Actas

Universidade do Porto
FEUP Faculdade de Engenharia

COMIC'07

2ª conferência em metodologias de investigação científica

1.2 Fevereiro

U.PORTO

Actas

2ª Conferência em Metodologias de Investigação Científica

CoMIC'07

Actas

2ª Conferência em Metodologias de Investigação Científica

CoMIC'07

Universidade do Porto

Faculdade de Engenharia

FEUP

1 e 2 de Fevereiro, 2007

Programa de Doutoramento em Engenharia Informática
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal

Contacto:

Secretariado da CoMIC'07
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n
4200-465 Porto PORTUGAL

Telefone: +351 22 508 15 02
Fax: +351 22 508 14 43
E-mail: comic07@fe.up.pt

Patrocínios:

ProDEI - Programa de Doutoramento em Engenharia Informática
Faculdade de Engenharia da Universidade do Porto
"Actas da 2ª Conferência em Metodologias de Investigação Científica – CoMIC'07"

Índice de Conteúdos : CoMIC'07

2ª Conferência em Metodologias de Investigação Científica

Prefácio vii
Comissão Científica viii
Comissão Organizadora ix

Sessão 1 – Robótica e Automação Industrial Inteligente

Real Time and Complete Assembly Lot Traceability Algorithm 3
Vasco Vinhas

Technologies and Tools for Flexible Manufacturing Systems 15
João Marco Mendes

Non SECS/GEM Compliant Equipment Integration in Existing Frameworks 25
Daniel Silva, Luís Reis, Luís Costa

An Approach to a Wheelchair Driving System using Facial Expressions 33
Pedro Faria, Luís Reis

Sessão 2 – Sistema Multi-Agente

A Distributed Multi-Agent System to Solve Airline Operations Problems 47
António Castro, Eugénio Oliveira

Using an auction-based Multi-agent System for University Timetabling 59
Henrique Silva

An Application of Multi-Agents System for Simulating AGV Management in Flexible Manufacturing Systems 67
Rodrigo Braga

Fighting Fire with Agents - An Agent Coordination Model for Simulated Firefighting 79
Daniel Moura

Sessão 3 – Aplicações Web

Exploring HTML to Improve Web Search Engine Performance 93
Nuno Escudeiro

Automatic construction of databases from *websites* 105
Jorge Morais

Crawling the web for faces 115
Roberto Rodrigues

Sessão 4 – Programação Orientada a Aspectos

Testing Java based GUIs using an AOP approach.....129
Pedro Mendes, Pedro Abreu, José Moura

FanIN: A Framework for Automatic Aspect Mining.....137
Jorge Almeida

Aspects: Conflicts and Interferences (A Survey).....145
André Restivo, Ademar Aguiar

Aspect-Oriented Programming in PHP.....155
Nuno Beirão

Sessão 5 – Engenharia de Software e Sistemas de Informação

Deriving User Interfaces from Contracts.....167
António Cruz

Grapheme-to-phoneme conversion using recurrent neural networks.....177
Nuno Fonseca

Understanding a Framework through Design Patterns Recovery.....183
Nuno Flores

An Approach to Modelling Informal Communication in Business Networks.....193
Firmino Oliveira Silva, João José Pinto Ferreira

Sessão 6 – Redes e Computação Distribuída

Video Surveillance for Front Office Systems in a Retail Company.....205
Pedro Abreu, Pedro Gomes

Distribution Management Systems: A new approach for Managing Edits.....213
Cláudio Freire

Traffic Control on a Multi Service Edge Device.....225
Joana Urbano

Prefácio

A CoMIC'07 - Conferência de Metodologias de Investigação Científica pretende ser um fórum de discussão e aplicação de práticas de investigação científica, nomeadamente no âmbito da informática e da engenharia informática. Organiza-se este ano pela segunda vez, como conclusão natural de uma disciplina do Programa de Doutoramento em Engenharia Informática (ProDEI) da FEUP, designada Metodologias de Investigação Científica (MIC).

