Assistance for the Construction of Critical Embedded Control Systems using Event-B

(Master Internship - Requirement Document, 04-2015)

Présentation M2 ALMA 15/09/2016

Christian Attiogbé

AeLoS Team
LINA CNRS UMR 6241 - University of Nantes
{firstname.lastname}@univ-nantes.fr

Abstract. This is a *requirement document*. It introduces guidelines to develop a software dedicated to the assistance of the construction of critical embedded control systems. The software will be used as a support of a method that we have previously proposed to construct critical embedded control systems with Event-B. It should be integrated in the Rodin platform. The functionalities of the desired software will be make precise in an agile development process.

1 Introduction

We have to develop a support software that will help and guide a specifier to build the main parts of a critical embedded control system.

In Fig. 1 we give the general principle of the main parts and the functioning principle of control systems.

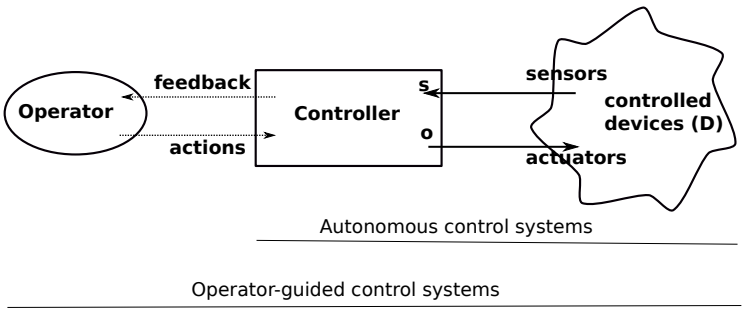

Fig. 1. Principle of control systems

In the following section we summarize as a pseudo-algorithm, the method that we have proposed to build the control systems using the Event-B formal method.

2 A Pseudo-Algorithm to Assist the Construction Method

The pseudo-algorithm gives the core of our construction method made of two processes described in the following. Therefore the construction of the software will be faithfull to this pseudo-algorithm.

2.1 Horizontal Process

Step 1: Characterize the abstract model of a considered system

On the basis of the interface between a control system and its environment, a user should provide the interface variables, the list of controlled devices of its system, the global and specific properties required by its systems. From these elements the specifier will be assisted in building an Event B machine M_0 and then a refined one M1. The machine M1 can be further refined until more concrete levels.

This document is the basis for the development of such a support software assistant.

Step 1.1: Elicitation of the interface

Let $\mathcal{X}_s$ be a set of input variables;
Let $\mathcal{X}_o$ be a set of output variables;
Let $\mathcal{X}_c$ be a set of control variables;

And additionnally, let $\mathcal{X}_i$ be a set of internal variables of the controller.
(A part of the control variables are used for the feedback, the internal variables are the part of the control variables used by the controller but which are not output as feedback).

Step 1.2: Elicitation of the global properties of the system Considering the requirement of the given system,

Let $\mathcal{P}_s$ be a set of safety properties;
Let $\mathcal{P}_n$ be a set of non-functional properties;
Let $\mathcal{X}_l$ be a set of liveness properties.

Consider the list of controlled devices of the system : $D_1, D_2, D_3, \cdots$

Step 1.3: Start with a first abstract model Build a context machine (Ctx0) made of all the carrier sets needed to describe the listed interface variables.

Machine M_0
SETS // via sees Ctx0
VARIABLES
List the variables from $\mathcal{X}_s$, $\mathcal{X}_o$, $\mathcal{X}_c$, $\mathcal{X}_i$
INVARIANTS
Build $Properties(\mathcal{X}_s, \mathcal{X}_o, \mathcal{X}_c, \mathcal{X}_i)$ from $P_s, ...$

INITIALISATION
Generate $Subst_{var \in \mathcal{X}_s \cup \mathcal{X}_o \cup \mathcal{X}_c \cup \mathcal{X}_i}(SubstInit(var)) \cdots$
END

Step 1.4: Systematic construction of the events of the abstract model

Sensing events family: Let $\mathcal{E}_s$ be a set of sensing events

Monitoring events family: Let $\mathcal{E}_m$ be a set of monitoring events

Control events family: Let $\mathcal{E}_c$ be a set of control events

Reaction events family: Let $\mathcal{E}_a$ be a set of reaction events

Step 2: Extend the abstract global model (with feature augmentation)

while $(\mathcal{E}_s \neq \varnothing) \wedge (\mathcal{E}_o \neq \varnothing) \wedge (\mathcal{E}_c \neq \varnothing) \wedge (\mathcal{P}_s \neq \varnothing)$ **do**
Select an event e from $\mathcal{E}_s$ and update $\mathcal{E}_s$ $(E_s = E_s - \{e\})$ **or**
Select an event e from $\mathcal{E}_m$ and update $\mathcal{E}_s$ **or**
Select an event e from $\mathcal{E}_a$ and update $\mathcal{E}_a$ **or**
$M_0 = Add_e(M_0, buildEVent(e))$
Select a property p from $\mathcal{P}_s$ and update $\mathcal{P}_s$
$M_0 = Add_p(M_0, formalise(p))$
Prove M_0
end while

Step 3: Introduce the specific properties

$M_1 = M_0$
while $(\mathcal{P}_n \neq \varnothing)$ **do**
Select a property p from $\mathcal{P}_n$ and update $\mathcal{P}_n$
$M_1 = Add_p(M_1, formalise(p))$
end while

2.2 Vertical Process

Step 4: Refine the global abstract model

Describe the behaviour of the devices $D_1, D_2, D_3, \cdots$ using automata or the appropriate models. Encode these models in Event-B (roughly, the transition of the automata are encoded as Event-B event).

Step 5: Decompose into software and physical parts

Control part events:
Let $\mathcal{E}_{soft} = \mathcal{E}_m$
$M_{control}$ = Projection($M_1, \mathcal{E}_{soft}$)

Physical part events:
Let $\mathcal{E}_{phys} = \mathcal{E}_s$ and related events $\cup$ $\mathcal{E}_a$ and related events
M_{physic} = Projection($M_1, \mathcal{E}_{phys}$)

Implement the decomposition with Rodin.

Step 6: Refine the control software and the physical environment

Step 6.1: Refining the control software

Event-B modelling skills are required.....
TODO

Step 6.2: Refining the controlled (physical) environment

Specific Event-B and Engineering expertises are required....
TODO

3 Work to be done

During the Master internship, the student(s) should develop a software that implement the above method.

- The software will be used by Event-B specifiers/engineers.
- The input of the software will be provided by the users (specifiers/engineers). A user-friendly graphical interface will be designed for this purpose.
- The output of the software will be at least: a development guidelines and some skeleton of Event-B models.

We will adopt an agile process to describe more in details the requirement document and also for the averall development of the software.

4 Perspectives

Generation of the skeleton of the models at deifferent levels.

Development of a tracability assistant, that checks the built models with respect to the required results (of the various steps).

5 References

TODO