

E-DART: A Specification Environment to the E-LOTOS Formal Technique

Lisandro Zambenedetti Granville and Maria Janilce Almeida

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Curso de Pós-Graduação em Ciência de Computação
Campus do Vale, Bloco IV – Av. Bento Gonçalves, 9500 – CEP 91591-970
Porto Alegre, RS - Brazil
{granvile, janilce}@inf.ufrgs.br

Abstract. This work presents the E-DART (*Enhancements to DART*), a graphical environment for the specification of systems based on graphic constructions for the formal description technique E-LOTOS. The use of diagrams reduces the complexity of the specifications turning E-LOTOS accessible even to users not familiarized with a formal description technique. A new graphic syntax was created as well as a specification tool, the E-DART Editor, that supports the new syntax. A translating module, from graphic specifications to the E-LOTOS syntax, is also part of the environment. The main purpose is to provide a new way to allow the user to specify time-dependent systems in a graphical approach, abstracting the largest complexities of the E-LOTOS textual syntax.

1 Introduction

In the 80's several efforts were made trying to create a technique to formally specify computational systems. From these efforts, three main techniques appeared: LOTOS, SDL and ESTELLE. Since then many changes and improvements have been incorporated to them, mainly respecting their easiness of use.

SDL (*Specification and Description Language*) [1] quickly became popular in the industrial environment because SDL is easy to understand and doesn't demand deep knowledge of the language. A large part of the SDL success is due to the existence of a graphical syntax for the language, what turns the systems specification process quite simpler and faster.

LOTOS (*Language Of Temporal Ordering Specification*) [2] and ESTELLE (*Extended Finite State Machine Language*) [3], on the other hand, had more success in the academic environment. In spite of having a strong formal base, such techniques didn't have success in the industrial environment because they are difficult to understand. For that reason the use of the FDT (*Formal Description Technique*) is limited.

Among the three techniques mentioned, LOTOS is the most powerful one and due to that the most complex too. With a strong mathematical base and an exclusively

textual syntax, LOTOS doesn't come as the most popular FDT. Several efforts to change this situation have been made. Graphical representations try to abstract the complexities of the textual syntax, while the temporal extensions of the language try to introduce the treatment and manipulation of time-dependent actions.

This work is based on the E-LOTOS (*Enhancements to LOTOS*) proposal of ISO, now still in draft proposal. The tool showed in this work supports a large part of these improvements through advanced functionalities in the user interface. The section 2 discusses the need of a temporal extension and the E-LOTOS proposal. In the section 3 the G-LOTOS and DART graphical extensions are described. The section 4 presents the E-DART environment. The section 5 exemplifies the use of E-DART through a specification example and the section 6 describes the E-DART Editor functionalities.

2 The LOTOS Temporal Extensions

Systems in LOTOS are defined through a group of processes that exchange information with each other. The most critical actions in LOTOS are the synchronizations. Such actions occurs when a LOTOS process, through synchronization channels, communicates with other process. This communication between two LOTOS processes is called synchronization. If a synchronization is not possible, then it is said that the related processes finished with no success. In LOTOS, a synchronization doesn't have duration and it is said instantaneous. However it doesn't happen in the practice when the LOTOS specifications become implemented systems, because the communications are not instantaneous, many times lasting a considerable time until its end.

The expression “temporal extension” refers to an improvement in the LOTOS language that facilitates the manipulation and the treatment of the time in the specifications. Several temporal extensions were proposed: Temporal LOTOS [4], TIC-LOTOS [5], T-LOTOS [6], RT-LOTOS [7] and E-LOTOS [8] are examples. The main purpose of all of them is to supply a mechanism able to measure the time elapsed in the LOTOS events. Thus, the synchronizations are not instantaneous anymore, and the time elapsed in a synchronization can be measured. In a general way, the main constructions incorporated to LOTOS by the temporal extensions are:

- Operator able to indicate the elapsed time of a synchronization. With such operator it is possible to determine how long a synchronization can take. It is possible to implement time-out and minimum synchronization time policies;
- Delay operator. The delay operator suspends the process for a defined period of time, forcing a delay in the execution.