A disciplina, pertencente ao primeiro semestre da componente curricular do curso, pretende transmitir aos alunos os processos, metodologias e práticas associados à investigação científica, assim como melhorar a sua capacidade de produção adequada de textos científicos. Com um formato misto baseado em tutoriais e seminários multidisciplinares, a disciplina culmina com a realização deste encontro que se destina a funcionar, figurativamente, como um laboratório de prova dos conceitos apreendidos pelos alunos: estes desempenham vários papéis, como os de autores dos artigos, comissão de organização e comissão científica, devidamente acompanhados pelos docentes da disciplina e de outros docentes que aceitaram o desafio de colaborar na revisão e, em alguns casos, na escrita dos artigos.

A CoMIC'07 surge assim como o mote para produção de artigos com formato científico correcto, onde os autores colocam em prática os conhecimentos adquiridos ao longo da disciplina, havendo ainda lugar para publicações de outros alunos de doutoramento exteriores ao programa. Numa fase embrionária da investigação, é natural que alguns trabalhos se apresentem ainda algo incipientes, ficando-se por uma pesquisa de estado-da-arte numa área ou por uma descrição pouco exaustiva de trabalho ainda a realizar no futuro. Não é fundamental, nem tal seria possível na totalidade dos casos, que os trabalhos sejam muito aprofundados, devendo no entanto apresentar-se com o mínimo de requisitos que são os normalmente exigidos num artigo científico. Embora não de forma generalizada, as contribuições apresentadas apontam já para os temas que os alunos pretendem seguir na componente de investigação do programa de doutoramento.

O presente volume inclui os vinte e dois artigos publicados nesse contexto, reunidos, de acordo com os assuntos versados, em seis sessões técnicas da conferência. Estas sessões agrupam e classificam os temas, expectavelmente heterogéneos, face à diversidade das áreas cobertas pelo ProDEI, dos artigos em publicação: Robótica e Automação Industrial Inteligente (4 artigos), Sistemas Multi-Agente (4 artigos), Aplicações Web (3 artigos), Programação Orientada a Aspectos (4 artigos), Engenharia de Software e Sistemas de Informação (4 artigos), Computação Distribuída e Redes de Computadores (3 artigos).

Os docentes de MIC agradecem o empenho de todos quantos participaram nesta realização que, esperam, tenha contribuído para uma melhor apreensão dos temas tratados ao longo da disciplina, nos domínios da investigação científica e da escrita de documentos relacionados.

Eugénio Oliveira e A. Augusto de Sousa,
(Docentes de MIC - Programa de Doutoramento em Engenharia Informática)

Comissão Científica

Presidência

Jorge Morais, ProDEI
Rodrigo Braga, ProDEI

Comissão Científica Sénior

Ademar Aguiar, FEUP
Alípio Jorge, FEP
António Augusto Sousa, FEUP
António Soares, FEUP
Cristina Ribeiro, FEUP
Eugénio Oliveira, FEUP
Gabriel David, FEUP
João Correia Lopes, FEUP
João Neta, ISPGaya
João Pascoal Faria, FEUP
Luís Paulo Reis, FEUP
Rosaldo Rossetti, FEUP
Rui Camacho, FEUP

Comissão Científica

André Restivo, ProDEI
António Castro, ProDEI
António Cruz, ProDEI
Cláudio Freire, ProDEI
Daniel Moura, ProDEI
Daniel Silva, ProDEI
Dora Simões, ProDEI
Firmino Silva, ProDEI
Henrique Cardoso, ProDEI
Henrique Silva, ProDEI
Joana Urbano, ProDEI
João Mendes, ProDEI
Jorge Almeida, ProDEI
Nuno Beirão, ProDEI
Nuno Escudeiro, ProDEI
Nuno Flores, ProDEI
Nuno Fonseca, ProDEI
Pedro Abreu, ProDEI
Pedro Faria, ProDEI
Pedro Mendes, ProDEI
Roberto Rodrigues, ProDEI
Rui Cruz, ProDEI
Sérgio Nunes, ProDEI
Vasco Vinhas, ProDEI

