

CQM: A Software Component Metric Classification Model

Joaquina Martín-Albo, Manuel F. Bertoa, Coral Calero, Antonio Vallecillo, Alejandra Cechich and Mario Piattini

Abstract— In the last few years component-based software development (CBSD) has been imposed as a new paradigm in software systems construction. CBSD is based on the idea that software systems can be developed by selecting and integrating appropriate components, which have already been developed, and then assembling them to obtain the functionality desired in the final application. Multiple authors have proposed metrics to quantify several components characteristics in order to help in its selection. Nevertheless, rather than helping developers, such proposals often provoke more confusion due to the fact that they do not systematically take into account different aspects of the components. Trying to achieve clarity in this line, we have developed the CQM model (Component Quality Model), whose first aim is to propose a classification of the defined metrics for software components. The model will also be able to be used for the evaluation of a component or a component system. Finally, it is necessary to indicate that this is the first version of the model, and it will need to be refined by means of its use and discussion in different forums.

Index Terms— Software measurement, measure properties, measurement theory, quality components, model components.

1 INTRODUCTION

THE increasing complexity of software applications is making traditional methods and tools of software development insufficient when dealing with new problems that appear in these types of systems [5]. This is reflected by an important decrease in the final quality, and by an increase in the complexity of maintenance tasks.

Component-based software development (CBSD) appears as a promising technique for solving the problems mentioned above. In general, this technique consists of modular design and development of applications based on software components, developed independently, and suitably combined to compose the final application [20]. The reuse of such components will allow us to reduce the development time and cost, and this will simultaneously allow us to increase productivity [13], [23].

Inside the CBSD, commercial software components are proliferating. These components are constructed by third parties and are previously tested, and their code cannot be modified. These components are known as COTS (Commercial-Off-The-Shelf). The reuse of the mentioned components will allow us to simplify the construction of applications software, since the traditional phases of development and codification are replaced with processes of component search and selection, which can be carried

out in less time and with less effort [2]. Component COTS selection is based on different criteria such as functionality, quality, price, confidence in the component manufacturer, cost of the formation of internal personnel for the use of the component, adjustment degree that needs to be carried out so that the component can be integrated with the requirements of our system, etc.

Improving component-based system quality assessments implies defining metrics, which allows us to quantify multiple component characteristics used for component selection. There has been a lot of research into this area and therefore a number of metrics provides us with a means of measuring component characteristics in diverse areas and stages of software engineering. Nevertheless, there are some classification criteria for all these metrics and every author defines them to measure the characteristics that he or she considers to be more relevant when working with component-based systems.

With the aim of classifying different existing models, we have elaborated the CQM model (Component Quality Model), which can be used to evaluate a given component (or a system formed by several components) and to propose new works of metric definition in those areas that still have not been looked into.

It is necessary to indicate that this is the first version that we do of the model, and for this reason it cannot be considered as a definitive proposal but as a new model that must be evolutionary by its application in different areas.

In the following section we present our model explaining in depth each one of the dimensions that it is composed of. In the third section, we carry out -as an example- the classification according to the CQM model of the metrics proposed for Use Facility and we will carry out a series of commentaries on the said classification. In the

- Joaquina Martín-Albo, Coral Calero and Mario Piattini are with the Alarcos Research Group, Department of Computer Science, University of Castilla-La Mancha. E-mail: {jmartin@proyectos.inf-cr.uclm.es, {coral.calero, mario.piattini}@uclm.es.
- Manuel F. Bertoa and Antonio Vallecillo are with the Department of Languages and Computer Science, University of Málaga. E-mail: {bertoa, av}@lcc.uma.es.
- Alejandra Cechich is with the Department of Computer Science, National University of Comahue. E-mail: acechich@uncoma.edu.ar.

last section the conclusions are presented and future work is proposed.

2 CQM MODEL

Our paper proposes a classification of different metrics that have been proposed for the characterization, evaluation and selection of software components. With this objective, we have elaborated the CQM model (Component Quality Model), taking as a basis the classification of quality models to component evaluation [9] and the WQM model [16], whose authors define a cube structure, which presents the three basic aspects to bear in mind when assessing the quality of a web site. Following this same idea, in our CQM model we propose four dimensions that must be considered in the evaluation of quality of component systems: Quality Characteristics (Distinguishing between External/Internal Quality and Quality in Use), Granularity/Visibility of a complete system and isolated components, Life Cycle Processes and the involved Stakeholder.

