A Systematic Review of the Application of Modern Software Engineering Techniques to the Development of Robotic Systems

Claudia Pons[1,2], Roxana Giandini[2], Gabriela Arévalo[1,3]

[1] CONICET
[2] Facultad de Informatica, UNLP
[3] Universidad Abierta Interamericana, UAI
{cpons, giandini}@info.unlp.edu.ar

Abstract. Robots have become usual collaborators in our daily life. While robotic systems grow to be more and more complex, the need to engineering their software development process grows as well. Traditional approaches that are used in the development process of these software systems are reaching their limits; currently used methodologies and toolsets fall short to address the needs of such complex software development process. Separating robotics knowledge from short-cycled implementation technologies is essential to foster reuse and maintenance. This paper presents a systematic review of the current use of modern software engineering techniques for the development of robotic software systems and their actual automation level. The goal of the survey is to summarize the existing evidence concerning the application of such technologies on the robotic systems field; to identify any gaps in current research in order to suggest areas for further investigation and to provide a background in order to appropriately position new research activities.

Keywords: survey, robotic software system, model-driven software development, software engineering, SOA, Component based software development.

1 Introduction

Robotic systems (RSs) play an increasing role in everyday life. Also the need for robotic systems in industrial settings increases and becomes more demanding. While robotic systems grow to be more and more complex, the need to engineering their software development process grows as well. Traditional approaches that are used in the development process of these software systems are reaching their limits; currently used methodologies and toolsets fall short to address the needs of such complex software development process.

It is widely accepted that new approaches should be established to meet the needs of the development process of today's complex RSs. In this direction, Component-based development (CBD) [5], Service Oriented Architecture (SOA) [12] [13], as well as Model Driven software Engineering (MDE) [15] [16] and Domain-Specific Modeling (DSM) [14] are among the key promising technologies in the RSs domain.

This paper presents a systematic review of the current use of those modern software engineering techniques for the development of robotic software systems and their actual automation level. The goal of the survey is to summarize the existing evidence concerning the application of such technologies on the robotic systems field. The paper is organized as follows. Section 2 describes the methodology we adopted to perform the review. Section 3 describes the needs for performing a review in this area; Section 4 presents the planning of the review. Section 5 reports data we extracted from each paper. Section 6 answers our research questions. Finally, we report our conclusions in Section 7.

2. Systematic Literature Reviews and Systematic Mapping Studies

A systematic literature review (SLR) [1] [2] is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest. Individual studies contributing to a systematic review are called primary studies; a systematic review is a form of secondary study.

Some of the features that differentiate a systematic review from a conventional expert literature review are that: SLRs start by defining a review protocol that specifies the research question being addressed and the methods that will be used to perform the review; SLRs are based on a defined search strategy that aims to detect as much of the relevant literature as possible; SLRs document their search strategy so that readers can assess their rigor and the completeness and repeatability of the process; SLRs require explicit inclusion and exclusion criteria to assess each potential primary study and specify the information to be obtained from each primary study including quality criteria by which to evaluate each primary study.

There are other types of review that complement systematic literature reviews such as the systematic mapping studies. If, during the initial examination of a domain prior to commissioning a systematic review, it is discovered that very little evidence is likely to exist or that the topic is very broad then a systematic mapping study may be a more appropriate exercise than a systematic review. A systematic mapping study allows the evidence in a domain to be plotted at a high level of granularity. This allows for the identification of evidence clusters and evidence deserts to direct the focus of future systematic reviews and to identify areas for more primary studies to be conducted.

A SLR involves several discrete activities. Existing guidelines have slightly different suggestions about the number and order of activities However, the medical guidelines and sociological text books are broadly in agreement about the major stages in the process. Kitchenham and colleagues in [1] summarize the stages in a systematic review into three main phases: Planning the Review (that includes the activities of identification of the need for a review, specifying the research questions, identification of research, selection of primary studies, study quality assessment, developing a review protocol, evaluating the review protocol); Conducting the Review (conformed by the following activities: data extraction and monitoring, data synthesis); Reporting the Review (conformed by the following activities: specifying dissemination mechanisms, formatting the main report, evaluating the report).

Due to the extensiveness of our topic of interest, in the present work we will perform a systematic literature review oriented to mapping studies.