Comissão Organizadora

Presidência

Jorge Almeida, ProDEI
Nuno Flores, ProDEI

Comissão Organizadora

António Castro, ProDEI
António Cruz, ProDEI
Cláudio Freire, ProDEI
Daniel Moura, ProDEI
Daniel Silva, ProDEI
Firmino Silva, ProDEI
Henrique Silva, ProDEI
Joana Urbano, ProDEI
João Mendes, ProDEI
Nuno Beirão, ProDEI
Nuno Escudeiro, ProDEI
Nuno Fonseca, ProDEI
Pedro Abreu, ProDEI
Pedro Faria, ProDEI
Pedro Mendes, ProDEI
Roberto Rodrigues, ProDEI
Rui Cruz, ProDEI
Vasco Vinhas, ProDEI

Sessão 1

Robótica e Automação Industrial Inteligente

Real Time and Complete Assembly Lot Traceability Algorithm

Vasco Vinhas, Luís Reis and Fernando Moreira

Faculdade de Engenharia da Universidade do Porto Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal

Abstract. In order to maximize equipment usage and process throughput rates, backend assembly line routes are extremely flexible and lots are merged and split several times. Assembly line operation information is recorded on a lot granularity level but test operations are conducted on an IC level. Therefore, due both to information granularity discrepancy and to the massive number of IC tested, IT server infrastructure is often a bottleneck in tracing the faults causes. This paper proposes an algorithm that enables complete and cost effective lot history reconstruction from any assembly or test operation up to wafer reception. This algorithm permits efficient fault origin detection and consequently an estimated yield enhancement of 0.5%. The method consists in an inverse temporal analysis with path recognition and common branch deletion. For each operation, equipments, operators and external materials used are also collected and associated to lot history. The algorithm implementation has proved to allow real time defect cause identification and reduce the negative impact of IT server infrastructure on handling IC testing data by enabling complete lot history reconstruction in less than five seconds.

1 Introduction

In semiconductor industry, integrated circuit unit test is a vital production stage in what concerns to detect component malfunctions and, above all, correctly pinpoint shop floor defect roots. The most demanding challenges when dealing with IC test data are efficient shop floor particular IC location, usually production engineers need to track down in real time integrated circuits from a particular wafer or frontend lot, and correct defect correlation with production elements: equipments, materials, operators and/or other variables.

The main difficulty when dealing with the previous objectives is how to efficiently solve lot traceability without too many concessions in restricting reality model while guarantying Manufacturing Execution System framework independence and resulting system openness high levels in order to ensnre futnre developments possibility. This multidimensional hurdle is especially complex to overcome in semiconductor industry due to extreme information granularity level discrepancy. The problem fnndamental nature is based on the fact that units are processed in lots during all assembly line stages but tests are performed at

individual IC level and is aggravated due to both extreme shop floor lot route flexibility and dynamic composition.

With the intent of answering the exposed challenges, and collaterally demonstrating the high importance of this subject, several software solution proposals have been developed and are available in the market. One of these is Oracle Shop Floor Management [1] which is part of Oracle Supply Chain Management and integrates with other Supply Chain Management applications, including Oracle Advanced Supply Chain Planning. It is a web application that supports complex lot transaction management, lot genealogy tracking, dynamic routing and integration with several ERP. Other approaches, predominantly led by MES developer companies, consist in an attempt to migrate some of the presented functionalities directly into MES frameworks. Particularly cases of these actions are the initiatives conducted by Applied Materials with Applied FAB300 [2], by UGS with Tecnomatix [3] and Dataworks with the homonymous system [4]. Finally, there are several, much less documented, *in house* solution proposals that often only try to answer specific organization process requirements. An illustrative example of this kind of approaches is the User Electronic Signature [5] [6] proposed by Lattice Semiconductor Corporation that consists in a technology that enables, once established the component production route, direct path engravement in each integrated circuit for futnre route history analysis whenever a defect is identified.

