ML DESIGNER™ as Simulation Tool in Robotics

Mario Schulz[1], Ivan Veličković[2], Saša Anđelković[2], Volker Zerbe[1], Goran S. Đorđević[2]

Abstract –***Robots are complex systems that require multidisciplinary approach to development. Hence, both research and production of a robotic system require a tool that would provide means for coordination between teams with different areas of expertise as well as the ability to integrate, simulate and debug the system in a comprehensive way. Also, with the growing popularity of service robotics, research and modeling of human-machine interaction gains the attention. From a simulation tool, it is expected to support the means for describing this interaction. For that purpose we are investigating the ML Designer™, not just as a possible simulation tool, but also as a modeling and design paradigm. For verification purposes we will compare it against MATLAB Simulink™. A pneumatic system will be modeled with both tools and a comparison will be performed.***

Keywords – **Simulation, Robotics, Pneumatics, Control systems.**

I. INTRODUCTION

The design of a robot is highly complex task opening several important research problems: mechanical design, design and control of the actuation system, high level control software design etc. All those problems are related to different technical disciplines and are usually investigated by teams with different areas of expertise. Although a common approach to reduce the costs of a research lead to extensive simulation and mathematical modeling of the investigated problem, simulation of a system as a whole, in this multidisciplinary case, imposes quite a challenge. Simulation models are constructed in a manner that provides the best insight into the behavior of interest. Therefore, for a system composed of various components, different behaviors of interest, or to say Quality of Service criteria, may be present. To illustrate this statement, take a robot vacuum cleaner for an example. Let it be a small robot with couple of electro-motors for actuation, vacuum cleaner system, a rechargeable battery power supply, and a microcomputer based control system. It is supposed to detect human absence in the room and start cleaning. When the cleaning is done or when the batteries are nearly empty it should go back into its docking station for recharging. For actuation system developers it is important that their drive accurately follows the required position-velocity-acceleration profiles with acceptable power consumption.Their simulation model would take into account dynamical properties of the system and the feedback loop parameters in order to determine time responses. Yet another simulation may be performed by the developers of the driver electronics that should provide an insight into the driver electronic power consumption. On the other hand, high level control system will take into account spatiotemporal representation of the cleaning task in order to test various cleaning algorithms. This model would probably take into account the behavioral description of the actuation and sensor system, but the power consumption behavior would probably be left out. A question arises: how can we determine, from a user point of view, that this robot would perform the cleaning task in a satisfying way? User doesn't care if the robot fails, whether it appeared because of the bad algorithm or because it ran out of power. For a user the system is a whole and it should be designed, simulated and tested in that way. It is important to notice that, theoretically, it is possible to produce the model that would take into account complete description of the system. Problem lies within the tools. For instance, although it is possible to simulate a mechanical system within the VHDL it is neither well supported nor recommendable.

Also, regardless of its complexity, design process for a system starts with a use-case specification. Potential misconceptions in this phase of the development may lead to devastating results. Therefore a tool that can somehow test the consistence of this early phase models of the system, but yet to be scalable enough to accept later detail refinement can significantly reduce costs. Baring this in mind the MLDesigner™ is developed [1].

This document is organized as follows: In Section 2 a brief introduction to MLDesigner™ as well as its modeling and design paradigm are exposed. Section 3 describes the pneumatic system we used to test MLDesigner™ simulation capabilities and discuses the performance. The same simulation is performed with MATLAB Simulink™ and the comparison is given.

II. BRIEF INTRODUCTION TO ML DESIGNER™

MLDesigner is a general purpose, Linux platform, tool for simulation and visualization of complex heterogeneous systems. Its multi-domain simulator permits the seamless integration of the design flow from mission/operational level to implementation handoff and test of complex designs. It provides means to facilitate and coordinate collaborative development. MLDesigner can be used for design and analysis for a broad range of applications, from complex systems like mobile/ fixed communication networks, satellite communication /navigation /observation systems, performance and architecture tradeoff of electronic and mechanic systems, VLSI integra-

[1]Mario Schulz and Volker Zerbe are with TU Ilmenau, Computer Science and Automation Faculty, System and Control Theory Department, Ilmenau, Germany, Email: Volker.Zerbe@TU-Ilmenau.De

[2]Ivan Velickovic, Sasa Andjelkovic and Goran Djordjevic are with University of Nis, Faculty of Electronics engineering, Robotics Lab, 18000 Nis, Serbia. Email: gorandj@elfak.ni.ac.yu

ted circuits, and automotive navigation/communication system design to simple logic design [2].