3. The needs for a review in this field

Prior to undertaking a systematic review it is necessary to confirm the need for such a review. Although the complexity of robotic software is high, in most cases reuse is still restricted to the level of libraries. At the lowest level, a multitude of libraries have been created for robot systems to perform tasks like mathematical computations for kinematics, dynamics and machine vision, such as [3]. Instead of composing systems out of building blocks with assured services, the overall software integration process for another robotic system often is still reimplementation of the glue logic to bring together the various libraries. Often, the kind of overall integration is completely driven by a certain middleware system and its capabilities. Middlewares are often used to hide complexity regarding inter-component communication, for example OpenRTM-aist [4] is a CORBA-based middleware for robot platforms that uses so-called robot technology components to model distribution of functionality. Obviously, this is not only expensive and wastes tremendous resources of highly skilled roboticists, but this also does not take advantage from a maturing process to enhance overall robustness.

We have faced this problem in our own practice. We have been programming educational robots for more than 10 years [24] [25] and we have observed in the last years the emergence of robotic kits oriented to non-expert users that gave rise to the development of a significant number of educational projects using robots. Those projects apply robots at different education levels, from kindergarten through higher education, especially in areas of physics and technology. In this context, one of the problems we encountered is that the hardware of the robotic kits is constantly changing; in addition its use is not uniform across different regions and even education levels. Therefore, the technical interfaces of these robots should hide these differences so that teachers are not required to change their educational material over and over again. An example of these interfaces is “Physical Etoys” [25], a project in which we participated and which proposes a standard teaching platform for programming robots, regardless of whether they are based on Arduino, Lego, or other technologies.

In this context, it is widely accepted that new approaches should be established to meet the needs of the development process of today’s complex RSs. Component-based development (CBD) [5], Service Oriented Architecture (SOA) [12] [13], as well as Model Driven software Engineering (MDE) [15] [16] and Domain-Specific Modeling (DSM) [14] are among the key promising technologies in the RSs domain.

In first place, the Component-based development paradigm [5] states that application development should be achieved by linking independent parts, the components. Strict component interfaces based on predefined interaction patterns decouple the sphere of influence and thus partition the overall complexity. This results in loosely coupled components that interact via services with contracts. Components such as architectural units allow specifying very precisely, using the concept of port, both the services provided and the services required by a given component and defining a composition theory based on the notion of a connector. Component technology offer high rates of reusability and ease of use, but little flexibility with regard to the implementation platform: most existing component are linked to C/ C++ and Linux (e.g. Microsoft robotics developer studio [6], EasyLab[7], Player/Stage project[10]), although some achieve more independence, thanks to the use of some middleware (e.g. Smart Software Component model[11], Orocos[3] Orca [8] CLARAty[9]).

In second place, we need a way to define interfaces and behavior at a higher level of abstraction so that they could be used in systems with different platforms. This is what prompted the idea of abstract components, which would be independent of the implementation platform but could be translated into an executable software or hardware component. Thus, the migration from code-driven designs to a model-driven development is mandatory in robotic components to overcome the current problems. A model-based description is a suitable mean to express contracts at component interfaces and to apply tools to verify the overall behavior of composed systems and to automatically derive the executable software. Instead of building tool support for each framework from scratch, one should now try to either express the needed models in standardized modeling languages like UML or any DSL, separating components from the underlying computer hardware. In the context of software engineering, the Model Driven Development (MDD) [15] [16] and Domain-Specific Modeling approach (DSM) [14] have emerged as a paradigm shift from code-centric software development to model-based development. Such approaches promote the systematization and automation of the construction of software artifacts. Models are considered as first-class constructs in software development, and developers' knowledge is encapsulated by means of model transformations. The essential characteristic of MDD and DSM is that software development's primary focus and work products are models. Its major advantage is that models can be expressed at different levels of abstraction and hence they are less bound to any underlying supporting technology. This is especially relevant for software systems within the ubiquitous computing domain, which consist of dynamic, distributed applications and heterogeneous hardware platforms, such as robotic systems.

Finally, Service-oriented architecture (SOA) is a flexible set of design principles used during the phases of systems development and integration in computing. A system based on a SOA will package functionality as a suite of interoperable services that can be used within multiple, separate systems from several business domains. SOA also generally provides a way for consumers of services, such as web-based applications, to be aware of available SOA-based services. SOA defines how to integrate widely disparate applications for a Web-based environment and uses multiple implementation platforms. Rather than defining an API, SOA defines the interface in terms of protocols and functionality. Service-orientation requires loose coupling of services with operating systems, and other technologies that underlie applications. SOA separates functions into distinct units, or services[12] which developers make accessible over a network in order to allow users to combine and reuse them in the production of applications. These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format [13]

Summarizing, we know that these software engineering techniques offer good potential for the development of robotic systems, so we need to search for proposals in these directions and we need to detect which work is already done and which work is pending. Additionally we want to know if there is any proposal taking advantage of the combined application of CBP, SOA and MDE to robotic software system development.

