# Modeling a tool for the generation of programming environments for adaptive formalisms

A.R. Camolesi

Departamento de Engenharia de Computação e Sistemas Digitais, Universidade de São Paulo, Brasil
Coordenadoria de Informática, Fundação Educacional do Município de Assis, Brasil
E-mail: camolesi@femanet.com.br

## Abstract

This paper aims to present the logical model that makes up the structure of a tool for the definition of environments for rule-driven adaptive formalisms.

## 1 Introduction

Adaptive applications need resources to adapt themselves to the environment's momentary needs and to foresee the internal and external demands, thus making up for a complex, robust, and fault-tolerant structure, yet flexible and responsive. Such applications offer modern capacities that are very difficult to be modeled by using present techniques of software development.

In order to solve the adaptive applications' modeling, it was proposed in [1] a generic formalism that allows (underlying) rule-driven non-adaptive devices to be extended to concepts of adaptive mechanisms. Such formalism is based on an Adaptive Mechanism (AM) that involves the kernel of an underlying non-adaptive device (ND). This way, an Adaptive Device (AD) is formally defined by AD = (C, AR, S, $c_0$, A, NA, BA, AA).

In this formulation C is the set of all the possible configurations of ND and c0 $\in$ C means its initial configuration. S is the set of all possible events that make up AD's entry chain and set A represents the acceptance configurations for ND.

Sets BA and AA are adaptive actions' sets. NA is a set of all symbols that can be generated with exits by AD, in response to the application of adaptive rules.

AR is the set of adaptive rules that define the adaptive behavior of AD and is given by the relationship $Ar \subseteq BA \times C \times S \times C \times NA \times AA$ in which adaptive actions modify the current set of AR adaptive rules from AD to a new AR set by adding and/or deleting adaptive rules in AR.

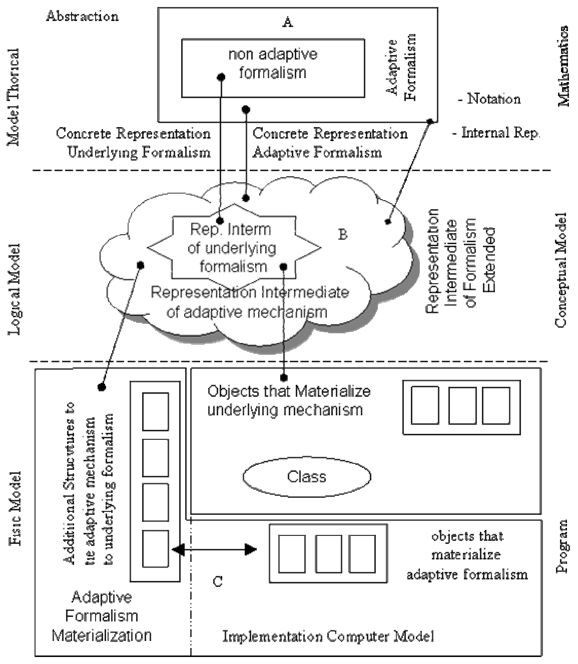
Based on these definitions, it is proposed in this paper a logical model for the representation of the formal elements shown in [1]. Such model is fundamental to the developing of tools that support a design methodology for adaptive applications. This paper is organized as follows: in section 2, the stages of extensions for adaptive devices and its use will be described. In section 3, the logical representation for adaptive devices is shown, and finally, in section 4, some conclusions and future papers are discussed.

## 2 Stages of extensions for non adaptive rule-driven devices.

When extending a formalism of an underlying device to the concepts of adaptive rule-driven mechanisms, a specialist should involve the non adaptive device with an adaptive layer. In order to develop this job, the specialist should possess good knowledge both of the underlying formalism and of the concepts of adaptive mechanisms. On the other hand, a planner that uses a device extended by a specialist does not need a formal knowledge as deep as the one needed by the specialist in extension of devices. The planner needs to know the extended specification language and how to use it in the project of his applications.

When extending a non adaptive rule-driven device to support adaptive characteristics there is the need to accomplish 3 stages: the stage of extension of the formal (mathematical) model, the stage of definition of the logical model and the stage of definition of the physical model. Figure 1 illustrates the stages and the existent relationship among them.