E-LOTOS is the result of the ISO efforts in the revision of the norm that describes LOTOS. Now there is a version, in draft proposal form, for the language improvements, where the main changes presented are: the addition of the temporal semantics, the change in the representation of abstract data types, the introduction of new operators and improvements in the behaviors representation. The improvements presented in E-LOTOS, besides presenting the solution for the temporal problems,

improve the understanding of the language and turn LOTOS more modern, incorporating for instance, object-orientation concepts and exceptions handling.

3 The LOTOS Graphical Extensions

The main purpose of the LOTOS graphical extensions is to decrease the complexities of the textual language, allowing the user to abstract the most difficult aspects of the language. Two main graphical syntaxes were important for the construction of the E-DART environment: G-LOTOS and DART.

The ISO definition of a graphical language to LOTOS originated G-LOTOS (*Graphical LOTOS*) [9]. The G-LOTOS syntax has been built from the direct mapping of the LOTOS structures to graphic diagrams. So, the syntax is able to represent any LOTOS construction. However, the legibility of G-LOTOS is harmed by the great symbols variety and the way they relate to each other. One can say that the textual complexities were all transferred to the new diagrams.

DART (Diagrams for Architectural Representation) [10] presents an approach where the main focus is the graphical definition of processes and the way they interact. This approach is much closer to the way the specifications are built using the textual syntax. The specification processes are defined together with its synchronization gates. Then, the interaction among these processes is built using the language operators. Finally, the internal behavior of each process is defined.

This approach allows a clearer visualization of the system that is being specified. However, DART doesn't have graphics corresponding to all the LOTOS constructions. Thus, not all the LOTOS systems can be represented by DART. The following figure shows the relationships between DART and LOTOS.

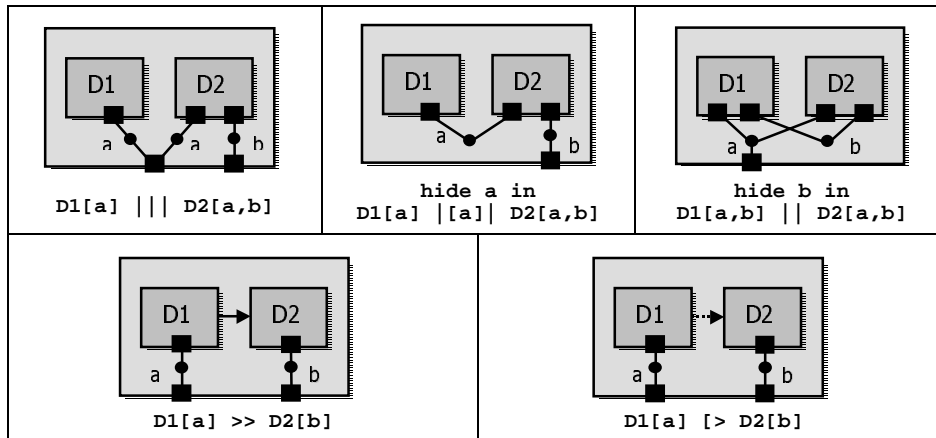


Fig. 1. Relationships between DART and LOTOS

4 The Proposed Environment

As previously shown, G-LOTOS and DART are important graphical syntaxes, but they don't satisfy completely the need of a textual syntax simplification. G-LOTOS supports all the LOTOS operators, then it is complete. However the presented diagrams are complexes due to the direct mapping of the textual syntax. DART, on the other hand, is more intuitive but it doesn't have elements corresponding to all the structures. That fact prevents the specification of more complex systems. Such situation requires the creation of a new graphical version. The syntax proposed is called E-DART, in analogy to E-LOTOS and for being largely based on DART.

This way, the proposed environment is formed by: the new graphic syntax E-DART, an editor that supports this syntax - the E-DART Editor, and a translator from the graphical diagrams to the textual syntax of E-LOTOS.

The new syntax is not based exclusively on graphics, but also in other forms of showing information offered by the created environment. More specifically, each symbol can have a table of properties. Thus, the syntax and the proposed environment cannot exist separately, since one complements the other.

