Project: COMPASS

Grant Agreement: 287829

Comprehensive Modelling for Advanced Systems of Systems

COMPASS

Case Studies of SysML to CML transformations

COMPASS White Paper 09

Date: March 2014

Public Document

http://www.compass-research.eu

Authors:
Lucas Lima, Alvaro Miyazawa, Ana Cavalcanti, University of York, UK

Abstract

This white paper presents the case studies used to exercise the SysML to CML transformation plug-in developed in the Artisan Studio tool. The modelled problems are the leadership election, which is a protocol for distributed systems, the dwarf signal, which is a mechanism to control rail traffic, and the classic producer-consumer problem. We provide descriptions of the problems and an overview on how they were modelled in SysML.

Case Studies of SysML to CML transformations

Lucas Lima Alvaro Miyazawa Ana Cavalcanti

March 2014

1 Introduction

In this document we present overviews of the case studies used to exercise the SysML to CML transformations. We performed three case studies: the leadership election protocol, the dwarf signal and the producer-consumer problem. The SysML models of each case study are available at `http://www.compass-research.eu/Project/Publications/SysML2CML/examples.zip`.

2 Leadership Election Protocol

Leadership election algorithms have been developed in distributed computing to designate a node as the organiser of a number of other nodes [6, 2]. Our case study corresponds to this protocol considering a multimedia network of devices. In this scenario, a network of multimedia devices needs to synchronise audio and video information in order to make the multimedia environment consistent. Such information must be managed by a control unit however the network does not have a centralised control unit. The solution is to elect a leader among the devices that should be responsible for managing this control information.

We provide two SysML models to this problem, an abstract version, which is simpler and corresponds to a conceptual model, and a concrete version, which is more complex and corresponds to the implementation model. For simplicity, we consider a system with three devices. The property that must be respected is that if there are active devices then exactly one leader must be chosen among them.

The abstract model only has one block `LE SoS`, which represents the whole system, and a data type `Device` modelling a specific device in the network. The system block has three parameters: `devices`, which represents the three devices of the system, `Active`, which is a set that keeps the identification of the devices that are on, and `Elected`, which is a set containing the identification of the devices chosen as leaders. The block has two operations: `turnOn(Integer)` that given a device identification turns such a device on, and `turnOff(Integer)` that given a device identification turns the device off. There is a signal called `tick` to represent the time passing. The verification of leadership only occurs after the occurrence of a `tick` event. The behaviour of the `LE SoS` block is described by the state machine `LE SoS STM`. This state machine can receive `turnOn` and `turnOff` requests and once a `tick` event occurs, if there is no leader (cardinality of `Elected` equals to zero) and there are active devices (cardinality of `Active` greater than zero) then exactly one leader is chosen from the `Active` set. In case there is no active device, then the set `Elected` is set to zero. This description of behaviour assures that exactly one leader is chosen when there is at least one active device.

The concrete model considers a system where each device communicates with the others asynchronously in order to maintain the current state of the network. Each device keeps its snapshot of the state of the system based on these communications. The block `SoS` represents the system, which is composed of three blocks `Device` and one block `Bus`. The choice on the leader is made according to the device with the highest petition. The longer a device stays active, higher is its petition. When turned on, a device can be in three states: Undecided, Follower and Leader. The device is in the Undecided state once it is turned on and it does not have information on the network yet, or when there is more than one leader in the network. In the latter situation all devices go to Undecided to agree in one new leader. A device is in the Follower state when the network already has a leader or it does not have a leader yet but the leader should be another device

due to its higher petition. Finally, the device is a leader when the network has no leader and its petition is the highest petition. This behaviour is described by the state machine of a device called `DeviceSTM`.

In order to gather information about the state of the network, the devices communicate with each other through a bus. They can broadcast information about its state and petition and receive these data from other devices. If the device is waiting a communication from another device that is turned off, a timeout event may occurs to make the device proceed to the next communication. The bus acts a relay of information sent by the devices. It just receives packs of data and transmit then to the proper device.

So far we have only exercised analysis considering the abstract version of this model due to the increased complexity of the concrete model. However, we are working with simplification strategies of the generated model in order to make such an analysis feasible for the concrete version as well.

