Design automation method for software development in complex embedded systems

Qiao, Ying

http://hdl.handle.net/10945/52653
DESIGN AUTOMATION METHOD FOR SOFTWARE DEVELOPMENT IN COMPLEX EMBEDDED SYSTEMS

Ying Qiao  
Computer Science Department  
Naval Postgraduate School  
Monterey, CA93943, USA  
yqiao@nps.navy.mil

Luqi  
Computer Science Department  
Naval Postgraduate School  
Monterey, CA93943, USA  
luqi@nps.navy.mil

Lynn Zhang  
Computer Science Department  
Naval Postgraduate School  
Monterey, CA93943, USA  
lzhang@nps.navy.mil

Abstract

Today, high confidence systems of embedded systems (SoES) are widely used in many fields where consequences of failures are serious. However, current approaches for software development of monolithic embedded system cannot meet the challenges existing in SoES development. In this paper, we present a new design automation method for software development in SoES. The core of this method is the model mapping which transforms high-level high-confidence properties identified in the requirement model into some low-level constraints in the design model. This method provides a quantitative way to handle the high confidence and keep the high confidence consistent and visible during the development process. It improves software quality and enables software automation.

Key words: Systems of embedded systems, High-confidence, Requirement model, Design model, Model mapping

1 Introduction

Complex embedded systems, known as systems of embedded systems (SoES), are widely used today and are usually deployed for long periods of time. They usually have mission critical requirements and demand real-time and high-confidence performance. There are two main problems in software development of SoES: (1) High-confidence properties of SoES are hard to keep consistent during the whole development process, making software quality difficult to ensure in the end product; and (2) Software development of SoES is time consuming and error-prone.

In recent years, many methods have been proposed for software development of high confidence monolithic embedded systems. Most of them are based on formal methods such as static verification [1-5], dynamic verification [6-8] and formal analysis [9, 10]. However, these approaches lack a systematic way to handle high confidence properties and lack the modeling capability to describe distinct characteristics of SoES such as emergent properties and independence of components. These shortcomings make them difficult to solve problems in software development of SoES identified above.

The contribution of this paper is to present a new design automation method for software development of SoES. The basic idea of this method is to use the model mapping between the requirement model and design model to keep the high confidence consistent and visible during the development process. This method provides a quantitative way to handle the high confidence. It improves the high confidence in resulting software and enables software automation which reduces the development time and costs.

The rest of paper is organized as follows: section 2 addresses the framework of the new design automation method; section 3 describes two formal models — requirement model and design model for SoES; section 4 presents a set of model mapping rules; section 5 is the conclusions and future works.

2 Overview of the Design Automation Method for SoES

Figure 1 describes the framework of the new design automation method for software development of SoES. This method spans the requirement stage and design stage. The main elements in this framework include a requirement model, a design model and the model mapping. The requirement model focuses on the customer’s view and mathematically describes the requirement of SoES by capturing its functional and non-functional aspects. It is the starting point for whole development process. Design model focuses on the designer’s view and describes SoES by capturing its structures and behaviors.

Model mapping is the core of this method. It transforms some key properties such as timing properties.
and high-confidence properties identified in the requirement model into concrete constraints described in the design model. The model mapping rules are organized in an open rule base. A reasoning engine is responsible to perform the model mapping based on this rule base. Although lots of efforts should be applied to accomplish the proposed method, this paper only focuses on addressing requirement model, design model and model mapping rules.

![Diagram](Figure 1 Framework of Design Automation Method for High-confidence SoES Development)

3 Formal Models for Design Automation

We have presented requirement model and design model for high confidence SoES [11, 12]. Formally, the requirement model, denoted as $\zeta'$, can be represented as follows:

$$\zeta' = (V, G, H)$$

$V = (v_1, v_2, \ldots, v_i)$ is a service vector which describes all functions of whole SoES. $v_i$ denotes a service which is a functional unit of SoES. $G = \{g_i | i \in [1, l]\}$ is a functional emergent property vector which represents the functional aspects of SoES requirements. $g_i$ is a set of functional emergent properties for service $v_i$. Typical functional emergent properties identified in the requirement model are timing properties such as maximum response time. $H = \{h_i | i \in [1, l]\}$ denotes a non-functional emergent property vector which describes the non-functional aspects of SoES requirements. $h_i$ is a set of high-confidence properties for service $v_i$. Furthermore, each $h_i \in h_i$ can be represented by a high-confidence metric vector.

The design model for SoES, denoted as $\zeta$, is formally represented as follows:

$$\zeta = (S, E, C, D, F_1, F_2)$$

$S$ is the component system set, $S = \{s_i | i \in [1, n]\}$. $s_i$ denotes the component system constituting SoES ($n$ is the number of component systems in the whole SoES). $E$ denotes the interaction sets between component systems, $E = \{e_{jk} | j, k \in [1, n]\}$, where $e_{jk}$ denotes the set of interactions from component system $s_j$ to component system $s_k$. $C$ denotes constraint sets on how the component systems are used in the given environment, $C = \{c_i | i \in [1, n]\}$. $D$ denotes constraint sets on interactions between component systems, $D = \{d_{jk} | j, k \in [1, n]\}$, where $d_{jk}$ is a set of constraints, each of which applies to interactions in $e_{jk}$.