In Figure 1, we can observe each of the mentioned dimensions and its characteristics.

2.1 Quality Characteristic Dimension

This dimension has been divided in two subcategories to distinguish between external/internal quality and quality in use of a component-based system [12].

External Quality refers to software execution. It is normally measured and evaluated during tests in a simulated environment. On the other hand, Internal Quality is the quality needed from the internal perspective of the product, and is in the habit of remaining without alterations unless the product is re-designed.

For the description of the External/Internal Quality sub-dimension, we use the quality model proposed by Bertoa and Vallecillo [3] as a basis for COTS components. We decided to work with this model because the authors, starting from the ISO 9126 quality model [12], have singled it out to be applicable to COTS components. It is basically the ISO quality model, where some of the Reliability, Maintainability and Portability sub-characteristics disappear. Moreover other characteristics have changed their meaning in the following context:

- **Functionality**, this characteristic maintains the same meaning for components rather than for a system composed by the integration of several software components. It tries to express the ability of a component to provide the required services and functions, when used under the specified conditions. It presents as sub-characteristics: Suitability, Accuracy, Interoperability, Compliance and Security.
- **Reliability**, this is a set of attributes that refer to the capability of software to maintain its level of performance under stated conditions for a stated period of time. It is directly applicable to components, and essential for reusing them. It

presents as sub-characteristics: Maturity and Recoverability.

- **Usability**, this characteristic and all its sub-characteristics have a completely different meaning for software components. The reason is that, in CBSD, the end-users of components are the application developers and designers that have to build applications with these components, rather than the people that have to interact with them. Thus, the usability of a component should be interpreted as its ability to be used by the application developer when constructing a software product or system with it. It presents as sub-characteristics: Learning capacity, Understandability and Operability.
- **Efficiency**, is a set of attributes that refer to the relationship between the level of performance of software and the amount of resources used, under stated conditions. It presents as sub-characteristics: Time behaviour and Resource behaviour.
- **Maintainability**, this characteristic describes the ability of a software product to be modified. Modifications include corrections, improvements or adaptations to software. The user of a component (the developer) does not need to carry out internal modifications but he or she does need to adapt it, re-configure it, and perform the testing of the component before it can be included in the final product. It presents as sub-characteristics: Changeability and Testability.
- **Portability**, this is defined as the ability of a software product to be transferred from one environment to another. In CBSD, portability is an intrinsic property of the nature of components, which are in principle designed and developed to be re-used in different environments. It presents as sub-characteristic: Replaceability.

From our point of view, the main concern of Quality in Use is the quality vision that the user has of a software product when he/she uses it in an environment and context of specific use. For the description of the sub-characteristics of Quality in Use, we have adopted the ISO 9126 model, which contemplates the following characteristics:

- **Effectiveness**, this is the capacity of product software to allow users to achieve specified aims in accuracy and completeness in a context of specified use.
- **Productivity**, this is the capacity of a software product to allow users to consume a suitable quantity of resources in relation to the level of efficiency achieved in a context of specified use. The time to complete the task, the user's effort and the materials or the financial cost of use can be relevant resources.