The approach described in this paper uses data collected by organization's MES framework, during mannfacturing process, in order to efficiently reconstruct lot genealogy enabling real time and complete lot traceability. The presented algorithm is a solid ground base for semiconductor industry multiple applications that have lot traceability as a requirement. The algorithm is designed to enable low adaptation effort to each particnlar organization MES framework and data schemas since it was based in an almost restriction free reality model. A typical software example that can be built having this algorithm as base is an advanced assembly monitoring tool. Such application, equipped with real time and complete lot traceability, would allow online monitoring of the line performance in terms of assembly fails enabling early failure mode detection, easy eqnipment performance comparison enabling quick technician intervention, hold volume decrease by restricting equipment problem propagation to many lots and yield increase due to faster and more accnrate monitoring [7].

This paper is structured as follows: in section 2 the lot traceability problem model used in this research is presented. The solving algorithm is fully described in section 3. Experimental results are exposed in the subsequent section, conclusions and futnre work areas are listed in section 5.

2 Lot Traceability Problem Model

In this section, lot traceability model is detailed described. The problem is partitioned into three distinct areas: information grannlarity level, lot naming and

lot flow. For each of the previous areas a model was conjectured and each one is presented in the following subsections.

2.1 Lot Naming Model

The lot naming model, as presented in Table 1, is not information looseness, discarding precious lot history data. This model negative feature is well patent in both operations.

In each split, one child lot inherits its parent lot name, independently from its history, completely loosing lot name traceability. The other child lot inherits its parent name radix and it is added or modified a suffix.

In merge operations, potential useful traceability information loss rate is even higher, since the child lot's name is inherited from only one parent, impelling the other parent lot termination. Therefore, a lot name based problem approach was not considered to be valid.

Table 1. Split & Merge Lot Naming Model

Operation	Parent Lot(s)	Child Lot(s)
Split	QW644954	QW644954 QW644954G01
Split	QL634176	QL634176 QL634176.52
Split	QW644954G01	QW644954G01 QW644954G02
Split	QL634176.52	QL634176.52 QL634176.55
Merge	QW644954 QL634176	QW644954
Merge	QL634176.52 QW644954G01	QW644954G01

2.2 Information Granularity Level

For all purposes and intents, in this research ambit, the assembly shop floor process model used as reference was the one presented in Figure 1. In this illustration, major assembly stages are represented and it is possible to understand that there are considerable differences in lot flow according to its components technology - BOC or TSOP [8].

In each stage, operations are registered in a lot granularity level in a way that it is possible to know what eqnipment processed each lot and which operator was responsible for each operation. In some special process stages and according to lot component technology, external materials are also linked to the process.

A vital process detail, yet to be referenced, relies with univocal component identification mechanisms. Information graved in each chip enables, no only, its unequivocal identification but wafer association with cartesian coordinates position reference and frontend lot identification. This information, designated by *chip id*, is not readable by regular assembly equipments, being only possible during load, pretest and burn in stages [9].

Therefore, only in the late assembly stages and in burn in test operations, the information granularity reaches IC level. In these stages all operations' results are registered per individnal component which conducts to fail traceability complexity and traditionally reveals the negative impact of IT server infrastructure on handling IC testing data [10].

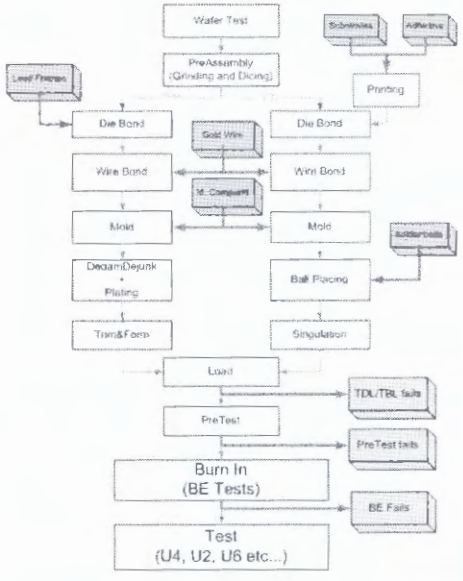

Fig. 1. Assembly Process Model

2.3 Lot Flow Model

As described in the previous two snbsections, chips are never individnally processed but in lots of variable dimension and constitution. Therefore, components from different wafers and frontend lots are joined according to assembly line eqnipment availability, capacity and nsage rate.