The stage of extension of the formal model Figure 1(A), offers a view in which a specialist with good mathematical knowledge of underlying formalism accomplishes the conceptual definition of the extended device to the concepts of adaptive mechanisms. In [1] and [2], extensions of underlying devices are presented to the concepts of adaptive devices. In this phase, the junction of the formal concepts of both (underlying and adaptive) formalisms is achieved, thus obtaining a new underlying device extended to concepts of adaptive mechanisms.



Abstraction
A
non adaptive formalism
Adaptive Formalism
Model Thorical
Mathematics
- Notation
- Internal Rep.
Concrete Representation Underlying Formalism
Concrete Representation Adaptive Formalism
Rep. Interm of underlying formalism
B
Representation Intermediate of adaptive mechanism
Representation Intermediate of Formalism Extended
Logical Model
Conceptual Model
Objects that Materialize underlying mechanism
Class
Additional Structures to tte adaptive mechanism to underlying formalism
Fisic Model
objects that materialize adaptive formalism
Program
C
Adaptive Formalism Materialization
Implementation Computer Model

Figure 1. Stages of extension of non adaptive rule-driven devices.

After obtaining the adaptive formalism, it is necessary a mapping of its concepts for an intermediate representation, as shown in Figure 1(B). Such stage consists of the definition of the logical structure that represents the formal concepts of the new adaptive device. Such structure is of fundamental importance, because it is part of the information storage structure necessary for the development of tools that will help the planner in designing adaptive applications.

In the stage of physical definition, as shown in Figure 1(C), a planner with knowledge of the developed adaptive formalism accomplishes the specification of his application. At this stage, yielded specifications are to be later analyzed and implemented.

When performing his work the planner instances the defined objects in the logical stage and he defines the physical elements that represent the behavior of the application.

In this phase, it can be observed that the instantiated objects belong to two different classes, i.e., the objects that represent the behavior of the developed application and other objects that represent the adaptive functions and actions responsible for modifications in the behavior of the application in execution. Based on the set of the defined objects in this phase, the presentation, the simulation, the verification and the execution of the projected application are allowed. During the simulation and execution process of specification in the adaptive kernel, adaptive actions can be executed and rules can be added or removed from the behavior represented by specification, thus modifying its structure.

In [3], a methodology was proposed to give support to the project of adaptive applications by using concepts presented in this paper. In Figure 2, it is shown the design methodology for adaptive applications formed by the following phases: specification phase, transformation specification phase, and validation and specification simulation phase. In the specification phase the application is accomplished by using either a text or a graphic tool. Soon afterwards, the transformation of the produced specification to an intermediate representation (logical model) is accomplished and, based on the obtained representation the planner can inform entry string sequences and evaluate its specification. If mistakes or inconsistencies occur, the planner can make changes in the specification and restart the process.

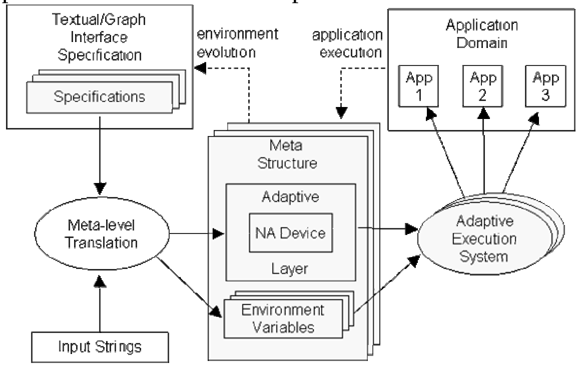


Figure 2. Methodology of Design of Adaptive Applications.

The proposed methodology is linked to the need to use tools for helping the planner in the performance of his job. During the specification phase there is the need of a text or graphic tool to aid the planner in the specification of an application. The phase of specification transformation of the application to an intermediate representation can be accomplished in two ways: automatic (generated by the editors at the moment of the edition), or through a translator that makes the transformation process after the specification process. And, finally, tools that allow the visualization, simulation and verification of the projected applications. In this phase, the planner, using an integrated environment informs the values regarding entry chain and submits their specification to the performer of the adaptive kernel. Initially, in case they exist, prior adaptive actions are performed, followed by elementary actions of the underlying device and finally the subsequent adaptive actions. This way, at each step the designer gets a new configuration (state of the system) and a new set of rules (behavior of the application) according to the adaptive actions that were performed. The obtained results should be displayed to the user, who can analyze them and, if necessary, make changes and restart the whole process.

