

An adaptive model for specification of distributed systems

Almir Rogério Camolesi^{1,2}

and

João José Neto¹

⁽¹⁾ Universidade de São Paulo, Escola Politécnica da USP
Departamento de Engenharia de Computação e Sistemas Digitais,
São Paulo-SP, Brasil, 05508-900

⁽²⁾ Instituto Municipal de Ensino Superior de Assis (IMESA), Coordenadoria de Informática.
Fundação Educacional do Município de Assis (FEMA)
Assis-SP, Brasil, 19807-634

almir.camolesi, joao.jose@poli.usp.br

Abstract

This paper presents the ISDL-Adp model, an architectural abstract self-modifying model for the specification of distributed systems. This model is the result of extending the concepts of non-adaptive ISDL devices by using the concepts of adaptive devices. This paper shows the basic structure by means of an illustrative example based on an adaptive hypermedia.

Keywords: ISDL, non adaptive devices, adaptive devices, adaptive automata, adaptive hypermedia

1. Introduction

Rady [1] says that computer technology has been developing fast, thus allowing its use in many different areas. Problems related to software development have increased and have become critical the more these applications embrace areas where a failure could lead to more drastic consequences, such as damages to human life and to property. This way, computer systems have become ever more complex. A growing number of new and sophisticated applications has been conducted in distributed machines that are interconnected through networks, that, in turn, have become even faster. The use of these applications in distributed environments makes the project an extremely complex task.

Due to the previously mentioned facts, specification and validation, including verification, simulation and test are of crucial importance in the lifecycle of the distributed systems, especially for the development of their applications. The specification cannot be ambiguous and it should be bug-free as much as possible. The existence of a successful integration of new products implies the performance of discerning tests related to the specification.

Current development methodologies present serious deficiencies in these areas. Frequently, informal specifications generate incompatible interpretations and usually, many errors are only detected after the release of the product. The later reasons justify the great effort that is being made for the use of formal models for the development of such methodologies.

Works have been developed aiming at the aid of specialists in the specification of their systems. The use of Formal Description Techniques (FDT), in the lifecycle of complex distributed systems, is the object of an increased interest from the involved communities. A proof of this interest is the great number of projects of the European Strategic Programme for Research and Development in Information Technology (ESPRIT) in the area of Formal Methods, such as the LOTOSPHERE Project [2].

Today, among the most commonly used FDTs for formal specification of distributed systems, the Extended State Transition Language (Estelle) [3] and the Language Temporary of Ordering Specification (LOTOS) [4] stand out. Both were developed and standardized by the International Standard Organization (ISO). Also, the Specification and Description Language (SDL) [5], developed and standardized by the International Telecommunication Union (ITU) stands out among these FDTs.

There are still non-standardized formal models (e.g., Nets of Petri [6], Statecharts [7], that are used in the specification and validation of distributed systems. [8] presents a group of architectural concepts that make up a formal model for the description of distributed systems. Supporting tools for this methodology can be found in [9]. [10] introduces the Interaction System Design Language (ISDL) that represents the main structures of the model. Such language and tools are being successfully employed in the construction of telematic systems and in the development of commercial projects.

Further studies, related to adaptive formalisms [11], [1] and [12], seek to use techniques and forms of representation of adaptive (dynamic) characteristics that are usually present in computation systems. Among several other reasons, the little use of the self-modifying formalisms is due to the complexity of existent notations, that turns its use a difficult one.

Neto [13] has presented a general proposal for the formulation of rule-driven adaptive devices. In spite of the inherent complexity of such devices, the proposed notation is clear, intuitive and easy to learn. The proposal is generic and it does not depend on the underlying non-adaptive formalism.

Stimulated by the previously presented works, this paper proposes the basic structure of the extension of the *Adaptive Interaction System Design Language* (ISDL-Adp) model. Such model aims at joining the ISDL model clearness and easiness [8][10] and the adaptive specification mechanism concepts (self-modifying). This paper is organized in the following way: section 2 briefly describes the structure of the non-adaptive ISDL device and its graphic representation. Section 3 introduces the ISDL-Adp model. The next section presents an illustrative example and finally, section 5 presents some conclusions and suggestions for future works.

2. The ISDL non-adaptive device

Between 1981 and 1986, specialists from the International Organization for Standardization (ISO), in its ISO/TC97/SC21/WG1/FDT Subgroup C developed the Language of Temporal Ordering Specification (LOTOS) [4], that performs a Formal Description Technique (FDT). This language was developed aiming to allow the specification and the development of protocols and open distributed systems. FDT LOTOS reached the status of International Standard ISO in 1989.