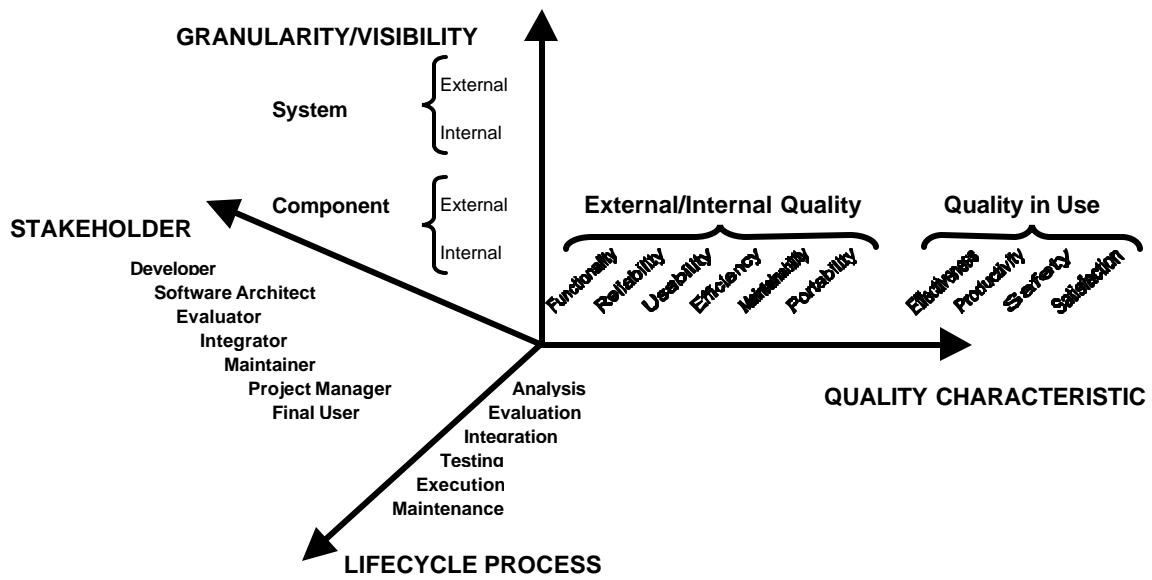


Fig. 1. CQM model dimensions

- **Safety**, this is the capacity of a software product to reach acceptable levels of safety in relation to the risk of causing harm to people, business, software, properties or environment in a context of specified use.
- **Satisfaction**, this is the capacity of a software product to satisfy the users' needs in a context of specified use. Satisfaction is the response of the user to interaction with the product, and includes the attitude towards product use.

2.2 Quality Characteristic Dimension

In this dimension, we consider the system as a whole, or an isolated component. As for the System, two approximations are considered: External, the user sees the system, and he or she only needs to know its functionality (what it does) and how to use it (its interface), and Internal, in that we see the system as a series of interconnected and related components.

In relation to the analogous form for an isolated Component, we distinguish the vision: External, in which the component offers a series of services across a set of interfaces, and Internal, which is the vision that the developer has of the component.

It is important to indicate that the components are "binary units of software application composition, which have a set of requirements, and should be developed, acquired, incorporated into the system and composed with other components, of independent form in time and space" [22]. Therefore, no component user is going to have access to its internal details of implementation that are reserved only for its developer. In general, component implementation adjustment and configuration mechanisms can

be integrated into different types of systems, but the user can never have access to the code of a component and modify it.

2.3 Quality Characteristic Dimension

Component-based software systems are developed by selecting various components and assembling them rather than programming an overall system from scratch, thus the life cycle of component-based software systems is different from that of traditional software systems. This way, based on the possibilities of life cycle models of CBSD, given by Cai et al. [6] and Chang et al. [8], the life cycle of component-based software systems can be summarized as follows:

- *System Requirement Analysis*: Process of discovering, understanding, documenting, validating and managing the requirements of a system.
- *Component Evaluation*: Selection of the suitable components.
- *System Integration*: System integration is the process of assembling components selected inside a whole system under the designed system architecture. The necessary changes should be carried out on a developed component, so it can be used in a specific or corporate environment with other components.
- *System Testing*: the process of evaluating a system to confirm that the system satisfies the specified requirements, and identify and correct defects in the implementation of the system.
- *System Execution*: Application use in its real environment, and for its final users.
- *System Maintenance*: Its objective is to provide services and maintenance activities needed to use

the software effectively after it has been implemented (correction of errors, adaptation of the system to changes in the environment, component substitution for its upgraded versions, etc).

2.4 User Dimension

In this dimension the stakeholders, that have a relationship with the system throughout its life cycle, are identified. Among them we can distinguish:

- *Component Developer*, foreign to the company if COTS components are used, or of the same organization if in-house repositories are used.
- *Software Architect*, which designs the internal architecture of the system, decides the structure of the system on the basis of components and their interrelationships.
- *Evaluator*, which selects most adapted components for the application, in agreement with the requirements (functional, of quality, etc.).
- *Integrator* takes on the task of combining and installing the components that have been selected inside the system, carrying out alterations in the components when necessary.
- *Maintainer*, who develops the necessary maintenance and evolution tasks after establishing the functioning of the system.
- *Project Manager*, who takes control of all the previous tasks and determines the quality level that the system needs to achieve, as well as all the aspects relating to investment costs, production times, etc.
- *Final User*, who only needs to know about system or component functionality (what it does) and how it is used (its interface).