## 3 Logical model for adaptive formalisms

Based on the concepts shown in [1], a logical model is proposed, so that it allows the construction of tools that help to plan adaptive applications. Such a model is represented by a data structure that gives support to the storage of the intermediate representation and allows the construction of a program that can manage the performance of the resulting specification by using the available facilities from adaptive devices.

In [3], a proposal was presented for the logical structuring of the formal definition of the concepts of adaptive devices. Figure 3 shows a diagram of entity relationship of the conceptual model for adaptive devices. Such a diagram is structured by objects of three types: Underlying Kernel (UK), Specification (S) and Adaptive Layer (AL) according to the characteristics they represent.

The objects of horizontal hachure (Device, Component Type, Connection Type and Attribute Type) are Underlying Kernel (UK) type and they correspond to the intermediate representation of the basic elements of an underlying device. In this structure, the conceptual elements of the underlying devices formally represented by set C are defined.

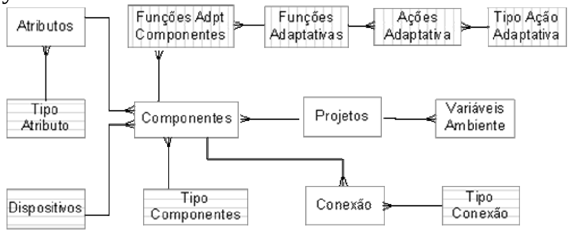


Figure 3. Diagram of entity relationship intermediate representation.

Solid color objects (Project, Attributes, Components, Connections and Variables Environment) are Specification (S) type and aim to represent the specifications yielded by a planner. Each object of this structure corresponds to elements of the formal definition, in which: each rule c that is part of the set of rules NR of an underlying device ND can be represented by the objects in S. The planner, when defining a specification, instances objects of the NS type (elements that constitute the underlying kernel) and defines the behavior of the application. This structure also stores the elements of set A that correspond to the rules of acceptance of an adaptive device and, furthermore, to the information on values of both the entry and exit chains in the Variable Environments object.

The objects with vertical hachure are Adaptive Layer (AL) type and they aim to provide the necessary resources to support the adaptive layer that involves the underlying kernel. The Adaptive Layer is structured in objects that correspond to the configuration of the adaptive device (Adaptive Action Type), and in objects that correspond to the AR conceptual elements that, in turn, correspond to adaptive functions and actions.

When defining the Underlying Kernel of a new device (Petri Nets, Automata, Grammar Free from Context, etc...) the specialist needs to store information related to the name of the device, the creation date and updating, etc… Such information is stored in the Devices object.

Information on the types of components (places and transitions of a Petri Net, final states, and non-final states of Automata, etc...) that represent the behavior of an application and that is used by a planner when specifying their application, can be represented by the Component Type object.

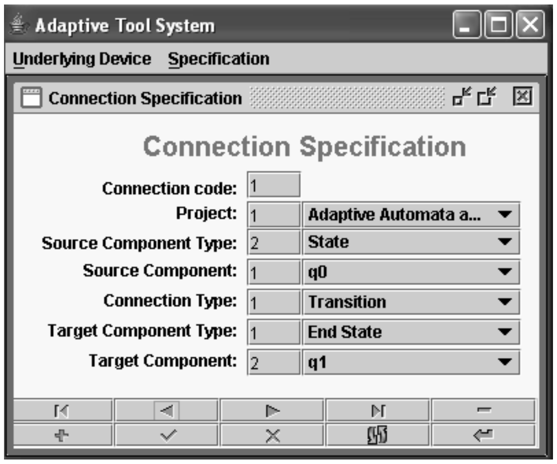
When specifying a rule that represents the behavior of an application it is necessary to represent the form of the existing connection between its components. The Connection Type object represents the information on the connection type for a device: transition for Automata, Petri Net connections, etc…, while the Attribute Type object contains information on the types of data that are available for attribution to a component of an application behavior.

