# A systematic review of applying modern software engineering techniques to developing robotic systems

## Revisión sistemática de la aplicación de técnicas modernas de ingeniería de software al desarrollo de sistemas robóticos

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

### RESUMEN

Los robots se han convertido en colaboradores habituales de nuestra vida diaria. Los sistemas robóticos son cada vez más complejos y, como consecuencia, crece la necesidad de aplicar nuevas técnicas ingenieriles a su proceso de desarrollo. Los enfoques tradicionales que se utilizan en el proceso de desarrollo de estos sistemas de *software* están alcanzando sus límites; las metodologías utilizadas actualmente y las herramientas de soporte no alcanzan para atender las necesidades de estos procesos complejos. Para fomentar la reutilización y el mantenimiento de código es esencial separar el conocimiento estable del dominio de robótica en las tecnologías de implementación, que varían rápidamente. Este artículo presenta una revisión sistemática de la utilización actual de técnicas modernas de ingeniería de *software* en el desarrollo de sistemas robóticos y su nivel de automatización. El objetivo del estudio es el de resumir la evidencia existente respecto a la aplicación de dichas tecnologías en el campo de los sistemas robóticos para identificar carencias en la investigación actual con el fin de sugerir áreas en futuras propuestas y proporcionar las bases para posicionar adecuadamente nuevas actividades de investigación.

**Palabras clave**: revisión, sistemas de *software* robóticos, desarrollo de *software* dirigido por modelos, ingeniería de *software*, SOA, desarrollo de *software* basado en componentes.

### ABSTRACT

Robots have become collaborators in our daily life. While robotic systems become more and more complex, the need to engineer their software development grows as well. The traditional approaches used in developing these software systems are reaching their limits; currently used methodologies and tools fall short of addressing the needs of such complex software development. Separating robotics' knowledge from short-cycled implementation technologies is essential to foster reuse and maintenance. This paper presents a systematic review (SLR) of the current use of modern software engineering techniques for developing robotic software systems and their actual automation level. The survey was aimed at summarizing existing evidence concerning applying such technologies to the field of robotic systems to identify any gaps in current research to suggest areas for further investigation and provide a background for positioning new research activities.

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

Received: September 28th 2011
Accepted: February 28th 2012

## Introduction

Robotic systems (RSs) play an increasing role in everyday life. The need for robotic systems in industrial settings is increasing and has become more demanding. While robotic systems become more and more complex, the need to engineer their software development grows as well. Traditional approaches used in developing such software systems are reaching their limits; currently used methodologies and tools fall short of addressing the needs of such complex software development.

It is widely accepted that new approaches should be established to meet the needs of developing today's complex RS. Component-based development (CBD) (Szyperski, 2002), service oriented architecture (SOA) (Bell 2008 and 2010), model driven software engineering (MDE) (Stahl, 2006) (Pons et al., 2010) and domain-specific modeling (DSM) (Steven and Juha-Pekka, 2008) represent promising technologies in the RS domain.

This paper gives a systematic review (SLR) of the current use of modern software engineering techniques for developing robotic software systems and their actual automation level. The survey aimed at summarising existing evidence concerning the application of such technologies in the field of robotic systems.

[1] Informatics Licentiate, Specialist in Universitary Teaching, PhD in Science (Informatics), Universidad Nacional de La Plata, Argentina. CONICET, Facultad de Informática, Universidad Nacional de La Plata y Universidad Abierta Interamericana, Argentina. E-mail: cpons@info.unlp.edu.ar
[2] Scientific Estimator, MSc in Software Engineering, PhD in Informatic Sciences, Universidad Nacional de La Plata, Argentina. Facultad de Informática, Universidad Nacional de La Plata, Argentina. E-mail: giandini@info.unlp.edu.ar
[3] Informatics Licentiate, Universidad Nacional de La Plata, Argentina. MSc in Software Engineering, Ecole des Mines de Nantes/Vrije Universiteit. Brussels. PhD in Computer Science, Universitaet Bern, Switzerland. Universidad Abierta Interamericana, CONICET, DCyT, Universidad Nacional de Quilmes, Argentina. E-mail: gabriela.b.arevalo@gmail.com

