CBR Applied to Development with Reuse Based on Mecanos

Francisco J. García[1], Juan M. Corchado[1] and Miguel A. Laguna[2]
[1]Dpto. Informática y Automática – Universidad de Salamanca
[2]Dpto. Informática – Universidad de Valladolid
{fgarcia, corchado}@usal.es, mlaguna@infor.uva.es

Abstract
Reuse is one of the most important disciplines looking for improving the productivity, the time-to-market and, of course, the product quality. However, this promising discipline has not given the expected results. Several reasons have appeared in literature to explain that, one of the most realistic ones is the high cost expressed in terms of money, time and efforts that is necessary to implant a reuse process.
The organisational and methodological aspects of this discipline have to be taken into account during the reuse process in order to obtain successful results. This paper presents a synergetic methodology based on a coarse-grained reusable element called *mecano* and the Case-based reasoning (CBR) system that facilitate the reuse of software components. Case-based reasoning systems provide a knowledge management method that supports a most intelligent and automated development-with-reuse process based on *mecanos*.
Key Words
Case-based reasoning systems, Mecano model, Development with reuse.

1. Introduction

Reuse is the systematic process to develop systems of components of a reasonable size and reuse them [14]. Software Engineering community accepts that organized reuse (not ad hoc reuse) allows enhancing productivity and quality in software development [9]. The reuse life cycle comprises two main activities, development for reuse and development with reuse as is presented in [8,11,13] for example.
Development for reuse is the process of preparing software artefacts that these can be reused in other contexts. In the other hand, development with reuse is related to the construction of new software systems using reusable elements. The process of development with reuse includes the activities of searching for a set of candidate elements from a software repository, evaluating the set of retrieved candidate reusable components to find the most suitable one, and, if it's necessary, adapting the selected element to fit the specific requirements.
The repository of reusable software elements is the link between the both reuse-processes, i.e., where the reusable elements are produced and where they are reused. But, when the repository is widely populated the activities of the development with reuse become more and more difficult, specially the searching and evaluating ones.
A coarse-grained structure of reusable software elements called *mecano* [4,5] has been defined to improve the reuse process at different abstraction levels.
A *mecano* is a reusable element that is composed of interrelated thin-grained components, usually called "assets" in reuse literature. The components can be included in one of these three abstraction levels: analysis, design and implementation; and at least two different abstraction levels at least have to be presented in every *mecano*.
The *mecano* notion provides a framework to organize product-lines because it can be used to develop software for and with reuse [4]. A formal methodology is necessary to guide the reuse process with *mecanos*, especially the retrieval operation in the development-with-reuse process. In this framework the Case-based reasoning systems can be used to provide the *mecano* model with the required formalism. Such formalism facilitates the automation of the reuse process and allows the adaptation of the components of the system (*mecanos*) to the changes in the environment.
CBR systems are Artificial Intelligent (AI) systems [12] which have been successfully used in problems in which is needed to establish relationships between past experiences and new problems [16]. These systems provide a framework for identifying problems similar to a given one and for using the solutions given to such problems on the present one after an adaptation, revision and retain process. These systems incorporate learning and adaptation mechanisms that can be used to eliminate, incorporate and adapt the *mecanos* of the reuse-base.
Following the paper includes an introduction to the *mecano* model and the CBR systems. Then it is shown how such models can be combined in order to facilitate the reuse, and finally we present an application of the system, the conclusions and some future work.

2. Mecanos: An Overview

A *mecano* is a special coarse-grained reusable software element. A *mecano* is always composed by a set of interrelated fine-grained elements classified in different abstraction levels, called reusable assets, these elements

have to be stored in an adequate repository [4,5]. The supported abstraction levels are requirements level, design level and implementation level.

Around the *mecano* a reuse model has been defined, which is articulated over three edges, the technical model, the process model and the qualification model. The technical reuse model presents a duality compositional/generative in the creation of *mecanos*. Firstly, *mecanos* can be built as composition of individual reusable assets based on their relationships. This operation is a typical development-for-reuse activity; the result of which is a reusable element that can be reused "as-is" in a development-with-reuse activity. However, when a *mecano* does not match the reuser' specifications it cannot be reused "as-is", consequently a *mecano* can be generated by composition of the assets stored in the repository, following the relationships among the assets, so the generative approach is a development-with-reuse activity.

The core *mecano* model[1] is presented in Figure 1, using UML 1.3 as modelling language [15]. It presents the main components of this structure: the assets and their relationships. The assets are the reuse-centred components, while the relationships build up the layer for automated retrieval processes through an entry point (usually an asset representing functional requirements) and trace ability across the assets mesh. Both the assets and the relationships must have a type that represents the share properties of these kind of assets or relationships.

