UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

A Parallel Programming Supporting Tool

Kalinka Regina Lucas Jaquie Castelo Branco
Marcos José Santana
Regina Helena Carlucci Santana
Nivaldi Calônego Junior

Nº 52

NOTAS

São Carlos - SP

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2577

A Parallel Programming Supporting Tool

Kalinka Regina Lucas Jaquie Castelo Branco
Marcos José Santana
Regina Helena Carlucci Santana
Nivaldi Calônego Junior

Nº 52

NOTAS

Série Computação

São Carlos – SP
Dez./2000

Resumo: Este artigo apresenta o desenvolvimento e a implementação de uma nova Ferramenta de Apoio à Programação Paralela baseada em um meta-modelo orientado a objetos da teoria dos grafos, que permite a construção de arcabouços de programas paralelos em P.V.M. e M.P.I.. O meta-modelo é baseado no metodo Fusion o que permite que sejam levados em consideração tanto a arquitetura do hardware quanto as peculiaridades do software na criação desses arcabouços. Este artigo apresenta o meta-modelo e a ferramenta desenvolvida.

A Parallel Programming Supporting Tool[1]

Kalinka Regina Lucas Jaquie Castelo Branco[1], Marcos José Santana[1], Regina Helena Carlucci Santana[1], Nivaldi Calônego Junior[2]

[1] Computer Science and Statistics Department. Institute of Mathematical Sciences and Computing. University of São Paulo – ICMC/USP. 13560-970 - São Carlos - SP – Brazil
{kalinka, mjs, rcs}@icmc.sc.usp.br
[2] University of Cuiabá. Centro de Ciências Exatas e Tecnológicas - Departamento de Informática. Av. Beira Rio, 3100 - Jardim Europa. 78015-480 - Cuiabá - MT - Brazil
nivaldi@zaz.com.br

Abstract. This paper approaches the development and implementation of a new parallel programming supporting tool, based on a graph-theory object-oriented meta-model, that allows the construction of program skeletons for PVM and MPI parallel programs. The meta-model is based on the Fusion methodology and allows taking into consideration both the hardware and the software architectures. This paper presents the meta-model and the tool developed.

1 Introduction

Although there are many fields with typical applications requiring high performance computing, the high cost of real parallel machines has been a constraint for the wide use of parallel computing.

The adoption of distributed computing systems to implement parallel computing environment offers the opportunity to realize parallel applications in a cost-effective approach. Virtual parallel environments are able to offer the computing power required by those applications that need more power than that available from the sequential platforms, but do not require an expensive real parallel machine. PVM (Parallel Virtual Machine) [7] and MPI (Message Passing Interface) [6] are two successful examples of virtual parallel environments and both have been widely used in many different fields.

Thus, the adoption of virtual parallel machines comprises an attractive approach that has been widely used over the last decade.

However, an additional bound in parallel computing is the lack of supporting tools for the development of parallel programs. This is an old well-known problem that, on one hand, has been completely solved for the sequential programming with many well-established tools available but, on the other hand, for parallel programming the reality is different.

---

[1] Article published in the Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA´2000. Las Vegas, Nevada, USA, CSREA Press, pp.2567-2573, June 26-29, 2000.

The existence of new problems concerning synchronization, dependence relationship among different parts of a parallel program, process placement, load distribution, etc, takes to new requirements.

The existence of supporting tools to help the development of parallel programs is clearly needed and they can make easier the parallel programming task in several ways [8]. For instance, the adoption of a support tool may allow reaching transparence in both task synchronization and program maintenance. This is actually useful for both the inexperienced users (they may implement their parallel application more easily, even with low knowledge of the programming environments and with low expertise in parallel programming, by means of the on-line helps provided in the tool) and also the expert users (they may increase their productivity because a supporting tool allows better codes being generated and therefore fewer debugging problems).

The approach discussed in this paper is based on the use of an object-oriented meta-model built for the graph theory [2]. The parallel programming-supporting tool developed allows the construction of parallel program skeletons for both PVM and MPI environments taking into consideration the hardware and the software architecture specifications supplied by the user (figure 1). Based on the skeleton supplied by the tool the user will specify the application details [1].