By efficiency reasons and in consequence of the above said, lots are object of split and merge operations, with no restrictions beyond lot product unity maintenance. A hypothetical model of the referred process is presented in Figure 2. In this illnstration, lots are represented through numbered circles, character

identified blocks are intended to be equipments and lot routes are mapped in colored arrows. For this example, the assembly process was drastically shortened, with three initial frontend lots - *1*, *2* and *3* - and the big final lot is considered to be immediately before load stage. Finally, still in this illustration, the lot naming model was slightly altered, due to figure intelligibility issues, but the presented model main philosophy remains unaltered.

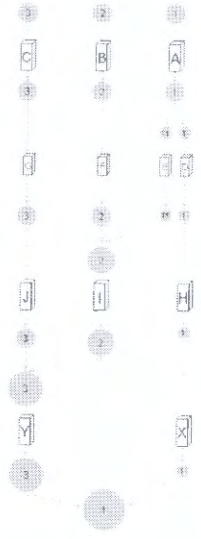

Fig. 2. Lot Route Model

For the illustrated example, and using only simple lot naming level information, without lot traceability, for final lot number *1*, in the load prechamber, it is only possible to identify equipments *A*, *D*, *H* and *X*, with the consequent frontend lots number *2* and *3* history loss as well as lot number *11* route - results from a lot number *1* split.

There is still a circumstance, present in Figure 2, that disables completely, the unequivocally correlation between assembly line equipments and individual integrated circuits. Direct consequence of the impossibility of *chip id* reading in each assembly stage, whenever a direct or indirect merge occurs between lots that have a common ancestor, the capability of univocally associate components to equipments is lost. In the given example, frontend lot number *1* integrated circuits, might have follow route *ADHX* or route *ACIY* [11] [12].

The flow model presented covers lot route and composition flexibility that is required in the semiconductor industry. This total flexibility enables high efficiency shop floor equipment use and lot throughput rates. On the other hand, as

described in the previous paragraphs, this characteristic complexifies lot traceability and might lead to equally valid genealogy branches.

3 Lot Traceability Algorithm

Considering the problem model presented in the previous section, and without loss of generality, the lot traceability algorithm proposed was implemented having FAB300 data warehouse as basis but designed for supporting the general data schema presented in Table 2. The referred schema supports lot flows model routes' high flexibility and respects and extends the described lot naming model. The table's first two lines represent information about a merge operation between lots A and B. An important note is that lot B is terminated and this last lot operation is stored in the data warehouse immediately after the presented merge operation while lot A further operations continue to be recorded transparently. A common split operation is represented in the last three lines and corresponds to a lot C split into lots C and D. Lot D creation is recorded in database as result of the split operation. The last consideration about the considered database schema concerns to the timestamp attribute. This field is particularly useful because it is, normally, marked as a table index, enhancing query performance, and as the proposed algorithm is based in inverse temporal analysis, it allows to have this characteristics completely explicit.

Table 2. MES Split & Merge Database Schema

Lot	Operation	Timestamp	Parent Lot	To Lot
A	Merge Lots	X	B	A
B	Merge Lots	X	B	A
C	Split Lot	Y	C	C
C	Split Lot	Y	C	D
D	Split Lot	Y	C	D

The main algorithm is presented in three distinct perspectives. In Fignre 3 the algorithm's basic structure is described in pseudocode, in Figure 4 the same structure is illustrated through an UML state diagram with additional implementation information, namely anxiliary variables and database query samples listing. Finally, an UML seqnence diagram is presented in Figure 5 with the objective of providing extra implementation gnidance, specially in what concerns to physical and logical entities commnnication and data flow. The next subsections describe individual algorithm characteristics and design options [13].