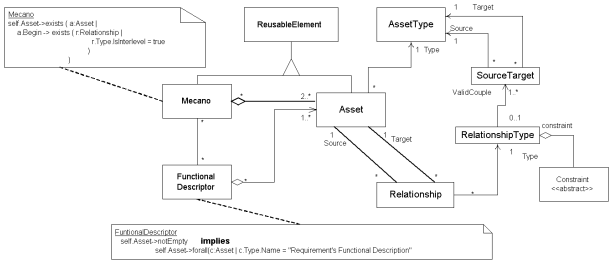

Figure 1. Mecano model

Due to the different abstraction levels supported by a *mecano*, there are two kinds of relationship between reusable assets: the *intralevel* relationships – among reusable assets classified in the same abstraction level – and the *interlevel* relationships – among assets classified in different abstraction levels. In this framework, a coarse-grained reusable software element can be considered as a *mecano*, if it incorporates *an interlevel relationship*.

The "reuse process" covers every activity, resource or procedure related to the organization's reuse policy. This model defines the activities to be carried out during the development-for-reuse and development-with-reuse processes. Figure 2 defines the basic development-with-reuse life-cycle, which will be used to present the novel software engineering reuse model introduced in this paper. Through the retrieval activity, potential *mecanos*, satisfying reuser's requirements, are searched and selected from the repository. However, not always a retrieved *mecano* can be reused as a black box, it could be adapted to customize it to a particular requirements of an application. Finally, the *mecanos* selected for reuse should be reassembled to build the subsystems of the final software product.

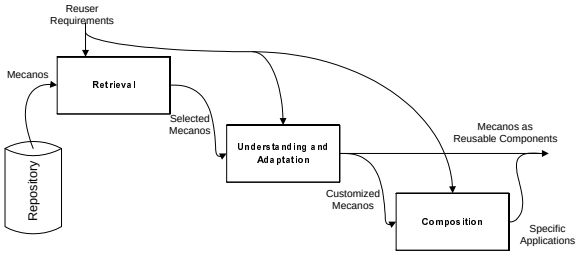

Figure 2. Classical development-with-reuse life-cycle reusing mecanos

The *mecano model* incorporates a qualification model, which defines the evaluation and certification criteria for each reusable software element. These criteria should be based on reuse metrics for the product *mecanos* and the processes. Some related metrics work can be found in [6,7].

Although the *mecano* model improves the abstraction level of the overall reuse process and supports the product-line concept it does not provides a method for identify which mecanos should be reused, eliminated or modified. A proposed solution is working with *mecanos* that present an analysis level, and try to select the suitable ones by their components asset representing functional requirements [4]. This solution is difficult to apply in real repository with a high number of *mecanos*, because there are too many functional requirements.

This model can be improved using a memory of the successful reuse actions performed in our repository. Thus, previous successful reuse experiences can be used when a new reuse situation appears. The Case-based reasoning life cycle is used, in this paper, to provide the *mecano* model with such capability and to define a reuse formalism based on the *mecano* concept.

3. Case Based Reasoning Systems

Knowledge-Based Systems (KBS) are one of the most popular and successful branches of the Artificial Intelligence (AI) world. According to [16] there are over 2000 KBS in commercial operation. Nevertheless, developers of these systems have met several problems: in many cases it can be difficult to deal with the knowledge elicitation part of a problem; the implementation of KBS is also complicated and once implemented, KBS are often slow and are unable to access or manage large volumes of knowledge and therefore they are normally difficult to maintain.

[1] A complete description of the *mecano* model appears in [4]. Figure 1 presents only an informal model described using UML class notation. [4] presents a formal model based on a graph grammar constructed with tube graphs.