4. Planning the review

A review planning specifies the methods that will be used to undertake a specific systematic review. A pre-defined planning is necessary to reduce the possibility of researcher bias.

4.1 The Research Questions

Specifying the research questions is the most important part of any systematic review. In this context, the right question is usually one that will lead either to changes in current software engineering practice or to increased confidence in the value of current practice and/or will identify discrepancies between commonly held beliefs and reality. The 5 research questions investigated in this study were:

RQ1 Have MDD techniques been applied to the development of robotic systems and how is the current tendency?
RQ2 Have CBD techniques been applied to the development of robotic systems and how is the current tendency?
RQ3 Have SOA techniques been applied to the development of robotic systems and how is the current tendency?
RQ4 Have those techniques been used in combination or in isolation?

RQ5 Which MDE techniques have been applied to the development of robotic systems and which is their automation level?

4.2 The Search strategy

A search strategy was used to search for primary studies. Such strategy includes search terms and resources to be searched. Resources include digital libraries, specific journals, and conference proceedings. We searched two digital libraries and one broad indexing service: IEEE Computer Society Digital Library; ACM Digital Library and SCOPUS indexing system. All searches were based on title, keywords and abstract. The searches took place in February 2011. We use the following Boolean query (adapted to the particular syntax of each library):

```
(robot*)
AND
(“software development” OR “system development” OR programming)
AND
(MDD OR MDE OR "model driven" OR "domain specific language" OR
"domain specific modeling" OR DSL OR "code generation" OR
"generative programming" OR "Component based” OR CBD OR "service
oriented" OR "service based" OR SOA OR "Web service")
```

Concerning the quality of the search strategy, general guidelines recommend considering the effectiveness of a question from five viewpoints (PICOC criteria):
Population: that is the application area.
Intervention: the intervention is the software methodology/tool/technology/procedure that addresses a specific issue.
Comparison: this is the software engineering methodology/tool/technology/procedure with which the intervention is being compared.
Outcomes: outcomes should relate to factors of importance to practitioners such as improved reliability, reduced production costs, and reduced time to market.
Context: this is the context in which the comparison takes place (e.g. academia or industry), the participants taking part in the study (e.g. practitioners, academics, consultants, students), and the tasks being performed (e.g. small scale, large scale).

According to this PICOC criteria our query is organized as follows,
Population: the population corresponds to the robotic domain. This is reflected in the first sub-expression of our query.
Intervention: the intervention of our survey comprises software and system development. This fact is specified in the second sub-expression of our query.
Comparison: in our case, the software engineering methodologies to be compared or analyzed are MDD, SOA and CBD. This is indicated in the third sub-expression of our query.
Outcome: we want to obtain as much outcomes as possible by collecting all the available information in the domain of study, so our query does not restrict the kind of outcomes.
Context: we apply no restriction to the context of our study.

4.3 Study Selection criteria

Study selection criteria are intended to identify those primary studies that provide direct evidence about the research question. Once the potentially relevant primary studies have been obtained, they need to be assessed for their actual relevance. Study selection criteria are used to determine which studies are included in, or excluded from, a systematic review. It is usually helpful to pilot the selection criteria on a subset of primary studies.

We undertook an initial screening of 171 papers found, based on title, abstract and keywords. In this screening we excluded studies that were obviously irrelevant, or duplicates. The remaining 104 papers were then subject to a second assessment: we obtained full copies of these remaining papers and undertook a more detailed second screening using the following inclusion and exclusion criteria: -The paper should be related to software engineering rather than mathematical modeling and/or math simulation. – "service oriented" should refer to SOA but not to "robots that perform a service". Based on those inclusion criteria we determined that 37 articles were excluded by the criteria. Finally, we analyzed the remaining 67 articles. All those sources can be retrieved from http://lifia.info.unlp.edu.ar/eclipse/robotsurvey2011.

