Concerning a Rational System-level Design Methodology for Autonomous Robotic Systems

Lars Dalgaard[*]
Danish Technological Institute
Centre for Robot Technology
Forskerparken 10
DK-5230 Odense M
Denmark
lars.dalgaard@teknologisk.dk

John Hallam
Maersk McKinney Moller Institute
University of Southern Denmark
Campusvej 55
DK-5230 Odense M
Denmark
john@mmmi.sdu.dk

ABSTRACT

As autonomous robotic systems become inherently more complex it becomes increasingly clear that ad-hoc design methods will fail to deliver predictable and guaranteed performance. This paper motivates the need for a rational system design process and gives the necessary properties and structure such a process must have.

1. INTRODUCTION

The field of robotics is inherently interdisciplinary covering most areas of the natural sciences and even various aspects of project management and social sciences. Each natural science discipline has through the past decades been the focus of intense research and has, at an ever growing pace, introduced increasingly more sophisticated and complex components to the market, supporting the equally growing demands for new consumer, service and industrial robot solutions. A large number of these solutions will in the future be based on mobile robots capable of autonomous behaviour in some specific context, be it household cleaning, the mowing of a golf course or the transportation of units in a hospital.

Traditionally mobile robots have been applied in two main scenarios: 1) As “simple” transportation units in structured environments – for example factory floors – or 2) as research platforms conceived in laboratories for the testing of various research specific components – for example new navigation systems or controller architectures. The new generation of mobile robots placed in unstructured and possibly unknown environments, have very specific tasks to carry out, and are constrained in various ways by firm requirements of safety, social and legal compliance, stability, robustness, maintainability, operability, cost, and the obvious performance criteria. In essence, the field of mobile robotics is shifting from a pure mechatronic domain to a more socio-technical domain where people who use and interact with the robots, the operational processes and the emergent properties of the *systems* as a whole become crucial. In this paper we refer to the constellation of the autonomous robots, their users, the environment in which they work and the tasks they carry out as an *autonomous robot system*; and, if the context is of a more service or industrially oriented character, as an *autonomous robotic system* abbreviated as *ARS*.

[*]LD is an Industrial PhD student at the Danish Technological Institute, Centre for Robot Technology, Odense, Denmark, with JH as university supervisor. LD is partially funded by stipend number 76303 from the Danish Council for Technology and Innovation.

©Lars Dalgaard, John Hallam, 2009
Published by the British Computer Society
Beyond Gray Droids: Domestic Robot Design for the 21st Century Workshop of HCI2009, September 2009, Cambridge, UK.

Today when robot researchers and developers have to conceive and design a new robot, the process is often guided by several “local” design approaches each linked to a specific domain, resulting in a natural focus on the components of the system rather that on the system as a whole. These traditional “component design approaches” are often sufficient when designing only parts of a system solution but, if a complete robot- or robotic *system* has to be conceived, chances are that the inevitable integration process will be guided solely by experience and ad-hoc approaches. In our experience this practice often leads to systems which do not obey their initial system requirements and may in the worst case result in a total project collapse – this, of course, depending on the complexity of the system.

As more and more robots enter our homes and become integral parts of service units in hospitals, social and legal compliance has to be *guaranteed* by the provider. Such issues cannot easily be addressed *after* a robot has been devised but need to be an inherent part of the design from the beginning. In these cases the need for a rational system design process becomes particularly pressing.

The ad-hoc robot design approach is therefore no longer sufficient and an approach is needed that is able to balance the design (or choice) of the system’s physical robot(s) and their components, their working environment, and the task they are required to solve, under the constraints imposed by the requirements of both the system and component levels. This view point is essential since only by keeping the design focus at the system-level whilst handling the component level can a design be conceived which can account for the tangled interdisciplinary interactions inherent in robotic systems.

The purpose of this paper is to offer the initial steps towards devising a rational design methodology for ARSs. This paper will focus mainly on the necessary properties and structure of such a design methodology and only lightly deal with *how* to apply it in real-life. This issue will be addressed more thoroughly in later publications.

