identify COTS products and classify them, (3) Define characterization attributes, (4) Assign attribute values for the identified COTS products, and (5) Data analysis.

Step 1 and 2 are presented in [5]. The goal of this paper is to discuss the proposed characterization attributes, which resulted from phase (3).

3. Software Characterization Attributes

This section first introduces our characterization attributes, which are an answer to the question: "*Which attributes are useful for characterizing COTS products for learning purposes?*" Then, it gives an explanation about the process to define them.

3.1 The Attributes

Here we provide a description of the attributes, their operational definition, how they can be measured and some examples. The examples are taken from the work done by the students during the course.

Product maturity: is described by means of a story related to the maturity of the product, e.g. in terms of years on the market and features stability.

Market Share: is the ratio/part of the market of similar products that is covered by this COTS product.

Performance: is here meant as the scalability, which can be measured as the number of users it can scale to without sensible decrease in response time.

Safety/Security: is the support offered by the product for developing secure or safety critical systems.

Reliability: a story describing how fault-tolerant the product is.

Hardware requirements: a list of the hardware characteristics required to use the product.

Product support: a summary of the facilities offered to support software development using the product.

Documentation: what kind (web, on-line, etc.) of documentation is provided together with the product and what is its size (e.g. num of manual pages).

Usability: degree of satisfaction on a scale from 1 to 5 and a story describing it in more detail.

Learnability: how much time does it takes to learn to use this component.

Modifiability: how easy it is to modify this product (non-modifiable, parameterizable, provides API).

Change frequency: how many releases/versions in the last year.

License Type: the type of license of the product.

Cost of use: cost of the product.

Software requirements: list of required software to run the component.

Conformance: list of standard to which the products are in conformance to.

Domain specific: if yes, list domains.

3.2 Attribute definition process

The process for defining attributes can be divided into two main phases: (1) students proposed an initial set of attributes, (2) teachers team select the final attributes.

Each student was asked to propose at least six attributes, which he/she believes are significant to describe/evaluate software items. Each attribute must be declared together with its measurement scale. The teacher team then processed the attributes proposed by the students. The selection process is made up of two phases: first merging of similar attributes, then screening of relevant attributes.

Since the students did not use a homogeneous terminology there were many synonyms or very similar attributes. The merging phase addressed this issue.

The screening was conducted as follows. At first the documents with the attribute proposals were fairly assigned to the different members of the teacher team. Each member had to write each attribute on a post-it note. The post-it notes were placed on a wall and then filtered by grouping together duplicates and synonyms.

The educational goal was the main concern when screening the attributes. Thus more generic and easily measurable attributes were selected.

Table 1 shows the results at the end of the selection process. The table shows the number of proposals for each attribute. These data may be used to deduce which of the properties the students weighted as most important.

Table 1: Characterization attributes.

Attribute	**Votes**	**Scale**
Product maturity	3	Qualitative
Market Share	15	Ratio
Performance	12	Ratio
Safety/Security	6	Qualitative
Reliability	9	Qualitative
Hardware requirements	14	Qualitative
Product support	6	Qualitative
Documentation	4	Qualitative
Usability	24	Qualitative
Learnability	7	Qualitative
Modifiability	14	Qualitative
Change frequency	6	Ratio
License Type	16	Nominal
Cost of use	24	Qualitative

Software requirements	3	Qualitative
Conformance	8	Qualitative
Domain specific	5	Qualitative

One of the most evident features of the final attributes is that the scale of 75% of them is qualitative. This fact can be easily explained. The sample attributes were devised by the teacher to be simple and easily understandable, thus they were chosen to be atomic and with a simple scale.

The final attributes are the result of a mediation process. Therefore they are more generic and complex. Instead of decomposing them into atomic attributes we preferred to keep them as they were and to ask for qualitative values.

Qualitative values are more valuable in the educational context where the attributes were defined.

In addition, when looking at the examples of the attributes, there are discrepancies between the type of the attributes and the values provided by the students. We observed that students tend to write stories rather than provide a simple value.

4. Related Work

Related work that deals with characterization of COTS products can be divided into three main categories: classification, evaluation and COTS selection.