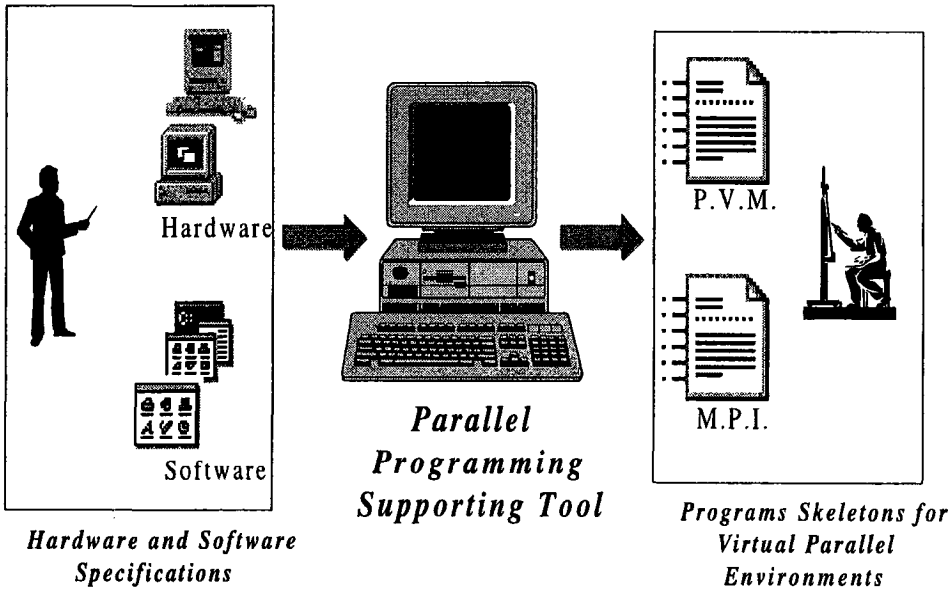

Fig. 1. Skeleton Generation Strategy

2 Existing Parallel-Programming Supporting-Tools

Several researches have been developed aiming at automating the development process of parallel programs by building environments and tools that give support to it. However, most of these environments and tools requires expert programmers to make the necessary changes in the code (solving the problem associated with synchronism, communication and load balancing), because it can not be totally automated in most of the cases. A good approach would be to provide environments and tools that starting from the specification would require the interference of the programmer only at the details about the application under implementation, hiding all the parallel programming traps [1].

There are few such tools available in the open literature including CAPTool [9] and Vienna Fortran [4] dedicated to parallel Fortran programs. P-RIO [3] is also available and supports the development of PVM applications. Meanwhile, neither of them can provide an appropriate code development, requiring an undesirable user involvement.

This paper proposes a novel tool to help users to construct program skeletons for the PVM and the MPI environments, following an object-oriented approach.

3 A Parallel Program Generation Machine

The computing systems, in general, as well as the programming languages and a large set of applications from different knowledge areas, can be represented as finite automatus or more generically by means of graphs.

A parallel program executing on a parallel architecture, or more generically in a parallel (or distributed) computing environment, is one of the applications that can take advantages in using the graph theory as the basis for constructing a formal model.

Thus, an object-oriented meta-model was developed (using the fusion method [5]) aiming at the representation of systems or entities that can be modeled by means of finite automatus or graphs.

This model allows the representation of virtual parallel environments where nodes represent tasks and arcs represent the communication between the tasks [1] [2].