The structure of this paper is as follows: Section 2 provides an overview of relevant previous work in system design, Section 3 outlines the necessary properties of the proposed robotics design methodology, Section 4 presents the resulting structure of the design methodology and Section 5 presents a comparison to existing methods and methodologies. Section 6 sums up the results and indicates what future work needs to be done.

2. PREVIOUS WORK

This section gives a short overview of some of the system-level design methods and methodologies that are used in scientific disciplines directly related to the field of robotics.

Software Engineering has a long history of broad systems-level thinking where user-interfaces, maintenance and economic considerations alongside the actual design of the software functionality often are key issues. Examples can be found by regarding for example the field of Object Oriented Analysis and Design (OOAD) [10] and Extreme Programming [3].

In the design of digital electronics and embedded systems containing for example Field Programmable Gate Arrays (FPGA) and micro-controllers, various system-level design methods exist. An example is an UML-based design methodology for real-time and embedded systems presented in [6].

In (traditional) Robotics- and Control Engineering, some broader system-level approaches exist but they are mostly concerned with designing kinematic configurations of modular robots to suit given application contexts, as is for example the case with the work of Farritor et. al. [8] and Paredis [13]. Other approaches more focused on the mechatronical parts of systems can be found in for example [7] and [5].

Within the field of Systems Engineering several systems-level methods do exist, for example [1, 2, 12, 18, 9], but have, to the best of our knowledge, not yet been directly applied to the robotics field at least with results accessible to the public.

The field of Product Development has traditionally much focus on the system-level as its development models treat all aspects from concept development to product maintenance [19]. Some commonly used tools hereof, such as Quality Function Deployment (QFD) [16], have been applied to robotics systems, for example [17]. However, we have found no references to complete robotic systems being developed solely using product development methodologies.

Only relatively few disciplines within the robotics research community seem to have actually dealt with system-level oriented design approaches for designing complete ARSs, and most of this work is based on the design and evolution of the robots' controller architecture or the controller itself as is the case for many disciplines within the AI field (see for example [4] for a discussion). Of these, Embodied AI is possibly the discipline with the focus closest to ours (see for example [15]).

Based on this brief outline of different design methods and methodologies and on our personal experiences, we do not believe that an actual system-level design methodology for complete ARSs has yet been devised.

3. PROPERTIES OF THE METHODOLOGY

The overall aim is a design methodology which will facilitate the design task by equipping the robotic system designer with tools and guidelines that allow him/her to transform 1) the task to be carried out, 2) the environment in which the system must operate and 3) the imposed constraints, into a *process description* leading to *sufficient system designs* meeting the initial requirements.

The *process description* will consist of techniques from various scientific fields facilitating a rational system-level design process while concurrently managing the organisational aspects of the project. The *sufficient system designs* will consist of sets of design recommendations of the robot(s) needed, of its/their functional design, of the possible alteration of the environment and of the possible adaptation of the initial task specification. These recommendations will describe different scenarios leading to solutions of the initial problem.

The overall structure of the design methodology is sketched in Fig. 1, where the altered Environment and the adapted Task are denoted Environment* and Task* respectively.

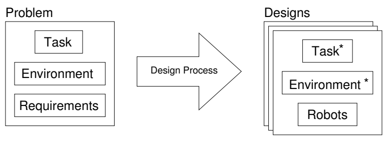

Figure 1: The overall structure of the design methodology.

In order to ensure that the proposed design methodology will provide the necessary means to handle effectively a design process that results in sufficient system designs, its structure must be well established. Seeking inspiration in Software Engineering, Mathiassen et. al. [11] state that a design process is about balancing the choice of a *mode of operation*, which can be either *analytical* or *experimental*, and a choice of *means of expression*, which can be either *specifications* or *prototypes*. Mathiassen et. al. summarise their results in what they call "The Principle of Limited Reduction" where they argue that a successful design process of a complex (software) system has to comprise both analytical and experimental modes of operation and both specifications and prototypes as means of expression. We believe this principle applies equally well to the design of ARSs and it can therefore be used as a basis for comparing and combining different design approaches.

