OETCS

openETCS

ITEA2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

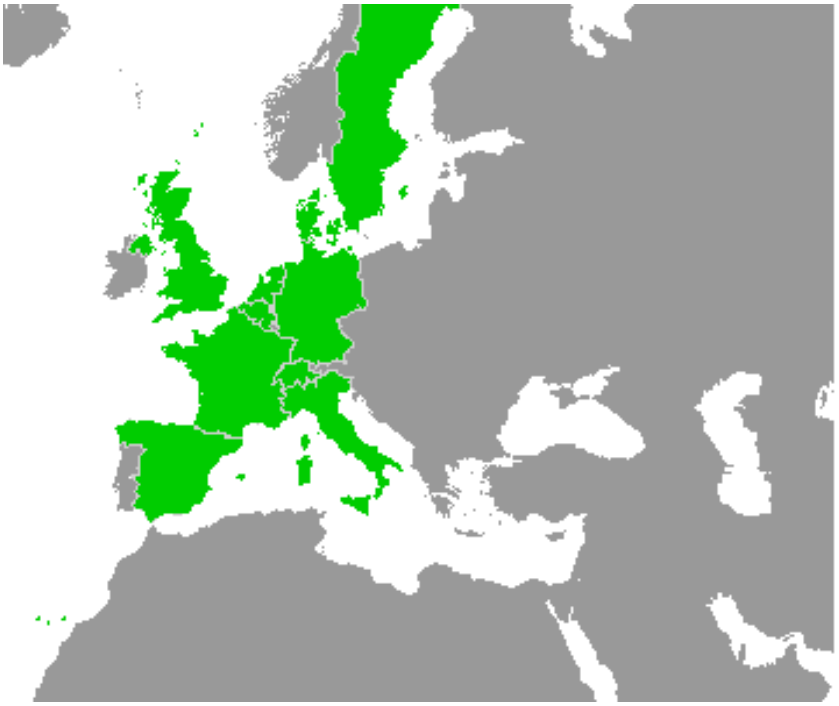
ITEA2 Project
2012 – 2015

Work-Package 7: “Toolchain”

SCADE Model of Subset 026, Section 3.5, Management of Radio Communication (MoRC)

Uwe Steinke, Siemens AG

February 2013



This page is intentionally left blank

SCADE Model of Subset 026, Section 3.5, Management of Radio Communication (MoRC)

Uwe Steinke, Siemens AG
Siemens AG

Model Description

Prepared for ITEA2 openETCS consortium
Europa

Table of Contents

Figures and Tables **Figures**

Tables

This document describes a formal model of the requirements of section 3.5 of subset 026 of the ETCS specification 3.3.0. This section describes the management of a communication session between onboard and on-track equipment from the view of the onboard unit (OBU).

The model is written in the (graphical) formal language SCADE by using the SCADE Suite, an integrated SCADE development environment.

At the beginning of this document a short overview about the SCADE language and the SCADE development suite will be given.

The following chapters introduce the "‘Management of Radio Communication"’ model to ease it’s unterstanding.

For a complete and detailed model description please refer to the MoRC document, generated from the model itself and located within this repository.

1 Short Introduction to Formalism and Tool

1.1 Motivation and Intention

The intention of SCADE is to provide a language and the appropriate development tool suite for embedded control software where high demands on dependability, reliability and safety are required.

Therefore SCADE targets especially on safety related systems in the avionics, rail and automotive domain.

SCADE provides certified code generators that transform the SCADE sources into executable C and ADA code.

1.2 SCADE Language and Formalism

SCADE is a formal textual and graphical language, where textual and graphical elements can be mixed with each other. The textual form is more applicable for automatized use, if the SCADE code itself should be generated from other tools. The graphical form is more intuitive and therefore suits better to manual modelling tasks. Within this document we will refer to the graphical representation.

1.2.1 The SCADE paradigm

The SCADE language is based on one basic paradigm, that must be understood by it's users.