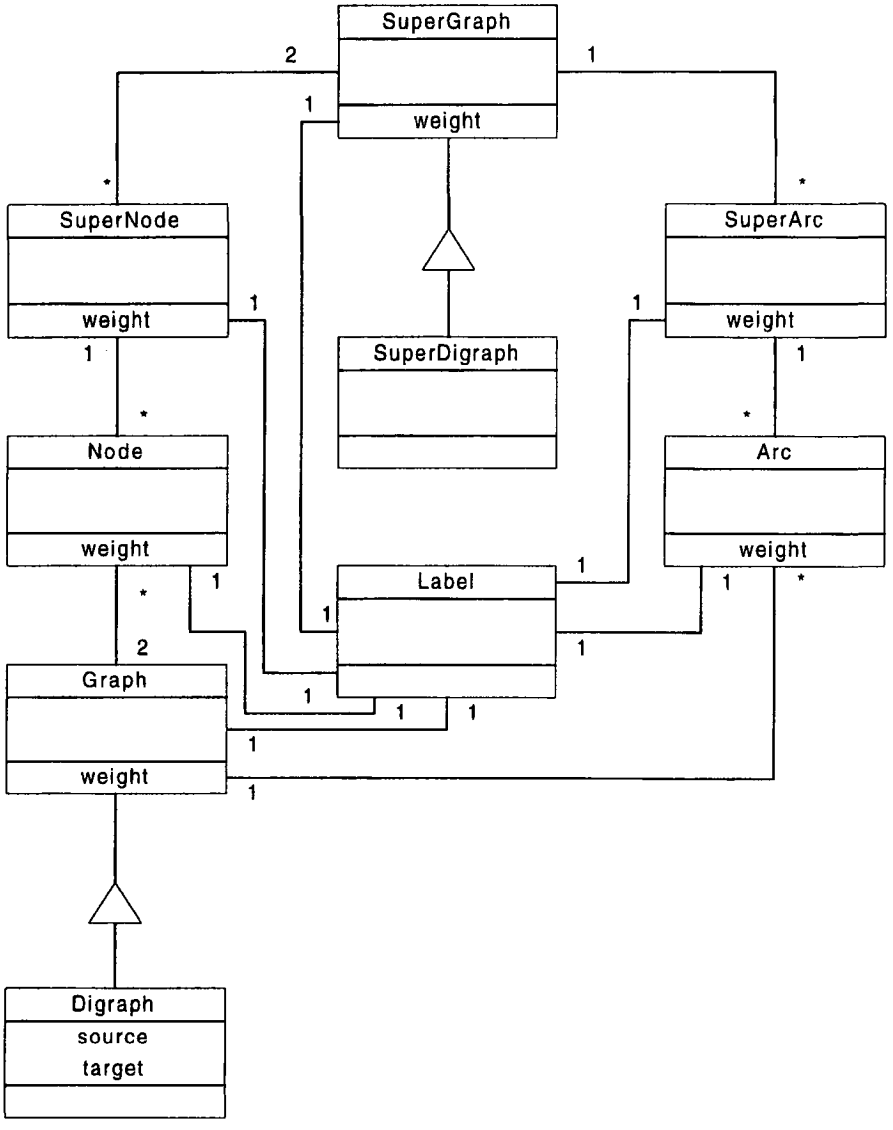

Fig. 2. Object-Oriented Meta-Model

Figure 2 shows some of the abstractions and specializations from the meta-model adopted incorporating labels, identifiers and weights.

The basis of the model is composed of nodes and arcs that can be used with or without weights. A graph is defined in the model by associating at least two nodes and one arc. A digraph can be defined as a specialization of a graph.

SuperNodes and SuperArcs are abstractions supplied to represent related sets of nodes and arcs. This concept allows the straight definition of SuperGraphs and SuperDigraphs.

The labels are used in the model to identify mnemonically an object aiming at facilitating the user-model relationship [2] [1].

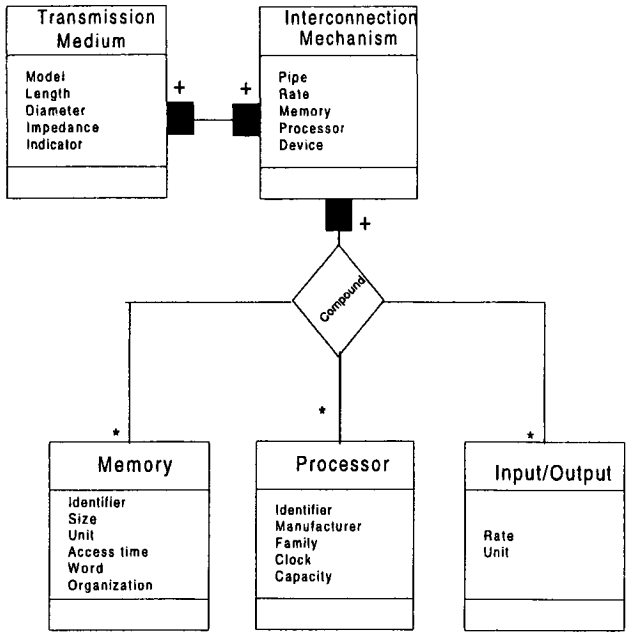

Fig. 3. Hardware Components Object-Model

In this paper, this meta-model is used to support the construction of a tool to help the development of PVM and MPI parallel programs. This tool is just an example of modeling by means of the graph theory associated with object orientation.