## The need for a review in this field

Although robotic software complexity is high, its reuse is still restricted to libraries. Many libraries have been created at the lowest level for robot systems to perform tasks like mathematical computation for kinematics, dynamics and machine vision (Bruyninckx, 2001). Instead of composing systems out of building blocks having assured services, overall software integration for another robotic system often relies on re-implementing glue logic to bring various libraries together. 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 (Ando *et al*., 2005) is a CORBA-based middleware for robot platforms that uses so-called robot technology components to model functionality distribution. Obviously this is not only expensive and wastes tremendous highly skilled robotics resources but does not take advantage of maturing to enhance overall robustness.

Educational robots have been programmed for more than 10 years (GIRA, 2011) (CAETI, 2011) and robotic kits oriented towards non-expert users have emerged in the last years, giving rise to the development of a significant number of educational projects using robots. Such projects apply robots from kindergarten to higher education, especially regarding physics and technology. However, robotic kits' hardware is constantly changing and its use is not uniform in different regions and even at similar education levels. These robots' technical interfaces should hide such differences so that teachers are not required to change their educational material over and over again. Physical Etoys (GIRA, 2011) is an example of these interfaces; it is a project proposing a standard teaching platform for programming robots, regardless of whether they are based on Arduino, Lego or other technologies.

It is widely accepted that new approaches should be established to meet the needs of developing today's complex RS. Component-based development (CBD) (Szyperski, 2002), service oriented architecture (SOA) (Bell 2008 and 2010), model driven software engineering (MDE) (Stahl, 2006), (Pons et al., 2010) and domain-specific modelling (DSM) (Steven and Juha-Pekka, 2008) represent promising technologies in the RS domain.

Component-based development (Szyperski, 2002) implies that application development should be achieved by linking independent parts (i.e. components). Strict component interfaces based on predefined interaction patterns decouple the sphere of influence and thus partition overall complexity. This results in loosely coupled components interacting via services with contracts. Components such as architectural units very precisely specify using the concept of port for 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 reusability rates and ease of use, but little flexibility regarding implementation platform; most existing components are linked to C/ C++ and Linux (e.g. Microsoft robotics developer studio (Microsoft, 2009), EasyLab (Barner *et al*., 2008), Player/Stage project (Gerkey *et al*., 2001)), although some achieve more independence by using middleware (e.g. smart software component model (Schlegel, 2007), Orocos (Bruyninckx, 2001) Orca (Brooks *et al*., 2005), CLARAty (Nesnas *et al*., 2003)).

Interfaces and behaviour must be defined at a higher level of abstraction so they can be used in systems having different platforms. This prompted the idea of abstract components which would be independent of an implementation platform but could be translated into executable software or hardware components. Migration from code-driven design to model-driven development is mandatory in robotic components to overcome current problems. A model-based description is a suitable means of expressing contracts at component interfaces and applying tools to verify composed systems' overall behaviour and automatically derive executable software. Instead of building tool support for each framework from scratch, one should now try to express required models in standardised modelling languages like UML or any DSL, separating components from the underlying computer hardware. Model driven development (MDD) (Stahl, 2006; Pons *et al*., 2010) and domain-specific modelling approach (DSM) (Steven and Juha-Pekka, 2008) have emerged as a paradigm shift from code-centric software development to model-based development; such approaches promote systematisation and automation in constructing software artefacts. Models are considered first-class constructs in software development and developers' knowledge is encapsulated by means of model transformations. MDD and DSM's essential characteristics are that software development's primary focus and work products are models; their major advantage is that models can be expressed at different levels of abstraction and hence they are less bound to any supporting technology. This is especially relevant for software systems within the ubiquitous computing domain, consisting of dynamic, distributed applications and heterogeneous hardware platforms, such as robotic systems.

