back to start

The challenges of using SDL for the development of wireless sensor networks

Klaus Ahrens, Ingmar Eveslage, Joachim Fischer,
Frank Kühnlenz, and Dorian Weber

Humboldt-Universität zu Berlin, Department of Computer Science,
Unter den Linden 6, 10099 Berlin, Germany
{ahrens,eveslage,fischer,kuehnlenz,weber}@informatik.hu-berlin.de

Abstract. In recent years, Wireless Sensor Networks (WSNs) have been primarily used to build ad-hoc telecommunication infrastructures from scratch or as low-cost alternatives to traditional networks. But the diversity of applications with typically narrow node resources and requirements of already existing information infrastructures sets hard constraints to WSN. The software development process becomes even more complicated when real-time constraints have to be taken into account. This is the case when the physical processes of the WSN environment have to be observed and are realized in space and time. For the development of such WSN we present a model-based framework (GAF4WSN), where the well-known techniques SDL, UML and ASN.1 are involved. The framework was already successfully used for the development of a new generation of Earthquake Early Warning Systems (EEWS). An Earthquake Synthesizer (ES) and an Experiment Management System (EMS) complete the framework, which supports the modelling, simulation, installation and administration of different EEWS approaches in combination with a Geographic Information System (GIS).

Key words: model-based development, sensor systems, wireless sensor networks, simulation, code generation, experiment management, SDL, UML, ASN.1

1 Introduction

Wireless Sensor Networks (WSN) become more and more popular for monitoring numerous physical phenomena and often they should be self-organized for easy installation and maintenance (e.g. to autonomously integrate new sensor nodes and react on failed ones). This emerging technology offers exciting potential for numerous application areas including environmental, medical, military, transportation and disaster management. The major challenges in the WSN domain include sensing and collecting data from sensors and then evaluating it to formulate meaningful information such as generating alarms or supporting decisions while minimizing energy consumption. But the development of such complex systems is a challenging task. In particular, the evaluation of their potential real-time behaviour is almost impossible or too expensive without prior modelling experiments, involving computer simulations. Besides the complexity of the system itself, its surrounding environment in real-time behaviour terms must be considered and modelled (e.g. load-models using synthetic sensor input data). In addition to this kind of requirement, it is quite difficult to port a WSN application to different platforms.

To solve these issues, system engineers need to be able to model applications using high-level abstractions and to simulate those using configurable and realistic topologies for the network itself. We follow a generally approved model-driven development paradigm using a technology mix of SDL/ASN.1/UML/C++ [1, 2] to generate the code for the target hardware platform (sensor nodes) and for different kinds of simulators supporting different experiment scenarios (including the system and its environment) in preparation for the implementation. A further specialty of our approach is the integration of the model-driven tool chain into a spatial-time-based Experiment Management System (EMS) in connection with a Geographic Information System (GIS). This allows us to describe the WSN topology and the distribution or movement of the physical phenomena in a geographic map. All tool components are integrated by our GIS-based Development and Administration Framework for Wireless Sensor Networks (GAF4WSN).

The presented paper illustrates the usage of GAF4WSN developing a new approach for Earthquake Early Warning Systems (EEWS) that uses self-organizing mesh WSN based on commercial off-the-shelf hardware where the WSN nodes are equipped with low-cost seismometers and GPS units. The most important installed software components are the Linux operating system, WLAN network stack and further protocol units for message routing and alarming through a meshed network. Additional general services are provided by middleware between the communication and application layer of the protocol stack. Significant and innovative aspects of that approach are connected with the fact that each sensing node performs on-site independent analysis of the ground motion and that the early warning is automatically carried out within the wireless mesh network itself by dedicated alarming processes. Moreover, since commercial off-the-shelf hardware is inexpensive, this also allows for more sensor nodes and hence much denser sensor networks. These can provide more detailed, higher resolution information than traditional seismic networks with only a few powerful seismological stations spread over a large area.

Structure and behaviour models of network topologies developed with GAF4WSN for specific geographic regions are coupled with seismometer sensor input data generation and convenient visualizations to form the basis for various types of simulation experiments ahead of system implementation and installation. The general objective of these studies is to test the functionality of an EEWS and to optimize it with respect to real-time, reliability and cost requirements of potential end-users. This approach is used for implementing a prototype-EEWS developed within the EU project SAFER (Seismic eArly warning For EuRope, [3]) in cooperation with the GeoForschungsZentrum Potsdam. A small installation of this prototype system is established in the mega-city Istanbul, a region threatened by strong earthquakes.

This paper focuses on a transcompiler (integrated in GAF4WSN) as an extension of PragmaDevs RealTimeDeveloperStudio [4] to generate specific code for different types of simulation experiments and for the sensor nodes' target platform out of SDL/UML models.

2 Related Work

Legacy computer network simulators, such as ns-2 [5], ns-3 [6], JistSwans [7] and OPNET [8] enable the simulation of wired or wireless network behaviour and protocol stack operation, but do not take into account WSN characteristics. This is overcome in the simulators proposed specifically for WSN, which were categorized by [9] into networking oriented and sensor node simulators. In addition [9] introduces a further simulator type, which is characterized by an integration of design, simulation, debugging and code generation tools under a unique GUI.

The **network-oriented simulators** model the transmission medium in detail and are more suitable for the large scale WSN simulations. Most of the proposed networking oriented simulators are based on legacy computer network simulators. SensorSim [10] and Naval Research Laboratory's sensor network simulator [11] extend ns-2 with general WSN features.

The **sensor node simulators** mainly simulate the operation of a single node but implement a lightweight communication model. Most of the proposed sensor node simulators are targeted to TinyOS motes. Complete TinyOS systems can be simulated with TinyOS SIMulator (TOSSIM) [12], and TinyOS Scalable Simulation Framework that is an extension to SWAN [13].

Currently some **integrated tool environments** are available that combine graphical design techniques with different kinds of simulators and code generators for different platforms. It is interesting that the most used modelling technique in that area is the ITU-T language SDL. Examples of that category are the WISENES framework [9] or the complete framework for modelling, simulation and multiplatform code generation based on MathWorks tools. They enable system engineers to effectively design WSN applications and map them on a wireless network. Application developers can automatically generate the complete application code for several target operating systems from the same simulated and debugged model, without thinking about the details of the target platform implementation. The SDL Environment Framework (SEnF), developed by the group of R. Gotzhein [14], is a further prominent example of that category.

Our integrated tool environment **GAF4WSN** supports time-dependent and timeless simulations in combination with virtual or real machine simulations based on a sensor network model in SDL/UML. Furthermore it is possible to use a GIS-based approach for modeling different net topologies in combination with models of environmental processes. For efficient transmission of messages, ASN.1 coding/decoding mechanisms are offered. GAF4WSN is not only a framework for the design, simulation and code generation; furthermore it is also prototyped as an operating and management platform for WSN. For that the WSN have to be equipped with a middleware in combination with an EMS, which has to be installed outside the WSN.

3 SOSEWIN: A Wireless Meshed Seismic Sensor Network

The Self-Organizing Seismic Early Warning Information Network (SOSEWIN) is technically a decentralised, wireless mesh sensor network, made up of low-cost components, with a special seismological application that supports earthquake early warning and rapid response tasks. It is being developed within the SAFER [15] project funded by the European Union.

