MODELING REAL-TIME APPLICATIONS FOR WIRELESS SENSOR NETWORKS USING STANDARDIZED TECHNIQUES

Andreas Blunk, Mihal Brumbulli, Ingmar Eveslage and Joachim Fischer
Department of Computer Science, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany

Keywords: Wireless sensor networks, Standardized languages, UML, SDL-RT, Code generation, Simulation, Performance evaluation.

Abstract: The development of applications for Wireless Sensor Networks is a challenging task. Any approach for developing such applications needs to provide means for describing their functionality, obtain an executable, and be able to evaluate their potential real-time behavior. This paper shows how standardized techniques can be used to address these challenges by giving a real application example. We evaluate our approach based on experiment results and provide some considerations on the used standardized languages.

1 INTRODUCTION

Wireless Sensor Networks (WSNs) have become popular for monitoring physical phenomena. Still, the development of applications for such complex systems remains a challenging task. For this the developer needs to describe the application's functionality, obtain an executable from it, and be able to evaluate its potential real-time behavior. These are general requirements that any approach for developing applications for WSNs needs to consider.

We focus on using standardized techniques as a possible solution, because they offer description means on a much higher level than general purpose languages and can be understood by large groups of people from different domains.

In our days the most recent and well-known standardized language is the Unified Modeling Language (UML) by the OMG (OMG, 2010). Although UML has proven successful for some applications, its semantics are sometimes unclear or incomplete (by so-called semantic variation points). They leave room for interpretation, making model execution a difficult task.

But standardization efforts did not start with UML. The Specification and Description Language (SDL) (ITU, 2007a) has a long history in modeling the functionality of telecommunication protocols. The strong side of SDL is a combination of the wide-used concept of communicating state automatons. These are represented by a graphical notation with a mathematical foundation of the static and dynamic semantics of the language. Because of these strengths, the standardized version SDL-2000 had an important influence on the UML 2.0 standard. Also efforts were made in defining SDL as a precised subset of UML (ITU, 2007b). A more pragmatic approach is SDL-RT (SDLRT, 2011), which is an extension of standard SDL by UML.

Our goal is to evaluate the standardized techniques using SDL-RT for developing applications for WSNs. We focus on (1) modeling their functional aspects, (2) generating executable code, and (3) evaluating performance based on their real-time behavior. For this we provide a real example of an earthquake early warning application.

In Section 2 we present some related work. Next, we start by introducing SDL-RT (Section 3) as standardized technique for describing communicating systems. In Section 4 we outline the functional aspects of our application example and our approach for code generation and performance evaluation. In Section 5 we provide our evaluation based on experiment results and some considerations regarding the used standardized languages. We present the conclusions of our work in Section 6.

2 RELATED WORK

Network simulators are considered a good tool for evaluating the potential real-time behavior of applica-

Blunk A., Brumbulli M., Eveslage I. and Fischer J..
MODELING REAL-TIME APPLICATIONS FOR WIRELESS SENSOR NETWORKS USING STANDARDIZED TECHNIQUES.
DOI: 10.5220/0003599601610167
In *Proceedings of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2011), pages 161-167
ISBN: 978-989-8425-78-2
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

tions running on a distributed and networked environment. Simulators like ns-2 (NS2, 2011), ns-3 (NS3, 2011), OMNET++ (OMNET, 2011), and JistSwans (JIST, 2011) provide such an environment for both wired and wireless networks. Nevertheless, the description of the functional behavior of the application remains a challenging and tedious task. In many cases this is done by using a general purpose language (e.g. C++ in ns-3 or Java in JistSwans). It is quite difficult to use these non-standardized techniques for performance evaluation because of their framework-dependency, which produces different models for the same application. Also, there is no way of using these models for generating code for a target platform (e.g. the operating system of a WSN-node).

In this context, some integrated tool environments, that combine standardized techniques with different kinds of simulators and code generators for different platforms, have been proposed.

The SPACE (Kraemer et al., 2009) method can be used for the rapid engineering of reactive systems. The method is based on reusable building blocks that express their behavior in terms of UML activities and collaborations.

The WISENES (Kuorilehto et al., 2008) framework can be used for the design, simulation, and evaluation of WSNs. The WSN protocols and applications are modeled in SDL. These models are compiled to executables used for both simulation and final implementation.

ns+SDL (Kuhn et al., 2005) enables developers to use SDL specifications as a common base for the generation of simulation and target code. The approach uses the ns-2 simulation framework for performance evaluation of SDL models.