SCADE models are

- synchronously
- clocked
- data flow and state machines and
- any nested combinations of these

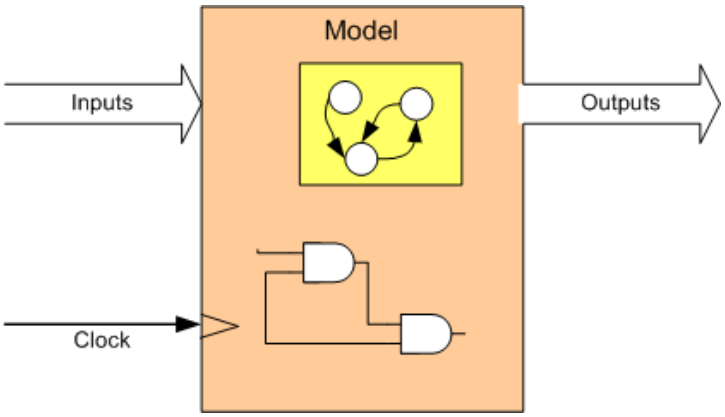


Figure 1. The SCADE Paradigm

A clock cycle consist of these actions:

- Apply input data to the model
- Compute / execute

- Disseminate model output data to external interfaces

SCADE does not make any assumptions on the clock timing. A continuous aequidistant period clocking leads to implicite dependable timing relations, while discontinuous clocking often suits better to existing hardware platforms.

In any case, the synchronous clocking principle causes a dependable timing behaviour of the model running on the target system without any racing and transient bug effects.

1.2.2 SCADE Language

While SCADE models can be written as text, the more friendly way of modelling is the graphical schematic entry with the editor that comes as part of the SCADE Suite.

The intention of the SCADE language is not only to model the structure of a software application, where the software has to be implemented with other languages afterwards. Instead, SCADE models are the software implementation itself and in any case executable and verifyable. This is very different from modelling languages like SysML. On the other hand, the level of abstraction is below SysML.

The SCADE language provides all capabilities typically required for embedded control applications. It is based on strong data types like bool, integer, real, enumerations and structures and arrays of these.

SCADE offers two fundamental concepts for modelling: (Data) flows and state machines.

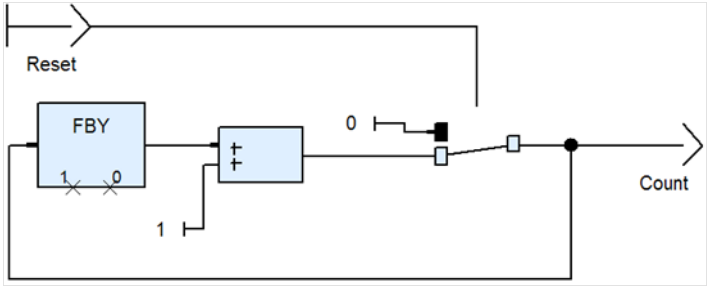


Figure 2. SCADE data flow example

The data flow design are similar to electrical schematic diagrams and can be understood intuitively. Elementary operators for data flows are

- boolean functions
- arithmetic functions
- choice (if/then/else, switch case, ...)
- iterators over arrays of data, state machines and functions (map, fold)
- temporal operators (access to previous values of data flows)

State machines and data flows have completely deterministic semantics. For example, state transitions have unambiguous priorities for the case when more than one guard condition should be met.

Data flows and state machines can be intermixed and parallelized and nested without limitations, which gives freedom to choose the more appropriate method for each model detail.

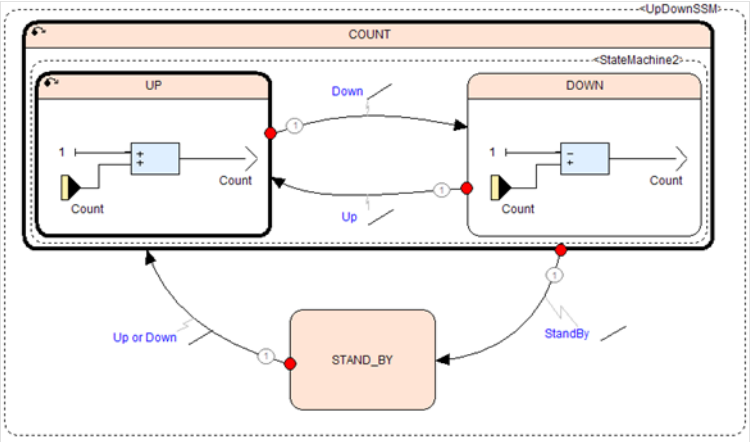


Figure 3. Example of 2 nested state machines with data flows

The SCADE language comprises some intentional restrictions that prevent the model designer from risky constructions in dependable and critical software applications:

- Static resource allocation only (no dynamic object creation at runtime)
- No recursions
- Loops / iterations with maximum repetition count only (no never ending loops)

2 Modeling Strategy

The model implements the SRS Subset-026, section 3.5 "‘Management of Radio Communication"’. This was achieved by modeling the subsections of the document one after the other subsequently in a clause-by-clause manner.