3.1 Earthquake Early Warning

Earthquakes belong to the most devastating natural hazards. They not only cause damage to economic infrastructures, but also cause the loss of human life. Several mega-cities like San Francisco, Mexico City, Tokyo or Istanbul are at risk. Such cities not only accommodate a large number of people, but they also constitute the economic heart of their regions. An EEWS is feasible because earthquakes generate two basic kinds of seismic waves: P-waves (primary waves) and S-waves (secondary waves). The harmless P-waves are almost twice as fast as the S-waves, which cause most of the destructive shaking. Therefore, the warning time (time interval between the detection of the fast P-waves and the arrival of the slower S-waves), depends on the distance of a target area, usually a city to be protected, from the hypocenter (the origin location of the earthquake). Although for the worst case scenario this short warning time is not enough for people to leave their houses, this can still be sufficient to mitigate secondary damages like gas explosions and fire outbreaks.

3.2 Rapid Response

Immediately after an earthquake event, several analyzing tasks of the recorded seismic waves must be carried out in order to understand the seismic impacts of the event. These analyzing tasks belong to the rapid response phase. A typical task in this phase is the fast generation of so-called ShakeMaps [16], which show the wave peaks (or intensity) in the seismic affected area in form of isobar lines or different colours. The combination of such ShakeMaps with information about building structures and population densities in the affected area is important for fast and proper disaster management. Thus the recording of aftershocks directly in the affected area is a typical task or earthquake task forces.

3.3 Characteristics and Purpose

In contrast to existing EEWS [17], SOSEWIN realizes the new approach of performing the decision whether an earthquake early warning should be raised, inside the network itself and not in a central data centre (two-level alarming protocol, Sect. 6.1).

A SOSEWIN reacts in a self-organizing way on newly added or removed nodes. Therefore it is more robust, easier to install and maintain than traditional, centralized EEWSs. In addition, with a relatively low price compared to common seismometer stations, a very dense network (hundreds or thousands of nodes) can be established directly in the threatened region. SOSEWIN nodes are also small and light-weight, which allows a working network to be established very quickly to support earthquake task forces during the rapid response phase.

The SOSEWIN nodes were already successfully used in two field tests for monitoring building structures: Recently during the aftershock phase of the Abruzzo event (Italy) and the Fatih bridge (Istanbul, Turkey) [18].

4 Framework Requirements for the Design and Administration of WSN

In the following section we postulate a list of requirements which we were able to identify by generalization of the SOSEWIN development in our case study and which were introduced step by step in our framework GAF4WSN as special features.

- To test SOSEWIN in the form of models or real installed network topologies, a framework feature is necessary as it allows us to simulate the **environmental processes** (characterized by time series for each node, identified by its precise local geographic position). In the case of earthquakes this data has to be generated by a synthesizer tool for which input data is defined by an event description (epicentre, depth, magnitude), a geological description of the ground area of the travelling waves (defined by fault characteristics), the WSN topology (given by geographic positions and link quality parameters) and the node equipment (processor, memory, GPS-unit, seismometer sensor, software components). Independent of the concrete use case, the framework should support the built-up of model repositories. Furthermore the WSN middleware should support a deployment of the test data and a principle switch between two kinds of node input: real sensor data or synthesized data.
- For the configuration of WSN models or real prototypes (network topology, software architecture of nodes, geographic area) influenced by environmental processes, a **graphical Topology Editor** based on a **Geographic Information System (GIS)** is necessary. Adding and removing nodes is one of the necessary basic features. The OGC standard Web Feature Service (WFS) [19] can be recommended to introduce an additional layer of spatial objects (e.g. points and lines with additional attributes) in an existing topographic map.
- The example SOSEWIN has shown that for the design of WSN as alarming systems, **time-based simulations** are absolutely essential to estimate the

relation between the speeds of the environmental process (travelling of seismic waves) and the information processing process by the WSN. This kind of performance prediction experiments allows for the evaluation of the chosen network topology, routing protocols, message coding methods and hierarchical alarming protocols before the WSN is installed as an early warning system.

- In view of the complexity of large-scale WSN, the spread of WSN simulators should allow a **step-wise transition of abstract to real components** during the test process of WSN components. Here it was very useful to offer the choice between different time concepts (free model time, model time related to real time and real time itself). Furthermore tests should be supported by execution of complete WSN protocol stacks on virtual machines, even if only small net topologies can be investigated by that approach. Larger topologies need a higher degree of model abstraction, so that GAF4WSN should support the use of ordinary network simulators.
- The WSN application protocols should be designed by **standardized modelling techniques**. This has an impact for maintaining and porting the WSN concepts. As already mentioned in Sect. 2, SDL was also the favourite modelling language for our framework. We decided to use the Real-time Developer Studio from PragmaDev, which supports the design of real-time systems combined with debugging and C code generation facilities for different real-time operating systems. In particular this tool allows for a powerful combination of the SDL agent concept (except some restrictions) with UML class, use case and sequence diagrams. The data and sequential action parts of PragmaDevs SDL/UML harmonization are based on C/C++. As a consequence pointers can be efficiently handled as message parameters independent of whether the message is used for internal node communication (with the parameter as a reference value) or if it is used for node-to-node communication (with the parameter as a dereferenced value).
  This piece of knowledge, whether the parameter is a reference or a dereferenced value, is always given by the respective sender agent. In addition it is possible to bypass the actual weakness of SDL-RT in message delivery between processes of block instances, where in SDL-RT all instances of a block share process instance sets: one shared instance set for each enclosed process definition (compared with SDL-2000 where each process instance set in a block is enclosed by the block and distinct from sets in other block instances).
- Commonly WSN have to deliver a great quantity of raw sensor data. Standardized techniques for coding and decoding improve the efficiency of the whole network. We use **ASN.1 basic encoding/decoding** rules with Mini-SEED [20], a standardized coding technique for seismometer raw data.
- Experiments as generalization of systematic simulator runs or real tests of WSN prototypes form the evaluation base of our use case example SOSEWIN. To ensure the consistency of all artefacts which are necessary for an experiment (concerning WSN model, WSN environment, tools, test input and output results), a tool-based information management is absolutely conve-

nient. Such an experiment management system should be integrated in the design and administration framework. Independent of the nature of the investigation object (model or real prototype), all events like state changes, timer events or message arrivals have to be logged into files. This guarantees a unique operational interface for the **experiment management system**. The collected data form the base for a further use-case-dependent information evaluation.

- Equally important are issues related to network deployment, interworking and the integration with existing network infrastructures. Of particular interest here is the choice of a suitable **communication middleware** that hides the complexity of the low-level networking protocols and facilitates the development of applications. In this context the data-centric communication approach, which is based on data content rather than receivers' addresses, appears to be more efficient and more appropriate for wireless sensor networks. This middleware can act as "glue" between software components of applications and the network or it is the "slash" in the Client/Server construct. To act in this way, middleware should offer services specified by well-defined interfaces.

5 Our Approach: GAF4WSN

5.1 Architecture Overview

Figure 1 shows an overview on the core components of GAF4WSN. For supporting the development task, this framework offers a centralized management of models, software artefacts and experiment results. This is enabled by several repositories that are implemented using database technologies with spatial extension (OpenGIS). This GIS-based technology also offers the possibility to administrate a WSN in terms of monitoring its health status or for deploying new or updated software components. The following gives a brief description of the framework components shown in Fig. 1.