We use the GAF4WSN framework (Ahrens et al., 2009) for the design, simulation, evaluation, and deployment of applications for WSNs. The same SDL-RT model is used for generating code for several simulators and the target platform.

3 MODELING WITH SDL-RT

SDL-RT is a pragmatic combination of the standardized languages SDL, UML, and C++ for modeling different aspects of real-time systems. These aspects include the modeling of a system's structure, its behavior, its initial configuration, an object-oriented description of data structures, and a description of actions in C++. The notation of SDL-RT is graphical on the SDL/UML-part and textual on the C++-part.

The following paragraphs provide a short description of the languages involved in SDL-RT with respect to the supported subsets of concepts.

3.1 SDL Subset

SDL-RT includes a subset of SDL-2000 with some additions and modifications, but these do not fundamentally change SDL's semantics. It adds a semaphore concept that can be used to protect access to global variables and changes the names of certain SDL constructs (e.g. a signal in SDL is renamed to a message in SDL-RT).

3.1.1 Basic Constructs

In SDL a system is modeled by a composition of agents and communication constructs. While agents model the structure and the behavior of the system, the communication constructs are used for modeling possible information exchange between them. The agents come in two flavors: block agents and process agents. A special block agent is the system agent, which is the outermost agent of the model. A brief overview of these concepts is depicted in Figure 1.

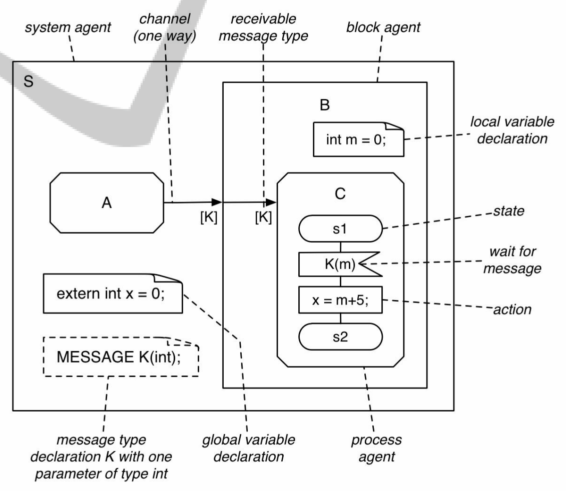

Figure 1: Excerpt of a sample system model using basic SDL concepts.

Block agents are composed of other agents and communication constructs. They are solely used for structuring the model in a hierarchical way and do not specify a behavior on their own.

Process agents model active elements of a system and are executed concurrently. Their behavior is specified by a communicating state machine, which may also include local variables and procedures. Process agents can react to a message reception or to an expiration of a timer. They act by invoking procedures, sending messages, starting timers, dynamically creating process agents or directly executing C++ code (e.g. changing variable values). The messages are consumed according to their priority first and then in FIFO order.

Communication constructs specify how information can be exchanged between agents. These constructs specify that communication takes place either by (1) exchanging messages through communication channels or (2) accessing global variables. If channel-based communication is used, channels have to be specified between agents including the types of messages to be exchanged in them. If on the other hand variable-based communication is used, agents have to protect concurrent access to global variables by semaphores.

3.1.2 Instance-oriented and Type-oriented Modeling

At the language level presented so far, each agent models a set of agent instances (1 instance by default). This style of modeling could be named instance-oriented modeling because agent definitions may only be used in the context where they are defined.

Reusing agent definitions is only possible by using type-oriented modeling. Here the modeler specifies agents as agent types and places them in packages from where they may be used in different agent contexts. The usage of an agent type represents an instantiation of the type in the context. Communication can be described in a context-independent way by using gates. They serve as communication end points where channels can be connected to at instantiation time. Agent instances may be given a name and an initial number of instances to be created. Additional instances may be created dynamically by process agents.

3.1.3 Object-oriented Modeling

Object-oriented modeling is only supported for process agents with single inheritance relations between them. Variables, procedures, and state machines can be inherited from a base process agent through specialization. In specialized agents, transitions may be added or they may overload existing ones. Also base agents may specify abstract transitions that have to be made concrete in specializations.

3.2 UML and C++

SDL-RT contains a subset of UML that is used for object-oriented modeling of data structures and system deployment. A subset of UML class diagrams can be used for modeling data structures. The subset includes classes (with attributes and operations), associations (as aggregation or composition and with uni or bidirectional relation), and class generalizations.