The $s_i$ and $e_{jk}$ describe structures of SoES while $c_i$ and $d_{jk}$ describe behaviors of systems of embedded systems.

The constraints defined in $C$ and $D$ have certain mapping relationships with the properties identified in $G$ and $H$. Theoretically, $C = F_1(G, H)$, $D = F_2(G, H)$, where $F_1$ and $F_2$ are two mappings that transform emergent properties in the requirement model into local constraints imposed on component systems and interactions that described in the design model. $F_1$ and $F_2$ reflect model mapping between the requirement model and design model which will be addressed in section 4. The main soundness property for these mappings is that conformance to the constraints in $C$ and $D$ should be sufficient to establish confidence in the goals represented by $G$ and $H$. Details of these two formal models can be found in [11] and [12].

4 Model Mapping Rules

In this section, we will develop a set of model mapping rules to transform two high-level emergent properties, i.e., one typical timing property – maximum response time (MRT) and one typical high-confidence property –
reliability identified in the requirement model into some low-level constraints imposed on component systems and interactions.

Although emergent properties are often realized bottom-up, it is still meaningful to decompose them and map them into the local constraints imposed on component systems and interactions. The decomposition of emergent properties can guide the designer to choose suitable component systems to build SoES that meet the requirements. This provides a systematic way to construct SoES to satisfy requirements.

4.1 MRT Mapping Rules

Timing properties have important impacts on high confidence in SoES. Maximum Response Time (MRT) is the most typical timing property identified in the requirement model. To ensure a given service SE delivers the final result within certain MRT, in design model, the corresponding component systems that realize service SE should finish tasks within certain deadlines. For this purpose, the period of component system, the maximum execution time of component system and the latency of the interaction between component systems should be carefully designed so that the schedulability of these component systems can be guaranteed. Based on this analysis, MRT identified in the requirement model can be mapped into some timing constraints imposed on the component systems and the interactions in the design model. These constraints include Deadline, Period and Maximum Execution Time (MET) of the component system and Latency of the interaction. Assume Com$_1$ and Com$_2$ are two component systems that realize service SE in the design model and there is an interaction e$_{12}$ between these two component systems. Thus, the MRT mapping rule is described as follows:

**Rule MRT Mapping is:**

if \( (MRT \in g_{SE}) \) then

\[
\begin{align*}
\text{Deadline} & \in c_1; \\
\text{Period} & \in c_1; \\
\text{MET} & \in c_1; \\
\text{Deadline} & \in c_1; \\
\text{Period} & \in c_1; \\
\text{MET} & \in c_1; \\
\text{Latency} & \in d_{12}; \\
\text{Com}_1.\text{period} + \text{Com}_1.\text{MET} + e_{12}.\text{latency} + \\
\text{Com}_2.\text{period} + \text{Com}_2.\text{MET} < SE.\text{MRT} \\
\text{Com}_2.\text{deadline} - \text{Com}_1.\text{deadline} \geq E_{12}.\text{Latency} + \\
\text{Com}_2.\text{MET}
\end{align*}
\]

Here, \( h_{SE} \) represents the value of emergent property or constraint \( B \) imposed on component \( A. \) \( g_{SE} \) represents a set of functional emergent properties for service \( SE. \) \( c_1 \) is a set of constraints imposed on \( \text{Com}_1 \) while \( c_2 \) is a set of constraints imposed on \( \text{Com}_2. \) \( d_{12} \) denotes a set of constraints for interaction \( e_{12}. \)

4.2 Reliability Mapping Rules

Reliability is one of the most typical high confidence properties. In [12], we have shown how reliability is transformed into some functional constraints related to fault tolerance such as the constraint of masking which indicates the basic style for the fault tolerance [13]. Based on this result, the reliability mapping rule can be illustrated as follows:

**Rule Reliability Mapping is:**

if \( (\text{Reliability} \in h_{SE}) \) then

\[
\begin{align*}
\text{FR} & \in c_1; \\
\text{FR} & \in c_1; \\
\text{FR} & \in d_{12}; \\
\text{Masking} & \in c_1; \\
\text{Masking} & \in c_1; \\
\text{Masking} & \in d_{12};
\end{align*}
\]

Here, \( h_{SE} \) denotes a set of high-confidence properties for service \( SE. \)

5 Conclusions and Future Works

Software development of high-confidence SoES is a great challenge. Many approaches have been proposed to construct high-confidence embedded systems. However, all these approaches were proposed for complex but monolithic embedded systems rather than SoES. They failed to model the distinct characteristic of SoES and didn't provide a systematic way to handle the high-confidence properties.

In this paper, we present a new design automation method for software development of SoES. The core of this method is the model mapping between the requirement model and design model. It provides a quantitative way to handle the high confidence and enables high-confidence properties which are identified and derived in the early stage (requirement stage) to be applied consistently in the later stage (design stage). This method improves the high confidence in resulting software and supports the software automation which reduces the development time and costs. It can be applied
to software development of typical systems of embedded systems such as home security systems [14].

However, further work still needs to be done. First, model mapping rules need to be extended and refined. The transformation of other high-confidence properties such as availability, safety and security should be also considered. Furthermore, the reasoning scheme used by the reasoning engine also needs to be studied. In this case, the reasoning algorithm supporting light-weight inference [15, 16] for model mapping will be considered.

References