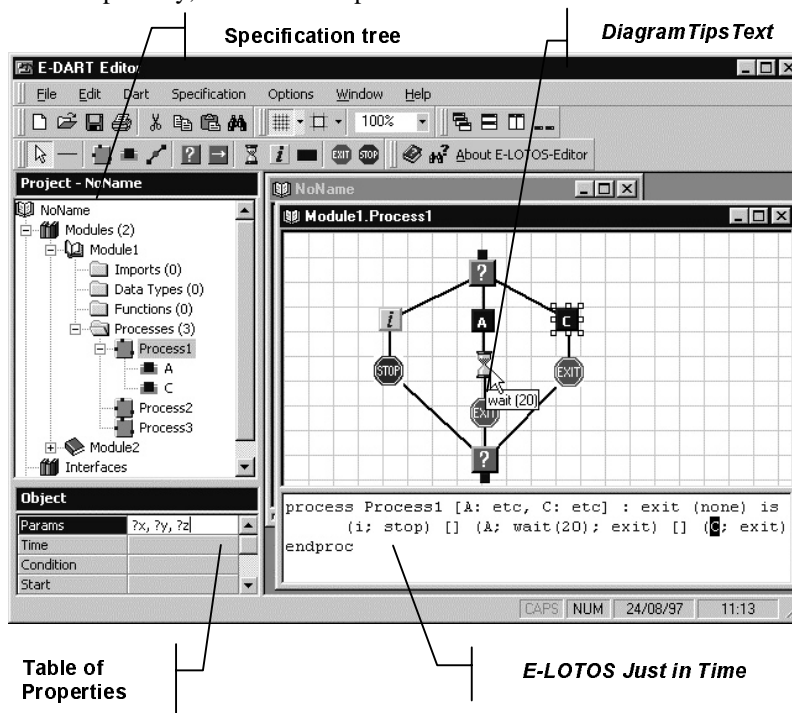


Fig. 2. E-DART Editor user interface

In the figure 2 we can observe the graphical interface of the proposed environment, supported by the E-DART Editor, with all its most important elements. From now on, the main E-LOTOS constructions will be described according to their respective supports in the tool.

Modules, interfaces and generic modules are elements introduced in E-LOTOS that don't have correspondent graphics. Such elements exist to turn a specification modular. Besides, they are mechanisms for code reusing in new specifications. Although they have no graphic representation, the editor supports such constructions through a specification tree.

We can observe that the tree organizes the specification in the three main elements. Each module presents folders for the data types, functions and processes of the module. The proposed syntax has its focus turned to the specification of the system behaviors in a graphical way. The data types and functions, therefore, continue being specified textually through an ordinary text window. In the future, the functions should also be specified in a graphic way, as well as the data types.

The system processes can also be accessed through the specification tree. Each process has a graphical window for the specification of its dynamic behavior. The gates, as well as the parameters, are created and managed in each process sub-tree. The process has a table of properties with its name and the return type.

The definition of the dynamic behavior of a process is made through the insertion of graphic symbols in the process window. As well as the process, each one of these diagrams also has a table of corresponding properties (figure 2).

In order to instantiate a process an instantiation frame should be defined. Such frame corresponds to a rectangle in which borders are the gates of the current process. Inside of the instantiation frame the instantiated processes are then placed. These processes are represented by rectangles similar to the instantiation frame. The gates should be linked by synchronization points that allow the definition of the synchronization among the instantiated processes. In the following figure, on the left, we observe the instantiation of the process "line".