Service-oriented architecture (SOA) is a flexible set of design principles used during systems development and integration in computing. A system based on a SOA will package functionality as a suite of interoperable services to be used within multiple, separate systems from several business domains. SOA also generally provides a way for service consumers (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 the loose coupling of services to operating systems and other technologies that underlie applications. SOA separates functions into distinct units, or services (Bell, 2008) which developers make accessible over a network to allow users to combine and reuse them in producing applications. These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format (Bell, 2010).

It is known that such software engineering techniques offer good potential for developing robotic systems so proposals in this area and detecting which work has already been done and which work is pending must be ascertained, as must any proposal taking advantage of the combined application of CBP, SOA and MDE for robotic software system development.

## Systematic literature reviews and systematic mapping studies

A systematic literature review (SLR) (Kitchenham and Charters, 2007; Dybå *et al*., 2003) is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, topic area or phenomenon of interest. Individual studies contributing towards a SLR are called primary studies; an SLR is a form of secondary study.

Other types of review complement SLR, such as systematic mapping studies (SMS). If, during the initial examination of a domain prior to commissioning an SLR it is discovered that very little evi-

dence is likely to exist or that the topic is very broad then an SMS may be more appropriate than an SLR. An SMS allows evidence in a domain to be plotted at a high level of granularity, thereby identifying evidence clusters and evidence deserts to direct the focus of future SLS and identify areas for more primary studies to be conducted. The present work consists of an SLR oriented towards mapping studies due to the extensiveness of our topic of interest.

An SLR involves several discrete activities; Kitchenham *et al*., (Kitchenham and Charters, 2007) summarised SLR stages as follows: planning (identifying the need for a review, specifying the research questions, identifying research, selecting primary studies, study quality assessment, developing a review protocol, evaluating the review protocol), conducting (data extraction and monitoring, data synthesis) and reporting (specifying dissemination mechanisms, formatting the main report, evaluating the report).

***Planning a review***

Review planning specifies the methods to be used when undertaking a specific SLR; pre-defined planning is needed to reduce researcher bias.

*The research questions*

Specifying research questions is the most important part of any SLR. 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 developing robotic systems and what is the current tendency?