This tool is actually a complete environment to support parallel programming for a target hardware architecture, which can be easily extended to different parallel architectures. It takes into consideration the actual specification of both hardware and software provided by the user. I.e., the user describes the hardware and the software architectures adopted, giving the information about process, interconnection, program paradigm (Single Program Multiple Data or Multiple Program Multiple Data), etc, with the aim at optimizing the performance reached (software and hardware are actually tuned).

Thus, based on these user information it is possible to define the hardware and the software configurations on which the target application will be executed.

Figures 3 and 4 show the specification for both the hardware and the software object-model components.

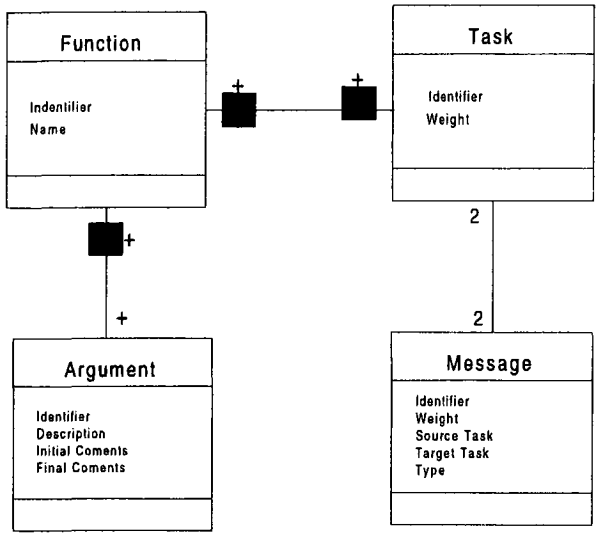

Fig. 4. Software Components Object-Model

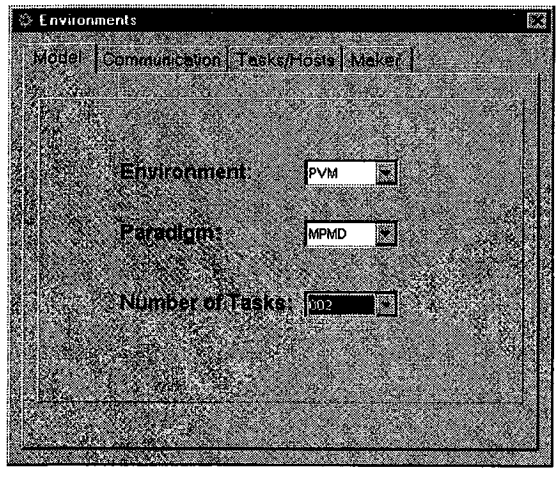

Fig. 5. Environment, Paradigm and Number of Tasks Definition

The hardware and the software architecture elements such as the processor, memory, data input and output devices, functions, etc. are all modeled because different architectural models are characterized by handling these elements.

Having the hardware and software object-models and, the templates for the chosen environments defined and with the assistance of the on-line helps, the program skeleton (for PVM or MPI) can be automatically generated.

Once the models are defined, the user must specify some application details, such as the model, the communication and the mapping of the tasks on the hosts considered.

The model adopted for the application is defined by the user, using a graphical interface, where the environment (PVM or MPI), the paradigm (SPMD or MPMD) and the number of tasks are specified. Figure 5 shows this graphical interface.

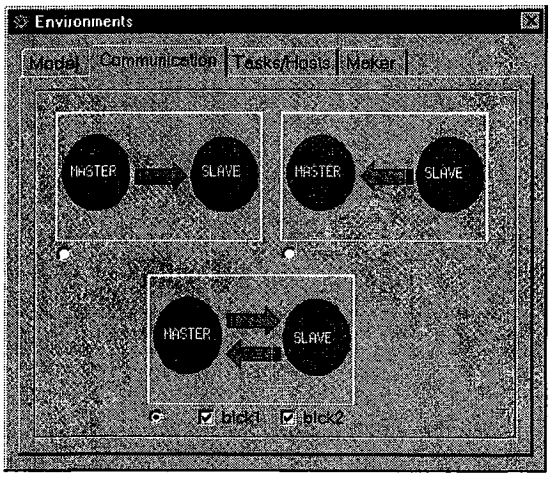

Fig. 6. Tasks Communication

The communication between the tasks is defined as shown in figure 6. In this step, the type of communication used is chosen (blocking and non-blocking).