MLDesigner models are defined graphically as hierarchical block diagrams. Blocks have defined inputs and outputs that are connected via visible links or via shared memories. Control and information is passed between blocks via data structures known as particles (tokens). To increase execution speed the bottom level blocks contain behavioral description of primitives in form of compiled C++ code. Source code of all primitives is provided for custom upgrade/modification purposes. Higher-level blocks are structurally described by means of interconnected bottom level blocks. There are more then 1400 core library modules and more then 260 example systems in MLDesigner. These library modules can easily be used to create new modules or whole systems. A system model can be constructed through the graphical editor or with the PTcl command language. You may specify the functionality of your modules by a hierarchical block diagram, a finite state machine, a module defined in the C/C++ like PL language, or by a PTcl module definition. Behavior of the model as well as functionality and appearance of the development environment can easily be extended by various supported scripting languages.

As can be seen from the Fig. 1 all supported tools, interfaces and mechanisms are available trough a graphical IDE environment [3]. It offers model editors which provide means to edit, configure and mange graphical and scripted models of a system, a model debugger, the library browser, CVS version control interface and various visualization tools to analyze simulation results and measure performance. The simulation results may be viewed through an animation view while the simulation is running and/or by the post-processing graphical plots. From the perspective of portability it is important to mention that conversion of MLDesigner models to other formats as well as importing of models made by other tools is well supported.

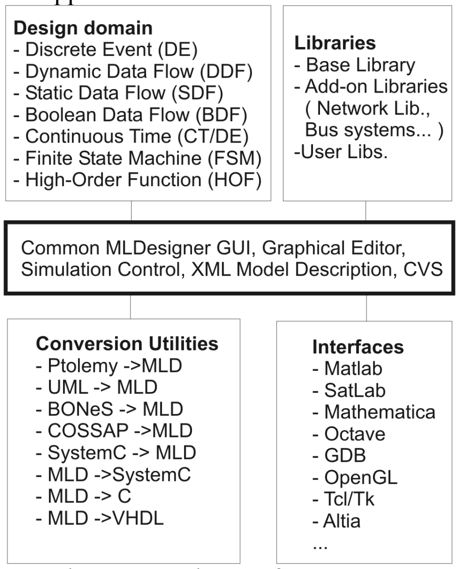

Fig. 1. MLDesigner software system

MLDesigner supports several different simulation domains. It is up to a user to choose a domain that best suit his/hers needs. Most important property of MLDesigner is that it allows for Multidomain simulation modules to be combined and simulated together. Following domains are available [1]. Discrete Event (DE) domain is based on event driven approach and it is well suited for modeling of large complex systems and for integrating and managing the execution of design elements developed in other domains. Dynamic Data Flow (DDF), Static Data Flow (SDF) and Boolean Data Flow (BDF) are based on the dataflow model of computation. These domains are intended for data processing applications. Continuous Time/Discrete Event (CT/DE) domain, still in experimental phase, allows simulation of the systems that can be described with linear and nonlinear algebraic equations. Based on Dormand-Prince variable step ODE solver, it is good for simulation of mechanical and electrical systems. The Finite State Machine (FSM) domain describes behavior in terms of states, transitions between states, and actions that affect or are affected by transitions. It can be used to define communication protocols or to describe the behavior of control units. Finally, the High-Order Function (HOF) domain enables high order function style of coding by allowing functions to be transferred to other functions as arguments.