Kolodner proposed a revolutionary model: the case-based reasoning model which is in fact a model of human reasoning [12], that overcomes some of the problems of KBS. The idea behind CBR is that people rely on concrete previous experiences when solving new problems. This fact can be tested on any day-to-day problem by simple observation or even by psychological experimentation. Since the CBR model was first proposed, it has proved successful in a wide range of application areas [3].
A case-based reasoning system solves new problems by adapting solutions that were used to solve old problems [10]. The case base holds a number of problems with their corresponding solutions. Once a new problem arises, retrieving similar cases from the case base and studying the similarity between them obtain the solution to it. A CBR system is a dynamic system in which new problems are added to the case base, redundant ones are eliminated and others are created by combining existing ones.
CBR systems record past problem solving experiences and, by means of indexing algorithms, retrieve previously stored cases, along with their solutions, and match them and adapt them to a given situation, to generate a solution. The intention of the CBR system is to abstract a solution from the knowledge stored in the case base in the form of cases. All of these actions are self-contained and can be represented by a cyclical sequence of processes in which human intervention may be needed. A case-base reasoning system can be used by itself or as part of another intelligent or conventional system. CBR systems are especially appropriate when the rules that define a knowledge domain are difficult to obtain or the number and the complexity of the rules affecting the problem are too large for the normal knowledge acquisition problem. This AI technology has been successfully used in disciplines such as law, medicine and other diagnostic systems with rich histories of cases [1].
A typical CBR system is composed of four sequential steps that are recalled every time that a problem needs to be solved [12,16].

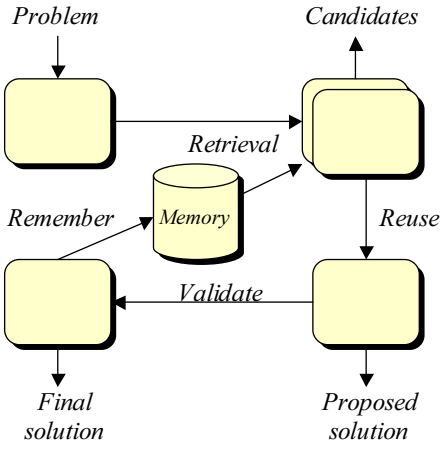

Figure 3. Classical CBR life-cycle

This cyclical process involves four major steps (Figure 3):

- Retrieve the most relevant case(s),
- Reuse the case(s) to attempt to solve the problem,
- Revise the proposed solution if necessary, and
- Retain the new solution as a part of a new case.

The mission of the retrieval algorithms is to search the case base and to select from it the most similar problems, together with their solutions, to the new problem. Cases should therefore represent, accurately, problems and their solutions. Once one or more cases are identified in the case base as being very similar to the new problem, they are selected for the solution of this particular problem. These cases are reused to generate a proposed solution (i.e. normally using an adaptation technique). This solution is revised (if possible) and finally the new case (the problem together with the obtained solution) is stored. Cases can also be deleted if they prove to be inaccurate; they can be merged together to create more generalised ones and they can be modified. For example, every time that an attempt to solve a problem fails, whenever possible, the reason for the failure is identified and logged in order to avoid the same mistake in the future. This is an effective way of learning that represents a general learning method used by humans [3]. CBR systems are able to utilise the specific knowledge of previously experienced problem situations rather than making associations along generalised relationships between problem descriptors and conclusions or relying on general knowledge of a problem domain such as rule-based reasoning systems. CBR is an incremental learning approach because every time that a problem is solved a new experience can be retained and made immediately available for future retrievals.
In the CBR cycle, there is normally human intervention [16]. Human experts often undertake case revision and retention. This can be a weakness of CBR systems and one of the major challenges of CBR research is to eliminate human intervention.
According to [1] there are five different types of CBR systems, and although they share similar features, each of them is more appropriate for a particular type of problem: exemplar based reasoning, instance based reasoning, memory-based reasoning, analogy-based reasoning and typical case-based reasoning.
Those CBR systems that focus on the learning of concept definitions are normally referred to as being exemplar-based. In the literature there are different views of concept definition [2]. A concept is defined extensionally as the set of its examples. PROTOS is an example of this type of CBR systems. In this case, solving a problem requires finding the right class for an unclassified exemplar. The class solution of the most similar retrieved case is the problem case solution. Instance-based reasoning (IBR) can be considered as exemplar-based reasoning is useful in highly syntactic problem [2]. This type of CBR system focuses on problems in which there are a large number of

instances which are needed to represent the whole range of the domain and where there is a lack of general background knowledge. The case representation can be made with feature vectors and the phases of the CBR cycle are normally automated as much as possible, eliminating human intervention.

4. Reusing Mecanos with a Case-Based Reasoning System

One of the main inconvenients of the reuse approach to software development is its high cost in terms of money, time and effort that is required to implant it. The *mecano* reuse model defines an abstraction level of the overall reuse process that reduces its cost in general terms by facilitating the organisation of the reuse process. This section describes another step towards the improvement and formalisation of the reuse process of software engineering. The reasoning process of Case-based reasoning systems is used in this context to guide the reuse of software components, in this case *mecanos*.
This cyclical process is composed of four steps that correspond with the four stages of the CBR systems presented in Figure 3. These four steps can be identified in Figure 4, which has been created combining the processes described in Figures 2 and 3. Figure 4 identifies the activities to be carried out, in sequential order, in the development-with-reuse process of a *mecano* reuse model.