- The **Experiment Management System** supports planning, configuration, automated execution of experiments and storage of experiment results. It provides additionally GIS-based visualization capabilities for experiment results and it can also be used for planning software deployment and monitoring of an installed WSN.
- The **Model Repository** stores used SDL(-RT), UML and C++ models defining the application for the WSN and underlying layers like middleware [21] or routing.
- The **Model Configurator** knows the target platform and uses platform dependent artefacts to configure the compiler (e.g. cross-compilation). It also specifies certain input parameters (e.g. threshold values or network clustering) and stores the whole configuration into the Experiment Repository.
- The **Transcompiler** is indeed a tool chain of several transcompilers, which accept SDL-RT models and compile C++ code at the end into different executable binaries (simulators, target code). See Sect. 5.3 for details.

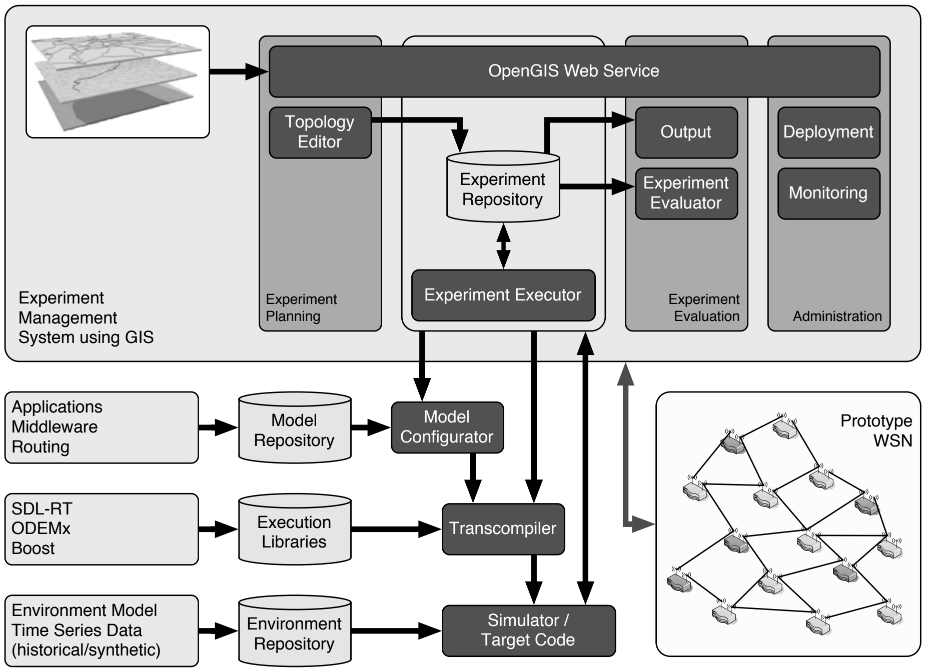


Fig. 1. Overview of GAF4WSN architecture

- The **Execution Libraries** are used by the transcompiler to generate the executable binaries for a simulator or target code. They comprise several libraries for the target code binary (e.g. threading and networking capabilities with the necessary functions to decode and encode the network messages using ASN.1). For simulator support it is also a pool of simulation frameworks.
- The **Environment Repository** comprises environment models to describe the setting of the WSN (e.g. network topology, link qualities) for simulation and also load models to provide synthetically or historically recorded sensor data to be applied as input to a sensor node in order to execute certain experiments (simulative or with an existing WSN).
- The **Simulator / Target Code** is the target binary representation created by the transcompiler tool chain. It is either a simulator out of a set of several simulator types realizing different functionality (derived from different simulation frameworks) or a binary intended to run on a node of a WSN.

Further on in this paper we focus on certain components of the GAF4WSN: mainly the transcompiler tool chain (Sect. 5.3) addressing the special needs to map SDL modelling concepts to simulator and target code. But in the following we first introduce the basic libraries for these target platforms.

5.2 Used Libraries / Target Platform

ODEMx [22] is a general purpose simulation C++ library that follows the well-known Simula-67 tradition of process-oriented model descriptions with model time based interleaved execution. ODEMx uses a highly portable coroutine implementation and has been developed by the authors in several versions along with the emergence of C++ features. ODEMx provides base classes for processes and events, so that it supports closed process life cycles as well as simple event actions. Processes are scheduled according to their model time consumption and executed as coroutines. Events are executed by simple function calls on top of some process' execution stack. The library contains special classes for synchronization utilities as limited resources and unlimited queues. It provides a broad range of random distribution classes and automatic reports on all resource utilizations. ODEMx also supports combined models with both discrete and continuous time processes. Since its version 2.0 ODEMx provides a special package for modelling protocol components. Using ODEMx allows for highly adaptable experiments with the signalling protocols specified in SDL with different variation points:

- various levels of details in the model representation of the net infrastructure ranging from ideal transmission without signal loss to tuning of transmission quality parameters including group separation and explicitly modelling of lower protocol layers,
- inclusion of real world node setup scenarios as well as all kinds of experiments on hypothetic node allocations,
- feeding synthesized (but real world equivalent shaped) or recorded real sensor data into the sensor nodes.

Boost [23] is a collection of C++ libraries usable across a broad spectrum of applications. It aims at establishing "existing practice" and provides reference implementations so that Boost libraries are suitable for eventual standardization. For the target platforms, the following particular components are used for the target code generation:

portable networking — including sockets;
timers;
hostname resolution;
socket iostreams;,
portable C++ multi-threading.

OpenWrt [24] is described as a Linux distribution for embedded devices. Instead of trying to create a single, static firmware, OpenWrt provides a fully writable file system with package management. This frees us from the application selection and configuration provided by the vendor and allows us to customize the device through the use of packages to suit any application. For the developer, OpenWrt is the framework to build an application without having to build the complete firmware around it.

5.3 SDL HUB Transcompiler

General Description: The core tool of our framework GAF4WSN is the extension of PragmaDevs RTDS to a new transcompiler which is able to generate C++ code artefacts from UML/SDL-RT. These different artefacts can be linked after their compilation with different libraries, which allows the construction of different binaries corresponding to Fig. 2:

A) RTDS-Simulator of small WSN topologies (as direct SDL interpretation)
B) ODEMx-Simulator of large-scale topology applications without a precise model of the underlying communication layer
C) OpenWrt-target code fragments representing the application functionality which has to be installed together with other software components on a node of a special type, whose functionality is described in SDL by a corresponding SDL block type and executable under OpenWrt.

In addition an SDL transcompiler to ns-3 [6] is in development by us.

PragmaDev approach: As illustrated in Fig. 2.A the PragmaDev approach is given by translation of correct SDL-RT models into valid C code in general. Some parts of the output are generated; others are static in order to support a wide range of operating systems. The static parts are interleaved by the RTDS transcompiler with generated code artefacts, which are parameterized by using macro functions. Each framework provided by PragmaDev or any third party developer both needs to define the macros used by the code generator as well as the static parts. These components are represented by "RTDS code templates".

During the transformation into C code, RTDS discards any information connected to the composition of processes into blocks. This implies a flat hierarchy among blocks and processes. Processes are always mapped to threads of the corresponding OS, procedures map to functions and messages are represented as C structures.

Due to the lack of a representation of blocks in the generated source code, RTDS is not able to instantiate blocks dynamically. This motivates the use of an additional compiler that gets called when the PragmaDev one finishes its work.

RTDS extension points: Figure 2.A also illustrates the principle way of compiler extension. First of all definitions of generated macro calls can be changed using customized RTDS code templates. In addition, the static files that are interleaved with actual generated code can be used as "markers" for our tool during examination of the generated files. The advantage of this approach is that the customised tool doesn't need to really understand the structure of the generated code, but instead can use the markings for orientation. Finally RTDS allows to partially redefine generated makefiles by the use of (RTDS generated) include directives. This mechanism can be used to integrate additional tools seamlessly into the chain without artificially changing how the toolchain works.