C++ is used for declaring data types and for specifying transition actions in state machines. Data types have to be declared for variables, procedure parameters, and message parameters. Here any valid C++ type declaration may be used. Actions are also specified with C++. This allows an efficient execution and opens the possibility to easily interface with existing C or C++ libraries.

4 APPLICATION EXAMPLE

Earthquakes produce different types of seismic waves: (1) P-waves and S-waves (called body waves); (2) Rayleigh waves and Love waves (called surface waves). P-waves (primary waves) travel faster than S-waves (secondary waves)[1]. They are less destructive than the S-waves and surface waves that follow them. Earthquake early warning is based on the detection of the harmless P-waves that precede the slower and destructive S-waves and surface waves.

Our application example consists in the development of an earthquake early warning application (Alarming Protocol) (Fischer et al., 2009). The Alarming Protocol (AP) runs on top of SOSEWIN[2] (Fleming et al., 2009), which is a decentralized WSN-based earthquake early warning system. The AP is a hierarchical alarming system, where the network is composed of node clusters. The organization of the network into clusters and the designation of corresponding cluster heads is done at installation time, but can change dynamically following the changes in the topology.

The functional aspects of the Alarming Protocol are described in SDL-RT (Figure 2).

4.1 The Alarming Protocol

As Figure 2 shows, the Alarming Protocol (AP) is modeled as a set of asynchronous communicating protocol entities:

- **Signal Analyzing Entity (SAE).**
  The SAE analyzes the incoming streams of data and informs the SE in case of a detected event.
- **Sensing Entity (SE).**
  The SE reacts on the results received from the SAE by informing its associated Leading Entity

[1]P-waves travel at 5-8 km/s, and S-waves at 3-7 km/s.

[2]Self-Organizing Seismic Early Warning Information Network.

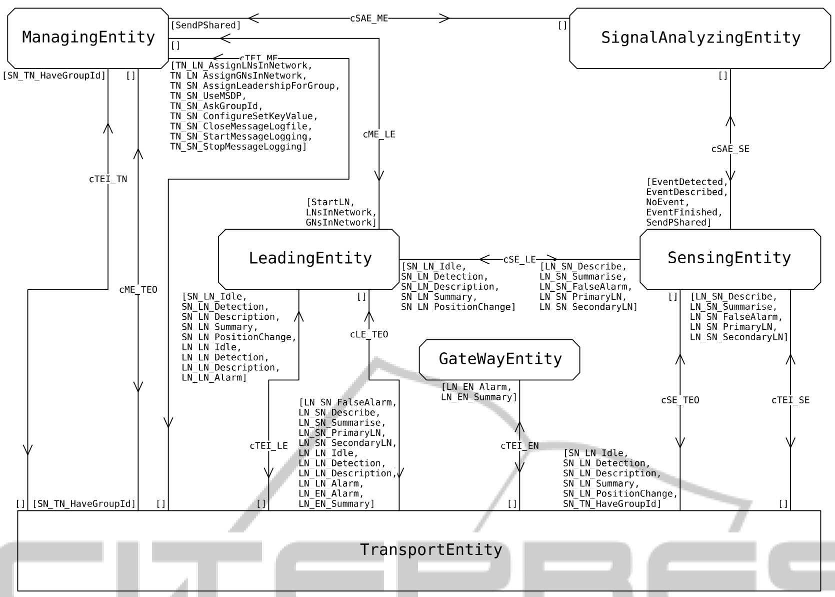
ManagingEntity
cSAE_ME
[SendPShared]
[]
SignalAnalyzingEntity
cTEI_ME
[TN_LN_AssignLNsInNetwork, TN_LN_AssignGNsInNetwork, TN_SN_AssignLeadershipForGroup, TN_SN_UseMSDP, TN_SN_AskGroupId, TN_SN_ConfigureSetKeyValue, TN_SN_CloseMessageLogfile, TN_SN_StartMessageLogging, TN_SN_StopMessageLogging]
[SN_TN_HaveGroupId]
cTEI_TN
cME_TEO
cME_LE
cSAE_SE
[StartLN, LNsInNetwork, GNsInNetwork]
[EventDetected, EventDescribed, NoEvent, EventFinished, SendPShared]
LeadingEntity
cSE_LE
[SN_LN_Idle, SN_LN_Detection, SN_LN_Description, SN_LN_Summary, SN_LN_PositionChange]
[LN_SN_Describe, LN_SN_Summarise, LN_SN_FalseAlarm, LN_SN_PrimaryLN, LN_SN_SecondaryLN]
SensingEntity
[SN_LN_Idle, SN_LN_Detection, SN_LN_Description, SN_LN_Summary, SN_LN_PositionChange, LN_LN_Idle, LN_LN_Detection, LN_LN_Description, LN_LN_Alarm]
cLE_TEO
GateWayEntity
[LN_EN_Alarm, LN_EN_Summary]
cSE_TEO
cTEI_SE
[LN_SN_FalseAlarm, LN_SN_Describe, LN_SN_Summarise, LN_SN_PrimaryLN, LN_SN_SecondaryLN, LN_LN_Idle, LN_LN_Detection, LN_LN_Description, LN_LN_Alarm, LN_EN_Alarm, LN_EN_Summary]
cTEI_LE
cTEI_EN
[SN_LN_Idle, SN_LN_Detection, SN_LN_Description, SN_LN_Summary, SN_LN_PositionChange, SN_TN_HaveGroupId]
TransportEntity