We have identified 12 criteria that we deem essential of a rational design methodology for ARSs where the first ten criteria concern the design process level and the last two the system design level. Each of the criteria will be presented in the following along with a brief elaboration on their background.

1. **Facilitate a rational design process:** A design process being rational implies that every step taken is always on the optimal path to the goal. Such processes obviously only exist in theory, however, it is possible to *fake* one by following the ideal process as closely as possible and producing the documentation we would have produced had it indeed been rational [14].
2. **Facilitate iterations:** Developing component prototypes and conducting experiments are integral parts of a robotic system design process as this is often the only way to gather the necessary information needed to make rational decisions. Iterations are therefore imperative to the design process.
3. **Offer the means to evaluate the fitness of a design:** A way of evaluating the fitness of a given system design is imper-

ative. Such a measure will enable a continuous tracking of the overall design progress thereby facilitating that the design process can be kept as rational as possible.

4. **Facilitate collaborative design:** Often several partners have to collaborate in the design and construction of a robotic system due to its inherently interdisciplinary nature. The means to establish what competences are needed to carry out a particular project are therefore necessary in a design methodology.
5. **Facilitate prototypes and experimental setups:** Prototypes and experimental setups often create undesired branches in a project process. The design methodology must therefore incorporate the means to determine what type of physical construction is feasible at any given point during the process.
6. **Produce process deliverables:** It is imperative to produce sufficient deliverables at the end of each design increment, containing at least the process documentation, the design documentation and possibly any constructed prototypes.
7. **Facilitate different design tools:** The design methodology should provide flexible interfaces to allow easy employment of project specific design tools needed during the design process.
8. **Be easy to explain and use:** The structure of the methodology, its tools and the project specific tools constitute the actual design method, and it is important that the individual design tasks can be readily communicated to and understood by the people actually carrying them out.
9. **Allow for different levels of generalisation:** The design methodology should support the development of robotic systems with various levels of generalisation, that is, from systems comprising generic robot platforms to systems comprising highly specialised robot platforms.
10. **Allow for different levels of autonomy:** The design methodology should support solutions employing different levels of autonomy, that is, from systems comprising remote controlled robots to systems comprising fully autonomous robots.
11. **Give a set of solutions:** Designing a robotic system is a complex task and may naturally result in several feasible design solutions obeying the initial requirements and solving the task. The design methodology should be able to produce these sets of solutions and not discard solutions prematurely.
12. **Produce design documentation:** Since the design process is rational, the *design documents* delivered at the end of the design process reflect a rational design process, meaning a recipe showing how to progress from the initial analysis to the final design in a single pass, thus masquerading as a strictly sequential process.

4. STRUCTURE OF THE METHODOLOGY

In order to be a useful tool for the robotic system designer the design methodology must provide a homogeneous and rational context setting in which both the design process and the actual design can be represented. That means that the various components of the methodology need to have clear interfaces to neighbouring components in order to guarantee an unambiguous, yet adaptable, methodology.

A natural choice is to divide the process into *phases* representing the sequential flow of the incremental process. Each phase may have several iterations internally and will at the end produce a set of *deliverables* comprising documentation, prototypes and possibly components. This overall structure is sketched in Fig. 2.