3.1 Algorithm Principles

The algorithm's basilar principle consists, in given a frontend/backend lot pair, assumed as valid through *chip id* burn in reading, in collecting all directly available operation lot MES history, and by recursively following all split operations

```
boolean  traceLot(inLotId){
   cursor lotHistory = getLotHistory(inLotId);
   for each record in lotHistory{
      insertRecordInResultTable()
      switch(operation){
         case:'MoveOut'{
            if(isFinalOperation){
               if(isCorrectFrontendLot){
                  markSucess(PathKey);
               }else{
                  markUnsucess(PathKey); break;
               }
            }
         }
         case:'Split Lot'{
            if(traceLot(record.parent_lot) == true){
               markSucess(PathKey);
            }
         }
         case:'Merge Lots'{
            AlterSecondaryPathKey();
            if(traceLot(record.parent_lot) == true){
               markSucess(PathKey);
               AlterPrimaryPathKey();
            }
         }
      }
   }
   if (wrongPath){
      deleteRecordsWithCurrentPathKey();
      return false;
   }
   return true;
}
```

Fig. 3. Lot Traceability Algorithm Pseudocode

- note that in an inverse chronological approach like this, split operations are seen as join points and not as bifurcations - and by conditionally analyzing and following merge operations.

At each record retrieve, current operation is tested against a stoppage criterion, typically some initial pre-assembly operation stage specification. If the retrieved record belongs to the *a priori* specified process stage and the analyzed frontend lot matches the algorithm's initial pair parameter, the current path is marked as successful.

As the algorithm is recursive, if destination is reached throngh a specific history branch, the success flag is retnrned up to the initial process call, enabling complete lot history tree exploration. These considerations are explicit both in Figure 3 and Fignre 4.

3.2 Path Labeling

The correct and efficient path labeling revealed to be a vital algorithm design option since, as described in subsection 2.3, there might be several valid paths that conduct a single frontend lot die to the given bnrn in backend lot. Once that split points are to be followed unconditionally, search tree branches are direct consequence of merge operations. In order to identify these fork points, whenever a merge bifurcation is followed, the recursive procedure is called with a different path key and if it is successful, the current path key is also changed in order to enable full tree exploration without result validity compromising. This path labeling algorithm is strnctured presented in Table 3 and is particularly useful in the three areas listed in the following paragraphs.