Figure 2: SDL-RT description of the Alarming Protocol.

(LE). The LE is either located on the same node as the SE if the node is a cluster head or on another node if it is not.

- **Leading Entity (LE).**
  The LE monitors all associated SEs. It is able to issue group alerts and system alerts. This entity is active when the node is a cluster head.
- **Gateway Entity (GE).**
  The GE is responsible for forwarding system alerts to end-users outside the network.
- **Transport Entity (TE).**
  The TE provides a bridge between AP entities and the underlying communication layer. In Figure 2 this is represented by a SDL-RT block, which is composed of two processes (not shown in the figure) for sending and receiving messages over the network.
- **Managing Entity (ME).**
  The ME is responsible for interpreting and processing network management messages (i.e. establishment of cluster structures).

The AP's functionality is defined in two layers, an intra and a inter-cluster protocol. The former handles (1) the communication between the SAE and SE and (2) the communication between LEs and their associated SEs. The inter-cluster protocol handles communication between all LEs. When a critical number of P-wave triggers have reached the LE, it informs its neighboring LEs. In the case that a LE has received enough cluster alarms, a system alarm will be sent as fast as possible to the GEs and other LEs, which will distributed it to all their cluster members (SEs). According to this hierarchical principle, three alarm levels are recognized by the nodes:

- *Pre-alarm* recognized by the LE, when a P-wave is detected by at least one SE in its cluster;
- *Group Alarm* recognized by the LE, when a certain number of SEs in its cluster have detected a P-wave;
- *System Alarm* recognized by the LE, when a certain number of LEs have triggered a group alarm.

4.2 Target Platforms

The SDL-RT model of the AP is used for generating code for different target platforms. These include several simulator frameworks, which are used for performance evaluation of the application, and also the operating system installed on the WSN-nodes.

The code generation is based on an extension of PragmaDev's RTDS[3] (PRAGMADEV, 2006) with a new transcompiler (Ahrens et al., 2009; Brumbulli and Fischer, 2010), which is able to generate C++ code artifacts from SDL-RT models. After compilation the artifacts are linked to different libraries (Figure 3), producing the corresponding binaries: RTDS built-in simulator, ODEMx (ODEMX, 2011), ns-3, and OpenWrt (OPENWRT, 2011).

The **RTDS built-in simulator** can just be used for evaluating the functional behavior of SDL-RT specifi-

[3]A SDL-RT tool including an editor, code generator, and debugger.

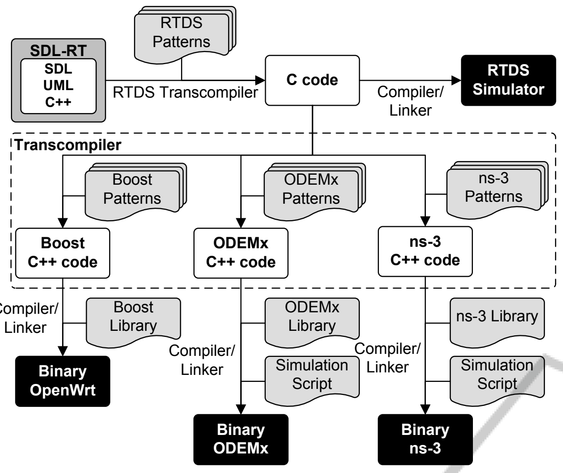

Figure 3: The transcompiler's architecture.

cations. It cannot be used for performance evaluation. For performance evaluation of large network topologies we use two simulation frameworks: ODEMx and ns-3.

We use **ODEMx** for simulations without a precise model of the underlying communication layers. ODEMx is a general purpose simulation library written in C++. It supports process- and event-oriented modeling of discrete and continuous event systems and also a combination of both.