The requirements text clauses were analyzed to identify the inputs and outputs of a subfunction und the task of the subfunction itself. The inputs and outputs were adopted as model in- and outputs directly and both interconnected by implementing the required task from inputs to outputs.

This proceeding lead to a lot of small subfunctions. Then, the inputs and outputs of these subfunctions were matched so that outputs of subfunctions were connected to inputs of other subfunctions that required these kinds of information as inputs. At the end, the remaining inputs and outputs not connected within the model were taken as inputs and outputs of the whole model.

3 Model Overview

The models top level operator is "‘managementOfRadioCommunication"’. It incorporates a state machine "‘CommunicationSession_SM"’ with its states reflecting the subsections of section 3.5 like "‘Establishing a communication session"’, "‘Maintaining a communication session"’ and "‘Terminating a communication session"’.

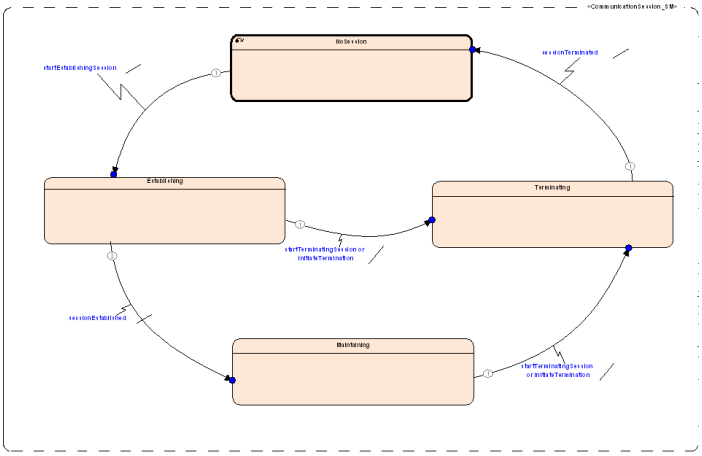


Figure 4. Session Management State Machine

In parallel to the "‘CommunicationSession_SM"’ state machine additional model nodes "‘registeringToTheRadioNetwork"’ and "‘safeRadioConnectionIndication"’ directly correspond to further subsections of 3.5. The node "‘initiateTerminatingASession"’ assists in "‘Terminating a communication session"’.

All 3.5 functions reside below the model top level layer as nested operators in "‘managementOfRadioCommunication"’.