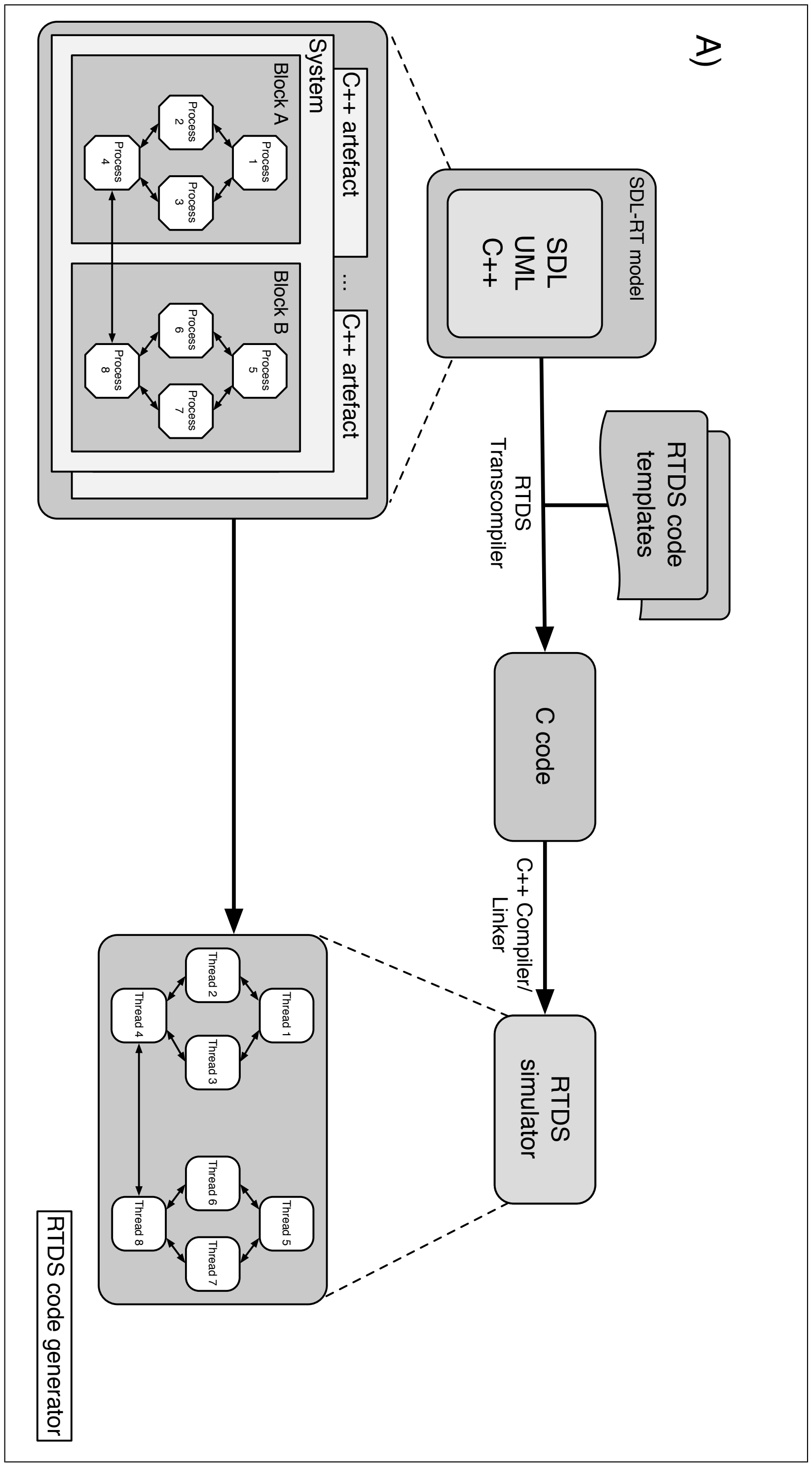


Fig. 2. A. Different code generation target types — RTDS-Simulator

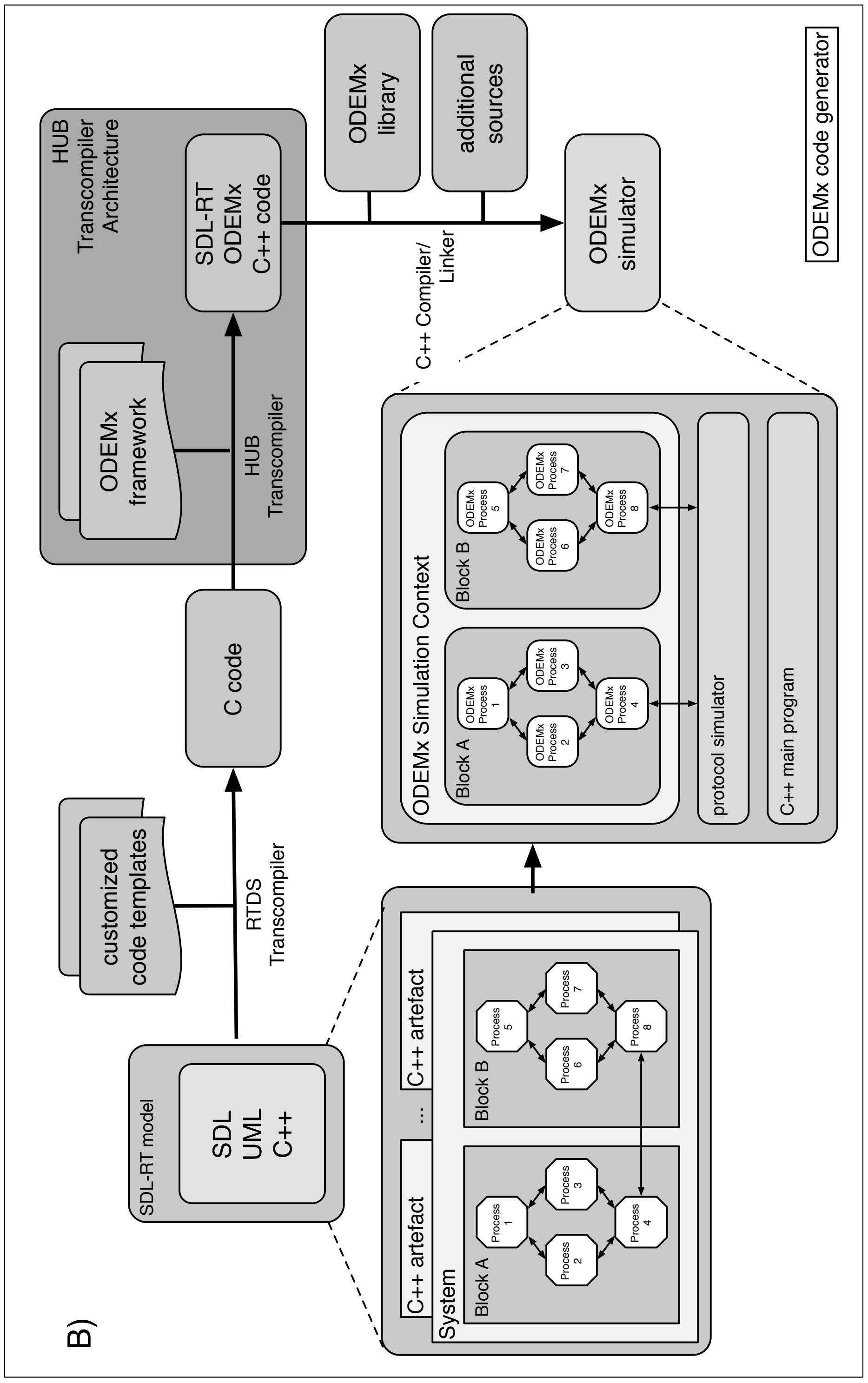


Fig.2. B. Different code generation target types — ODEMx simulator generation

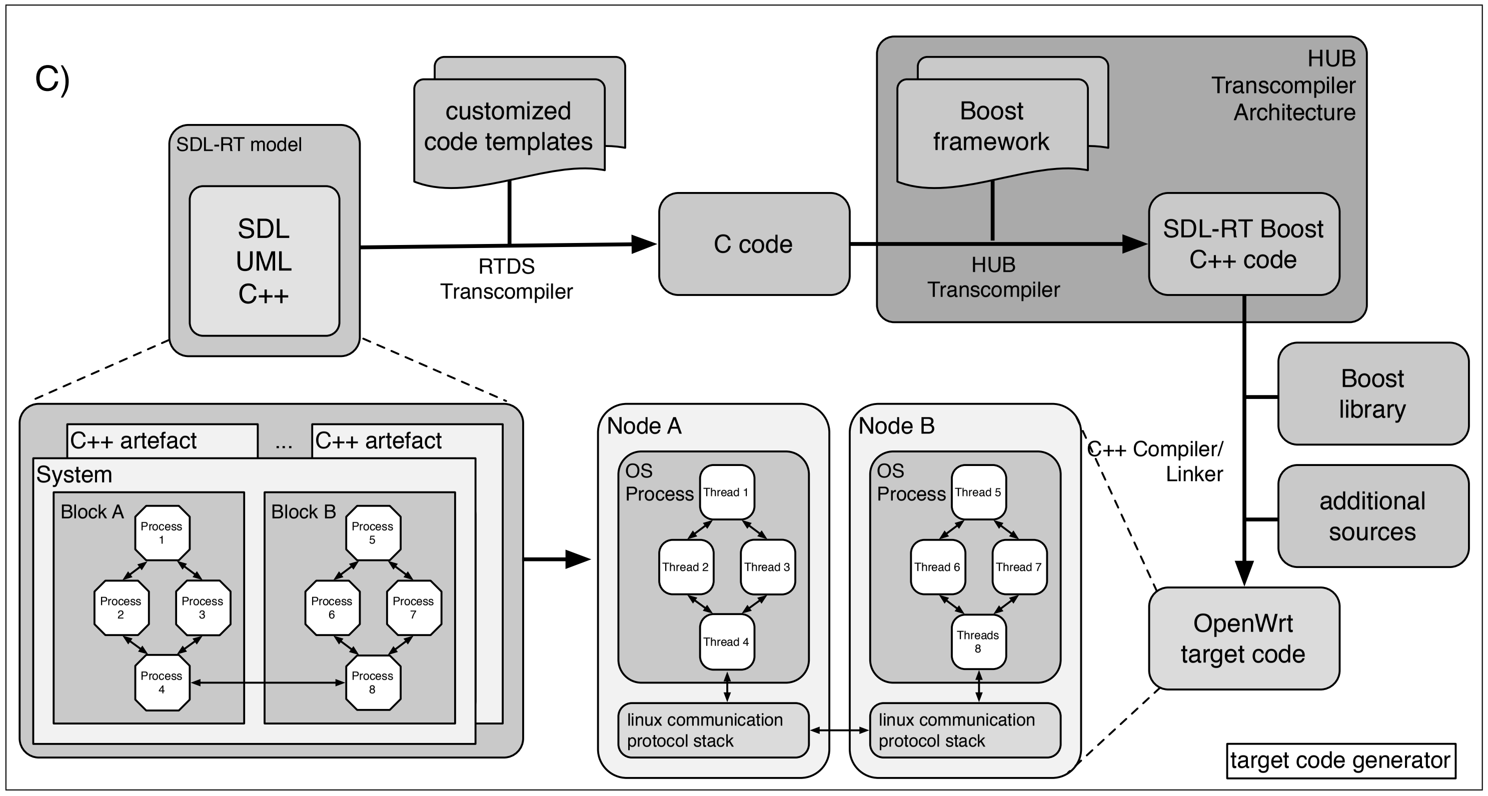


Fig.2. C. Different code generation target types — OpenWrt-target generation

HUB SDL approach: Due to the structure of the generated code, our additional transcompiler is still able to reconstruct missing information. Blocks are mapped to C++ classes that are populated with other blocks or processes, and processes become C++ classes that have procedures as their methods. In order to allow the sending of messages among different processes, all processes derive from a common base class that implements an interface for resolving process type IDs. Since we require dynamic instantiation of blocks in our large-scale topology projects, a block needs to provide mechanisms for internal block communication. This implies a common base class. Unfortunately we were unable to distinguish SDL-RT blocks from user defined include files just by analyzing C code without loss of generality. Therefore we needed to introduce naming conventions for blocks and block classes if those should be instantiated dynamically. This doesn't break compatibility with the original SDL-RT, because compilation will still work even if the naming conventions are ignored.

ODEMx simulator/target code: Following the above principles, the modified compiler pipelines for an ODEMx simulator generation (Fig. 2.B) and for the target code generation (Fig. 2.C) can be constructed.

5.4 Experiment Management System

Experiment management ensures the consistency of artefacts that are necessary for an experiment, so that during analysis, the modeller is provided with all information that characterizes experiments: e.g. used model, parameter settings, results, characteristics of the simulator/prototype (code generation process, software version, algorithms used etc.) and the machine (account, operating system, execution time, etc.). Therefore EMS [25] provides support for the three phases: experiment planning, experiment execution and evaluation of the experiment results.