We use **ns-3** for simulations with a detailed model of the underlying communication layers. It is a discrete-event network simulator for internet systems written entirely written in C++; simulations are also C++ executables.

Our target platform for the nodes is **OpenWrt**. It is a Linux distribution for embedded devices that provides a fully writable file system with package management. This allows us to customize the device through the use of packages to suit any application.

5 EVALUATION

We evaluate the standardized techniques using UML and SDL-RT for developing applications for WSNs based on three criteria:

1. modeling of their functional aspects,
2. generating executable code, and
3. evaluating performance based on their real-time behavior.

First we present some experiment results for the Alarming Protocol to show that our pragmatic approach of using SDL-RT in combination with code generation and performance evaluation provides a solution for the defined criteria.

Another approach could be to include some new concepts in the modeling language itself. We think that this could also provide a solution for the defined criteria. These proposed concepts are listed in the second part of this evaluation.

5.1 Experiment Results

We define two types of experiments: real-world and simulation based.

Real-world experiments consist in running the Alarming Protocol with synthesized earthquake data on the Humboldt Wireless Lab (HWL, 2011) testbed. As a result of these experiments, we provide the times available for early warning in Figure 4. These times are given as difference between S-wave detection and System Alarm for different earthquake distances. The figure shows that it is possible to use our AP for early warning.

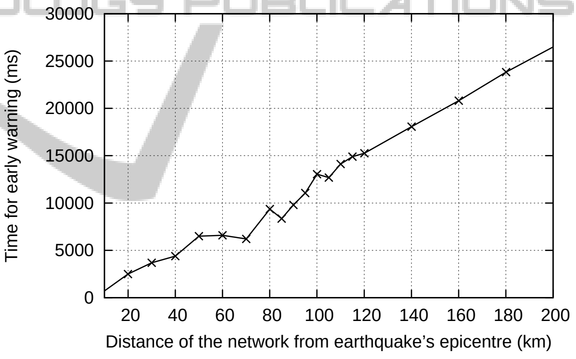

Figure 4: Time available for early warning in real-world experiments.

Simulation based experiments use the ns-3 library for evaluating potential real-time behavior of the Alarming Protocol. For a comparison to be possible, we use the same configuration (network model + earthquake data) as in our real-world experiments. A comparison between real-world and simulation based experiments is shown in Figure 5. Because of the small differences (less than 150 ms), we can assume that simulation experiments with larger topologies can be used to predict the behavior of the AP.

5.2 Modeling Methodology

SDL-RT was created for developing real-time and embedded systems. We showed in our example that its pragmatic nature (the combination of UML, SDL, and C++) can be used successfully for the development of applications for WSNs. Nevertheless, we

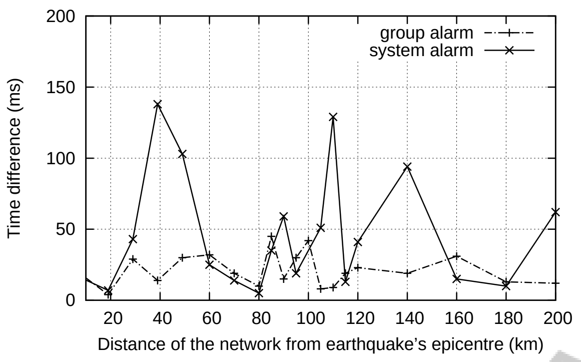

Figure 5: Difference in alarm times between real-world and ns-3 simulation.

identified a number of concepts that we believe can provide better means for describing functional aspects and evaluating potential real-time behavior of these applications.

5.2.1 Time Consumption

Modeling the time consumption of actions is an important aspect when a system's timed behavior needs to be evaluated. In SDL-RT, timers can be used for modeling time-dependent action execution. But when using SDL-RT timers, functional and timing aspects are not separated in the model. From such a specification, code for either target platform execution or simulation cannot be generated automatically.

5.2.2 Inter-node Communication

SDL-RT supports communication between agents in a local one-to-one fashion: one sender to one receiver in a local context (i.e. sender and receiver are running on the same node). Although this type of communication is suitable for embedded systems, the same cannot be stated for distributed ones (e.g. WSNs). In these systems inter-communication between nodes plays an important role, but there are no language constructs in SDL-RT that allow an agent to address or send a message to a remote one (agent running on another node).

