A Modelling Method for Embedded Systems

Ed Brinksma, Angelika Mader, Jelena Marincic, Roel Wieringa
University of Twente, The Netherlands

Abstract

We suggest a systematic modelling method for embedded systems. The goal is to derive models (1) that share the relevant properties with the original system, (2) that are suitable for computer aided analysis, and (3) where the modelling process itself is transparent and efficient, which is necessary to detect modelling errors early and to produce model versions (e.g. for product families).

Our aim is to find techniques to enhance the quality of the model and of the informal argument that it accurately represents the system. Our approach is to use joint decomposition of the system model and the correctness property, guided by the structure of the physical environment, following, e.g., engineering blueprints. In this short note we describe our approach to combine Jackson’s problem frame approach [1, 2] with a stepwise refinement method to arrive at provably correct designs of embedded systems.

1 Introduction

Our long-term goal is to improve the quality of control software in embedded systems. One way to achieve this goal is by application of verification techniques, such as model checking, to a model of an embedded system. A prerequisite for a successful verification are good models, i.e. models that share the relevant properties with the original systems (truthful), that are small enough to be analyzed by verification tools (feasible), and, that can be derived in an efficient and transparent way (trustworthy). In state-of-the-art verification most focus is on verification algorithms and tools, and less on modelling issues (model hacking). This motivates our work.

For embedded systems modelling decomposes in a natural way into modelling of the control and modelling of the physical environment. In general, modelling software is easier, because software in itself is already a formal object. For the physical environment modelling has to bridge between an informal, physical object and a formal one. Therefore, modelling the environment cannot be purely formal. This process is far from understood, i.e. different people tend to produce different models of the embedded system, and our trust in the models often depends upon our trust in the people who made them. Our aim in this project is to improve the quality of the modelling process that is at the heart of the verification process. By this we mean that (1) In the modelling process, reusable and validated knowledge about the modelled system is used and (2) The justification that the model accurately represents the system (wrt a correctness property) is clear and understandable.

The first goal makes the modelling process more efficient and less dependent on the genius of the modeller and the second goal makes the resulting model more reusable.

Our solution idea is to construct the system model and the proof of its correctness wrt the property of interest at the same time. We produce the model by means of stepwise decomposition of both the environment and the property of interest, so that the accuracy of the model wrt the property is justified by a structural similarity between the model and the system.

In section 2 we describe our suggested modelling method. It will be illuminated by an example section.

2 The Method

Our goal is to derive good models of embedded systems that can be used for verification. Quality criteria for a good model are: (1) The model shares the properties of interest with the original system. (2) The model is suitable for computer aided analysis, i.e. small sized models. (3) The derivation process is transparent and efficient, such that design errors can be detected easily and variations of the model (for, e.g., product families) can be applied straightforwardly.

The systems we consider are control applications, consisting of a control machine interacting with a physical environment. The control machine itself consists of a physical part, such as a PLC, sensors and actuators, and a software part.

We want to achieve the quality criteria mentioned above.
by a systematic modelling method with the following ingredients:

A verification theorem that describes the logical relation between environment, control and properties, and that also guides the construction of the proof: $E \land S \models P$, where $E$ describes the potential or assumed behaviour of the environment (without the control machine) and $P$ describes the desirable behaviour of the environment (interacting with the control machine). $S$ is a specification of the control machine.

Decomposition of environment, control and properties with respect to the desired behavior. The justification of the decomposition steps comes from different sources, such as engineering diagrams, problem patterns, experiments. The decomposition steps are often non-monotonic in the sense that adding new details can require more assumptions, natural laws etc, such that the properties of the refined model may no longer imply those at the previous level.

Figure 1: Stepwise decomposition of the environment model and the correctness proof. $P$ is a property of desired behavior $D$ of the environment. $D$ is decomposed into $D_i$ and and $P_i$ is a property of $D_i$.

Describing all possible behaviour of the environment typically ends up in models that are too large for computer aided analysis. Therefore, a relevant aspect here is that we restrict the description to the desired behaviour (and the error cases we do want to consider).

Patterns. Another relevant ingredient of our method is the use of problem and modelling patterns, following the idea of problem frames introduced by Jackson [1, 2]. Using patterns gives (at least) two advantages: first, it makes a decomposition argument more transparent, because we can use some form of standard argument, and, second, it makes the decomposition steps more reliable, because we make use of well-proven solution fragments. To make the idea of problem frames useful we need a library of identified patterns with modelling solutions.

3 An example

In this section we will illustrate our ideas by an example. Our example is not worked out in detail, we only demonstrate the refinement steps, show the problem diagrams and show what the verification theorem should look like. However, in earlier work we have proven a description of this example with PVS [3]. Due to lack of space our example covers not all aspects of the modelling method suggested. In the first place, we focus here on decomposition. The elaboration of verification theorems requires more formal involvement and space, which also holds for the use of patterns. The example is a small PLC-controlled Lego-plant that sorts yellow and blue blocks according to their colour (see figure 2). A number of blocks is in a queue. When the belt is moving the first block of the queue moves to the belt. At the scanner the colour of a block can be detected. A block moves further on into the sorter. The sorter consists of two fork-like arms, each driven by its own motor and equipped by an angle-sensor that can be used to check whether the sorter arm has made a full rotation. The task of the control program is in principle simple: the belt has to move unless there is a block being sorted and at the same time a new block at the scanner. If there is a block in the sorter it has to be moved to the correct side.