The mapping of tasks to hosts is defined using the graphical interface shown in figure 7. The user can visualize each host available (identification, memory, etc.) and he is able to define where to execute each task.

After the definition of the model, the communication and the mapping, the user can ask for the generation of the application skeleton, pressing the button "maker".

Once the complete skeleton code was generated, the user has just to complete the code with the specific characteristics of the application, having the guarantee that the code produced is correct (mainly in terms of communication, where send and receive routines must be carefully defined in order to avoid lack of synchronism in the program).

Figure 8 shows a piece of code related to the communication among the master and slave tasks.

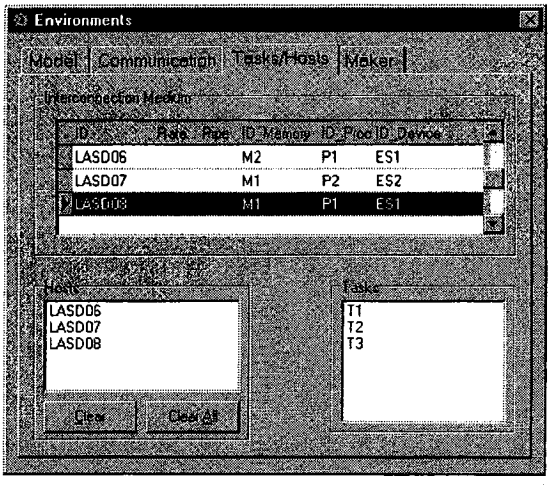

Fig. 7. Mapping Tasks on Hosts

The main advantages provided by this tool can be summarized as follows: (a) for PVM applications, although communication procedures do not represent a huge difficult for users, with the tool a user is didactically conducted to choose the suitable primitives both in terms of communication primitives and also in terms of the packing and unpacking procedures; (b) for MPI applications, the tool minimizes the problem involved in choosing the appropriate communication primitives (actually MPI offers a large and complex number of communication primitives including routines such as point-to-point, broadcast, collective, etc.); (c) for both PVM and MPI, the tool gives support for correct code generation reducing the maintenance requirements and making easier the debugging steps; (d) additionally, users can state the placement of the tasks in both cases (PVM and MPI) directly in the specification of the application, through the graphical interface available.

```
//Routines to send messages

//Start send buffer

for (i=0;i<ntask-1;i++) {

pvm_initsend(PvmDataDefault);

//Pack Data to Send

// pvm_packf (const    char *);

// pvm_pkbyte (char*, int,int);

.........

//Send Messages

pvm_send(tids[i+1],MSGTAG);}

//Routines to Receive Messages

//Receive Message

for (i=0;i<ntask-1;i++) {

info=pvm_recv(-1,MSGTAG);

if (info<0) { pvm_perror("Erro
FAPP: ");

          pvm_exit();

          return -1;} // if

//Unpack Received Data

// pvm_unpackf (const  char *);

// pvm_upkbyte (char*, int,
int);
```

```
//Unpack Received Data

// pvm_unpackf  (const
char *);

// pvm_upkbyte  (char   *,
int, int);

.........

//Define the routines to
be executed

//with the unpacked data

//Define the routines that
will generate

// the data to be send

//Routine to send data

//Start Send Buffer

pvm_initsend(PvmDataDefaul
t);

//Pack Data

// pvm_packf   (const
char *);

// pvm_pkbyte  (char
*, int, int);

.......

//Send Messages

pvm_send(myparent,MSGTAG);
```

Fig. 8. Communication Specification Among the Tasks

4 Conclusions

The object-orientation applied as a technique to model the graph theory produced in this paper a general meta-model that can potentially be used as the basis for several applications. The graph-theory meta-model proposed is very useful because it allows the generation of parallel programs in which future technological evolutions will be easily added on, due to the object-orientation concepts incorporated into the meta-model.

This tool is very simple to use and offers a useful set of facilities carefully chosen to supply the specific needs of PVM and MPI environments. Furthermore, considering the existence of on-line helps the tool is also very suitable as a learning supporting for PVM and MPI program development, playing an important hole as a teaching tool.

The tool is also suitable to be used by advanced parallel programmers, because it can improve their confidence on the production of correct parallel codes.