In our use case study the results of the simulator runs (event traces) or application-level-messages transmitted within the WSN are stored within the relational database, the Experiment Repository (by importing log files or direct access through an API). Experimental results can then be evaluated manually by the Visualizer. This tool allows the presentation of a P-wave travelling through the network, with its detection (or non-detection) being marked by the sensor nodes changing their colour.

But there are much more possibilities to evaluate experiment results such as visualizing the messages between software processes running on the Sensing Node as a UML sequence diagrams or doing statistical analysis, e.g. to compute the average message length of messages sent during a certain time period.

6 SOSEWIN Application

The main purpose of the SOSEWIN application we developed with our GAF4WSN is to perform the tasks of an EEWS (Sect. 3.1). Therefore in this chapter the basic principles of the Alarming Protocol (AP), implemented in the WSN application layer, are introduced. To go more into detail the Earthquake Synthesizer (ES) is explained, followed by a description of the integration of the signal analysis performed by a single node to detect a P-wave.

6.1 Cooperative Alarming Protocol

To establish a hierarchical alarming system, each node in the WSN runs the AP with different roles at runtime [26]. The SOSEWIN nodes are organized into clusters using criteria that determine the optimum communication efficiency. Each cluster is headed by a Sensing Node (SN) that is designated as a Leading Node (LN), with whom the other SNs within its cluster communicate general "housekeeping/status" information and initial alarms. The LN in turn communicates with other LNs, including issuing of system alarms, based on each LN knowing the status of the nodes that make up its respective cluster.