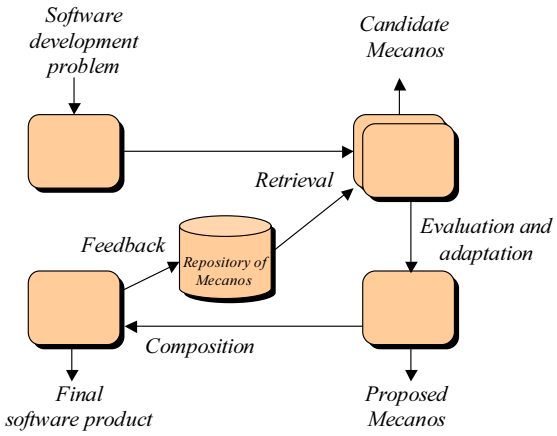

Figure 4. Mecano-CBR reuse model

In the present paper the term "*Mecano-CBR reuse model*" will be used to refer to the novel reuse model presented here. To construct a new software system using this model it is necessary to create a repository of *mecanos* first. Then, on cyclical steps, the four steps of this model should be run to identify reusable component for each of the fragments in which the new software development process is divided. The four steps are: Retrieval, Evaluation and Adaptation, Composition and Feedback. Although this reuse process can be used to build any software system, the metrics used in any of the steps may be adapted to each type of problem.
The main reuse-process, shown in Figure 4, could be briefly described as follows. Once identified a particular software development problem, a metric is used to select those *mecanos* that can be reused during the retrieval step. Since probably such *mecanos* cannot be reused as they are, they have to be evaluated and adapted in the following step of the process using other predefined metrics. The initial solution may be reviewed by human experts or by other predefined metrics. After implementing the system and analysing its performance, conclusions can be extracted about the efficiency of the reuse process and the final system. During the Feedback step, the extracted conclusions can be used to carry on actions in order to avoid future errors and to improve the system by adding/eliminating *mecanos* from the repository of *mecanos*, modifying them or the metrics used for their retrieval, evaluation or adaptation.
In a typical incremental software life-cycle, the process shown in Figure 4 could be done several times, so many times as subsystems or increments are defined in the global system, where each subsytem could be supported or represented by a set of *mecanos*, in a coarse approximation. After a *mecano* is proposed, it needs to be integrated with the previously found subsystems during the composition step of the Mecano-CBR reuse cycle.

5. Case Study

A CBR reuse-prototype has been constructed with the ideas presented in this paper. An experiment has been carried out using a small number of laboratory-made *mecanos* stored in the GIRO Reuse Repository (http://marte.dcs.fi.uva.es/) and classified in the robotic-workcell domain.
Every *mecano* represents an active component in a robotic workcell (i.e. a robot, a conveyor belt and so on). Each one is composed by four main assets, as shown in Figure 5, representing the Software Requirements Specification (SRS) document (composed by the atomic functional requirements expressed in XML), the UML class packages, the sequence diagram and the IDL and C++ implementation.

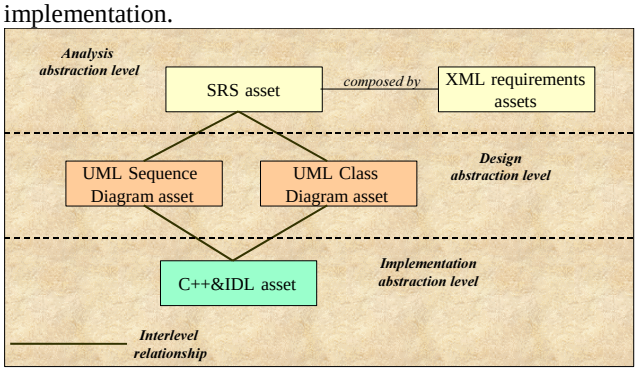

Figure 5. Mecano representing an active robotic-workcell component

During the retrieval activity, firstly, we use the consulting facilities of the repository, basically by navigating through the requirements specifications of the *mecanos* classified in a concrete domain, the robotic-workcell domain in this case. This way you can select the mecanos you are interested in. If there are many *mecanos* that represent the

same component you should select the better one, in this case we use the metrics framework for UML diagram classes defined in [7].
The adaptation and composition are manual activities, but the effort made in them should be measured to take conclusions and to express the feedback information.
The feedback activity is the key of the *Mecano-CBR reuse model* because it represents the result of the overall reuse process and can determine management action to modify the repository population, introducing new *mecanos* or deleting existing ones because they are not useful for an effective reuse process.
This initial prototype can be used to compose new systems by reusing existing subsystems in a systematic way. The Mecano-CBR tool only guides software engineers through the four steps presented above. This tool as it is does not include any option that automates the adaptation, the composition, and the feedback; it is just a framework that facilitates such processes by now.