**RQ2** Have CBD techniques been applied to developing robotic systems and what is the current trend?

**RQ3** Have SOA techniques been applied to developing robotic systems and what is the current trend?

**RQ4** Have these techniques been used in combination or in isolation?

**RQ5** Which MDE techniques have been applied to developing robotic systems and what is their automation level?

*The search strategy*

A search strategy was used for primary studies; it included the search terms and resources to be searched. Resources included digital libraries, specific journals and conference proceedings. Two digital libraries and one broad indexing service were searched: the 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 May and June 2011 using Boolean query (adapted to each library's particular syntax):

```
(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 search strategy quality, general guidelines recommend considering a question's effectiveness from five viewpoints (PICOC criteria):

**Population**: application area

**Intervention**: the software methodology/tool/technology/procedure addressing a specific issue

**Comparison**: the software engineering methodology/tool/technology/procedure with which the intervention is being compared

**Outcomes**: factors of importance for 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)

The current query was thus organised as follows (i.e. following PICOC criteria)

**Population**: the robotic domain, reflected in the query's first sub-expression

**Intervention**: software and system development specified in the query's second sub-expression

**Comparison**: MDD, SOA and CBD software engineering methodologies were compared or analysed, indicated in the query's third sub-expression

**Outcome**: the query did not restrict the kind of outcomes (as many outcomes as possible were needed by collecting all the available information in the study domain)

**Context**: no restriction was applied

*Study selection criteria*

Study selection criteria are intended to identify primary studies by providing direct evidence about a particular research question. Once potentially relevant primary studies have been obtained, they must be assessed regarding their actual relevance. Study selection criteria are used for determining which studies are included in, or excluded from, an SLR. It is usually helpful to pilot the selection criteria on a subset of primary studies.

Title, abstract and keywords were used in initial screening (giving 195 papers); IEEE computer society digital library contributed 55 articles (28%) and ACM digital library another 140 (72%). The results from searching the SCOPUS indexing system were included in previous results, so SCOPUS contributed no new articles. Studies were excluded that were obviously irrelevant or duplicates (91 articles were eliminated). The remaining 104 papers were then subjected to a second assessment: full copies of these remaining papers were obtained and a more detailed second screening used the following inclusion and exclusion criteria: the paper should be related to software engineering rather than mathematical modelling and/or math simulation, "service oriented" should refer to SOA but not to robots performing a service (37 articles were ex-

cluded). The remaining 67 articles were analysed (http://lifia.info.unlp.edu.ar/eclipse/robotsurvey2011).

***Data extraction strategy***

This strategy defined how information required from each primary study would be obtained; data extraction forms were designed to accurately record the information researchers obtained from primary studies:

| Field name | Type |
|---|---|
| Paper identification | Integer |
| Year of publication | Date |
| Was SOA applied | Boolean |
| Was CBD applied | Boolean |
| Was MDD applied | Boolean |

If the value of the last field was true (i.e., the paper applied MDD), then the following form was filled in:

| Field name | Type |
|---|---|
| Modelling 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 "*modelling language*" field specified the language used to express platform independent models. Some projects used standard UML language while other projects considered that UML was not expressive enough and defined extension by creating a profile. Other proposals did not use UML but defined their own domain-specific modelling language (DSML).

The "*programming language*" field identified the implementation language used as model transformation target. Most PIM models were translated to different languages, being one of the principles of MDD. However, in other cases, PIM models were mapped to a specific high level language, such as Urby, or specific middleware, such as MSRS.

The "*model transformation technique*" field indicated which strategy was used to transform PIM to PSM or code. Some projects implemented the transformation by just using general purpose programming language (GPL), such as Java, while other proposals used existing transformation DSLs, such as ATL, JET or QVT. Most proposals used black-box transformations.

The "*tools*" field denoted which kind of software tools were being used in the project: existing tools (such as EMF or MS DSL tools) or a new specific tool was created.

The "*automation level*" field stated how much work was done automatically. "Full" indicated that code was automatically generated from models, "medium" that a code was partially generated and had to be completed manually and "low" indicated that transformation from models to code was mostly carried out by hand.

***Data synthesis strategy: answering the questions***

Data synthesis involved collecting and summarising the results of the primary studies so included.

Figure 1 shows the response to the first three questions, i.e. "Have MDD/CBD/SOA techniques been applied to the development of robotic systems and what is the current trend? An increasing trend in using all these techniques was observed, CBD being most applied in the robotics field.

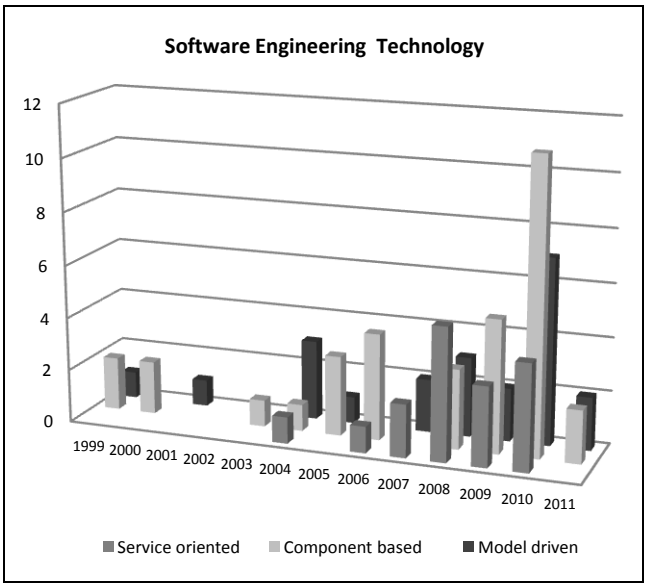


Figure 1. Software engineering technology

Figure 2 shows answers to question 4, showing the distribution of articles in each field. Little interaction amongst the technologies was observed. However, there was promising intersection between MDD and CBD, showing the potential of combining them.

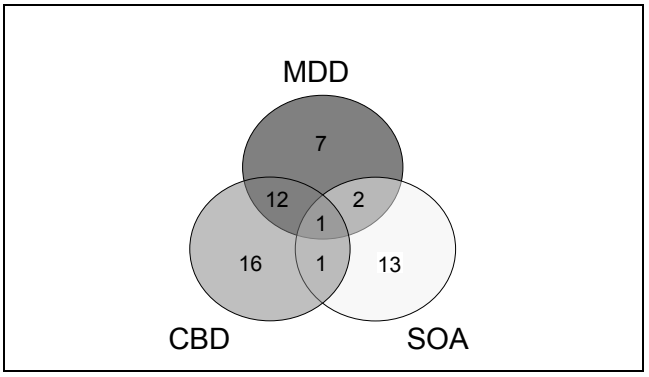


Figure 2. Field intersection

Concerning question 5, Figure 3 illustrates which modelling languages were being applied in robotic projects. Defining new domain specific languages was the most applied technique (64%), followed by UML (27%) and its profiles (9%). Figure 4 shows that 65% of MDD projects took advantage of existing MDD tools, such as ATL, EMF and DSL, while 35% implemented their own modelling and transformation tools. A reasonable expectation is that existing tools will be increasingly reused in the near future.

Figure 5 shows MDD project automation level distribution; only 55% had full MDD automation while 27% had intermediate automation implying abstract model creation and automatic code skeleton generation to be manually completed by the developers. 18% of MDD robotic projects only had a low level of automation, consisting of creating abstract models but manual code derivation.