The SNs continuously monitor the ground shaking and perform a signal analysis with the goal to detect/trigger a P-wave. If a critical number of P-wave triggers have reached the cluster's LN, this node informs its neighbouring LNs. In the case that a LN has received enough cluster alarms, a so-called system alarm will be sent as fast as possible to the sinks in the SOSEWIN (Gateway Node (GN)) that are responsible for forwarding those alarms the potential end-users (e.g. public organizations responsible for disaster management).

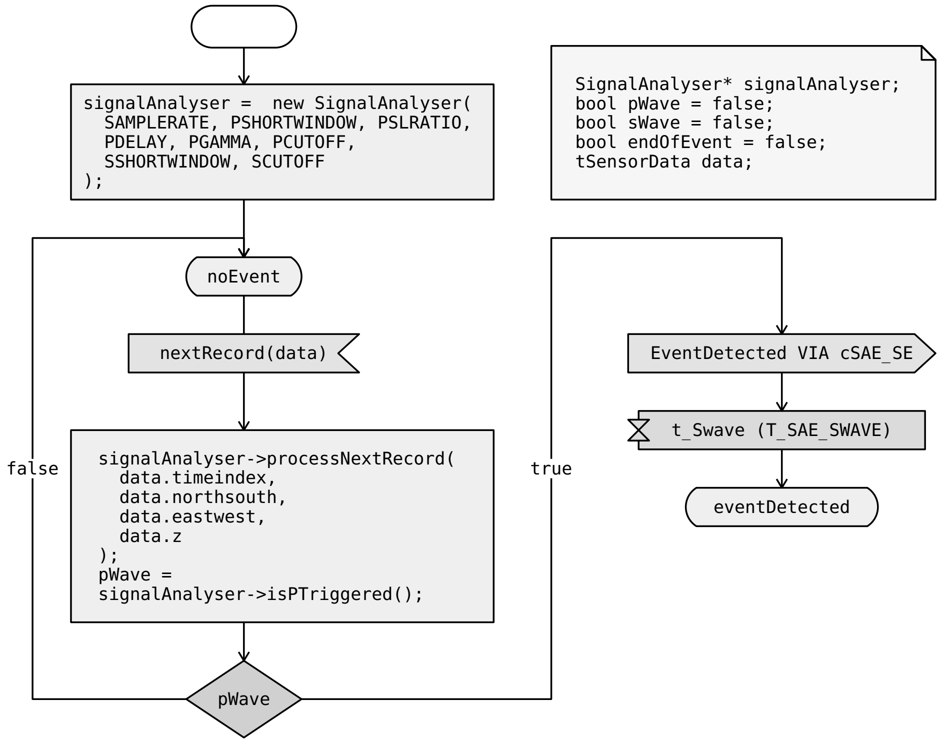
6.2 Earthquake Synthesizer

The ES generates synthetic seismograms for each location of a node based on the method of Wang [27]. Synthetic seismograms offer the opportunity to test different methods of event detection and classification, with the freedom to introduce as much (or as little) "noise" to the data as required.

6.3 Integration of Streaming Analysis

The Signal Analysis (SA) [28] of the incoming sensor data stream is originally performed by algorithms developed at the GFZ using the Matlab program. These algorithms where re-implemented by hand into C++ to be highly efficient running on each SN. In the SDL model within the part of a SN this code is included as passive UML classes.

Figure 3 shows a simplified SDL-RT state machine representing the Streaming Analysis within the AP (the corresponding generated code for the target platform is presented in Listing 1). Figure 4 shows a combined specification of the SDL process und a passive UML class related to the `SignalAnalyser` class performing the signal analysing algorithms. It processes a data record provided by the SDL signal input queue via `nextRecord(data)`. If a P-wave is detected, it informs another state machine and goes into state `eventDetected`.

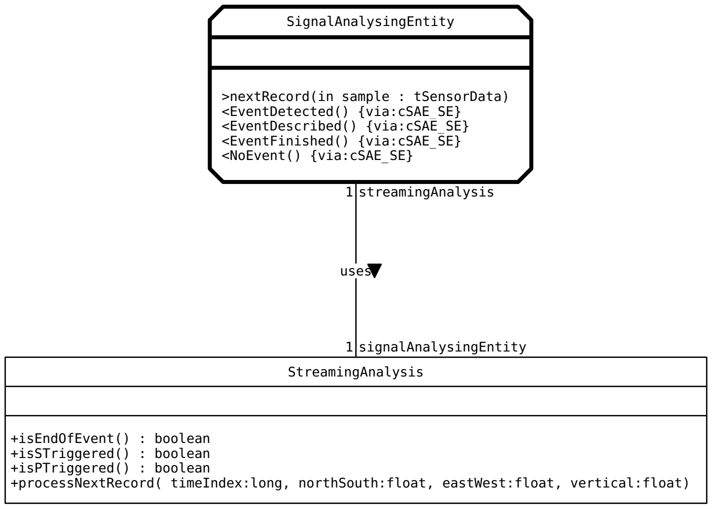


```
signalAnalyser =  new SignalAnalyser(
  SAMPLERATE, PSHORTWINDOW, PSLRATIO,
  PDELAY, PGAMMA, PCUTOFF,
  SSHORTWINDOW, SCUTOFF
);
```

```
SignalAnalyser* signalAnalyser;
bool pWave = false;
bool sWave = false;
bool endOfEvent = false;
tSensorData data;
```

```
signalAnalyser->processNextRecord(
  data.timeindex,
  data.northsouth,
  data.eastwest,
  data.z
);
pWave =
signalAnalyser->isPTriggered();
```

Fig. 3. Simplified SDL-RT state machine for Streaming Analysis



```
>nextRecord(in sample : tSensorData)
<EventDetected() {via:cSAE_SE}
<EventDescribed() {via:cSAE_SE}
<EventFinished() {via:cSAE_SE}
<NoEvent() {via:cSAE_SE}
```

```
+isEndOfEvent() : boolean
+isSTriggered() : boolean
+isPTriggered() : boolean
+processNextRecord( timeIndex:long, northSouth:float, eastWest:float, vertical:float)
```

Fig. 4. SDL-UML specification for the Signal Analysing Entity process and the Streaming Analysis class