3 METRIC CLASSIFICATION FOR USE FACILITY

As we have already indicated, the primary goal of our model is to help classify metrics proposed for component-based systems. For brevity reasons, only the Usability metrics are presented in this paper as a way of exemplifying the model. These metrics are shown in Table 1, in which some characteristics have been used as descriptors. However, besides this classification, it is fundamental to identify some other characteristics of the metrics [7] as follows:

- *Objectivity/Subjectivity*. The degree of objectivity depends on whether an algorithm or mathematical formula provides us with the value of the metrics, or the mentioned value is given directly by an expert. We must point out that, with the exception of the metrics "User's Documentation" and "Strengthen to Operate" [3], and "Accessibility and Administration" [15], the rest of the metrics of Table 1 are objective. By reasoning about the implications of subjectivity on our examples of

Table 1, we must observe that, as a subjective aspect of a metric, it is important to consider who carries out the evaluation, i.e., who provides the metric value. This has special relevancy in some of the Use Facility metrics. For example, the evaluation of "Correct Use Period" or "Configuration Period" greatly depends on the level of knowledge and experience that a user has of the company that is going to use or to configure the product, and therefore this metric may have very different values in different organizations.

- *Theoretical Validation*. This validation help us to know when and how to apply metrics.
- *Empirical Validation*. Here the objective is to show the practical utility of the proposed metrics by developing some experimental cases. Concerning the metrics summarised in Table 1, it is necessary to emphasize that the models by Sook Cho et al. [21] have been empirically validated, and that these models have been applied to several bank systems. In the other cases, Schach and Yang [17] carried out an experiment on 7 software modules previously chosen; Nguyen et al. [14] developed a case-study on components implemented in C; and B. Bohem [4] carried out a validation on some aspects considered relevant (databases, compilers, emulators, etc.).
- *Automated Support*, i.e., whether or not there is a supporting tool that facilitates the calculation of the metrics. Finally, for the same group of metrics, there is no supporting tool for metrics calculation.

4 CONCLUSIONS AND FUTURE WORK

The current trends in the investigation and development of software engineering indicate that component-based software development (CBSD) is being imposed more and more. Users should select one or several components of a repository and should integrate them in their systems, with the consequent saving in time and cost. However to develop components, as well as to lead its selection, we require a knowledge of the characteristics of quality of such components, so that metrics acquire a cardinal importance.

There have been many metric models for software components, but no consensus has been reached for its classification. To advance in this area, it is essential to rely on a model that allows us to classify and systematize metric use.

Classification of diverse metric models for software components though it can also be used for the evaluation of a component or a component system.

It is necessary to indicate that this is the first approximation, and it needs to be checked to be a definitive and complete version, that can be used with guarantee.

TABLE 1
METRICS CLASSIFICATION FOR USE FACILITY

MÉTRIC	DESCRIPTION	QUALITY CHARACTERISTICS		GRANULARITY/ VISIBILITY		LIFECYCLE PROCESS	STAKE HOLD ER ¹	REF.
		External / Internal	Use	System	Component			
Time to Use	Average time needed for a developer to learn how to correctly use the component.	{U(Le)} ²	Efficiency/ Satisfaction	External	External	Evaluation/ Integration	Evaluat or/ Integrat or	<i>Bertoa and Vallecillo [3]</i>
Time to Configure	Average time needed for a developer to learn how to correctly configure the component, and for properly understanding its configuration parameters.	{U(Le)}	Satisfaction	External/ Internal	External	Evaluation / Integration	Evaluat or/ Integrat or	
User Documentation	Quality of the user documentation, in terms of its completeness, clarity and usefulness.	{U(Un)} ³	Satisfaction	External	External	Evaluation / Integration	Evaluat or/ Integrat or	
Required Interfaces	Number of interfaces that the component requires from other components to operate.	{U(Ope)} ⁴		Internal		Integration	Integrat or	
Provided Interfaces	Number of provided interfaces by the component as an indirect measure of its complexity.	{U(Ope)}	Efficiency/ Satisfaction	External	External	Integration/ Execution	Integrat or/ Final User	
Complexity Ratio	Average number of operations per provided interface.	{U(Ope)}	Efficiency	External	External	Integration/ Execution	Integrat or/ Final User	
Effort for Operating	Level of effort needed to properly operate the component.	{U(Ope)}	Satisfaction	External/ Internal	External	Integration/ Execution	Integrat or/ Final User	

¹ Bertoa y Vallecillo consider as users software architects and designers, which need to evaluate the components available in order to be incorporated into the software product.

