"Integrare", a Collaborative Environment for Behavior-Oriented Design

Lian Wen[1], Robert Colvin[2], Kai Lin[1], John Seagrott[1], Nisansala Yatapanage[1], and Geoff Dromey[1]

[1] Griffith University, Brisbane, Qld, Australia
[2] University of Queensland, Brisbane, Qld, Ausralia
{l.wen,j.seagrott,g.dromey}@griffith.edu.au,
{robert,nisansala}@itee.uq.edu.au,
kai.lin@student.griffith.edu.au

Abstract. In this paper, we introduce a new cooperative design and visualization environment, called "Integrare", which supports designers and developers in building dependable, component-based systems using a new behavior-oriented design method. This method has advantages in terms of its abilities to manage complexity, find defects and make checks of dependability. The environment integrates and unifies several tools that support multiple phases of the design process, allowing them to interact and exchange information, as well as providing efficient editing capabilities. It can help formalize individual natural language functional requirements as Behavior Trees. These trees can be composed to create an integrated tree-like view of all the formalized requirements. The environment manages complexity by allowing multiple users to work independently on requirements translation and tree editing in a collaborative mode. Once a design is constructed from the requirements, it can be visually simulated with respect to an underlying operational semantics, and formally verified by way of a model checker.

Keywords: behavior-oriented design, behavior tree, software environment.

1 Introduction

Software tools, from editors and compilers to software engineering environments, which are integrated collections of different tools, have been developed and used from the very early days of software engineering [4]. As software systems are becoming larger and more complex, selecting the right tools and environments is critical to the quality and speed of developing these systems [6]. In this paper, we introduce a new collaborative environment "Integrare", which can be used throughout multiple phases in the software design cycle, such as requirement engineering, simulation, formal specification, and model checking.

Integrare is built to support the Behavior Tree (BT) design method [1], which is a process that constructs a component-based software design from the system's functional requirements. This process is a systematic method for translating informal natural language functional requirements into a formal BT representation, in a

Y. Luo (Ed.): CDVE 2007, LNCS 4674, pp. 122–131, 2007.
© Springer-Verlag Berlin Heidelberg 2007

straightforward and traceable manner. Validation of the system model is one of the most important tasks in developing software that meets the client's needs, and Integrare supports this in a rigorous manner by including simulation and model checking facilities, in contrast to many commercially available modeling tools based on UML [24]- [27] and requirement engineering tools [28]-[30]. The first version, which has been released for internal testing, includes the following functions:

- Visio-styled user interface.
- A collaborative (multi-user) working mode.
- A Requirement Translation Assistant (RTA).
- Simulation.
- Translation of BT to SAL for model checking.

Integrare was developed using C++, employing Visual Studio [21] and Microsoft Foundation Classes [22]. It uses XD++ [23] as the library to support graphical editing. The architecture is a hybrid of model-view [33] and event-driven [32] architectures.

The paper is organized as follows: in section 2 we briefly introduce the BT design process and notations. The architecture and GUI are described in section 3, and from section 4 to section 7, we present four major features of the tool, which are its collaborative working mode, the requirement translation assistant, simulation, and SAL translation. Related work is discussed in section 8.

2 The Behavior Tree Design Approach

The Behavior Tree (BT) approach is a software design process that constructs a component-based software design from the system's functional requirements. This process is a systematic method for translating informal natural language functional requirements into a formal BT representation, in a straightforward and traceable way [1] [7]. The constructed BT can be used to support different stages and different aspects of software engineering such as requirements engineering [11], architecture and component design, software change [3], architecture normalization [2], model checking [9] safety [14], reliability issues [10], verification [15] and simulation.

Compared with UML, independent researchers find that the lack of precise [40], formal [36] and unambiguous [37] semantic models is one of the major difficulties in checking the consistency between different UML diagrams [38], translating UML into formal languages [39], and simulating UML models [40]. In contrast, the formal semantics of the BT notation has been stressed from the beginning; a formal semantic language Behavior Tree Specific Language (BTSL) has been developed [11], and a BT can be automatically translated into formal languages such as CSP [9] and SAL [10], and described by a metamodel [8]. Even though a BT is a formal specification, unlike formal languages such as CSP, SAL or B notation [41], the flowchart-styled graphic notation of BT can be easily understood by non-experts. Therefore, the BT notation has both advantages as a formal language with precise semantics so it can be mechanically checked, analyzed and simulated, as well as a soft and casual modeling [5] that non-technical stakeholders find appealing.

The BT approach also provides a systematic way to transform the natural language described user requirements into component-based designs, in contrast to approaches