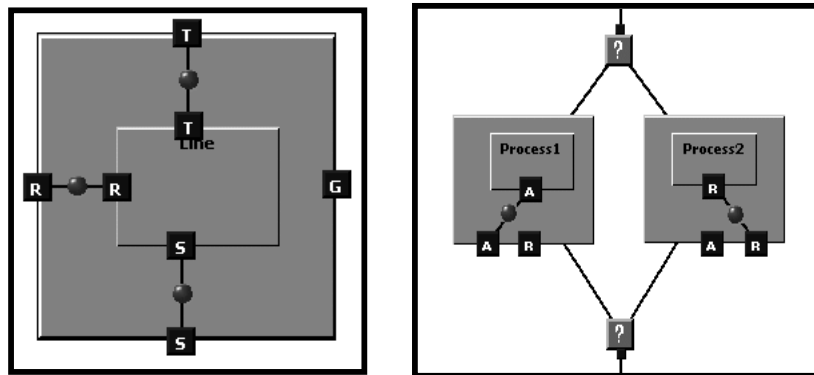


Fig. 3. Process instantiation and choice operator

The choice operator is represented by two similar symbols. One indicates the beginning of the operator and the other the end of a choice. In the figure above, on the right, we can observe the choice operator with its two diagrams. In this example we have two processes instantiations: *Process1* and *Process2*. The instantiated process will be the one that first offers an event to the environment.

The preemption operator is similar to the choice operator, except that in the preemption only two branches of graphics can exist between the beginning and the

end symbols of the preemption. The right branch is executed and can be interrupted by the left branch.

The wait operator is represented by an hourglass and its delay time is defined in the table of properties. The non-observable internal action operator i is represented by a rectangle with the letter i inside of it. Such operators can be observed in the following figure.

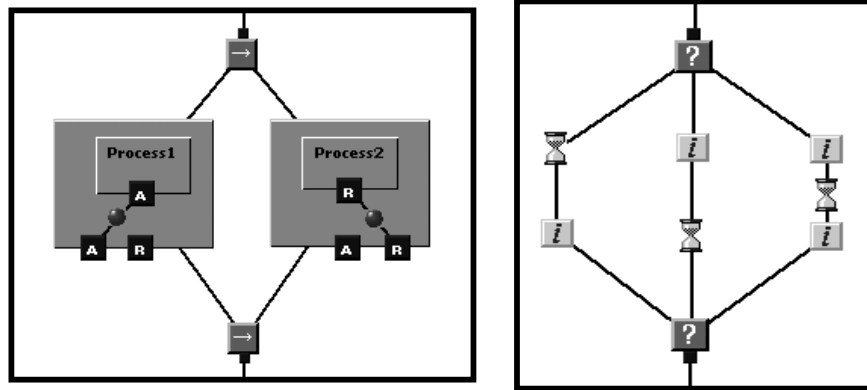


Fig. 4. Preemption, Wait and Internal Action operator

The synchronization is represented by the same graphic symbol which identifies the gates of a process. However such symbol isn't linked to any processes instantiation frame. The table of properties of a synchronization presents the necessary data to specify a synchronization according to the new E-LOTOS syntax. Finally, the sequence is represented by a line that links two successive symbols.

5 An Example of E-DART Specification

This section presents the HTTP protocol specified in E-DART. The example models the client and server applications defining processes that exchange HTTP information.

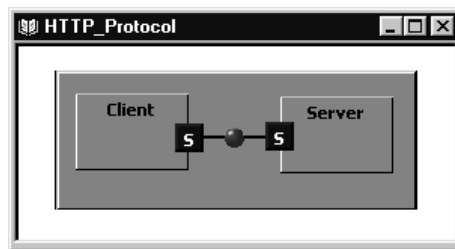


Fig. 5. Interaction between *Client* and *Server* processes. Both processes communicate each other through the synchronization gate S . This figure shows the highest system abstraction level

Two modules define the client and server entities. *WebServer* defines the following types: HTTP messages, HTML pages and time units for the timeout control. The

function *Build_Reply* is responsible for generating the answers to the requests sent by the clients.

In the *WebClient* module there are two functions: *Build_Request*, that builds a request message that will be sent by the client to the server and *Extract_HTML_Page* that is responsible for separating the HTML pages from the HTTP messages sent by the server to the client. We must observe that the *Imports* branch of the Client module indicates a reference to the Server module, what allows the client module to have access to the data types, functions and processes of server module.