Based on the acquired experience in the development of tools for FDT LOTOS, a group of researchers from the University of Twente (Netherlands) developed a set of architectural concepts for the specification of distributed systems [8][10]. The ISDL model was originally developed to support the project of telematic systems, for example, services and protocols of the OSI model. A methodology for the project of services and protocols based on this model can be found in [14]. Camolesi [15], developed a methodology for the project of Interactive TV services by using the model in one of its phases.

Based on concepts presented in [13] the set of architectural concepts of the ISDL model forms a non-adaptive device driven by rules that behave exclusively according to a finite group of rules (actions and relations) that determine, in turn, for each possible configuration of the device, the next configuration. This way, the ISDL non-adaptive device is described as being a sextuple, and is formally represented by $ISDL = (Ac, RC, \Sigma, c_0, A, NA)$, where:

- ISDL is a non-adaptive device driven by rules, the operation of which defines a behavior constituted by a group of causality relations RC;
- Ac is a set of all possible action occurrences, and $c_0 \in Ac$ determines the set of actions of the initial behavior. The possible achieved behaviors are represented through the cross conjunction of their partial executions. The cross conjunction and the partial executions are formally denoted by \otimes and $e\chi$. Let us consider the cross conjunction of two partial behaviors $EE1$ and $EE2$ that form the EE behavior. EE consists of all possible executions $e\chi$, being $e\chi$ the conjunction of two compatible executions $e\chi1$ and $e\chi2$, being $e\chi1 \in EE1$ and $e\chi2 \in EE2$. Therefore, the EE behavior consists of all possible alternative constructions that meet the requirements of an $EE1$ (sub-) construction and an $EE2$ (sub-) construction at the same time.
- Σ is the finite set of all possible events that form the valid input strings to ISDL, being $\varepsilon \in \Sigma$;
- $A \subseteq Ac$ is the set of action occurrences;
- $F = Ac - A$ is the set of action non-occurrences;
- ε denotes “empty”, and represents the null element of the set to which it belongs, in relation to the concatenation operation
- $w = w_1... w_n$ is a stream of input stimuli, where $w_k \in \Sigma - \{\varepsilon\}$, $k = 1, \dots, n$ with $n \geq 0$;
- NA is a finite set (with $\varepsilon \in NA$) of all possible symbols to be generated as output by the ISDL mechanism. In practice, the output symbols in NA may be mapped into procedure calls, so an output generated by applying any rule may be interpreted as a call to its corresponding procedure;
- RC is a defined behavior defined by a set of behavior actions and interactions and their causality relations. A behavior represents a set of activities that the entity (an abstract concept that models a system, or part of a system) can perform. The action concept was introduced in order to represent an activity executed by a single system in a specific abstraction level.

Although an infinite variety of activities can exist, a single action concept can exist is enough to model all of them, as the essential characteristics of an activity are represented through three attributes of the action that models this activity: *information*, *time* and *location*. The attribute of information represents the result of the execution of the activity that is being modeled. The attribute of time represents the moment when the result becomes available, while the location attribute represents the place (physical or logical) where such result is available.

An action either occurs only once, or it doesn't occur at all. When it does, it means that the activity has successfully finished. An action is an indivisible unit of activity, at a certain abstraction level (an action is atomic). Therefore, it can be considered as a behavior unit.

An interaction represents an activity jointly executed by two or more systems. Consequently, an interaction is common to the entities that represent the involved systems. As they represent the activities, interactions carry the same attributes of actions. The difference between an interaction and an action is in its partakers; while an action is always executed by a single participant, the interaction is jointly executed by two or more partakers that can affect it in different ways.

An action and an interaction can be graphically represented by a circle and a semi-circle, respectively, both connected to a text box describing their related attributes. Figure 1a graphically represents a login action and Figure 1b illustrates a login interaction that had its functionality modeled into two different cooperative behaviors (user and system).

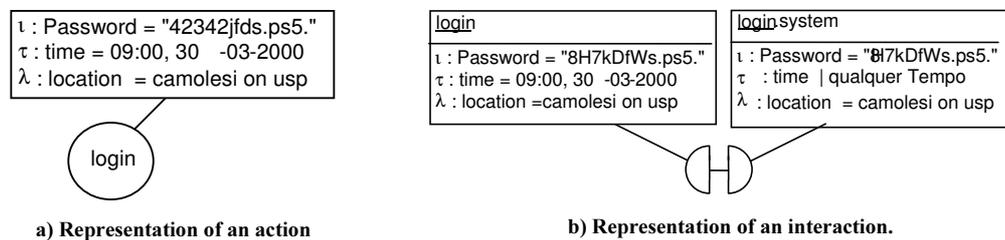


Figure 1- Graphic representation of ISDL action and interaction.