The flexibility offered by the object-oriented meta-model allows the insertion of codes and techniques such as scheduling and load balancing, taking into consideration both the software and the hardware information supplied by the users.

This tool can also support future integration of modules providing the simulation of the applications specified by the users and supplying the information about communication and performance.

Therefore, the design of this tool establishes a common kernel for parallel-programming supporting tools that allows future technological evolutions being dealt as specializations of abstract elements.

5 Acknowledgments

The authors would like to thank the Brazilian foundations FAPESP, CAPES and CNPq for their support to the researches developed by the Distributed Systems and Concurrent Programming Group at ICMC/USP, São Carlos, Brazil.

References

1. Branco, K. R. L. J. C., *Extensão da Ferramenta de Apoio à Programação Paralela para Ambientes Paralelos Virtuais*, Dissertação de Mestrado, Instituto de Ciências Matemáticas e Computação da Universidade de São Paulo (USP), Março, 1999.
2. Calônego JR., N., *Uma Abordagem Orientada a Objetos de Uma Ferramenta de Auxílio a Programação Paralela*, Tese (Doutorado), Instituto de Física e Química da Universidade de São Paulo (USP), Outubro, 1997.
3. Carrera, E., Loques, O., *P-Rio: Graphic and Modular Programming on PVM*, II EuroPVM Users' Meeting, pp, 89-94, Luon-France, September, 1995.
4. Chapman, B. M., Mehrotra, P. and Zima H., *Vienna Fortran Compilation System User Guide Version 1.0*, Dep. of Statistics and Computer Science, University of Vienna, Australia, 1994.
5. Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., Jeremaes, P., *Desenvolvimento Orientado a Objetos: O Método Fusion*, Rio de Janeiro, Editora Campus, 1996.

6. Dongarra, J. J., *et al, An introduction to the MPI standard*, University of Tennessee Technical Report CS-95-274, http://www.netlib.org/utk/papers/intro-mpi/intro-mpi.html, 1995.
7. Sunderam, V. S., Geist, A., Dongarra, J., Manchek, R., *The PVM concurrent computing system: evolution, experiences and trends,* Parallel Computing, v. 20, pp. 531-545, 1994.
8. Foster, I., *Designing and Building Parallel Program*, Addison-Wesley Publishing Company, s. 1, 1995.
9. Ierotheou, C. S., Johnson, S. P., Cross, M, Leggett, P. F., *Computer aided parallelisation tools (CAPTools) - conceptual overview and performance on the parallelisation of structured mesh codes,* Parallel Computing, v. 22 (1996), December, 1995.

NOTAS DO ICMC

SÉRIE COMPUTAÇÃO

051/2000 FRANCÊS, C.R.L.; VIJAYKUMAR, N.L.; SANTANA, M.J.; CARVALHO, S.V. de; SANTANA, R.H.C. – Stochastic statecharts for obtaining performance measurements of a file server model.

050/2000 FARIA,G.; ROMERO, R.A.F. – Incorporating fuzzy logic to reinforcement learning.

049/2000 OLIVEIRA, P.R.; ROMERO, R.F.; NONATO, L.G.; MAZUCHELI, J. – Techniques for image compression: a comparative analysis.

048/2000 LINS, L.; LINS, S.; MORABITO, R. – Na n-tet graph approach for non-guillotine packings of n-dimensional boxes into na n-container.

047/2000 OLIVEIRA JR., O. N.; MARCHI, A.R.; MARTINS, M.S.; MARTINS, R.T.; NUNES, M.G.V. – A critical analysis of the performance of english-portuguese-english MT systems.

046/2000 SOUZA, P.S.L.; SANTANA, M.J.; SANTANA, R.H.C. – Escalonamento de processos: uma contribuição para a convergência da área.

045/2000 MARTINS, R.T.; RINO, L.H.M.; NUNES, M.G.V.; MONTILHA, G.; OLIVEIRA JR., O. N. – An interlingua diming at communication on the Web: how language-independent can it be?

044/99 FELTRIM, V.D.; FORTES,R.P.M. – Evaluation of a reverse engineering method through the application in a hypermedia system.

043/99 PANSANATO, L.T.E.; NUNES, M.G.V. – EHDM: Método para projeto de hiperdocumentos para ensino.

042/99 MORABITO, R.; ARENALES, M. – Optimizing the cutting of stock plates in a furniture company.