5. Data extraction strategy

This strategy defines how the information required from each primary study will be obtained. The objective of this stage is to design data extraction forms to accurately record the information researchers obtain from the primary studies.

We elaborated a form comprising the following fields:

Field name	Type
Paper identification	Integer
Year of publication	Date
It applies SOA	Boolean
It applies CBD	Boolean
It applies MDD	Boolean

If the value of the last field is True (i.e., the paper applies MDD), then the following form is filled:

Field name	Type
Modeling Language	{UML, Profile, DSML}
Programming Language	{Any language, robotic-high-level}
Model Transformation Technique	{GPL, DSL, Black-box}
Tools	{existing tool, new tool }
Automation Level	{Full, Medium, Low}

The field named "*Modeling Language*" specifies which language is used to express the platform independent models. We found that some projects use the standard UML language, while other projects consider that UML is not expressive enough and then they define an extension through the creation of a profile. Finally, other proposals do not use UML but instead they define their own domain specific modeling language (DSML).

The field named "*Programming Language*" identifies the implementation language that is used as the target of model transformations. We observed that in most cases the PIM models are translated to different languages, which is one of the principles of MDD. However in other cases the PIM models are mapped to a specific high level language, such as Urby, or to a specific middleware, such as MSRS.

The field named "*Model Transformation Technique*" indicates which is the strategy used to transform the PIM to the PSM or to the code. Some projects implement the transformation just using a general purpose programming language (GPL) such as Java, while other proposals use existing transformation DSLs, such as ATL, JET or QVT. Finally, most proposals use black-box transformations.

The field named "*Tools*" denotes which kind of software tools are being used in the project. The options are as follows, using existing tools (such as EMF or MS DSL tools) or creating a new specific tool.

The field named "*Automation Level*" states how much work is made automatically. The value "Full" indicates that code is fully automatically generated from models. The "Medium" value states that code is partially generated and it should be completed manually, while the "Low" value indicates that the transformation from models to code is carried out mostly by hand.

6. Data synthesis strategy: answering the questions

Data synthesis involves collecting and summarizing the results of the included primary studies. We present here the answers to our research questions and we display quantitative foundations.

Figure 1 shows the answer to the first three questions. We observe an increasing tendency in the use of all these techniques, being CBD the most applied in the robotics field.

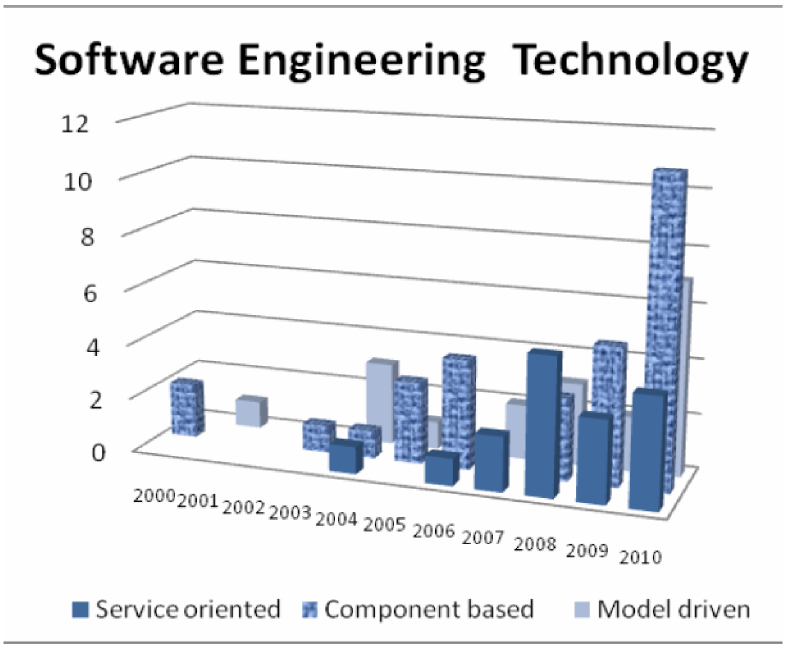

Fig. 1. Software Engineering Technology

Figure 2 answers the question number 4 showing the distribution of articles in each field. We observe little intersection among the different technologies. However there is a promising intersection between MDD and CBD showing the good potential of combining these two technologies.

Concerning question number 5, the figure 3 illustrates which modeling languages are being applied in the robotic projects. We observe that the definition of new domain specific languages is the most applied technique, while the use of UML and its profiles come later.