An action can either occur unconditionally, starting from the beginning of the execution of a behavior (the so-called *initial action*), or it can be started when a specific condition is met, which, in turn, may involve or not the occurrence of other actions during the execution of a behavior. A causality relation, that sets the condition for the occurrence of an action, is made up of:

- a *causality condition*, which defines the way the occurrence of an action can depend on the occurrence or on the non-occurrence of other actions;
- *action attributes' constrains*, which define how the values of information, time and location attributes of result actions, established by the actions of the causality condition, can influence in the occurrence of the action and the respective values of its attributes; and
- a probability attribute, which defines the occurrence probability of the result action when the causality condition and the action attributes' constrains are met at the execution of a behavior. Value 1 is taken for the attribute of probability, i.e., when the two previous constructions are met, the action must occur.

This way, the set of actions and interactions which define an ISDL behavior is given by a causality relation:

$\langle a, I, T, \Lambda, \zeta \rangle, \Gamma, \nu, \langle I\text{-Refs}, T\text{-Refs}, L\text{-Refs}, ITL\text{Refs} \rangle, \langle I\text{-Caus}, T\text{-Caus}, L\text{-Caus}, ITL\text{-Caus} \rangle$, where:

- $a \in A$ is the identification name of the action in the system and it belongs to A (set of action occurrences);
- I, T, Λ , respectively; are the information, time and location value domains of action a ;
- $\zeta \subseteq I \times T \times \Lambda$ is the mixed value domains of action a ;
- $\Gamma \in CC$ is the causality condition of action a and belongs to CC (set of disjunctive causality conditions);

- I-Refs, T-Refs, L-Refs, ITLRefs, respectively, define the set of information, time, location and mixed reference relations of action a ;
- I-Caus, T-Caus, L-Caus, ITL-Caus, respectively, are the set of information, time, location and mixed causality relations of action a ;

A causality relation allows the modeling of the temporal order of the occurrences of actions. If we take two actions a and b , which occur, respectively, in instants τ_a and τ_b the following basic conditions of causality can be defined:

- *enable* (a enables $b - \tau_a < \tau_b$) in which the occurrence of action a is a necessary condition for the occurrence of the action b .
- *disable* (a disables $b - \tau_a > \tau_b$), defines that the non-occurrence of action a until b takes place is a necessary condition for the occurrence of action b . Assuming that a occurs and that b still has not occurred, b is disabled;
- *choice* (either a or b must occur), defines a choice between a and b , such that one of both actions must occur. This choice is modelled as a mutual disabling. The corresponding textual notation is $\{-b! \rightarrow a, -a! \rightarrow b\}$.
- *interlace* (a and b can happen the same time), define that the action a and b can happen in parallel, not needing to be simultaneous.
- *synchronize* (a synchronizes with $b - \tau_a = \tau_b$), defines that the occurrence of action a is a condition for the occurrence of action b , so that a must happen simultaneously with b .

Figure 2 presents some causality relations commonly found between two actions a and b . Behaviors made up of multiple actions can be defined through the conjunction and disjunction of these relations.

Relation Between actions	Graphic Representation
a enable b	
a disable b	
a choice b	
a interlace b	
a synchronize b	

Figure 2 – Common relations between actions.

In some behaviors, multiple basic conditions involving different actions should be met so that a certain action can occur. Such situation can be represented through the conjunction of enabling, disabling or synchronization conditions involving different actions. In other behaviors, at least a basic condition or a conjunction of conditions should be met so that a certain action can happen. Such situation can be represented through the disjunction of two or more basic causality conditions or conjunctions of conditions.

A behavior and its actions can be represented formally in the following way:

$$[B] = * \otimes \{ [\rho] (B) \mid \rho \in B \};$$

$$[\langle a, \Gamma a, \nu a \rangle] (B) = [\langle a, \Gamma a \rangle] (B) \otimes [\nu a] (\Gamma a).$$

Symbols “[” and “]” represent the function that defines the (part of the) semantic of the execution of the causality relation, or the set of relations. For example, $[B]$, $[\langle a, \Gamma, \nu \rangle]$, $[\langle a, \Gamma \rangle]$, $[\langle a,$

$\gamma \}] e [\upsilon]$, respectively, represent the semantic for the execution of behavior B , the causality relation of action a , a 's causality condition, an alternative causality condition for a , and a 's uncertainty attribute.