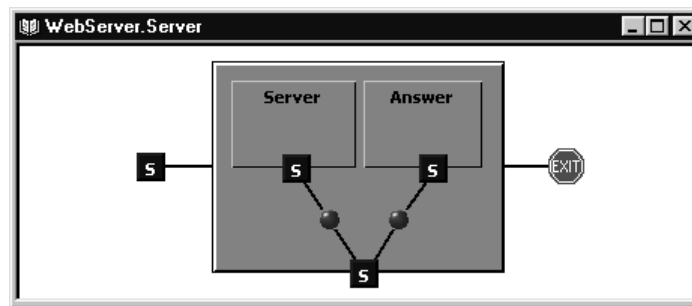


Fig. 6. The *Server* process. It waits indefinitely for a synchronization in the gate *S*. When that happens, a new *Server* process is instantiated, allowing new requests to be answered. *Answer* process, also instantiated by *Server*, indeed answers the client requests

The *Answer* process firstly generates a return message that is sent to the client through a new synchronization on *S*. We must notice that this last synchronization has a timeout indicated by the client, when the request is sent. The variable *t* controls the elapsed time of the synchronization.

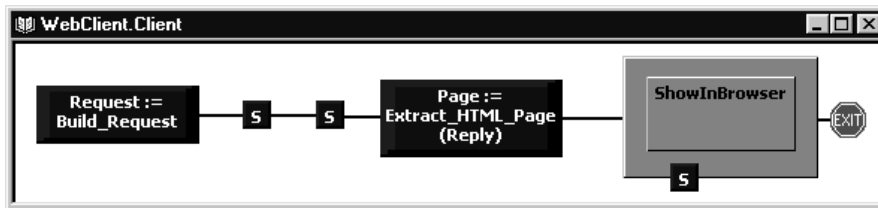


Fig. 7. The *Client* process. It generates an HTTP request and sends the message to the server, together with a timeout value, through a first synchronization. Then, another synchronization is waited by the client, when the answer to the previous request shall return. From this answer, an HTML page is extracted and forwarded to the process *ShowInBrowser*, which finally shows the HTML page to the user

We must notice that the time of all the synchronizations made by the client are controlled, while in the server it only happens in the answers synchronizations. It happens because the first message sent to the server has an uncertain duration time and it can occurs at any instant.

6 Functionalities of the E-DART Editor

The E-DART Editor is a tool where an E-LOTOS system can be specified through the elements of the previously presented E-DART graphical syntax. The natural deficiencies that a graphical syntax comes to present are lived up by the functionalities of the environment. The common functionalities of a programming environment are present in the editor: the creation, opening and management of projects for instance. The graphics manipulation functionalities are present too.

The whole editor is based on the user interaction through context-sensitive menus. At any moment one can click with the right button of the mouse on an element of the editor and a menu will appear, showing which operations can be executed on the clicked element. Optionally, the user can also use the conventional menus present in the top part of the interface.

The process of specifying a system is constituted mainly by the definition of the system modules. Each new module comes as a folder in the specification tree. In such a folder new data types, functions and processes can be created. The creation of a data type or function makes a conventional text window to be presented to the user, where the new type or function can be specified. The window can be normally closed and accessed again by clicking twice in the element in the specification tree. The processes are created in a similar way. For each created process a new sub-tree is also presented, where then we can specify the gates, parameters and local variables of the process.

The specification modules can be stored in independent files. The specification as a whole is constituted by a main file plus the group of files that contains the modules of the specification.

The independent storage of modules gives an important characteristic to the environment. Modules already tested, debugged and homologated by other specifications can be reused without the need of new tests. That guarantees quickness in the construction of new specifications besides increasing its reliability, because they are based on already tested components.

An important characteristic of the environment is the fact that the users don't need to know E-LOTOS deeply to specify complex systems. That is reached with the use of the E-DART graphics and the facilities of the environment. However, users familiarized with E-LOTOS nevertheless have instantaneous access to the textual syntax corresponding to the graphics through the functionalities of the environment.

The editor presents the functionality *DiagramTipsText*, that supplies a textual E-LOTOS translation for a specific E-DART graphic element. In a conventional programming environment this functionality allows the user to know the functions of each button of the environment.

The editor also has the resource *E-LOTOS Just in Time*. Through a text window, placed just below the E-DART graphics area, it is possible to visualize the complete E-LOTOS code for the process. In this window it is possible to copy the text to the system clipboard, so allowing the edition and manipulation of specific parts of code using other tools.