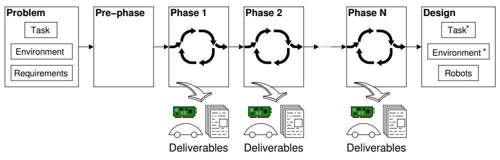

Figure 2: The phases of the design methodology

Table 1: A comparison of our design methodology to other system-level design methods

	UP	WM	SM	XP	NASA	DoD	INCOSE	SMC
C.1	×		×	×	×	×	×	×
C.2	×		×	×	×	×	×	×
C.3					×	×	×	×
C.4	×		×	×	×	×	×	×
C.5	×		×	×		×	×	×
C.6	×	×	×	×	×	×	×	×
C.7	×	×	×	×	×	×	×	×
C.8		×	×	×				
C.9	×	×	×	×	×	×	×	×
C.10	×	×	×	×	×	×	×	×
C.11					×	×	×	×
C.12	×	×	×	×	×	×	×	×

An initial phase, *pre-phase*, is needed to process the information from the problem domain and make preliminary analysis needed throughout the design process – for example stakeholder analysis and analysis of the competences needed for realising the project. The information will also help set up the management structure of the project.

The subsequent phases, *Phase 1–Phase N*, will each comprise the disciplines needed to carry out the design process for the system in question. Several standard tools and methods from Product Development, Systems Engineering and Software Engineering can easily be used to handle the more general process specific tasks such as customer requirements, test and experiment planning, and project management, alongside the more domain specific tasks like simulations tools, CAD applications and rapid prototyping equipment. Also a per-system customisation will often be required in order to accommodate, for instance, project specific requirements. If a robotic system based on mobile robots is to be realised such that it can carry out tasks in the home of a disabled person, knowledge regarding, for example, the design, construction and control of specialised grasping equipment may be crucial. Tools to support this will be directly applicable in the design methodology.

5. COMPARISONS

As presented in Section 2, Previous Work, not much work has been done previously in rational system-level design methodologies or methods for robotic systems. However, many contributions exist in the related fields of Software Engineering and Systems Engineering. Both have for many years contributed to theory and practice on methods and methodologies and thus form a natural basis of comparison to our proposal.

Eight methods have been chosen for comparison: The Unified Process (UP), the Waterfall model (WM), the Spiral model (SM), Extreme Programming (XP), NASA's-, DoD's-, INCOSE's- and SMC's Systems Engineering methodologies. In Table 1 each method is evaluated with respect to the required properties presented in Section 3, and are marked with an "×" if the method supports the property.

The comparison reveals that within Software Engineering, the

Unified Process (UP) and Extreme programming seem to be the methods fulfilling the criteria best. The Waterfall model falls through and the Spiral Model is more of a reference model than a fixed model – it can therefore be adapted to the required needs.

Within Systems Engineering, the identified test methodologies are actually all full development models that all deal with the required criteria in more or less the same way. However, their application scope being very broad and primarily dedicated to high-performance military equipment, potentially makes them cumbersome to apply in our context. This does not disqualify them in any way, but our impression is that as our research progress we will find specific issues that call for more dedicated design methodology.

The conclusion is, that methods and methodologies exist in both Software Engineering and Systems Engineering that may be used and adapted – at least partially – to the design of ARSs. That implies that several concepts, methods and ideas from the test cases will help shape our methodology and may well become directly part of its final tool box.

6. CONCLUSIONS AND FUTURE WORK

This paper makes a first contribution towards the realisation of a rational design methodology for autonomous robotic systems. We have argued that such a methodology is increasingly necessary as robotic systems are targeted toward more complex tasks and environments. We have presented the properties required of such a methodology and have proposed a structure that will support them. The structure is based on an iterative development model implemented as several sequential phases each containing one or more development iterations, producing deliverables comprising documentation, prototypes and possibly components.

We have compared our methodology to eight existing methods and methodologies from Software Engineering and Systems Engineering and have concluded that no one existing method will suffice in our case. However, several concepts, methods and ideas from some of them may well become part of the final design methodology tool box.

Much work is still needed before a finished design methodology is ready. The next period of our research will be concentrated on refining the methodology by 1) establishing more specifically what the design phases shall contain, 2) conducting further work on the proposed design tool and the knowledge representations supporting it, and 3) choosing a representation of the task, environment and requirements used as input parameters to our methodology, and transforming them into processable information.