3. Adaptive ISDL Device

Figure 4 shows the basic architecture of the ISDL-Adp model. Such device is made up of a non-adaptive ISDL kernel, enclosed by an adaptive layer. The adaptive layer is formed by the set of before and after adaptive actions. The adaptive actions aim to accomplish changes in the behavior of the ISDL specification. Prior to the execution of an ISDL non-adaptive behavior, the set of precedent adaptive actions must be executed, as these actions may modify the structure of the non-adaptive behavior. The ISDL non-adaptive actions follow and finally, the subsequent adaptive actions are executed.

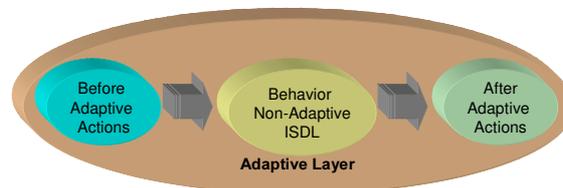


Figure 3 - ISDL Adaptive Device.

Taking as a base the general adaptive device presented in [13], the model ISDL-Adp can be defined by beginning its operation at the initial configuration c_0 in the form $ISDL-Adp_0 = (C_0, AR_0, \Sigma, c_0, A, NA, BA, AA)$. At step $k \geq 0$, an input incentive moves ISDL-Adp to the next configuration and its operation proceeds to step $k+1$ if and only if a non-null adaptive action is executed. So, as ISDL-Adp is at its step k , in the form $ISDL-Adp_k = (C_k, ARC_k, \Sigma, c_k, A, NA, BA, AA)$, the execution of a non-null adaptive action turns it into the form $ISDL-Adp_{k+1} = (C_{k+1}, AR_{k+1}, \Sigma, c_{k+1}, A, NA, BA, AA)$. In this formula:

- $ISDL-Adp = (ISDL_0, AM)$ is an adaptive device made up of an $ISDL_0$ initial subjacent device and an AM adaptive device;
- $ISDL$ is the subjacent non-adaptive device, that had its operation described in section 2, at step k . $ISDL_0$ is the subjacent device, defined in the initial set RC_0 (set of actions or causality relations that represent a non-adaptive behavior). By definition, any action or non-adaptive causality relation in RC_k has its corresponding adaptive rule in ARC_k ;
- C_k is the set of all possible ISDL behaviors at step k , and $c_k \in C_k$ is its initial behavior at step k . For $k=0$, we have, respectively, C_0 , the initial set of actions and valid causality relations and $c_0 \in C_0$, the initial $ISDL_0$ and ISDL configuration.
- ϵ ("empty") denotes the absence of any other valid element of the corresponding set;
- Σ is the (finite, fixe) set of all possible events(including ϵ) that are valid input stimuli to AD ($\epsilon \in \Sigma$);
- $A \subseteq C$ is the subset of both action and acceptance causality relations of the ISDL behavior;
- $F = C - A$ is the set of rejection configuration;
- BA and AA are both adaptive actions' set, that include the null action ($\epsilon \in BA \cap AA$)
- $w = w_1 w_2 \dots w_n$, is a string of stimuli, where $w_k \in \Sigma - \{\epsilon\}$, $k = 1, \dots, n$ with $n \geq 0$;

- NA , with $\epsilon \in NA$, is a (finite, fixed) set of all possible symbols that can be generated as output by ISDL-Adp as a side effect to the use of adaptive rules (actions and relations). Just like in non-adaptive devices, such output string may(whenever convenient) be interpreted as a sequence of the corresponding procedure calls;
- ARC_k is the set of adaptive rules that define ISDL-Adp adaptive behavior at step K . ARC_k is given by a ratio $ARC_k \subseteq BA \times C \times \Sigma \times C \times ISDL \times AA$.
- ARC_0 especially defines ISDL-Adp initial behavior. Adaptive actions change ISDL-Adp ARC_k current adaptive behavior to a new ARC_{k+1} behavior by adding and/or deleting adaptive actions and interactions in ARC_k . Rules $arc \in ARC_k$ have the form:

$\langle\langle ba \rangle, \langle\langle a, I, T, \Lambda, \zeta \rangle, \Gamma, \nu, \langle I\text{-Refs}, T\text{-Refs}, L\text{-Refs}, ITL\text{Refs} \rangle, \langle I\text{-Caus}, T\text{-Caus}, L\text{-Caus}, ITL\text{-Caus} \rangle\rangle, \langle aa \rangle\rangle$

meaning that, in response to some input stimulus $\sigma \in \Sigma$, arc initially executes the adaptive action $ba \in BA$. If the execution of ba eliminates ar from ARC_k , the execution of arc is aborted; otherwise, one uses the subjacent non-adaptive rule:

$arc = \langle\langle a, I, T, \Lambda, \zeta \rangle, \Gamma, \nu, \langle I\text{-Refs}, T\text{-Refs}, L\text{-Refs}, ITL\text{Refs} \rangle, \langle I\text{-Caus}, T\text{-Caus}, L\text{-Caus}, ITL\text{-Caus} \rangle\rangle \in RC_k$,

as previously described; and finally, the adaptive action $aa \in AA$ is executed.

- ARC is defined as the set of all possible adaptive rules for ISDL-Adp;
- RC is defined as the set of all possible actions and non-adaptive causality relations for ISDL-Adp;
- $AM \subseteq BA \times RC \times AA$, defined for a particular ISDL-Adp adaptive device, is an adaptive mechanism to be applied to all rules at step k in $RC_k \subseteq RC$. AM must be interpreted in the same way as if it were applied to any sub domain $RC_k \subseteq RC$. This will determine a single pair of adaptive actions associated to each non-adaptive rule.

The $ARC_k \subseteq ARC$ set can be obtained by collecting all adaptive rules built by the association of each pair of adaptive actions to the correspondent non-adaptive rules in RC_k .

ISDL_i non-adaptive behavior, at each step of the adaptive device's execution forms the ISDL-Adp model through its actions, its basic causality conditions and the relations between actions. An *ISDL-Adp* action can be represented by:

$\langle\langle a, I, T, \Lambda, \zeta \rangle : A, \rightarrow \langle\langle a', I', T', \Lambda', \zeta' \rangle, \Gamma', \nu', \langle I\text{-Refs}', T\text{-Refs}', L\text{-Refs}', ITL\text{Refs}' \rangle, \langle I\text{-Caus}', T\text{-Caus}', L\text{-Caus}', ITL\text{-Caus}' \rangle\rangle : B$

Where:

$\langle\langle a', I', T', \Lambda', \zeta' \rangle, \Gamma', \nu', \langle I\text{-Refs}', T\text{-Refs}', L\text{-Refs}', ITL\text{Refs}' \rangle, \langle I\text{-Caus}', T\text{-Caus}', L\text{-Caus}', ITL\text{-Caus}' \rangle\rangle$ stands for the situation of C_i behavior prior to the execution of an adaptive action. It is calculated according to the previous definition for ISDL behaviors;

Similar to the definition proposed in [1] for Adaptive Statecharts, ISDL elementary adaptive actions assume the following format: *preset* [*pattern of the production*], where *preset* represents one of the three types of elementary adaptive actions to be executed: "?" (inspection action), "-" (elimination action) and "+" (insert action), while the *pattern of the production* corresponds to an ISDL production:

$\langle\langle a', I', T', \Lambda', \zeta' \rangle, \Gamma', \nu', \langle I\text{-Refs}', T\text{-Refs}', L\text{-Refs}', ITL\text{Refs}' \rangle, \langle I\text{-Caus}', T\text{-Caus}', L\text{-Caus}', ITL\text{-Caus}' \rangle\rangle$

Figure 3 shows the action *Write_Lesson* modeled in ISDL-Adp and its previous(A) and subsequent (B) adaptive actions. The adaptive actions are inserted in boxes located above and below the traditional action of the ISDL model.

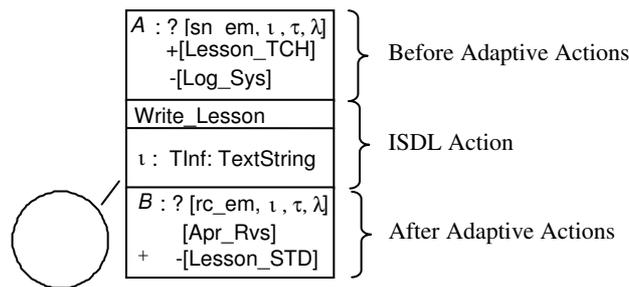


Figure 4 – Representation of Adaptive Actions.

The operation of the ISDL-Adp model happens through the modification of the actions belonging to a behavior, through the gradual evolution of the set of rules, by adding or removing actions made through the execution of the adaptive actions.