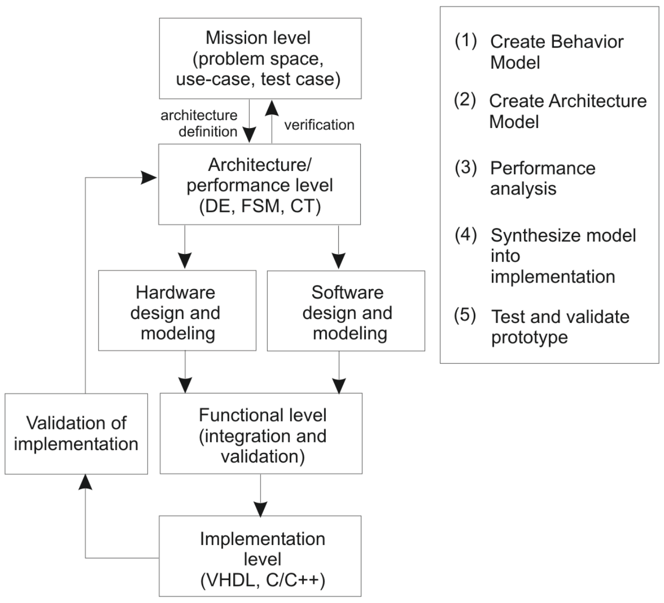
Mission level (problem space, use-case, test case)
architecture definition
verification
Architecture/ performance level (DE, FSM, CT)
Hardware design and modeling
Software design and modeling
Validation of implementation
Functional level (integration and validation)
Implementation level (VHDL, C/C++)
(1) Create Behavior Model
(2) Create Architecture Model
(3) Performance analysis
(4) Synthesize model into implementation
(5) Test and validate prototype

Fig. 2. Mission level design flow. MLDesigner approach

MLDesigner™ modeling paradigm is based on mission oriented approach to development (ML in the name stands for Mission Level). Although MLDesigner supports bottom-up design, it is intended for top-down iterative design process (also known as a spiral design process) [1]. Therefore it supports both behavioral and structural description of subsystem models, allowing a user to choose the level of detail that should be considered at the particular phase of the design. On Fig. 2 the design flow proposed by MLDesigner is depicted. Modeling of a system, regardless of its complexity, starts with informal use-case and test-case specification. At this phase the Behavioral Model of the system is composed. The system is described through services that it provides to the outside world. According to this specification, the Architectural Model is constructed. Considering required services of the system, it is coarsely modeled as a collection of interconnected blocks with established protocols of data/resource flow. With each following phase this model can be upgraded with technical and implementation details. At each level of development test of consistency and plausibility, as well as performance estimation could be performed. To facilitate the final-implementation phase of the development MLDesigner offers tools for conversion of the model into appropriate implementation-ready formats.

III. SIMULATION IN ROBOTICS BY MLDESIGNER™

For a modeling and simulation tool to be used throughout the design of a robot system it is important that it provides the means to describe effects relevant to various technical disciplines involved in the design process. First, such tool should be able to describe electromechanical processes related to the robot mechanical structure, actuation and drives. Coordinated motion of the robot joints is achieved by sophisticated control algorithms that should, also, be integrated into this description. So far, this description is related to time domain. However, high level software, usually implemented in the event-driven manner, is more related to data domain. It deals with robot's reaction to the environment based in the current state, interpreted sensor data and some built-in inference/decision mechanism. Yet another design approach is applied to sensors. It involves implementation of data processing and pattern recognition algorithms. To include all effects related to robot system design the modeling tool should be able use various design domains interchangeably. As can be inferred from the previous chapter and judging by numerous applications in communication and networking [1, 2, 5], MLDesigner™ should be able to successfully meet these demands.

To verify this we have performed a comparison between MLDesigner and MATLAB Simulink software packages. The tests are performed only for continuous time domain which is available in both tools. Knowing that this domain is still in the experimental phase, the results of this test should give us the good insight into applicability of this tool to simulation of electromechanical systems. The Simulink is chosen because it is commonly used for this type of simulations in various technical disciplines.

The benchmark system is made of a pneumatic actuator, pneumatic valves used, a source of clean compressed air, a set of hoses used for air transport, and a set of sensors and specialized hardware added to control desired behavior of the system. Few parts of the system are shown in Fig. 3. The piston of the cylinder pushes and pulls mass-spring payload. The air pressure in the chambers of the cylinder is controlled by a couple of PCM valves: one, designated as IN, controls the air flow from a source of compressed air (not presented on the figure), and the other, designated as EX, controls the air exhaust by connecting the chamber to the atmosphere. Air pressure in cylinder chambers is monitored via pressure sensors attached to the cylinder ports, while the position of the load is measured by an incremental encoder. Mathematical modeling as well as parameter identification is performed as described in [4]. Complexity and nonlinearity of the pneumatic system was the primary motive for choosing it as a benchmark case. Graphical representation of both models is given on Fig 4.