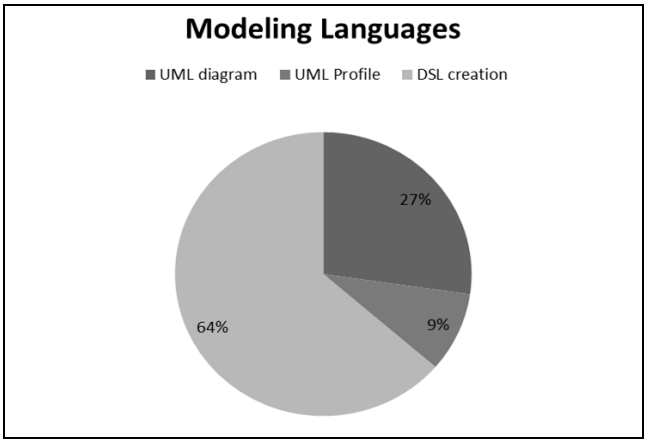


Figure 3. Modelling languages

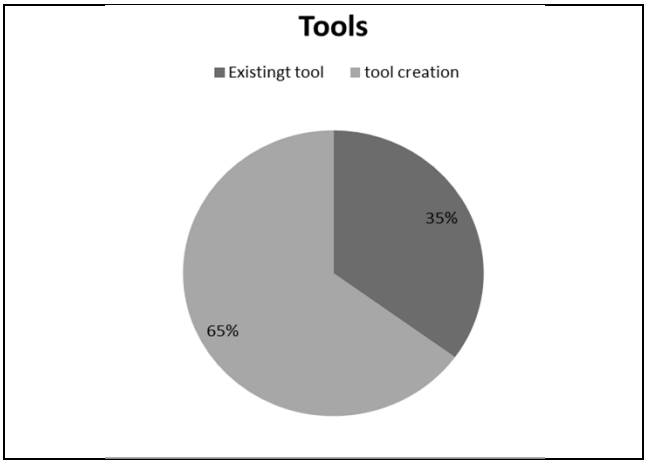


Figure 4. Tools

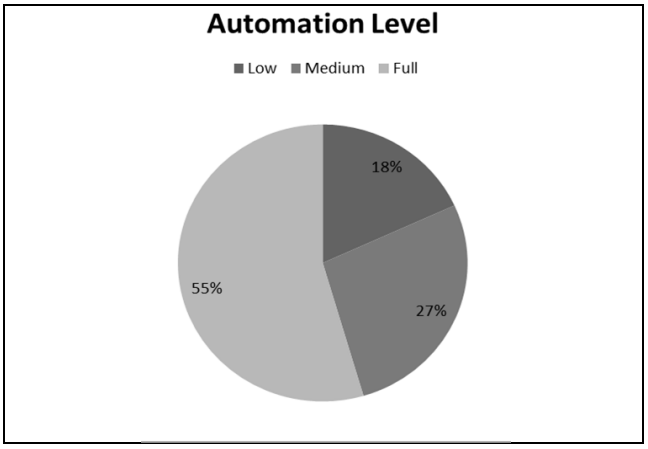


Figure 5. Automation level

## Conclusions

The robotics' community has sufficient experience regarding how to build complex robotic systems. However, one cannot expect significant growth with hand-crafted single-unit systems and it is mandatory to work towards applying engineering principles to cope with the complexity of robotic software systems. Knowledge about what has proved to be a good solution in the software engineering field is usually available. Such knowledge must be explicit and easily accessible for new systems. Applying existing technology would save time and effort which is better put into what is specific in robotics.

This paper has presented an overview of ongoing activities regarding the application of modern software engineering techniques to robotic software development. A growing tendency was identified regarding applying component-based development as well as service-based architecture and model-driven software development, although such techniques have mostly been applied in isolation.

Some work (Basu *et al.*, 2011; Biggs, 2010; Brooks *et al.*, 2005; Jawawi *et al.*, 2008; Min Yang Jung *et al.*, 2010) has taken advantage of CBD for developing robotic systems whilst other proposals (Amoretti *et al.*, 2007; Cesetti *et al.*, 2010) have applied SOA to building autonomic robot systems. Only preliminary proposals were found for applying model-driven development to robotics (Arney *et al.*, 2010; Baer *et al.*, 2007; Brugali and Scandurra, 2009; Brugali and Shakhimardanov, 2010; Hyun Seung Son *et al.*, 2008; Iborra *et al.*, 2009; Jorges *et al.*, 2007; Jung *et al.*, 2005; Sanchez *et al.*, 2010; Schlegel, 2009; Wei *et al.*, 2009) while only one work combined all three technologies (Tsai *et al.*, 2008).

Gaps were identified in current research leading to further investigation after reviewing more than 100 papers on the subject, thereby providing background for appropriately positioning new research activities.

## References

Amoretti, M.; Zanichelli, F.; Conte, G.; A service-oriented approach for building autonomic peer-to-peer robot systems enabling technologies: infrastructure for collaborative enterprises, 2007. WETICE 2007. 16th IEEE International Workshops on, pp.137 - 142

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), 2005, pp. 3933–3938.

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), 2010, pp.: 66 – 74.

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.

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.

Basu, A.; Bensalem, B.; Bozga, M.; Combaz, J.; Jaber, M.; Nguyen, T.; Sifakis, J.; Rigorous component-based system design using the BIP framework software, IEEE Volume: 28 , Issue: 3, 2011, pp. 41 – 48.

Bell, M., "Introduction to Service-Oriented Modeling". Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley & Sons. 2008, pp. 3. ISBN 978-0-470-14111-3.

Bell, M., SOA Modeling patterns for service-oriented discovery and analysis. Wiley & Sons. 2010. pp. 390. ISBN 978-0470481974.

Biggs, G.; Flexible, adaptable utility components for component-based robot software. Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 4615 – 4620.

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

Brugali, D.; Scandurra, P.; Component-based robotic engineering (Part I) [Tutorial] Robotics & Automation Magazine, IEEE Vol. 16 , Issue 4, 2009, pp. 84 – 96.