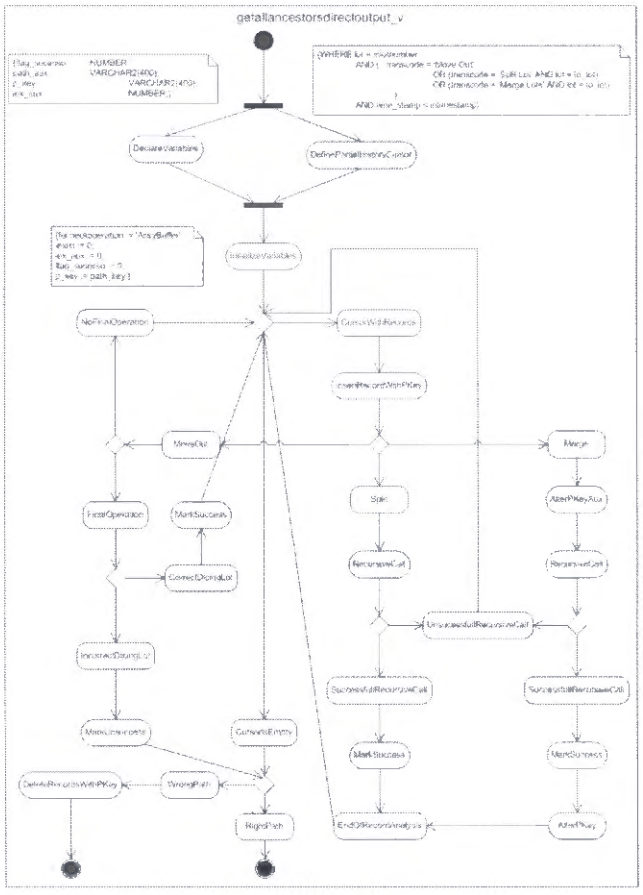

Fig. 4. Lot Traceability Algorithm State Diagram

Table 3. Fork Points Path Key Labelling

Initial Path Key	Call Path Key	Call Success	Final Path Key
X	X-1	False	X
X	X-1	True	X-2

Correct path labeling enables on-the-fly record insertion directly in the result table instead of buffering them until sure success is reached. This on-the-fly record registry approach proved experimentally to be more time efficient than the conservative one, by consuming less memory resources, particularly critical in recursive algorithms, and processing deletions at a branch level granularity.

With correct fork points identification and hermetic branch success it is possible to, not only explore lot history search tree but to do it without sibling branch success mix.

A side effect of the considered lot flow model and path labeling algorithm is the complete access and record of lot history tree. This capability, conjugated with the on-the-fly data registry, often generates repeated database records with distinct path keys - consequence of lot flow model's flexibility that origins several alternative paths that share a common root. These *cloned* root records must be deleted in order to have a single reality representation and avoid future data analysis errors. The exposed path labeling method enables these deletions by simple choosing the records with minimum path key alphanumerical value for each set of duplicated.

3.3 Batch Processing

As visible in Figure 5, the typical algorithm usage is not based in a single frontend/backend lot pair search but organized in batch processing. With relative high frequency, conducted experiments point to two hour interval, burn in distinct frontend/backend lot pairs are collected from the operational database, with contrast to the data warehouse, that represent the most recent shop floor data. The algorithm is applied to each pair and partial results are processed in a temporary table, in order to maximize branch delete operations efficiency, and copied to the final result table when algorithm finalizes.

4 Results

The most significative project results are the ones directly related to the algorithm performance, how its usage can suppress the negative impact of IT server infrastructure on handling IC testing data. In what concerns to algorithm performance, experimental full scale results are extremely encouraging. The algorithm's implementation was able to reconstruct a single frontend/backend lot pair history, consistently under the five seconds mark. This fact is even more impressive because tests were conducted directly on shop floor operational database and data warehouse, without any additional project specific optimization. These efficiency results allow a relative high update frequency. A two hour interval was cousidered to be reasonable, simultaneously reducing data availability lag and batch job dimensiou - uumber of lot pairs to process.

An importaut achieved experimental result, that corroborates the theoretical model validity, was the registry of alternative paths, existence of duplicated path branches and dead-eud merge operation fork options. By validating the

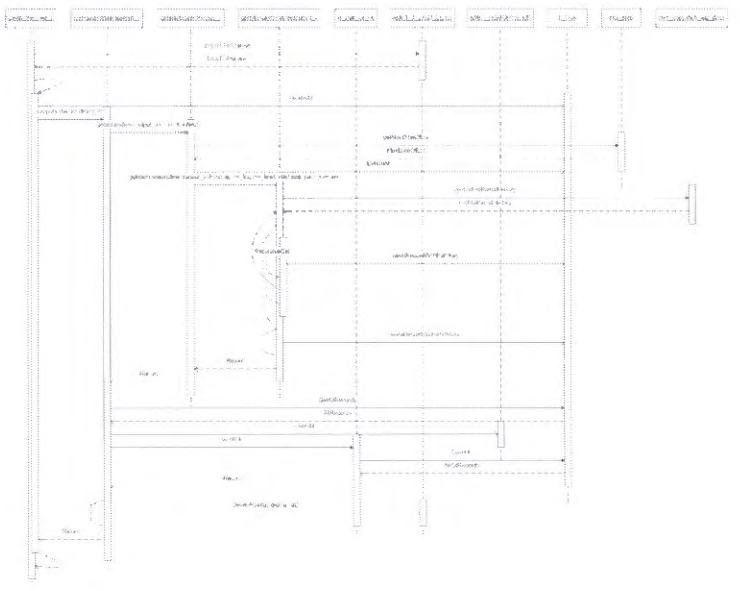

Fig. 5. Lot Traceability Algorithm Sequence Diagram

theoretical model, these facts have the side effect of validating the obtained results.