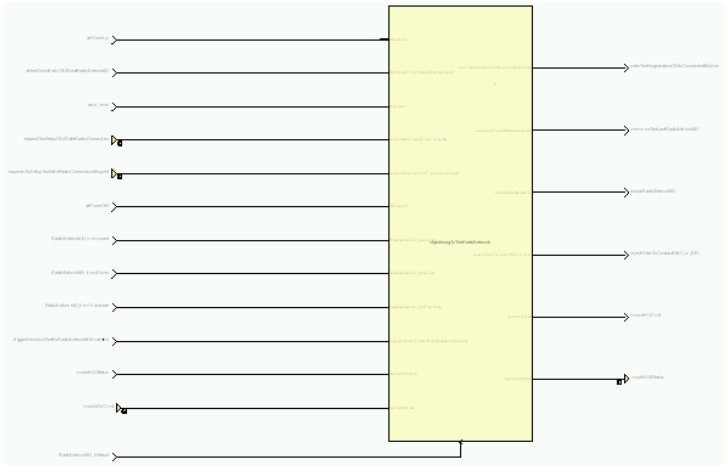


Figure 5. Registering to the Radio Network

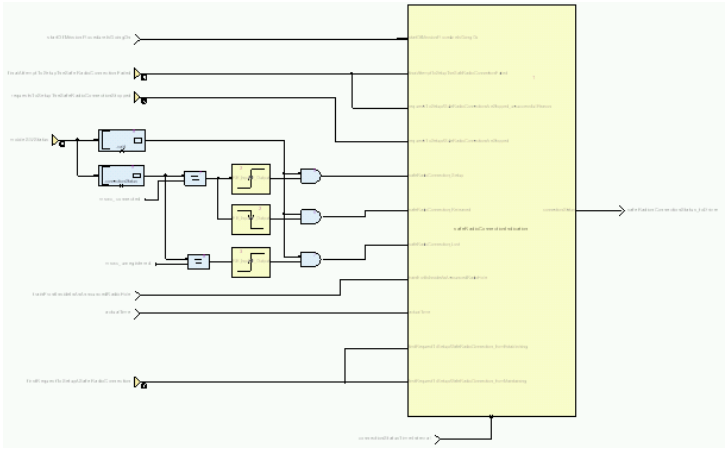


Figure 6. Indication of the Safe Radio Connection

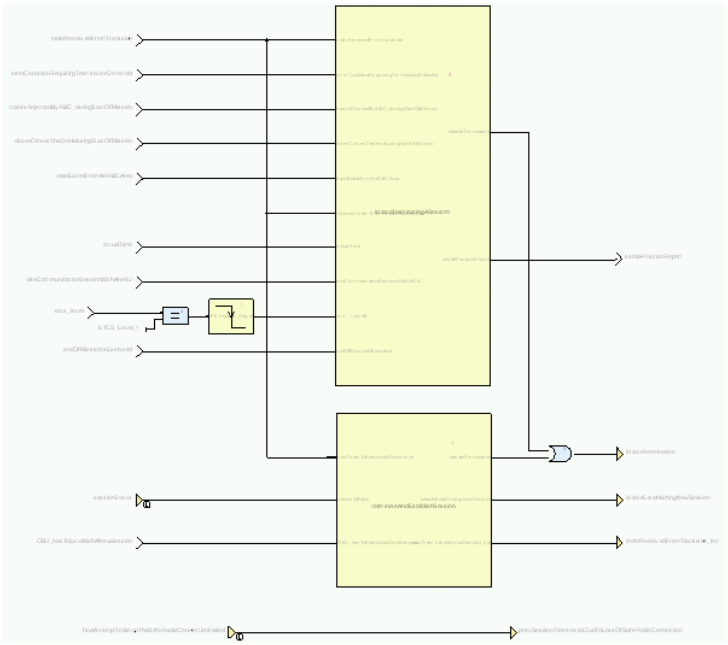


Figure 7. Initiate terminating a Session

4 Model Benefits

The described way of transformation from the written textual requirement specification in conjunction with the modelling technique leads to the following benefits:

1. The model is strictly formal and concrete. In opposite to semiformal modelling languages there is no room for unambiguousness or weaknesses.
2. The model is the implementation. It is the implementation of the required functions (the textual requirement specification) as understood by the modelling persons.
3. The model is executable, simulatable and verifiable. With the SCADE code generator, a runnable can be generated from the model. It is ready for debugging, simulation and test on (graphical) model level, so that the behavior of the modelled functions can be investigated practically. It can be executed on any platform for which a C- or ADA compiler exists. This applies to the whole model as well to its subfunctions, which eases incremental and iterative development processes.
4. To achieve requirements traceability, the model artefacts can be linked with the requirements located in the textual specification in fine granularity
5. By applying description and explanations within the model, a documentation can be generated from it automatically.
6. For integration into different toolchains, the tools are usable via command line and automatization interfaces, providing model information and meta-information, and there are plugins for eclipse.
7. The tools, especially the code generator, are qualified for safety-related software development compliant to DO-178B/Level a and CENELEC EN50128/SIL4.
8. The language and tool chain are mature, efficient and complete.

Some drawbacks of the shown proceeding:

1. Since the model originates from written requirements document, its structure is close to the documents structure. This cannot be expected to be optimal from the system or architectural point of view.
2. The modeling language and technique is very suitable for the chosen target. For more complex systems like the whole ETCS OBU, a more abstract, less formal model and language intermediate layer like SysML might be appropriate between the textual specification and the SCADE model layer.

5 Detailed Model Description

A detailed descripted was generated from the model itself: Management_of_Radio_Communication