² Characteristic Usability, sub-characteristic Learnability

³ Characteristic Usability, sub-characteristic Understandability

⁴ Characteristic Usability, sub-characteristic Operability

⁵ Characteristic Functionality

⁶ Characteristic Reliability

⁷ Characteristic Efficiency

⁸ Characteristic Maintainability

⁹ Characteristic Portability

MÉTRIC	DESCRIPTION	QUALITY CHARACTERISTICS		GRANULARITY/ VISIBILITY		LIFECYCLE PROCESS	STAKE HOLD ER ¹	REF.
		External / Internal	Use	System	Component			
Cost	Overall expenses incurred during the course of software development (software development tools, manpower, licensing fees, etc.). Include the costs of component acquisition and integration and quality improvements to the system.	{Fu ⁵ , Re ⁶ , U, E ⁷ , M ⁸ , P ⁹ }	Productivity	External/ Internal	External	Evaluation	Evaluat or/ Project Boss	<i>Sedigh-Ali, Ghafoor and Paul [18]</i>
Software Engineering Environment	Capability of producing high-quality software.	{Fu, Fb, U, E, M, }	Efficiency/ Productivity/ Physical Security/ Satisfaction	External/ Internal	External/ Internal	Analysis/ Evaluation/ Integration/ Testing/ Maintenance	Develo per/ Evaluat or/ Integrat or/ Maintai ner/ Project Manag er	
Complexity of Interfaces and Integration	Comparable to the metrics for the Bertoa and Vallecillo's sub-characteristic "Operability".							
Required Service for Primitive/Compl ex/ System Components	Number of required services that are 'satisfied' by other basic components. Notice: In the component, interface and service environment is the same.	{Fu, U(Ope)}		Internal		Evaluation/ Integration	Evaluat or/ Integrat or	<i>Dincel, Medvidovie and Van der Hock [11]</i>
Error of Propagation	Probability that a random error in the data transmitted from Component A to B results in an erroneous state of B.	{Fu, Fb, U, E, M, P}	Physical Security	Internal		Execution	Evaluat or/ Integrat or/ Maintai ner	<i>Shereshevsky, Ammari, Gradetsky, Mili, and Ammar [19]</i>
Coupling	Shared Information flow among the component A and B.	{Fu, Fb, U, E, M, P}		Internal		Evaluation/ Integration	Integrat or/ Maintai ner	

MÉTRIC	DESCRIPTION	QUALITY CHARACTERISTICS		GRANULARITY/ VISIBILITY		LIFECYCLE PROCESS	STAKE HOLD ER ¹	REF.
		External / Internal	Use	System	Component			
Cohesion	Relation among the elements of a component.	{Fu, Fb, U, E, M, P}		Internal		Evaluation/ Integration	Integrat or/ Maintai ner	<i>Shereshevsky, Ammari, Gradetsky, Mili, and Ammar [19]</i>
Volume	Measure of the size of a component.	{U, M}			Internal	Evaluation	Develo per/ Evaluat or	<i>Schach and Yang [17]</i>
Regularity	Ratio of the actual length of a component to the predicted length.	{U, M}			Internal	Evaluation	Develo per/ Evaluat or	
Number of comment words, global variables, formal parameters (Objective - Perceived)	Number of comment words, global variables, formal parameters. Measure expressed in an objective and perceived mode.	{U, M}			Internal	Evaluation/ Maintenance	Evaluat or/ Maintai ner	
Component Capacity	Information quantity available on the component outputs from its inputs.	{Fu, U, M}	Satisfaction	External	External	Execution	Maintai ner/ Project Manag er/ Final User	<i>Nguyen, Delaunay and Robach [14]</i>
Control Measures	Facility of generating the component outputs from the system outputs.	{Fu, U, M}	Satisfaction	External	External	Execution	Maintai ner/ Project Manag er/ Final User	