Figure 2: Top view on the Lego plant

The Lego plant is controlled by a programmable logic controller (PLC), a P8 from Nyquist. The control program structure is written in the language Sequential Function Charts (Petri-net like), extended by the low level language Instruction Lists.

The first design step is to state explicitly which fragment of the system is modelled, which is, due to the size, the whole system here. Additionally, we start with some assumptions: E.g., there are only blue and yellow blocks in the queue, no other colour.

We start with a description of the system by a problem diagram in figure 3, which corresponds to an architectural description.

We have a Requirement $R_0$ and the interface description $a$. In the verification theorem the requirement $R_0$ represents the desired property. The interface description $a$ contains the phenomena that should occur in both models, the one of the plant and the one of the machine. Informally, we have $R_0$: All blocks will be sorted eventually by the plant.
The verification theorem in its first version is:
Plant $\land$ Machine $\implies R_0$

Because at this level of abstraction we do not have enough details for a meaningful description we continue with the first refinement. Figure 4 contains the first refinement step.

The informal component description is:
Queue: Stores unsorted yellow and blue blocks and moves a block on the Belt.
Belt: Transports blocks alongside the Scanner, to the Sorter. The speed of the belt is either $v$ or 0. The block is present in front of the Scanner for a minimum time of $t = \frac{v}{l}$ seconds, where $v$ is the speed of the belt (in meters per second) and $l$ is the length of a block (in meters).
Scanner: Scans the part of the path through which a block is transported from the Queue to the Sorter, recognizes blue, yellow and absence of blocks.
Sorter: Sorts a block outside the plant on the appropriate side by rotating only one of the two arms depending of the block’s colour. After a full rotation a block is gone and the sorter is idle again.
Machine: The Machine switches the Belt on if there is no block at the Scanner. The Machine switches the Belt on if the Sorter is idle. It switches the Belt off, if there is a block in the Sorter and a block at the Scanner. The Machine moves the Sorter arms according to the colour of the block in the Sorter. A Sorter arm is moved only for one full rotation. The Machine continuously receives one of three possible values from the Scanner.

The interface descriptions (for the example here, informally) are:

a1: The machine switches the belt on and off.
a2: The scanner sends 3 values to the machine, blue, yellow and nothing.
a3: The sorters sends the angle positions of the arms to the machine; the machine switches each sorter arm on and off.
p1: The first block of the queue moves to the belt, or nothing moves to the belt.
p2: The scanner detects whether there is either nothing or a yellow block or a blue one at the scanner position.
p3: If a block is at the end of the belt and the belt is moving (and the sorter is idle), then the block moves to the sorter.

The verification theorem at this stage has the form
Queue $\land$ Belt $\land$ Scanner $\land$ Sorter $\land$ Machine $\land$ Natural Laws $\implies R_1$
where all elements should be filled in by a description that should formalize the informal description above. Note that in the verification theorem we do not have explicit interfaces. The composition is done by the logical $\land$, and all phenomena taking place on the interfaces should be part of the component descriptions sharing the interface.

The Natural Laws typically contain facts as: If the belt is not moving a block on the belt remains in its position. or A block does not change its colour.

We perform a second refinement step, introducing the actuators (motors), and the sensors (scanner, anglesensor).

$R_2$ remains unchanged wrt $R_1$, and also $r1$ and $r2$ remain the same.
The component descriptions remain almost the same. The extensions with the additional elements are more or less obvious, and we omit them due to lack of space. The interface descriptions, again informally, are:

- **a21:** The machine switches Motor A on and off.
- **a22:** The scanner sends 3 values to the machine, blue, yellow and nothing.
- **a231:** The machine switches Motor C on and off.
- **a22:** The scanner sends 3 values to the machine, blue, yellow and nothing.

Patterns. We can identify two sources for patterns: the first is the decomposition of physical environments in general. We can have an architectural, a functional or a process-oriented decomposition as in [5], where the method suggested here has been performed in detail.

The second source for patterns comes from assumptions in PLC modelling. Roughly, the assumptions concern the relative times of the PLC scan-cycle in comparison to the real-time requirements of the environment. This allows for different modelling decisions for the control, as instantaneous, periodic or lazy reaction, [4].

**Control problem frames.** The idea of problem frames (rather than solution patterns) was introduced by Jackson [1, 2] as a means to identify reusable problem structures in different problem domains. In our example we used a simple control frame, as also in Jackson’s sluice control. Specific issues to be addressed for frames are: (1) The compositionality of problem (solution) frames (2) The robustness of decomposition trees under small changes in properties and/or environments (3) Systematic analysis of environment assumptions and consequences of their failure to hold.

**Model justification.** Another research question is if informal engineering argumentation (physical, graphical, ...) can be combined with formal reasoning. We believe that the decomposition process should provide the justification of the accuracy of the model wrt the required property. The decomposition process may be constrained by the education of the plant engineer and available formal languages. Visual representations of the formal decomposition that are understandable to the plant engineer may help.

**Acknowledgment.** This paper benefited from extensive discussions with Michael Jackson.

### 4 Discussion and conclusions

Our modelling method progresses from a top-level understanding of the system in its environment into more detailed understanding, decomposing the environment model and its desired properties in parallel. We stop decomposing when we reach environment phenomena that can be observed or controlled by the control machine. At that level, we have obtained a correct specification of the controller. We have applied this method on several cases, including the one discussed in this paper [3, 5]).

Many research questions remain, some of which we mention here.

**Patterns.** We can identify two sources for patterns: the first is the decomposition of physical environments...