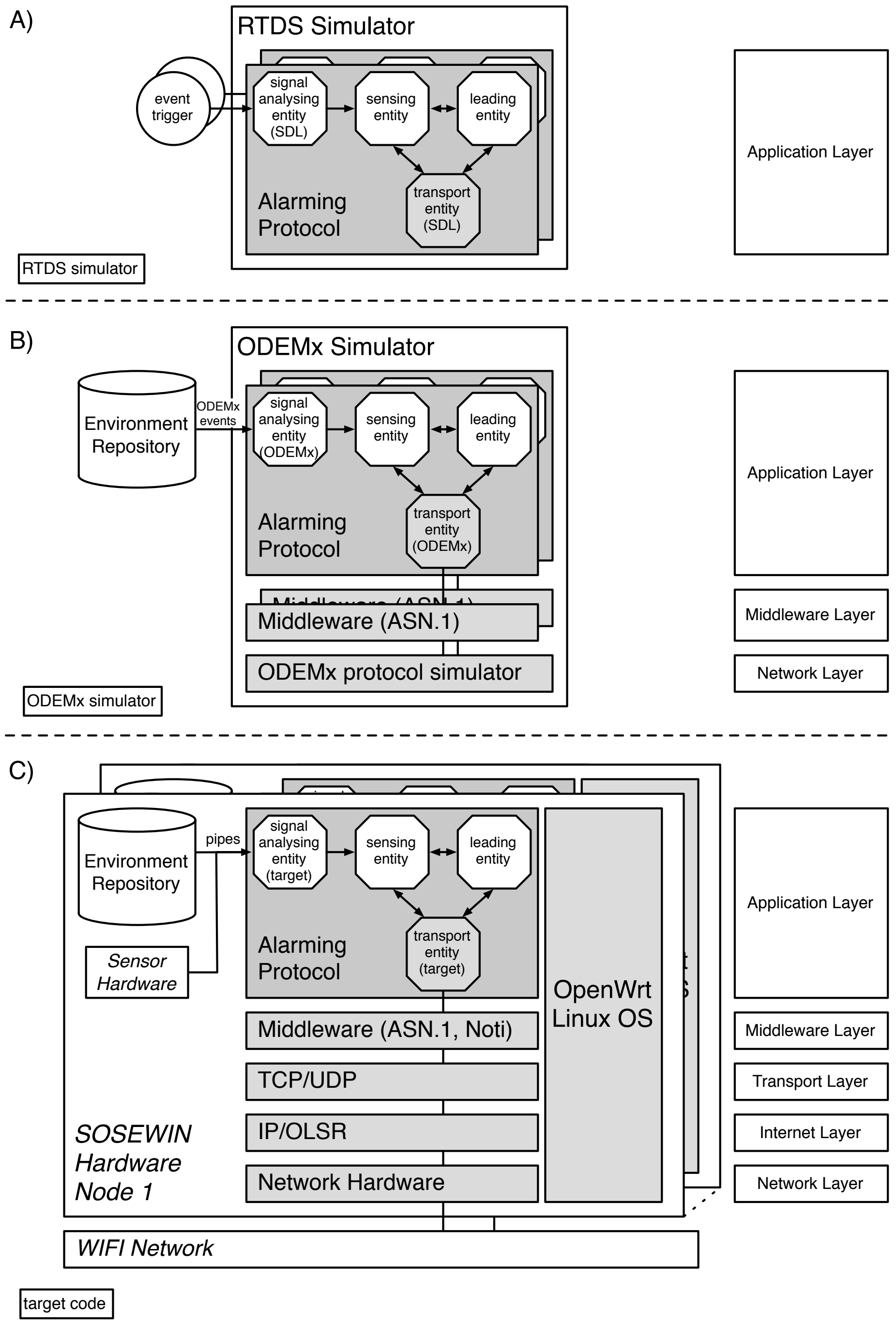


Fig. 5. The GAF4WSN simulators and testbeds

6.4 Simulators and Target Code

As motivated in Sect. 4 and described by their generation tools in Sect. 5.3 GAF4WSN supports code generation for SDL simulator, ODEMx simulator and target code. In Fig. 5 each subsection represents one of the three HUB SDL Transcompiler code generation target types and shows the specific input data, the generated parts of the AP, the substituted entities and the relation to the corresponding layers. In the development of the SOSEWIN software architecture each code generation target fulfils a specific purpose and provides a different level of abstraction.

The **SDL simulator** (Fig. 5.A) is used to test the functional behaviour of smaller ensembles of the SOSEWIN nodes. We abstract from concrete earthquakes, and underlying protocol layers. One further important preposition is perfect transmission behaviour of used communication channels over the air. The results of functional tests allow us to evaluate and improve the business logic of our AP. Typical outputs here are Sequence Diagrams (SD) and alarming signals without useful time constraints.

The **ODEMx simulator** (Fig. 5.B) uses the capability of our general-purpose ODEMx library, which supports next event simulation of parallel processes. It allows an integrated test of all developed software components (incl. SA, ASN.1 encoding and decoding) in a large network of sensor nodes. Every node processes its own set of sensor data provided by our ES. The outputs are general trace files which can be used to generate SD or ShakeMaps. Input, output and network configuration data come with precise time and spatial references. This simulator allows the estimation of required transmission times and transmission quality of alternative SOSEWIN configurations, which guarantees the early warning functionality is independent of different earthquake scenarios.

Target code (Fig. 5.C) is a generated software binary which is, after deployment and installation using the EMS, running on every node in the SOSEWIN network (Fig. 5.C shows two nodes communicating over WIFI network). For testing purposes it is possible to switch between the real sensor data input and the synthetic sensor data. The target code is integrated in a special package which could be installed remotely into the OpenWrt operating system.

Listing 1 shows the generated target C++ code based on the simplified Streaming Analysis state machine presented in Fig. 3. It realises the modeled state changes within an endless-running loop (lines 42-88) consuming signals depending on the current state. For example, if it has consumed a `nextRecord` signal (line 59) this is processed (line 61) and may result in the detection of a P-wave (line 68) and a switch to state `eventDetected` (line 73).

The **XEN simulator** [29] is an additional code generation target, not shown in Fig. 5, which allows to run several virtualized node hardware instances on a XEN server. XEN is a virtual machine monitor for different computer architectures. It allows several guest operating systems to be executed on the same computer hardware concurrently. In contrast to the SDL and ODEMx simulator the XEN simulator allows the complete integration test of all software components running on the real nodes (incl. the operating system and the complete communication stack). The link between the nodes is implemented by a network bridge without any packet loss. Due to the absence of a sensor in the virtualization the raw sensor data is provided by the Environment Repository.

```
#include <boost/thread/thread.hpp>
#include <boost/bind.hpp>
#include "SignalAnalysingEntity.h"

#define RTDS_PROCESS_NUMBER RTDS_process_SignalAnalysingEntity
#define RTDS_PROCESS_NAME SignalAnalysingEntity

SignalAnalysingEntity::SignalAnalysingEntity(RTDS::Logger& logger)
: SDLProcess(logger)
{
  RTDS_LOG_PROCESS_CREATION((int)msgQueue.writer, "SignalAnalysingEntity",
    (int)cover);
}

void SignalAnalysingEntity::main()
{
  RTDS_MSG_DATA_DECL
  short RTDS_transitionExecuted;

  SignalAnalyser* signalAnalyser;
  bool pWave = false;
  bool sWave = false;
  bool endOfEvent = false;
  tSensorData data;

  /* declare framework-side variables for the process
   * starts stdio.run() asynchronous stops when isRunnings
   * destructor is destroyed
   */
  boost::thread _workThread_t(boost::bind(
    &boost::asio::io_service::run, &ioService));
  auto_ptr<boost::asio::io_service::work> _workThread_ptr(isRunning);
  isRunning = 0;

  /* Initial transition */
  signalAnalyser =  new SignalAnalyser(
    SAMPLERATE, PSHORTWINDOW,
    PDELAY, PGAMMA,
    SSHORTWINDOW, SCUTOFF
  );
  RTDS_SDL_STATE_SET(noEvent);

  do
  {
    /* peek new message from queue */
    currentMessage = msgQRead();
    RTDS_LOG_MESSAGE_RECEIVE((int)&currentMessage->sender,
      (int)msgQueue.writer, currentMessage->sequenceNumber,
      getCurrentTime());
    /* Double switch state / signal */
    RTDS_transitionExecuted = 1;
    switch(RTDS_currentContext->sdlState)
    {
        /* ... */
        /* Transitions for state noEvent */
    case noEvent:
      switch(RTDS_currentContext->currentMessage->messageNumber)
      {
        /* Transition for state noEvent - message nextRecord */
      case nextRecord:
        RTDS_MSG_RECEIVE_nextRecord(data);
        signalAnalyser->processNextRecord(
          data.timeindex,
          data.northsouth,
          data.eastwest,
          data.z
        );
        pWave = signalAnalyser->isPTriggered();
        if ( pWave )
        {
          RTDS_MSG_QUEUE_SEND_TO_NAME(EventDetected, 0, NULL,
            "SensingEntity", RTDS_process_SensingEntity);
          RTDS_SET_TIMER(t_Swave, T_SAE_SWAVE);
          RTDS_SDL_STATE_SET(eventDetected);
          break;
        }
        RTDS_SDL_STATE_SET(noEvent);
        break;
      default:
        RTDS_transitionExecuted = 0;
        break;
      }
      break;
    default:
      RTDS_transitionExecuted = 0;
      break;
    } /* End of switch(RTDS_currentContext->sdlState) */
    delete currentMessage;
    } while (1);
}
```