Brugali, D.; Shakhimardanov, A.; Component-Based Robotic Engineering (Part II) Robotics & Automation Magazine, IEEE Vol. 17, Issue 1, 2010, pp. 100 – 112.

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

CAETI (Centro de Altos Estudios en Tecnología Informática). Proyectos del área robótica. http://www.caeti.uai.edu.ar. Visited June 2011.

Cesetti, A.; Scotti, C. P.; Di Buo, G.; Longhi, S.; A service oriented architecture supporting an autonomous mobile robot for industrial applications Control & Automation (MED), 2010 18th Mediterranean Conference on Pages: 604 – 609.

Dybå, T., Kitchenham, B.A., Jørgensen M., Evidence-based software engineering for practitioners, IEEE Software 22 (1) (2005) 58–65.

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

GIRA Grupo de Investigación en Robótica Autónoma del CAETI. http://tecnodacta.com.ar/gira/ (Visited May 2011).

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, pp. 93 – 98

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 2009, pp. 24 – 33.

Jawawi, D.N.A.; Deris, S.; Mamat, R.; Early-life cycle reuse approach for component- based software of autonomous mobile robot system. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on, Pages: 263 – 268.

Jorges, Sven; Kubczak, Christian; Pageau, Felix; Margaria, Tiziana; Model driven design of reliable robot control programs using the JABC. Engineering of Autonomic and Autonomous Systems, 2007. EASe '07. Fourth IEEE International Workshop on, Pages: 137-148

Jung, E.; Kapoor, C.; Batory, D.; Automatic code generation for actuator interfacing from a declarative specification Intelligent Robots and Systems, 2005. (IROS 2005). IEEE/RSJ International Conference on. Pages: 2839 - 2844

Kitchenham, B.A., Charters, S., Guidelines for performing systematic literature reviews in software Engineering Technical Report EBSE-2007-01, 2007.

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

Min Yang Jung; Deguet, A.; Kazanzides, P.; A component-based architecture for flexible integration of robotic systems Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, Pages: 6107 – 6112.

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.

Pons, C., Giandini R., Pérez, G., "Desarrollo de software dirigido por modelos. Teorías, Metodologías y Herramientas", Ed: McGraw-Hill Education. ISBN: 978-950-34-0630-4.

Sanchez, P; Alonso, D; Rosique, F; Alvarez, B; Pastor, J; Introducing safety requirements traceability support in model-driven development of robotic applications. Computers, IEEE Transactions on Issue: 99 (2010)

Schlegel, C., ''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)

Schlegel, C., Haßler, T., Lotz, A., Steck, A., Robotic software systems: from code-driven to model-driven designs. In procs. Of ICAR 2009. International Conference on Advanced Robotics. IEEE Press (2009)

Stahl, M Voelter. Model Driven Software Development. John Wiley. 2006.

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

Szyperski, C., Component software: beyond object-oriented programming. 2nd ed. Addison-Wesley Professional, Boston ISBN 0-201-74572-0, 2002.

Tsai, W.T., Qian Huang, Xin Sun. A collaborative service-oriented simulation framework with Microsoft robotic studio simulation symposium, 2008. ANSS 2008. 41st Annual Digital Object Identifier: 10.1109/ANSS-41.2008.32, pp. 263 – 270, 2008.

Wei Hongxing; Duan Xinming; Li Shiyi; Tong Guofeng; Wang Tianmiao; A component based design framework for robot software architecture. Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, Pages: 3429 - 3434 (2009)