MÉTRIC	DESCRIPTION	QUALITY CHARACTERISTICS		GRANULARITY/ VISIBILITY		LIFECYCLE PROCESS	STAKE HOLD ER ¹	REF.
		External / Internal	Use	System	Component			
Functional Density	Percentage of functionality of the system delivered to the whole of COTS components of the system.	{Fu, U}		External	External	Execution	Project Manager/ Final User	<i>Abts [1]</i>
Adjustment Complexity	Summary of individual complexity values for factors that influence in the process of component integration.	{Fu, U, M}		Internal		Integration	Integrator	<i>Boehm, Abts y Bailey [4]</i>
Accessibility		{U (FA), (FC)}	Satisfaction	External	External	Execution	Integrator/ Final User	<i>Preiss, Wegmann and Wong [15]</i>
Administración		{U (FA)}	Satisfaction	External/ Internal	External	Execution	Integrator/ Final User	

REFERENCES

- [1] C. Abts, "COTS-Based Systems (CBS) Functional Density - A Heuristic for better CBS Design," *In Proceedings of the First International Conference on COTS-Based Software Systems*, Springer-Verlag Berlin Heidelberg New York, pp. 1-9, 2002.
- [2] A. Andrews, S. Ghosh, and E. Man Choi, "A Model for Understanding Software Components," *In Proceedings of the IEEE International Conference on Software Maintenance (ICSM'02)*, pp. 359-370, 2002.
- [3] M.F. Bertoa, and A. Vallecillo, "Quality Attributes for COTS Components," *In Proceedings of the Sixth ECOOP International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002)*, pp. 54-66, 2002.
- [4] B. Boehm, C. Abts, and E. Bailey, "COCOTS Software Integra Cost Model: an Overview," *In Proceedings of the California Software Symposium*, 1998.
- [5] J. Bosch, "Design and use of software architectures: Adopting and evolving a product line approach," Addison Wesley, 2000.
- [6] X. Cai, M.R. Lyu, K. Wong, and R. Ko, "Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes," *Proceedings of the Seventh Asia-Pacific Software Engineering Conference (APSEC'00)*, pp. 372-382, 2000.
- [7] C. Calero, M. Piattini, and M. Genero, "Empirical Validation of Referential Integrity Metrics," *Information Software and Technology*, special Issue on Controlled Experiments in Software and Technology, vol. 43, no. 15, pp. 949-957, 2001.
- [8] C.K. Chang, J. Cleland-Huang, S. Hua, and A. Kuntzmann-Combelles, "Function-Class Decomposition: A Hybrid Software Engineering Method," *IEEE Computer*, pp. 87-93, 2001.
- [9] A. Cechich, and M. Piattini, "Issues for Assessing Component-Based Systems," *VIII Congreso Argentino en Ciencias de la Computación (CACIC 2002)*, pp. 253-262, 2002.
- [10] COCOTS, "Constructive COTS Model," available at <http://sunset.usc.edu/research/COCOTS/2001>.
- [11] E. Dincel, N. Medvidovic, and A. Van der Hock, "Measuring Product Line Architectures," *In Proceedings of the Internatinal Workshop on Product Family Engineering (PFE-4)*, 2001.
- [12] ISO/IEC 9126-1:2001, "Software Engineering - Product quality - Part 1: Quality Model".
- [13] P. Kallio, and E. Niemelä, "Documented Quality of COTS and COM components," *In Proceedings of the 4th ICSE Workshop on Component-Based Software Engineering*, 2001.
- [14] T.B. Nguyen, M. Delaunay, and C. Robach, "Testability Analysis Based Software Engineering," *In Proceedings on The 6th IASTED International Conference Software Engineering and Applicatins (SEA 2002)*, 2002.
- [15] O. Preiss, A. Wegmann, and J. Wong, "On Quality Attribute Based Software Engineering," *In Proceedings of the 27th EUROMICRO Conference 2001: A Net Odyssey (EUROMICRO'01)*, pp. 114-120, 2001.
- [16] J. Ruiz, C. Calero, M. Piattini, "A Three Dimensional Web Quality Model," *International Conference on Web Engineering (ICWE)*, 2003.
- [17] S.R. Schach, and X. Yang, "Metrics for Targeting Candidates for Reuse: An Experimental Approach," *In Proceedings of the 1995 ACM Symposium on Applied Computing*, pp. 379-383, 1995.
- [18] S. Sedigh-Ali, A. Ghafoor, and R. Paul, "Metrics-Guided Quality Management for Component-Based Software Systems," *In Proceedings of the 25th Annual International Computer Software and Applications Conference*, IEEE Computer Society, pp. 303-310, 2001.
- [19] M. Shereshevsky, H. Ammari, N. Gradetsky, A. Mili, and H.H. Ammar, "Information Theoretic Metrics for Software Architectures," *In Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC'01)*, IEEE Computer Society, 2002.
- [20] R. Simao, and A. Belchior, "Quality Characteristics for Software Components: Hierarchy and Quality Guides", *In Component-Based Software Quality: Methods and Techniques*. LNCS 2693. pp. 188-211, 2003.
- [21] E. Sook Cho, M. Sun Kim, and S. Dong Kim, "Component Metrics to Measure Component Quality," *In Proceedings of the Eighth Asia-Pacific Software Engineering Conference (APSEC'01)*, pp. 419-427, 2001.
- [22] C. Szyperski, "Component Software Beyond Object-Oriented Programming," Addison-Wesley and ACM Press, 2002.
- [23] M. Woodman, O. Benediktsson, B. Lefever, and F. Stallinger, "Issues of CBD product quality and process quality," *In Proceedings of the 4th ICSE Workshop on Component-Based Software Engineering*, 2001.