Regarding the application of MDD tools we observe in figure 4 that 72% of MDD projects take advantage of existing MDD tools such as ATL, EMF and DSL tools, while the 28% implements their own modeling and transformation tools. The reasonable tendency is that existing tools will be increasingly reused in the near future.

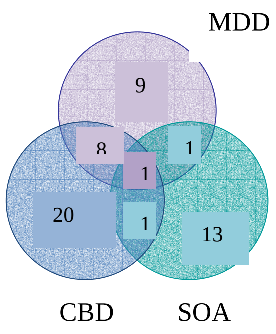

Fig. 2. Field intersection

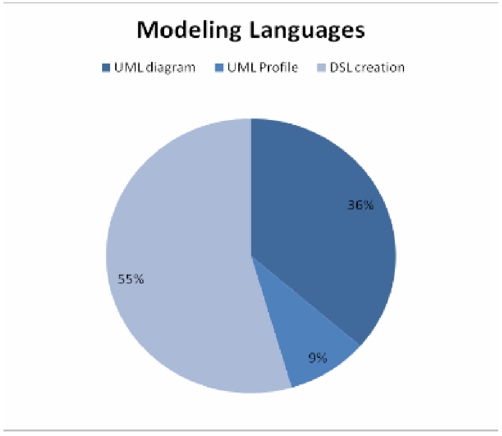

Fig. 3. Modeling languages

Finally, figure 5 shows the distribution of levels of automation in the MDD projects. Only 33% have achieve full automation in their MDD process, while 50% present an intermediate level of automation, that implies the creation of abstract models and the automatic generation of code skeleton that should be manually completed by the developers. Finally, 17% of the MDD robotic projects only reach a low level of automation consisting in the creation of abstract models but the manual derivation of code.

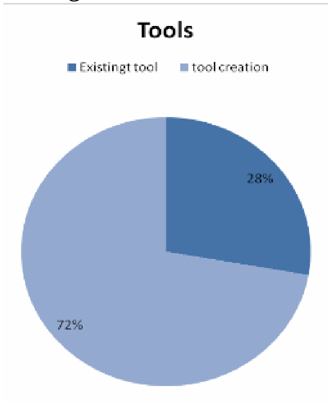

Fig. 4. Tools

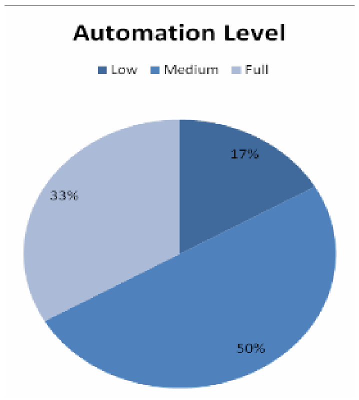

Fig. 5. Automation Level

7. Conclusion

The robotics community has a sufficient amount of experience on how to build complex robotics systems. However, we cannot expect significant growth with hand-crafted singleunit systems and it is mandatory to work towards applying engineering principles to cope with the complexity of robotics software systems. In most cases, we already have the knowledge about what proved to be a good solution in the software engineering field. The next step is to make this knowledge explicit and easily accessible for new systems. Applying existing technology would safe time and effort that is better put into what is specific in robotics.

In this paper we have presented an overview of ongoing activities regarding the application of modern software engineering techniques on the robotic software development process. We observe a growing tendency on the application of Component based development as well as Service based architecture and Model driven software development, although those techniques have been mostly applied in isolation. After reviewing more than 100 papers on the subject we have identified gaps in current research that open the door to further investigation. Our analysis provides a background in order to appropriately position new research activities.

References

[1] B.A. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering Technical Report EBSE-2007-01, 2007.

[2] T. Dybå, B.A. Kitchenham, M. Jørgensen, Evidence-based software engineering for practitioners, IEEE Software 22 (1) (2005) 58–65. [4] Khan S. Khalid, Regina Kunz, Jos Kleijnen, Gerd Antes, Systematic Reviews to Support Evidence-based Medicine, Springer, 2003.

[3] Bruyninckx, H.: Open robot control software: The OROCOS project. In: Proceedings of 2001 IEEE International Conference on Robotics and Automation (ICRA'01), vol. 3, pp. 2523–2528. Seoul, Korea (2001)

[4] Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T., Yoon, W.K.: RT-middleware: Distributed component middleware for RT (robot technology). In: International Conference on Intelligent Robots and Systems 2005 (IROS 2005), pp. 3933–3938 (2005)

[5] Clemens Szyperski Component Software: Beyond Object-Oriented Programming. 2nd ed. Addison-Wesley Professional, Boston ISBN 0-201-74572-0 (2002).