When accomplishing the Specification of an application it is necessary to store information on the description of the specification, on the planner in charge, etc…. Such information is represented by the Projects object. At first, when defining the behavior of a project, one should define the components that constitute the application behavior. Such components are parts of the NR rules and they are represented by the Components object. One can mention the description of the states that constitute a specification of Automata or the description of the places and transitions of a Petri Net, etc… as examples of such components.

Following the definition of the components, one defines the rules (set c of the formal representation) that constitute the behavior of an application (formally acted by NR). Such structure establishes the relationship among the defined elements in both Component and Connection Type objects and defines the behavior of an application.

The value of each attribute associated to a (Component or Connection) object is represented by the Attributes object. The values of stimulus, and related information to the exit and other necessary information during

execution are acted by the Environment Variables object.
The Adaptive Layer is associated to the elements of specification of an application. This results, at first, in the definition of the information on the type of adaptive action that can occur: consultation action, insert or removal. Such information is stored in the Adaptive Action Type object.
When the adaptive mechanism is joined to the underlying kernel it is necessary to define the adaptive functions (the conceptual elements BA and AA) that should be associated to the elements of the Components object. The Adaptive Functions object allows the extension of the underlying kernel to have the features of adaptive mechanisms and it makes the connection between the elements of the underlying kernel and their respective adaptive actions that are represented by the Adaptive Actions object.
The Adaptive Actions object represents the set of adaptive actions belonging to AR that has the function of accomplishing changes in the behavior of the projected application.
Based on the logical structure, a tool is being developed that will allow a specialist to configure the conceptual elements of a non adaptive device and to accomplish its extension for the adaptive mechanisms. Such tool will also allow a planner, by using a textual language (intermediate representation), to develop the project of their applications.
In a second stage other tools will be developed that will allow the specification and display of graphic elements of the extended adaptive devices. The tool development is being made in Java [3] due to the portability and reuse features inherent to this programming language. Figure 3 shows the interface of the tool that is responsible for the definition of the connections of a specification.



Connection Specification

| | | |
|---|---|---|
| Connection code: | 1 | |
| Project: | 1 | Adaptive Automata a... |
| Source Component Type: | 2 | State |
| Source Component: | 1 | q0 |
| Connection Type: | 1 | Transition |
| Target Component Type: | 1 | End State |
| Target Component: | 2 | q1 |

Figura 4. Interface of an Adaptive Tool System.

## 4 Conclusion.

This work aimed to present how to make the extension of a non adaptive device to support the characteristics of adaptive mechanisms.
Initially, the general structure of an adaptive mechanism was presented, followed by the stages for the extension of a non adaptive formalism to support the characteristics of adaptive mechanisms. Following, the methodology for the design of adaptive applications was shown by using these concepts. Finally, a logical model was presented for the construction of tools that will give support to a design methodology of adaptive applications.
The proposed methodology was used in [2] to modeling of applications that has support the use of graphic interface and tools are being implemented to facilitate specialists and planners in their job with adaptive technology.
In relationship the stages of definition of adaptive formalisms several works were accomplished in relation to formal definition and as resulted adaptive formalisms were developed. Such works served as base for the definition of the extension stages for adaptive formalisms and they were to base the proposal of a logical model that it seeks to represent adaptive rule-driven formalisms.
The defined logical structure represents the conceptual elements for adaptive formalisms and it constitutes an intermediate representation for the definition of tools that it will support the methodology of design of adaptive applications.
As a continuation to this work, it is suggested a deeper study for the validation of the proposed logical model and the definition of a physical model (computational) for the validation of the proposed structure.

## References

[1] Neto J.J.(2001) Adaptive rule-driven devices - general formulation and case study, Sixth International Conference on Implementation and Application of Automata, Pretoria-South Africa.

[2] Camolesi, A.R. and Neto, J.J. (2004) Modelagem Adaptativa de Aplicações Complexas, XXX Conferencia Latino Americana de Informatica (CLEI), Arequipa, Peru.

[3] Camolesi, A.R. e Neto, J.J (2003) An adaptive model for modelling of distributed system, *Conference Argentina in la Ciência da Computacion (CACIC)*, La Plata, Argentina.

[4] Programming Language JAVA in http://www.sun.com (September 2004).