In its initial situation, ISDL-Adp receives a sequence of external events. The transitions are performed according to that sequence, by consuming the external events and repeating the process until the end of the sequence. At each received event a check is done to determine which production(or productions) is next in execution. As far as the ISDL-Adp execution is concerned, the existence of adaptive action *A* means that this will be the first to be executed, whereas if *B* is present, it will be the last to be executed.

4. An Illustrative Example

According to [16], the latter years of the 20th century saw one of the great triumphs of the human skill: the consolidation of the Internet as a shared virtual space of distributed architecture and global reach, where a growing variety of information, resources and services are available. The Internet topped the use of hypertext and hypermedia in the exchange of ideas, information and services, because its structural and navigation core is based on the hypertext notion.

Thus began one of the technologies that increased and made easier the interface between the user and the available information in computers [17]. Nowadays, thousands of documents are daily put into the WEB. However, those documents are prepared according to a single style, thus ignoring the great variety of users that use them. Studies have been made, in an attempt to supply the users' specific demands, i. e., the systems must adapt themselves to the users' needs and profiles.

According to [16], the Adaptive Hypermedia has lately been the object of a great interest, thus increasing the number of events worldwide where this matter is a highlight. Several areas of Adaptive Hypermedia have been identified, such as e-commerce, marketing, education, information systems, advertising and leisure. [18], shows that the educational hypermedia area leads in research and applications It is fueled by Distance Learning and Information Systems.

[19] presents a proposal for the development of an adaptive course on computer concepts, a self-teaching adaptive course that uses Adaptive Hypermedia adaptation methods to offer a larger interaction to the student. In this context the use of ISDL-Adp model is proposed for the modeling of this course.

Figure 5 [19] shows the “Welcome scene” of this course. Such scene is introduced to the user after his registration in the system, or after a successful login. If it is the first time the user access the adaptive course on computer concepts, he can go to the “Start course” option. Users that have already began the course, can go to a “Last session” shortcut that directs them to resume the course from the point they had left. They do not need to go through previously studied modules. The system also shows a summary of previously studied subjects in the visual format of a traffic light.

The traffic light shows the current situation (formalized, non-formalized, in process) of each studied module. The green light indicates that the module was successfully finished; the yellow light indicates that the module is in process and the red light shows that the module has not been initiated yet.

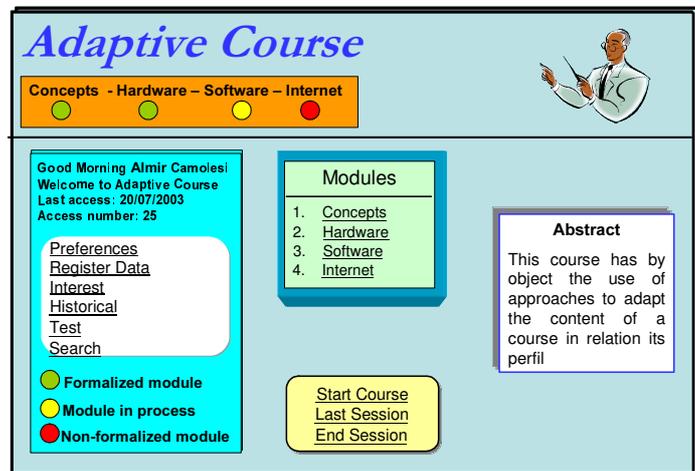


Figure 5 - The Welcome Scene from the Adaptive Course on Computer Science.

The Welcome Scene sub-behavior is shown in Figure 6; this diagram represents the ISDL-Adp modelling of the Welcome scene. The Welcome Scene sub-behavior is composed up of the parallel (interlace) execution of actions *Shw_Title*, *Shw_Mascot*, *Shw_Abstract*, *Shw_Menu*, *Shw_Button* and *Shw_TrafficLht*. Such actions are responsible, respectively, for the presentation of title of scene, the presentation of the responsible mascot for aiding the user in the accomplishment of the course; the introduction of the label with information on the abstract of the course that is being taken, introduction of the course's option menu, presentation of the key that sends the user to the section (lesson) to be performed, and introduction of the traffic light that indicates the current situation of each module of the chosen course. Actions *Shw_Button* and *Shw_TrafficLht*, have precedent adaptive actions. The preceding adaptive actions along with the *Shw_Button* action enable the user to check which of the course's corresponding scene has been executed. Thus, these joint actions allow the execution of the module's first lesson or the last lesson done by a student. The preceding adaptive actions along with the *Shw_TrafficLht* action are responsible for the verification of each module's situation in the course taken, and, this way, the adaptation of the interface can be done, by showing the current situation of each module of the accessed course.