[6] Microsoft, "Microsoft robotics developer studio," 2009,http://msdn.microsoft.com/en-us/robotics/default.aspx, visited on March 11th 2009.

[7] Barner, S., Geisinger, M., Buckl, C., Knoll, A.: EasyLab: Model-based development of software for mechatronic systems. In: IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications. Beijing, China (2008)

[8] Brooks, A., Kaupp, T., Makarenko, A., Oreback, A., Williams, S.: Towards component-based robotics. In: Proc. of 2005 IEEE/RSJ Int. Conf. on Intellegent Robots and Systems (IROS'05), pp. 163–168. Alberta, Canada (2005)

[9] Nesnas, I., Wright, A., Bajracharya, M., Simmons, R., Estlin, T.: CLARAty and challenges of developing interoperable robotic software. In: Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 3, pp. 2428–2435 (2003).

[10] Gerkey, B.P., Vaughan, R.T., Howard, A.: Most valuable player: a robot device server for distributed control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1226–1231. Wailea, Hawaii (2001) Player Stage.

[11] C. Schlegel, ‘‘Communication patterns as key towards component interoperability,’’ in Software Engineering for Experimental Robotics (Series STAR, vol. 30), D. Brugali, Ed. Berlin, Heidelberg: Springer-Verlag, , pp. 183–210. Smartsoftware (2007)

[12] Bell, Michael. "Introduction to Service-Oriented Modeling". Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley & Sons. pp. 3. ISBN 978-0-470-14111-3. (2008).

[13] Bell, Michael. SOA Modeling Patterns for Service-Oriented Discovery and Analysis. Wiley & Sons. pp. 390. ISBN 978-0470481974. (2010).

[14] Steven K., Juha-Pekka T. Domain-Specific Modeling. John Wiley &Sons, Inc. 2008.

[15] Stahl, M Voelter. Model Driven Software Development. John Wiley. (2006).

[16] Claudia Pons, Roxana Giandini, Gabriela Pérez. “Desarrollo de Software Dirigido por Modelos. Teorías, Metodologías y Herramientas”, Ed: McGraw-Hill Education. (2010).

[17] Jacobson, Ivar; Grady Booch; James Rumbaugh. The Unified Software Development Process. Addison Wesley Longman. ISBN 0-201-57169-2. (1998).

[18] Christian Schlegel, Thomas Haßler, Alex Lotz and Andreas Steck. Robotic Software Systems: From Code-Driven to Model-Driven Designs. In procs. Of ICAR 2009. International Conference on Advanced Robotics. IEEE Press (2009)

[19] Iborra, A.; Caceres, D.; Ortiz, F.; Franco, J.; Palma, P.;Alvarez, B.; Design of Service Robots. Experiences Using Software Engineering. IEEE Robotics & Automation Magazine 1070-9932/09/ IEEE Page(s): 24 – 33. MARCH 2009

[20] Barner, S., Geisinger, M., Buckl, C., Knoll, A.: EasyLab: Model-based development of software for mechatronic systems. In: IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications. Beijing, China (2008)

[21] Hyun Seung Son; Woo Yeol Kim; Kim, R.; Semi-automatic Software Development Based on MDD for Heterogeneous Multi-joint Robots. In Future Generation Communication and Networking Symposia, 2008. FGCNS '08. : 2008 , Page(s): 93 – 98 (2008)

[22] Baer, P. A.; Reichle, R.; Zapf, M.; Weise,T.; Geihs, K.; A Generative Approach to the Development of Autonomous Robot Software.EASe '07. Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems. (2007).

[23] Arney, D.; Fischmeister, S.; Lee, I.; Takashima, Y.; Yim, M.;Model-Based Programming of Modular Robots. 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), Page(s): 66 – 74 (2010).

[24] Physical Etoys. GIRA Grupo de Investigación en Robótica Autónoma del CAETI. http://tecnodacta.com.ar/gira/ (accedido en May 2011).

[25] Centro de Altos Estudios en Tecnología Informática (CAETI). Proyectos del área robótica. http://www.caeti.uai.edu.ar. Accedido Junio 2011.