5.2.3 Broadcast Communication

When modeling protocols for wireless networks, it is a common case that information should be send remotely to multiple identifiable receivers or even be broad-casted to every receiver in range. But SDL-RT only supports the sending of information to exactly one identifiable receiver, that must also be reachable by a complete communication path. We believe that modeling wireless network protocols could be done in a more concise way if multi-send or broadcast-send would be supported by the description language.

5.2.4 Addressing Gates in Message Receives

When entities are able to receive the same type of message via multiple gates, then it may be necessary to identify the gate if different gate-dependent actions are to be taken. This can be done by addressing the gate where a message is received. Such a feature is currently not available in SDL-RT.

5.2.5 Dynamic Block Instantiation

Evaluating timed behavior of an application in a distributed environment may require the creation of several application instances depending on the size of the network. Although this can be achieved in SDL-RT by static block instantiation, it becomes almost impossible for large networks. In this context, we believe that creating block instances and channels dynamically can be helpful.

In SDL-2000 we already have the concept of dynamic block instantiation. Nevertheless, this can only be done in an existing agent context and there are no means of dynamically creating channels.

6 CONCLUSIONS

The development of applications for WSNs is a challenging task. This task is handled in different ways, varying from general purpose languages to domain specific ones. Although these techniques do provide a solution, they can only be used within their specific frameworks. This makes it almost impossible to evaluate the potential behavior of applications because different framework-dependent models are used.

In this paper we showed that standardized techniques can be used to describe the functional aspects of applications for WSNs. Also it was possible to derive from the same model executables for different platforms so that potential real-time behavior could be evaluated.

Based on a real application example and our experience, we identified a number of concepts that can provide better means for describing such applications. We propose to extend the existing modeling language in order to include these concepts. If such a solution is really feasible, has to be checked in future work.

REFERENCES

Ahrens, K., Eveslage, I., Fischer, J., Kühnlenz, F., and Weber, D. (2009). The Challenges of Using SDL for the Development of Wireless Sensor Networks. In Reed, R., Bilgic, A., and Gotzhein, R., editors, *SDL 2009: Design for Motes and Mobiles*, volume 5719 of *Lecture Notes in Computer Science*, pages 200–221. Springer Berlin / Heidelberg.

Brumbulli, M. and Fischer, J. (2010). SDL Code Generation for Network Simulators. In *System Analysis and Modeling - SAM 2010*.

Fischer, J., Kühnlenz, F., Ahrens, K., and Eveslage, I. (2009). Model-based Development of Self-organizing Earthquake Early Warning Systems. In *Proceedings MATHMOD 09 Vienna*.

Fleming, K., Picozzi, M., Milkereit, C., Kuhnlenz, F., Lichtblau, B., Fischer, J., Zulfikar, C., Ozel, O., et al. (2009). The Self-organizing Seismic Early Warning Information Network (SOSEWIN). *Seismological Research Letters*, 80(5):755.

HWL (2011). http://hwl.hu-berlin.de/.

ITU (2007a). ITU-T Recommendation Z.100: Specification and Description Language (SDL).

ITU (2007b). ITU-T Recommendation Z.109: SDL-2000 combined with UML.

JIST (2011). http://jist.ece.cornell.edu/.

Kraemer, F., Slåtten, V., and Herrmann, P. (2009). Model-Driven Construction of Embedded Applications Based on Reusable Building Blocks - An Example. In Reed, R., Bilgic, A., and Gotzhein, R., editors, *SDL 2009: Design for Motes and Mobiles*, volume 5719 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg.

Kuhn, T., Geraldy, A., Gotzhein, R., and Rothländer, F. (2005). ns+SDL - The Network Simulator for SDL Systems. In Prinz, A., Reed, R., and Reed, J., editors, *SDL 2005: Model Driven Systems Design*, volume 3530 of *Lecture Notes in Computer Science*, pages 103–116. Springer Berlin / Heidelberg.

Kuorilehto, M., Hännikäinen, M., and Hämäläinen, T. D. (2008). Rapid Design and Evaluation Framework for Wireless Sensor Networks. *Ad Hoc Netw.*, 6:909–935.

NS2 (2011). http://www.isi.edu/nsnam/ns/.

NS3 (2011). http://www.nsnam.org/.

ODEMX (2011). http://odemx.sourceforge.net/.

OMG (2010). OMG Unified Modeling Language, Infrastructure.

OMNET (2011). http://www.omnetpp.org/.

OPENWRT (2011). http://www.openwrt.org/.

PRAGMADEV (2006). http://www.pragmadev.com.

SDLRT (2011). http://www.sdl-rt.org/.