Joaquina Martín-Albo. PhD student in Computer Science at the University of Castilla-La Mancha, Ciudad Real, Spain. His research interests are focused on component systems, web services, software metrics, conceptual data models quality, empirical software engineering.

Manuel F. Bertoa. He is a Telecommunications Engineer from the Polytechnic University of Madrid (Spain). Since 1984 he has worked in several national and international companies, mainly within the computer and telecommunications businesses, with responsibilities ranging from software design and development in Fujitsu, to heading the Customer Support Service of IngeniA. He has also worked for the Andalucian Health Service (SAS), first as Head of the IT Services for the Costa del Sol Helath Care District, and then as Director of IT Services for the Escuela Andaluza de Salud Pública. Since 1987 he has simultaneously lectured at the University of Málaga, where in 2002 became a full-time lecturer. His current research interests focus on the quality aspects of component-based software development, and their application to the software industry.

Coral Calero. She is Associate Professor at the Department of Computer Science in the University of Castilla-La Mancha, Ciudad Real, Spain. She received her MSc degree Computer Science in the Department of Computer Science of the University of Seville, and the PhD in Computer Science by the University of Castilla-La Mancha, Ciudad Real, Spain. Her research interests are: database quality, metrics for advanced databases, formal verification and empirical validation of software metrics.

Antonio Vallecillo. He holds the BSc and MSc degrees in mathematics, and the PhD degree in computer science. Most of his professional experience comes from the computer industry, where he has worked for more than 10 years for several international companies, both in Spain and in UK. Since 1996, he works at Málaga University, where he is currently an associate professor, and has also been the Head of the University IT Services. His research interests include component-based software development, open distributed processing, and the industrial use of formal methods. He is the representative of the Málaga University at ISO, the OMG, and AENOR (the Spanish National Body for Standardization), and a member of several professional organizations, including the ACM and the IEEE.

Alejandra Cechich. She is MSc in Computer Science by the University of South, Argentina and a PhD student at the University of Castilla-La Mancha, Spain. Full Professor at the Department of Computer Science at the University of Comahue, in Neuquén, Argentina. Her research interests are: software architectures, software quality, and conceptual modelling.

Mario Piattini. He is MSc and PhD in Computer Science by the Politechnical University of Madrid. Certified Information System Auditor by ISACA (Information System Audit and Control Association). Full Professor at the Department of Computer Science at the University of Castilla-La Mancha, in Ciudad Real, Spain. Author of several books and papers on databases, software

engineering and information systems. He leads the ALARCOS research group of the Department of Computer Science at the University of Castilla-La Mancha, in Ciudad Real, Spain. His research interests are: advanced database design, database quality, software metrics, object oriented metrics, software maintenance.