3 Dwarf Signal

Dwarf Signal [3, 5] is a kind of railway signal which is used at the side of track when space is limited. It has three lamps which are displayed in different configurations to give instructions to train drivers. The signal's three lamps are named L1−L3 and different configurations of a signal can be written using set notation, for instance this signal is in {L1, L2} which means Stop. The signal has a total of four proper states which are the well-defined commands a signal can convey to a driver. When all lamps are off (indicated by the empty set {}) the signal is in the Dark state, which is a power saving mode for when the signal is not in use. The three remaining proper states Stop ({L1,L2}), Warning ({L1,L3}) and Drive ({L2,L3}) are indicated by a combination of two lamps and correspond to the positions of an old-style semaphore signal. Along with the four proper states there are also three transient states which describe the unstable states when a signal is moving from one proper state to another, since the signal may only light or extinguish one lamp at a time. These states are {L1}, {L2} and {L3}. Finally there is the ambiguous state {L1, L2, L3} which a signal should never display as it is meaningless.

To ensure the safety of this system, four safety properties are given:

1. Only one lamp may be changed at once
2. All three lamps must never be on concurrently
3. The signal must never be Dark except if the Dark aspect has to be shown or there is lamp failure
4. A change to or from Dark is allowed only from Stop or to Stop, respectively

These properties essentially form a contract with the signal, or the administrator of the signal, which ensures that the driver never observes an ambiguous state, and therefore never has to make an ill-informed decision.

This problem is modelled by a single block `Dwarf` representing the system. As previously described, it has three lamps and they are initialised to the stop state (L1 and L2 are on, and L3 is OFF). It also has a property named `desiredState` that represents the state that the lamps must become. This property can only be set to one of the four proper states. The block has four operations: `shine` only displays the lamps that are on, `setDesiredState` changes the value of the property `desiredState`, `light` turns a lamp on and `extinguish` turns a lamp off. Two versions of the behaviour were designed. The first only has one state machine that models the possible transitions of the system. The second version uses an activity (`ActShine`) to model the behaviour of the `shine` operation and the state machine is a simplified version of the previous one because it calls the cited activity to answer `shine` calls. This latter version was designed to exercise the integration between block, state machine and activity.

4 Producer-Consumer

This case study is a model of the well known producer-consumer problem. The system is designed as a composition of three blocks, `Producer`, `Buffer` and `Consumer`. While the producer only adds items to the buffer, the consumer only removes them. The buffer stores items. It has an upper bound of five items and it provides two operations, `add(Item)` that inserts a item in the buffer and `rem()` that removes an item from

the buffer and return it. These operations are modelled by activities that use opaque actions to change the state of the buffer. The behaviour of the `Buffer` is described by a state machine, which only allows the `add` operation if the upper bound is not reached and, conversely, the `rem` operation can only be invoked when the buffer is not empty. For the other cases, the calls are deferred by the state machine.

5 Conclusions

We have explained the three case studies used to exercise the SysML to CML transformations. The SysML models were designed in the Artisan Studio tool [1]. The models were exported in pdf files and they are available at `http://www.compass-research.eu/Project/Publications/SysML2CML/examples.zip`. They were used to exercise the CML generator plug-in, which was developed to extract CML specifications from SysML models designed according to the guidelines described in [4].

References

[1] Artisan Studio. `atego.com/products/artisan-studio/`, 2013.

[2] Shlomi Dolev and Nir Tzachar. Empire of colonies self-stabilizing and self-organizing distributed algorithms. In MariamMomenzadehAlexanderA. Shvartsman, editor, *Principles of Distributed Systems*, volume 4305 of *Lecture Notes in Computer Science*, pages 230–243. Springer Berlin Heidelberg, 2006.

[3] AlistairA. McEwan and J.C.P. Woodcock. a refinement based approach to calculating a fault-tolerant railway signal device. In Ren Jacquart, editor, *Building the Information Society*, volume 156 of *IFIP International Federation for Information Processing*, pages 621–627. Springer US, 2004.

[4] A. Miyazawa, L. Albertins, J. Iyoda, M. Cornélio, R. Payne, and A. Cavalcanti. Final report on combining SysML and CML. Technical report, 2013.

[5] J.C.P. Woodcock. Montigels dwarf, a treatment of the dwarf signal problem using csp/fdr. In *Proceedings of the 5th FMERail Workshop*, 1999.

[6] David Zuckerman. Randomness-optimal sampling, extractors, and constructive leader election. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 286–295, New York, NY, USA, 1996. ACM.