From there the focus will be mainly on robotic systems containing mobile robots as part of the design solution. This is to focus our work effort and create a more well-defined basis for real-world test cases. It is our belief that only through field tests can our design methodology really prove its worth.

We are also currently in the process of developing a design tool that will aid the robotic system designer in rationally selecting physical system components by allowing her/him to easily observe their system wide impacts. We expect this tool to become a natural part of the tool-box when designing future ARSs.

7. REFERENCES

[1] *Systems Engineering Handbook: A "How to" Guide For All Engineers*. International Council on Systems Engineering (INCOSE), second edition, July 2000.

[2] *SMC Systems Engineering Primer & Handbook. Concetps, Processes, and Techniques*. Space & Missile Systems Center, U.S. Air Force, third edition, April 2005.

[3] K. Beck. *Extreme Programming Explained: Embrace Change*. Pearson Education, Inc., publishing as Addison Wesley Longman, 2000.

[4] P. M.-O. celik and A. K. Mackworth. Situated Robot Design with Prioritized Constraints. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2004*, Sendai, Japan, 2004.

[5] P. Coste, F. Hessel, P. L. Marrec, Z. Sugar, M. Romdhani, R. Suescun, N. Zergainoh, and A. A. Jerraya. Multilanguage Design of Heterogeneous Systems. In *Proceedings of the Seventh International Workshop on Hardware/Software Codesign, 1999. (CODES '99)*, pages 54–58, 1999.

[6] G. de Jong. A UML-Based Design Methodology for Real-Time and Embedded Systems. In *Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE'02)*, pages 776–779, March 2002.

[7] C. W. de Silva. *Mechatronics: An Integrated Approach*. CRC Press, 2005.

[8] S. Farritor, S. Dubowsky, N. Rutman, and J. Cole. A Systems-Level Modular Design Approach to Field Robotics. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, volume 4, pages 2890–2895, Minneapolis, MN, USA, April 1996.

[9] D. K. Hitchins. *Systems Engineering: A 21st Century Systems Methodology*. Wiley Series in Systems Engineering and Management. John Wiley & Sons, Ltd, 2007.

[10] L. Mathiassen, A. Munk-Madsen, P. A. Nielsen, and J. Stage. *Objekt Orienteret Analyse og Design*. Marko, Aalborg, Denmark, 1998.

[11] L. Mathiassen and J. Stage. *Information, Technology and People*, volume 6, chapter The Principle of Limited Reduction in Software Design. 1992.

[12] NASA. NASA Systems Engineering Handbook. Technical report, NASA, June 1995.

[13] C. J. J. Paredis. An Agent-Based Approach to the Design of Rapidly Deployable Fault Tolerant Manipulators, August 1996.

[14] D. L. Parnas and P. C. Clements. A Rational Design Process: How and Why to Fake It. *IEEE Transactions on Software Engineering*, SE-12(2):251–257, february 1986.

[15] R. Pfeifer. Building Fungus Eaters: Design Principles of Autonomous Agents. In Maes, Mataric, Meyer, Pollack, and Wilson, editors, *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, 1996.

[16] J. B. ReVelle, J. W. Moran, and C. A. Cox. *The QFD Handbook*. John Wiley & Sons, Inc., 1998.

[17] C. G. Sørensen, R. N. Jørgensen, J. M. Pedersen, and M. Nørremark. Hortibot: Application of Quality Function Deployment (QFD) Method for Horticultural Robotic Tool Carrier Design Planning. Presentation at the 2006 American Society of Agricultural and Biological Engineers (ASABE) Annual Meeting, July 2006.

[18] D. o. D. Systems Management College. *Systems Engineering Fundamentals*. Defense Acquisition University Press, Fort Belvoir, Virginia 22060-5565, January 2001. Supplementary text.

[19] K. T. Ulrich and S. D. Eppinger. *Product Design and Development*. McGraw-Hill, fourth edition, 2008.