Listing 1. Generated target C++ code for simplified Streaming Analysis state machine

7 Conclusion

SDL in combination with UML and ASN.1 is one of the most powerful technologies for modelling complex distributed reactive systems. In this paper we have demonstrated the extension of PragmaDev's RTDS into the core of model-driven development and administration approach of WSN, offered by one integrated framework GAF4WSN. Its architecture is based on OGC, OMG and ITU-T standards and combines different technologies for GIS, databases, behaviour modelling, information coding and code generation for target platforms and for different types of simulation. In addition, with experiment management, simulation and prototyping test cycles are handled independently, which leads to a quality improvement by applying these two evaluation techniques on different abstractions of the same real system. The advantages of GAF4WSN comprise not only the general support for modelling and simulation of environmental processes with geo-specific spatial contexts, but also the unification of information management of model experiments with tests of the prototyped WSN that allows us to use GAF4WSN for modelling as well as for administration. With the development of the first prototype of an EEWS for Istanbul we were able to demonstrate a proof of concepts. In the up-coming consolidation period of SOSEWIN in Istanbul we will extend our simulation experiments for preparing the extension of the real network with general network simulators like ns-3 to investigate large-scale topologies.

8 Acknowledgement

The work was supported by the SAFER project partners at the GFZ J. Zschau, C. Milkereit, K. Fleming and M. Picozzi, by our colleagues from the Humboldt University J.-P. Redlich, B. Lichtblau and J. Nachtigall (Systems Architecture Group) and our students T. Niebuhr, A. Jäkel, R. Kluth, D. Reinert and J. Wegener.

References

1. International Telecommunications Union: Recommendation Z.100 (11/07), Specification and Description Language (SDL), http://www.itu.int/rec/T-REC-Z.100/en
2. International Telecommunications Union: Recommendation X.690 (11/08), Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), http://www.itu.int/rec/T-REC-X.690/en
3. Seismic eArly warning For EuRope (SAFER), http://www.saferproject.net
4. PragmaDev RTDS V3.4, http://www.pragmadev.com
5. The Network Simulator – ns-2, http://www.isi.edu/nsnam/ns
6. Henderson, T.R., Roy, S., Floyd, S., Riley, G.F.: ns-3 project goals. In: WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator. ACM New York (2006)
7. JiST – Java in Simulation Time / SWANS – Scalable Wireless Ad hoc Network Simulator, http://jist.ece.cornell.edu
8. OPNET Modeler. http://www.opnet.com/products/modeler/home.html
9. Kuorilehto, M., Hännikäinen, M., Hämäläinen, T.D.: Rapid design and evaluation framework for wireless sensor networks. Ad Hoc Netw. 6(6), 909–935 (2008)
10. Park, S., Savvides, A., Srivastava, M.B.: Simulating networks of wireless sensors. In: WSC '01: Proceedings of the 33nd conference on Winter simulation, pp. 1330–1338. IEEE Computer Society (2001)
11. Downard, I.: Simulating sensor networks in ns-2, Naval Research Laboratory (2003)
12. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: accurate and scalable simulation of entire TinyOS applications. In: SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems, pp. 126–137. ACM, New York (2003)
13. Perrone, L., Nicol, D.: A scalable simulator for TinyOS applications. In: Proc. Winter Simulation Conference 2002., vol. 1., pp. 679–687. IEEE (2002)
14. Kuhn, T., Geraldy, A., Gotzhein, R., Rothländer, F.: ns+SDL – The Network Simulator for SDL Systems In: Prinz, A, Reed, R, Reed, J. (eds.) SDL 2005. LNCS, vol. 3530, pp. 103–116 Springer, Heidelberg (2005)
15. Milkereit, C., Fleming, K., Picozzi, M., Jäckel, K.H., Hönig, M.: Deliverable D4.5 Development of seismological analysis of single station recordings and network trigger for Early Warning purposes, http://casablanca.informatik.hu-berlin.de/wiki/images/4/41/D4.04_4.05_SAFER_GFZ.pdf
16. ShakeMap Manual; Technical Manual, Users Guide and Software Guide, http://pubs.usgs.gov/tm/2005/12A01/pdf/508TM12-A1.pdf
17. Erdik, M., Fahjan, Y., Ozel, O., Alcik, H., Mert, A., Gul, M.: Istanbul Earthquake Rapid Response and the Early Warning System. Bulletin of Earthquake Engineering 1(01), 157–163 (2003)

18. Picozzi, M., Milkereit, C., Zulfikar, C., Ditommaso, R., Erdik, M., Safak, E., Fleming, K., Ozel, O., Zschau, J., Apaydin, N.: Wireless technologies for the monitoring of strategic civil infrastructures: an ambient vibration test of the Faith Bridge, Istanbul, Turkey, http://casablanca.informatik.hu-berlin.de/wiki/images/2/23/Picozzi_et_al._Ambient_Vibration_Fatih_Bridge_BEE.pdf
19. Open Geospatial Consortium (OGC), www.opengeospatial.org/standards/wfs
20. SEEDLink. http://www.iris.washington.edu/data/dmc-seedlink.htm
21. Fischer, J., Kühnlenz, F., Eveslage, I., Ahrens, K., Nachtigall, J., Lichtblau, B., Heglmeier, S., Milkereit, C., Fleming, K., Picozzi, M.: Deliverable D4.22 Middleware for Geographical Applications, http://casablanca.informatik.hu-berlin.de/wiki/images/2/2f/D4.22_SAFER_UBER.pdf
22. ODEMx - Object oriented Discrete Event Modelling. http://odemx.sourceforge.net
23. Boost. http://www.boost.org
24. OpenWrt. http://www.openwrt.org
25. Kühnlenz, F., Theisselmann, F., Fischer, J.: Model-driven Engineering for Transparent Environmental Modeling and Simulation. In: Troch, I., Breitenecker, F. (eds.) Proceedings MathMod Vienna 2009. ISBN 978-3-901608-35-3. (2009)
26. Fischer, J., Kühnlenz, F., Eveslage, I., Ahrens, K., Nachtigall, J., Lichtblau, B., Heglmeier, S., Milkereit, C., Fleming, K., Picozzi, M.: Deliverable D4.21 Develop Software for Network Connectivity, http://casablanca.informatik.hu-berlin.de/wiki/images/7/73/D4.21_SAFER-UBER.pdf
27. Wang, R.: A simple orthonormalization method for stable and efficient computation of Green's functions. Bulletin of the Seismological Society of America 89(3), 733–741 (1999)
28. Milkereit, C., Fleming, K., Picozzi, M., Jäckel, K.H., Hónig, M.: Deliverable D4.4 Development of the seismological analysis software to be implemented in the Low Cost Network, http://casablanca.informatik.hu-berlin.de/wiki/images/4/41/D4.04_4.05_SAFER_GFZ.pdf
29. XEN. www.xen.org