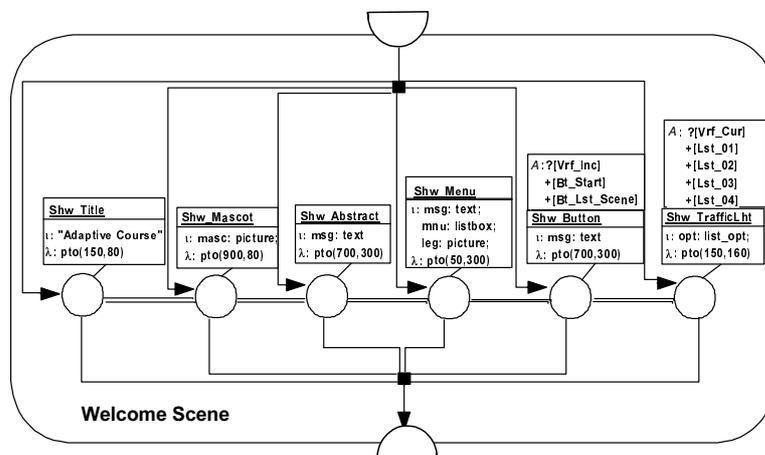


Figure 6 - ISDL-Adp Modeling - Welcome Scene.

Figure 7 show the new behavior *Welcome Scene* after running before adaptive actions. The action *Shw_Button* went to change to present the last scene that a student used and the last class that the student was doing. The action *Shw_TrafficLht* had changed to show which modules that student finished, non started or it is studing. The *lst_01* until *lst_04* represent this situation. The value equal 1 (green) indicates that student finished the module, the value 2 (yellow) illustrates that student was making this module and the value 0 (red) represents that he didn't start this module. It can be observed in this behavior the adaptive before action were to used to represent the changes in behavior of this scene.

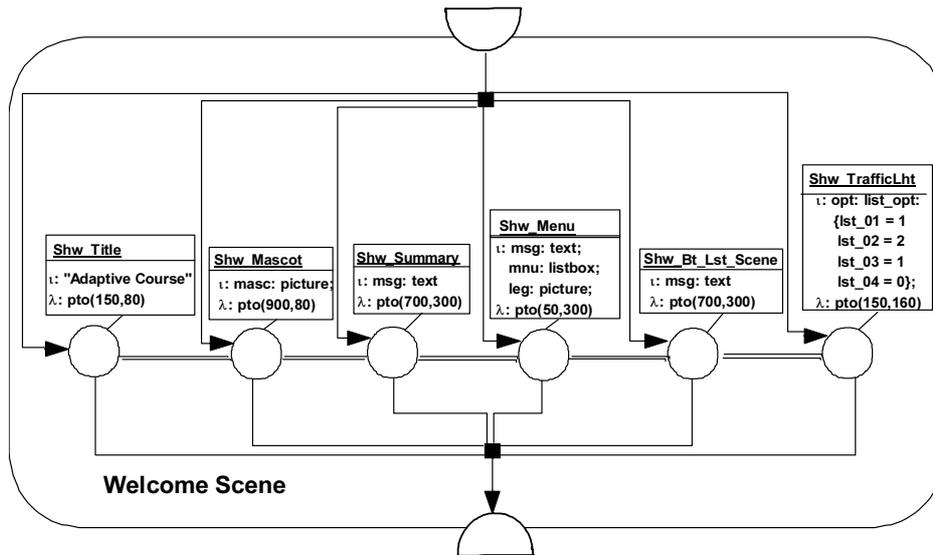


Figure 7 - Welcome Scene after running before adaptive actions ISDL-Adp.

5. Conclusion

This paper has shown the basic structure of the ISDL-Adp Model that is being developed for the specification of adaptive distributed systems at modeling of Adaptive Hypermedia. Such model results from joining the concepts of a non-adaptive ISDL model and the concepts of adaptive devices.

Just as in the ISDL model, in the ISDL-Adp model an entity models an object and features its behavior (methods) and state (attributes). That is the reason why a system can be structured, for example, as a hierarchy of abstractions (entities) and the behavior of entities can be encapsulated, as these are only accessible through the defined interaction points in its interface.

The definition of adaptive actions in the structuring of behaviors, in turn, allows the use of the addition and of the optimization of actions and causality relations in a dynamic way. This makes the ISDL model more flexible and one with a larger expression power for problems that show self-modifying characteristics. When using the ISDL-Adp model the specialist can model behaviors that have their structure modified during the execution of the system. The adaptive resources allow the insertion or elimination of actions and causality relations on each execution of the behavior, thus making it possible, with the aid of this model, the _expression of applications with great representation complexity.