The risk of relying on a widely used acronym, such as ”COTS” is that it identifies a very heterogeneous set of items. Therefore it becomes difficult to generalize conclusions and select techniques and methods. An initial attempt to classify COTS products has been done by Carney and Long in [2]. They use a bi-dimensional cartesian space. The dimensions they define are origin and modifiability. A more complex classification of COTS products is presented by Morisio and Torchiano in [9]. The purpose of their proposal is twofold: first it is a tool to precisely define what is the meaning of a COTS product, second it represents a way of distinguishing different sub-classes of products in order to characterize them better. They propose ten attributes, grouped into four areas: source, customization, bundle, and role.

Several methods have been proposed for COTS product characterization. Boloix et al. [1] propose a framework for evaluating technology; it is based on attributes grouped into three *dimension*s: the project, the system and the environment. They allow the attributes to assume only three values: basic, intermediate, or advanced. Recently Ochs et al. [11] proposed a new approach. The COTS Acquisition Process (CAP) provides a more general framework for product characterization. It aims at reducing the effort needed for characterization and in providing the basis for reusing the information acquired during the process. The process is made up of three assets: Initialization, Execution and Reuse. The attributes adopted for the characterization form a super-set of the ISO 9126 set of attributes. They are fixed in order to be reusable.

5. Discussion

In this section, we present our comparisons of the attributes for describing software in our own framework with the ISO 9126 standard for software quality characteristics and metrics. The ISO 9126 standard focuses on software quality and metrics that can be used to evaluate own software or software that have documentation and source code available. This means that the software quality will be evaluated mostly from an internal view.

Our software description framework mostly focuses on describing software through external attributes. We consider external those attributes that can be measured without accessing the intermediate artifact (e.g. design and code) or the development process documentation.

The ISO/IEC standard 9126 [4] provides a set of attributes for evaluating the quality of software systems.

The ISO 9126 standard is made up of six main attributes:

functionality The capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.

reliability The capability of the software to maintain the level of performance of the system when used under specified conditions

usability The capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.

efficiency The capability of the software to provide the required performance relative to the amount of resources used, under stated conditions.

maintainability The capability of the software to be modified.

portability The capability of software to be transferred from one environment to another.

We will now go through the main attributes of the ISO 9126 standard to see how our proposed attributes relate to them. Table 2 compares the ISO 9126 framework with our framework.

Functionality Our attribute *Security and safety* corresponds to the *Security* attribute. We do not cover *accuracy*, *suitability*, *interoperability*, and *compliance*. The rest of the attributes were not considered practical since they require insight into system code and documentation. This depends from the fact that these attributes are related to the internal characteristics of a software system, which are not of interest when evaluating COTS products.

Reliability Our set of attributes covers reliability in general. Moreover it includes the *maturity* attribute. However, we describe maturity from a business point-of-view (mature in the market), while ISO 9126 analyses the intrinsic maturity of the software.

Usability Our attributes include *usability*; in addition they describe it through the attributes *learnability* and *documentation*. The learnability attribute is also defined in ISO 9126. The ISO 9126 does not have the *documentation* attribute, but this attribute cover mostly of the ISO’s attributes *understandability* and *operability*. In addition ISO 9126 have an attribute to describe the capability of the software product to be liked by the user *(likability*); which has not any corresponding within our attributes.

Efficiency Our framework uses the attribute *performance* to describe the same software quality. ISO 9126 further specifies efficiency in terms of *time behavior* and *resource utilization*. Our attribute is measured in a different way; therefore it has no relation to these two more specific attributes.

Maintainability We propose the attribute *modifiability*, which maps easily into the standard’s *changeability* attribute. ISO 9126 does in addition use *analyzability*, *stability*, and *testability* to describe maintainability. Since these attributes require internal access to the product, they are not suitable to our perspective.

Portability This main attribute requires access to source code and other design artifacts, and is therefore not directly included in our attributes. On the other end we propose the *conformance* attribute, which maps to the same attribute in ISO 9126. While the other standard sub-attributes (i.e. *adaptability*, *installability*, *co-existence*, and *replaceability*) have no correspondents.