Both tools were able to perform the simulation successfully. There were differences related to both model construction and model execution. First of all there was a difference in the available support. MLDesigner was not supported with some basic building blocks such as continuous derivative. However, we were able to easily overcome this by model reconfiguration. There was another method we could use: we could make our own continuous derivative block either by modifying and compiling some of the available C++ block templates, or by scripting it in some of the available script languages. Unlike Simulink which is mostly graphically oriented, for MLDesigner model description and extension through scripting appears more natural especially for members of the GNU community. Although very flexible and scalable the scripting approach posses a drawback – it may prolong the simulation time.

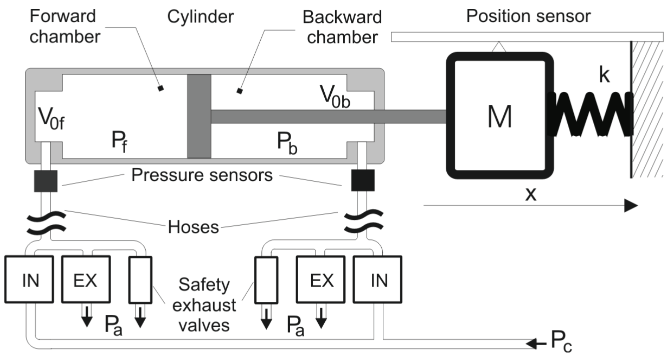

Fig. 3. Benchmark pneumatic system

From the model execution perspective, there was one notable difference. Unlike Simulink that is supplied with a variety of fixed-step and variable step solvers, MLDesigner comes with only Dormand-Prince variable step solver (denoted in Simulink as ode45, currently default). This is not a limiting factor because this algorithm is fast and stable, except for debugging purposes. For example, within the early phases of model construction we encountered some artifacts in the model response which were difficult to physically explain. We suspected that those artifacts were induced by the solver as the numerical noise. The easiest way to check this hypothesis would be to change the solver itself, because the artifacts appeared to be immune to changing in solver time parameters. As later appeared it was a construction flow rather then a numerical noise.

There was also a notable difference in model execution time. We are making this statement with reserve because the testing conditions were not the same. The models were executed on different machines with different operating systems. Regardless, our subjective impression was that, given the circumstances, MLDesigner model execution is significantly slower. Evaluation of other resources required for model execution was not performed.

IV. Conclusion

According to results presented in this paper MLDesigner shows good potential as a simulation tool in robotics. All required resources are available and are easy to use. The tool is well documented with many helpful examples and experiences from experts from all around the world.

Our tests were oversimplified for a program like ML Designer^{TM}. We neither tested its full potential of multidomain simulation nor followed its Mission Level design methodology. Our intent was to test it against a tool which performances are well known. From our personal perspective, it excided our expectations. Our further work will be focused on integration of constructed model into a robot with 2DOF. It is important to note that MLDesigner^{TM} is not just a simulation tool. It is a platform that supports and generalizes a design method which is present and tested in software industry for years. Finally, MLDesigner^{TM} is a free tool for education purposes which may make it more available to students interested in this domain.

References

[1] www.mldesigner.com

[2] Gunar Schorch, at. al., “*System-Level Simulation Modeling with MLDesigner™*”, MASCOTS 2003.

[3] Horst Salzwedel, "*Large scale networked system simulation using MLDesigner*", 6th Biennial Ptolemy Microconference, Berkley 2005.

[4] I.Veličković, S.Anđelković, M.Rašić, G.Đorđević, "*Mathematical Model of a Pneumatic System. Simulation and Experiments*", ETRAN 2007

[5] T. Liebezeit, V.Zerbe, "*Mission Level Design of Autonomous Underwater Vehicle*", ICAIS 2002