The illustrative example shown above referring to the modeling of a Adaptive Course of Introduction Computer Science, was intended to show the use of the model in the project of these applications. Such example proves that the model can take the representation of specifications that have adaptive features, as shown in the specification of the *Welcome Scene* sub-behavior.

As a continuation of this work, adaptive extensions are being formalized for all the architectural concepts of the ISDL model. Such extension can turn the project of its specifications a much easier task for the specialist. Furthermore, specifications with larger consistence and easier understanding will be able to be achieved. Studies are also being done regarding the development of a tool that will use the new model to aid the specialist in the development of his specifications.

Bibliography

- [1] J. Rady, "STAD-Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos", Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 1995.
- [2] T. Bolognesi, J. Lagemaat, and C.A. Vissers, "LOTOSphere: software development with LOTOS", Kluwer Academic Publishers, Netherlands, 1995.
- [3] ISO/IEC 9074, "Information Processing Systems - Open System Interconnection - Estelle - A Formal Description Technique Based on an Extended State Transition Model", ISO, 1989.
- [4] ISO IS 8807, "Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", ISO, 1989.
- [5] ITU-T, "Recommendation Z.100", White Book and Annexes, 1994.
- [6] T. Agerwala, "Putting Petri nets to work", Computer, v.12, n.12, December, 1979, p.p. 85-94.
- [7] D. Harel, A. Pnueli, J.P.Schmidit, and R. Sherman,. "On the formal semantics of statecharts", In: Symposium on logic in Computer Science, 2º, Ithaca, Proceedings, IEEE Press, New York, 1987, p.p.54-64.
- [8] D. Quartel, "Actions Relations – Basic design concepts for behaviour modelling and refinement". Ph.D Thesis, Netherlands, 1997.
- [9] Testbed Studio Tool, <http://www.bizzdesign.com>, 2003.
- [10] C.G. Farias, and L.F. Pires, "Uma linguagem para desenvolvimento de sistemas distribuídos". XIX Simpósio Brasileiro de Redes de Computadores, Florianópolis, Maio, 2001, p.p. 82-97.
- [11] J.J. Neto, "Contribuição à metodologia de construção de compiladores. Tese de Livre Docência", Escola Politécnica, Universidade de São Paulo, São Paulo, 1993.
- [12] M. Iwai, "Um formalismo gramatical adaptativo para linguagens dependentes de contexto", Tese de Doutorado. Escola Politécnica, Universidade de São Paulo, São Paulo, 2000.
- [13] J.J. Neto, "Adaptive rule-driven devices - general formulation and case study", CIAA 2001 - Sixth International Conference on Implementation and Application of Automata, Pretoria University, Pretoria-África do Sul, July,2001.
- [14] C.A.Vissers, L.F. Pires, D. Quartel, and M.V. Sinderen, "The architectural design of distributed systems", Lecture notes for The design of telematic systems, University of Twente, Enschede-Netherlands, 1998.
- [15] A.R. Camolesi, "Uma metodologia para o Design de Serviços de TV-Interativa", Dissertação de Mestrado, PPG-CC, UFSCar, Fevereiro, 2000.

- [16] L.A.M. Palazzo, “Modelos Proativos para Hipermedia Adaptativa”, Tese de Doutorado. Universidade Federal do Rio Grande do Sul, Instituto de Informatica. Porto Alegre (RS), janeiro de 2000. Disponível em: www.inf.ufrgs.br/~palazzo/docs/Artigos/00_XISBIE.pdf
- [17] B.R. Marques, L.F. Melo, M.T. Chella, and T.R. Quirino, “Tecnologias Para Ambientes Colaborativos de Ensino – Hipermedia”, Faculdade de Engenharia Elétrica e de Computação – FEEC. Universidade Estadual de Campinas – UNICAMP, janeiro de 2001. Available: www.decom.fee.unicamp.br/~leonimer/hipermidia.html
- [18] L.A.M. Palazzo, “Sistemas de Hipermedia Adaptativa”, XXI Jornada de Atualização em Informatica. SBC 2002. Florianópolis-SC, Julho, 2002.
- [19] E. Hayashida, “Uma proposta para um curso adaptativo de Introdução à Informatica”, Monografia apresentada no curso de Especialização Desenvolvimento de Software para Web, Instituto Municipal de Ensino Superior de Assis, Junho, 2003.