MIRELA: A Language for Modeling and Analyzing Mixed Reality Applications Using Timed Automata

Jean-Yves Didier  *  Bachir Djafri  †  Hanna Klaudel  ‡

Laboratoire IBISC - Université d’Evry Val d’Essone / CNRS FRE 2873 - FRANCE

ABSTRACT

We propose a compositional modeling framework for Mixed Reality (MR) software architectures in order to express, simulate and validate formally the time depending properties of such systems. Our approach is first based on a functional decomposition of such systems into generic components. The obtained elements as well as their typical interactions give rise to generic representations in terms of timed automata. A whole application is then obtained as a composition of such defined components. To ease writing specifications, we propose a textual language (named MIRELA: Mixed Reality Language) along with the corresponding compilation tools. The generated output contains timed automata in UPPAAL format for simulation and verification of time constraints, and which also may be used to generate source code skeletons for an implementation on a MR platform.

Keywords:  Mixed reality systems modeling, time constraints, timed automata, formal analysis.

Index Terms:  D.2.4 [Software Engineering]: Software/Program Verification—Model checking; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; I.6.2 [Simulation and Modeling]: Simulation Languages

1 INTRODUCTION

Mixed Reality (MR) hardware systems tend to be more and more complex. This is partly due to new ways of interacting with such systems and the wide range of available human computer hardware interfaces.

MR developers have to interface a wide range of heterogeneous devices (like data-gloves, motion trackers, force-feedback devices like haptic arms, inertial sensors, cameras, D-GPS, GPS, . . . ) with their own MR applications. However, the challenge is not only restricted to sensor’s heterogeneity, but it also relies on variety and novelty of algorithms and techniques developed in parallel with new hardware. It explains why during the past years almost thirty different projects of modular software architectures and frameworks have been developed for MR to cope with these constraints [3]. Moreover, those new fields of research tend to enrich the feeling of presence of the user in VR environment and also to fuse VR with reality. One lead is to propose several kinds of interactions modalities to the end user by combining visual, aural and haptic feedbacks, each of these sensory modalities giving rise to its own rendering loop. Indeed, sensors as well as rendering loops have their own time constraints, generally different from each other.

Nowadays, the current process for developing MR applications relies mostly on fast response and high hardware performances to cope with real-time constraints. However, for some applications (for example, teleoperation or haptic ones) the respect of time constraints may be critical. Therefore, it may be worth to validate the application, before testing it on actual hardware, by modeling it and applying formal method techniques to prove its robustness in terms of temporal integrity. The benefits may be twofold: it may avoid unnecessary cost related to a possible deterioration of hardware, and in the case of design errors, it allows to identify their source and to be corrected and validated again.

2 MODELING TIME DEPENDING PROPERTIES

In this work we are interested in specification and validation of heterogeneous, reconfigurable, open but not hot pluggable MR systems. We propose a compositional modeling framework for software architectures in order to express, simulate and validate formally their time dependent properties. Our intention here is not proposing a new formalism, but taking advantage of known real-time specification and verification techniques in the design and the programming of MR systems. We chose for this purpose to use a prominent model of timed automata [1], as represented in Fig. 1, and its associated tool UPPAAL [4]. It allows simulating systems and verifying, through model-checking, various reachability properties. Typically, it can answer the designer questions which may look like “starting from its initial state, can the system reach a given state in a given delay?”, or “starting from the initial state and given concrete time constraints, can the system deadlock (meaning that the specified time constraints cannot be met)?”.

Figure 1: Example of a timed automaton modeling the processing of a task, where clk is a clock. After the reception of a signal work_processing!, the automaton spends at least $t_{processing\_min}$ time in the location Init. Then, it sends the signal free_processing! if the processing time does not exceed $t_{processing\_max}$, otherwise, it emits error_processing!.

Our approach is based on a functional decomposition of such systems into generic components. The obtained elements as well as their typical interactions give rise to generic representations in terms of timed automata and the whole application is then obtained as a composition of such defined components. The MR architectures addressed here are organized according to a data flow-oriented scheme, from sensors to the actual result produced by the rendering loops. As represented in figure 2, data are produced by sensors (cameras, GPS, motion trackers,...), then they are processed by processing units (for example, in charge of noise filtering, image processing,...) and stored in a shared memory where they are picked up by the rendering loops. A rendering loop is a pipeline which
processes such data and transforms them to the actual rendering result (for example, images on a screen or force feedback according to the rendering device).

The elements of such a functional decomposition giving rise to timed automata, are:

- Periodic and aperiodic sensors capturing data from the environment;
- Four basic sorts of data processing units (PU) processing data received from sensors, each of them abstracting actual processing and focusing on time constraints:
  - A Unary PU starts processing the data as soon as it is received on its unique input,
  - An AtLeast PU starts processing when data are received at least on one of its two inputs,
  - A Both PU has two buffered inputs and starts processing when data are ready in both buffers,
  - A Priority PU has one master input and one buffered slave input. It starts processing when data are ready in master input and possibly uses buffered data from the slave input.
- A shared memory composed of registers supporting mutually exclusive reading and writing operations, and possibly several periodic rendering loops;
- Various sorts of controllers needed to compose together automata for the previously introduced elements. In particular, it is possible to define groups of sensors which may be suspended once another sensor is running.

3 The MIRELA LANGUAGE

To ease the writing of MR architecture specifications, we define a textual language called MIRELA(for Mixed REality Language) provided with timed automata semantics as introduced above.

A MIRELA application description is composed of two parts: Component and Sheet. The first one corresponds to the declaration of all the elements like sensors and processing units, while the second one describes how these elements should work together. A Sheet is composed of three elements: a non empty set of controllers, the shared memory and one or several rendering loops. Declarations of sensors, processing units, shared memory and rendering loops are parametrized and relate to bounds and delays needed for the actual implementation.

Figure 2 gives an example of a MIRELA specification for a classical setup for an MR application involving a camera (C) and a graphical user interface (G) with one visual rendering loop. C is modeled as a periodic sensor while G as an aperiodic one. We assume that the camera C may be suspended by the graphical user interface G supervising it. Constant values referring to time constraints are expressed in microseconds.

The semantics is obtained by associating to each syntactical element the corresponding timed automaton. The whole specification is given by a parallel composition of all these automata. More precisely, the translation instantiates the templates defined for each component of the application. For each line, we have an instantiated automaton, and two complementary automata are also generated to manage initializations and error management for periodic sensors. The resulting model comprises in this case 13 timed automata synchronizing through channels and including a large number of time constraints. The obtained set of timed automata representing the given system has been analyzed and validated using existing tools (UPPAAL). It has been simulated and checked against basic behavioral properties, permitting to show in particular, that the system was deadlock free and was meeting its expected behavior.

4 Conclusion and Perspectives

We introduced in this paper a compositional modeling framework for MR software architectures dedicated to specify and validate formally the time-dependent properties of such systems. The framework offers a simple language, called MIRELA, allowing to express in a concise way the elements of such architectures with their connections, delays, periods...using generic components. It allows also to be automatically translated into a corresponding set of timed automata in order to be analyzed and validated using standard tools.

In our future work we will be interested in developing a method and tools allowing to use the obtained timed automata model for automatically generating source code skeletons for the ARCS implementation platform [2], following the scheme in figure 4.

References