In addition to the main attributes above, we have added these attributes that are not directly related to the quality in use: Market share, software requirements, hardware requirements, product support, license type, acquisition cost, and domain specific. All these attributes focus on external software attributes that are often used to describe software components when buying off-the-shelf. These attributes are important when evaluating different COTS for determining the actual cost of software acquisition.

Table 2: Mapping of Attributes.

ISO-9126	**Our**
Functionality	-
Accuracy, Suitability, Interoperability, Compliance	-
Security	Safety/Security
Reliability	Reliability
Maturity	Product maturity
Fault tolerance, Recoverability, Availability	-
Usability	Usability
Understandability	Documentation
Learnability	Learnability
Efficiency	Performance
Operability, Time behaviour, Resource utilization	-
Maintainability	
Changeability	Modifiability
Analysability, Stability, Testability	-
Portability	-
Adaptability, Installability, Co-existence, Replaceability	-
Conformance	Conformance
NO ISO 9126 correspondent attribute	Market share, Product support, Change frequency, License type, Cost of use, Domain specific, Hardware requirements, Software requirements,

6. Conclusions

In this paper we presented the definition of a COTS characterization framework. We discussed the framework in light of the ISO 9126 standard, as it is a well-known characterization standard for software systems. This comparison enables us to reflect about the peculiarities of COTS versus those of standard software systems.

34 students have used our framework. We have now collected 4 characterizations for 34 products and two master students are in the process of coding and analyzing these data. The purpose of coding the qualitative data is to provide a searchable database, which can be used for further education and COTS research.

The perspectives of the two frameworks are different. ISO specifies software quality of a software system, used to define the quality of a system whose requirements, architecture, design, and source code are available. Our framework defines attributes that can be used to select a specific component, technology or tool.

ISO 9126 defines each attributes specifically and how to measure them. Some of these measurements are very hard or maybe impossible to do on COTS (e.g. measurements that requires number of functions, number of lines of code, design specifications etc...) Our framework focuses on external quality attributes. ISO 9126 attributes that were adopted in our framework are observable even under the black-box perspective, which is mandatory when dealing with COTS products.

Many interesting attributes are not included in the ISO framework. A problem with ISO is that the framework has a conservative way of looking at software, e.g. traditional software process (waterfall) vs. incremental development, functions vs. objects, heavy and well-defined procedures vs. agile methodologies. In addition ISO does not cover the trivial attributes that are easy to access (hardware/software requirements, acquisition cost etc.).

The challenges we are facing now are: it happens that some products that belong to the same class are in fact not comparable. Moreover, some attributes have been assigned to subjective histories. In those cases where the products were too complex (see Oracle for example) or completely unknown to students or even not installable on their platforms, the collected stories are cut and pasted from the product web sites, thus resulting in marketing oriented stories.

7. Acknowledgements

This work has been supported by the Research Council of Norway funded INCO project. We are grate to the students of SIF80AT course at Norwegian University of Science and Technology.

8. References

[1] Boloix, G. and Robillard, P. A Software System Evaluation Framework. IEEE Computer, 12(8), 1995, 17-26.

[2] Carney, D. and Long, F. What Do You Mean by COTS? Finally a Useful Answer. IEEE Software, 17(2), 2000, 83-86.

[3] ISO. Information Technology – Software Product evaluation – Quality Characteristics and Guidelines for their Use. Int. Standard ISO/IEC 9126, ISO, 1991.

[4] Jaccheri, L. and Torchiano, M. Classifying COTS products. In European Conference on Software Quality, Helsinky, 2002.

[5] Morisio, M. and Torchiano, M. Definition and Classification of COTS: a proposal. In [8], 165–175.

[6] Ochs, M.A., Pfahl, D. and Chrobok-Diening, G. A method for efficient measurement-based COTS assessment and selection - method description and evaluation results. In IEEE 7th Int. Software Metrics Symposium, pp. 285–296, London, England, 2001.

[7] SIF80AT - A course in new software technologies, home page at http://www.idi.ntnu.no/emner/sif80at/.

[8] John Dean, Andre Gravel, eds. COTS-Based Software Systems. Proc. International Conference on COTS Based Software Systems (ICCBSS), Orlando (FL), LNCS vol. 2255, Springer, 2002.