Object-based Language for Generalized State Machines

Dimitris Dranidis, George Eleftherakis, Petros Kefalas
Computer Science Department
CITY LIBERAL STUDIES
Affiliated Institution of the University of Sheffield
Tsimiski 13, 54624 Thessaloniki, Greece
{dranidis, eleftherakis, kefalas}@city.academic.gr

***Abstract*— X-machines is a formal method for modeling the behaviour of a software system. Although X-machines proved to be very useful in modelling and testing software systems, complex systems tend to have a complicated unstructured memory which furthermore may vary from developer to developer, depending on the modelling skills of the developer. Motivated by the fact that a UML class diagram can model the structure of a software system according to object-oriented principles, this paper presents OX-machines (Object X-machines), a specialization of X-machines, in which the memory structure is implicitly defined from class definitions. Thus, after modelling a system's structure with a class diagram, a developer can concentrate on the modeling of the system's behaviour. In this paper, the formal definition of OX-machines is presented. Furthermore, XMDL-O as a language extending XMDL (a markup language used to model X-machines) is presented. XMDL-O specifications can be transformed to XMDL specifications, for which several supporting tools exist, including a parser, a type checker, and an animator. Finally, examples from case studies illustrate the advantages of this object-based modeling approach.**

***Index Terms*— Formal methods, languages, state machines, object-oriented technology.**

I. INTRODUCTION

X-MACHINES [4], [7] is a formal method for modeling the behaviour of a software system. One of the main advantages of X-machines compared to the standard finite state machines is the existence of an extra structure called memory. Memory is a tuple including any information that is necessary to describe the data of a system and it helps in modelling complex systems by using a small number of essential states. However, it is not always straightforward to define the memory type. Another difference of X-machines from finite state machines is that transitions are labeled with functions and not simple inputs. Depending on the representation selected for the memory, the definition of the functions may vary. In the majority of the cases the function definitions imply a suitable memory structure. Changing the function definitions may cause changes in the memory representation. Modelling of large systems, like the steam-boiler case study [1] using X-machines [5], has shown that the memory structure could be very difficult to write, understand and correspondingly it is easy to make mistakes or omissions, and difficult to identify and fix logical errors.

Lately, X-machines have been utilized to enhance or support agile software engineering techniques like use-case modeling [3] and acceptance test generation for extreme programming [2]. Their efficient application requires a formal model and a notation which is closer to the object-oriented paradigm which is usually used in these agile techniques. Another attempt to adapt X-machines to the object paradigm can be found in [10].

In this paper we present the XMDL-O language, an extension of the XMDL language. XMDL-O allows the specification of the memory element in an object-oriented flavour. The memory structure does not depend on the definition of functions or the skills of the developer but it is derived by standard object-oriented design techniques. Actually, a class diagram provides sufficient information for constructing the memory structure. Thus the modeler is not concerned with the representation of the memory but only with the definition of the classes, the state diagram of the system, and the definition of the functions. XMDL-O is more appealing to developers of object-oriented programs. The resulting specification is easily readable by someone who is acquainted with the usual object-oriented notation.

In section II, we briefly present X-machines. We propose a structured way for representing the memory element of an X-machine specification based on object-oriented concepts in section III. In section IV we present XMDL, a practical language for X-machine specification suitable for the development of tools. In section V, we present XMDL-O, an extension of XMDL that supports the ideas proposed in section III and

ISSN 1109-9305