Finally, the environment is able to generate E-LOTOS code for a complete specification. The generated code is in the form of ASCII text files and they can be used as input for other E-LOTOS validation or test tools. The generation of the

complete E-LOTOS code is done by selecting E-LOTOS files (*.els) as storage format instead of the standard E-DART files (*.eds).

7 Conclusions

This work describes the E-DART environment, composed by a new graphical syntax for the E-LOTOS FDT, a tool to design the specification of systems using this new syntax and a translator to E-LOTOS textual syntax. The main benefit of the environment is to turn the specification of E-LOTOS systems more intuitive and easier to understand. That will allow E-LOTOS to be used by a larger number of users, turning it more popular.

The need of temporal and graphical extensions for LOTOS was presented. The E-DART uses structures that deal explicitly with the notion of quantitative time, allowing the specification of time-dependent systems. Since E-DART is graphic-oriented, it allows a larger and better interactivity among E-LOTOS and the users that specify their systems. With E-DART, many of the textual complexities are abstracted and the understanding of the specifications becomes easier, what nevertheless allows non-expert professionals to build complex specifications using such a powerful FDT.

To allow the use of the new graphic syntax created, the E-DART Editor was developed. It is a graphic tool that supplies the necessary functionalities to turn the specification of systems in E-DART possible. Through the several functionalities of the tool, like the mechanism of modules reusing, *DiagramTipsText* and the *E-LOTOS just in time* functionalities, the systems specification process can be simplified, providing less development time and a better quality of the specifications.

The E-DART graphical syntax and the E-DART Editor tool integrates the DAMD, an environment to develop distributed multimedia applications, where it is possible to validate, animate and execute the systems specifications. Such integration will allow the specifications built with the E-DART to be validated by other tools. Besides, DAMD will facilitate the generation of source code to the specifications created with the E-DART.

The future works related to the environment involve the development of a E-LOTOS to E-DART translator, that allows the automatic creation of diagrams from a textual specification. The translator will do the reverse process to the current one: besides generating textual specifications from graphics, it will be possible to generate graphics from textual specifications. A modules browser will also be developed. It will allow the cataloguing and the management of modules created by the user. New modules created by other users could be incorporated and the browser will be responsible for the creation and maintenance of modules libraries.

References

1. Saracco, R. e Tilanus, P.A.J. CCITT SDL: Overview of the language and its applications. *Computer Networks and ISDN Systems*, 13(2):65-74, 1987.
2. Bolognesi, Tommaso e Brinksma, Ed. Introduction to the ISO specification language LOTOS. In Chris A. Vissers Peter H.J. van Eijk e Michel Diaz, editors, *The Formal Description Technique LOTOS*, pg 303-326. Elsevier Science Publisher B.V. North-Holland, 1989.
3. Budkowski, S. e Dembinski, P. An introduction to ESTELLE: A specification language for distributed systems. *Computer Networks and ISDN Systems*, 14(1):3-24, January 1987.
4. Regan, T. Multimedia in Temporal LOTOS: a Lip-Synchronization Algorithm. *IFIP'93 - Protocol Specification, Testing and Verification*, pg. 127-142, 1993.
5. Azcorra, A.; Quemada, J. e Frutos, D. A Timed Calculus for LOTOS. In *Proceedings of the FORTE'89*, pp. 245-264, Vancouver, Canada, 1989.
6. Bolognesi, T. Lucidi, F. e Trigila, S. Converging towards a Timed-LOTOS Standard. 42 pg., 1993.
7. Quemada, J. e Fernandes, A. Introduction of Quantitative Relative Time into LOTOS. In *IFIP Workshop on Protocol Specification, Testing and Verification: VII*, Zurich, 1987.
8. ISO/IEC JTC1/SC21/WG7. Working draft on enhancements to LOTOS. Project WI 1.21.20.2.3, January 1997.
9. G-LOTOS: a Graphical Syntax for LOTOS, 1989. ISO/IEC JTC1/SC21 N3253.
10. Allende, Jesús S. Graphical Designer for LOTOS. Dept. of Telematic Systems Engineering, Technical University of Madrid, 1994. (User Manual). Available at <http://www.dit.upm.es/~jallende> (May 1997).