6. Conclusions and Future Work

The Mecano-CBR reuse model presented in this paper is a first step toward the development a more automated repository-environment that facilitates the software engineering process of reuse. This model combines the advantages of an Artificial Intelligent model with one of the most productive branches of the Software Engineering, as reuse is.
The developed prototype has shown promising results and presently is been improving to be used in a real product-line engineering problem.
Further work will be focussed in defining the methods that facilitate the automation of every step of the Mecano-CBR reuse model that have not been yet automated. In this way, it is compulsory consider the generation part in the creation of the *mecanos* in reuse time, as is explained in [4].

7. Acknowledgements

This work has the partial support of the Castilla y Leon Council (SA02/00F Project) and the CICYT (Dolmen Project - TIC2000-1673-C06-05).

8. References

[1] Aamodt, A. and Plaza, E. "*Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*". AICOM, 7(1). March, 1994.

[1] Corchado, J. M. and Lees B. "Adaptation of Cases for Case-Based Forecasting with Neural Network Support". S.K Pal, T.S. Dillon and D.S. Yeung (Eds.), *Soft Computing in Case Based Reasoning*, Springer Verlag, London, 2000.

[2] Corchado, J. M. and Lees, B. "*A Hybrid Case-based Model for Forecasting*". Applyed Artificial Intelligence (In press). 2001.

[3] Corchado, J. M., Lees, B., Fyfe, C., Rees, N. and Aiken, J. "*Neuro-Adaptation Method for a Case-based Reasoning System*". Computing and Information Systems Journal, 5(1):15-20. February, 1998.

[4] García, F. J. "*Modelo de Reutilización Soportado por Estructuras Complejas de Reutilización Denominadas Mecanos*". PhD Thesis, University of Salamanca, 2000.

[5] García, F., Marqués, J. and Maudes, J. "*Mecanos as Basis of a Compositional/Generative Mixed Reuse Model*". Proceedings of the 2nd ed. of the European Reuse Workshop. Madrid - Spain, 2, 17-20. 1998.

[6] Genero, M., Manso, Mª E., Piattini M. and García, F. J. "Assessing the Quality and the Complexity of OMT Models". 2nd European Software Measurement Conference - FESMA 99, Amsterdam,The Netherlands, pages 99-109, 1999.

[7] Genero, M., Manso, Mª E., Piattini M. and García, F. J. "Early Metrics for Object Oriented Information Systems". In proceeding of the 6th International Conference on Object Oriented Information Systems (OOIS'2000). London (United Kingdom) – December, 2000. D. Patel, I. Choudhury, S. Patel and S. de Cesare (eds). Pages 414-426. Springer-Verlag London. 2000.

[8] Girardi, M. R. "Application Engineering: Putting Reuse To Work". In Tsichritzis, D. editor, Object Frameworks, pages 137-149. Centre Universitaire d'Informatique, University of Geneva.

[9] Griss, M. L. "Software Reuse: A Process of Getting Organized". Object Magazine, 4(12). 1995.

[10] Joh, D. Y. "CBR in a Changing Environment". Case Based Reasoning Research and Development. ICCBR-97. Providence, IR, USA. 1997.

[11] Karlsson, E.-A., editor. "Software Reuse. A Holistic Approach". Wiley Series in Software Based Systems. John Wiley & Sons Ltd, 1995.

[12] Kolodner, J. "Case-Based Reasoning". Morgan Kaufmann, 1993.

[13] McClure, C. "Software Reuse Techniques: Adding Reuse to the System Development Process". Prentice-Hall, 1997.

[14] McIlroy, D. "Mass Produced Software Components". Proceedings of the 1968 NATO Conference on Software Engineering.

[15] OMG. "*OMG Unified Modeling Language Specification. Version 1.3*". Object Management Group Inc. http://uml.shl.com:80/docs/UML1.3/99-06-08-pdf. June, 1999.

[16] Watson, I. and Marir, F. "Case-Based Reasoning: A Review". Cambridge University Press. The knowledge Engineering Review, 9(3). 1994.