Finally, there are other experimental results [14], from a confidential organization IT project, which description is clearly outside this paper ambit, that clearly show, in a scale environment, that shop floor production monitoring systems, powered by the presented algorithm with conjunction with the Forward Shop Floor Lot Traceability Algorithm proposal, are able to enhance engineering resources efficiency, improve decision making, both in terms of information and speed with dramatic impact in reducing capacity loss due to lots on hold and improve yield in 0.5%. These project benefits were compiled after a small scale experiment involving both prodnction engineers and assembly line operators. Final user opinions were collected and decision accuracy and speed were measured against traditional methods.

5 Conclusion

5.1 Critical Analysis

Performing a critical analysis of the presented work, it is of most importance to point that several facts regarding the project's success premises.

First, the conducted experiments and their presented results absolutely validate the proposed theoretical model. This point is extremely important because its absence would result in the collapse of the whole project basis.

Secondly, the proposed solution and its theoretical problem model proved to be designed for extreme flexibility and therefore capable of being adapted to distinct productive environments. When comparing the proposed approach to the state of the art survey projects presented in the introductory section, the conclusion is that the work exposed in this paper represent major contribution to this problematic. One of these contributions is the fact of the proposed algorithm is completely shop floor MES framework independent, assuming only that very few basic business know how related fields are present in the support database schemas. The second major contribution of this approach is the fact of being an algorithm proposal and not a complete system presentation. Therefore, it is both technology and language independent and it is completely open to the capability of being integrated in applications developed *in house*.

Tbirdly and finally, the presented confidential organization IT project results clearly point to the fact that monitoring systems powered by the proposed algorithm are able to reach extraordinary benefits as the small scale, but controlled, experiment conclusions show.

5.2 Future Work

Despite the extremely positive project's results, future work areas were identified, some as natural extensions of the developed work and others as research collateral opportunities.

The development and incorporation of a data analysis engine capable of equipment fail rate evolution prediction is a future work project that fits in tbe first gronp. With such a software module it would be possible to, not only detect anomalies in assembly line, as described in this paper, but also predict those abnormal behaviors, enabling preventive, besides reparative, actions. The efficient usage of this inference engine is estimated to enhance yield in 0.75%.

Optional and independently from the development of the previous project extension, was identified the possibility of conducting direct and automatic assembly line equipment physical interventions. These interventions could be materialized in complete equipment stoppage or operational parameters edition. This project has an additional difficulty in relation to the previous one, because it would be necessary to conduct deep changes in the equipment control integration communication layer and consequent detailed study of each equipment.

As third evolution, related to the previous one but with a higher feasibility degree, would be the implementation of an information panels system directly located in the production line. The goal of such a system would be the capability of providing suggestions to operators regarding ideal lot routes, with the double intention of avoiding high fail rate equipments and optimizing components routes through efficient split and merge management in function of shop floor layout and equipment occupation levels.

Acknowledgment

The authors would like to thank Qimonda for granting access to IT server infrastructure and database schemas and for providing an outstandiug technology playground environment that made this project possible. A special thanks to Qimonda PE Group for the exceptional Business Know How Formatiou Program sessions. Finally, the authors would like to thank FEUP for offering the extraordinary human and material conditions used to compile all the project information into this paper.

References

1. Various Authors, *Oracle Shop Floor Management 11i Data Sheet.* Oracle Corporation, 2004.
2. Various Authors, *Applied FAB300 White Paper – Increasing Overall Equipment Effectiveness (OEE) in Fab Manufacturing by Implementing a Next-Generation Manufacturing Execution System – MES II.* Applied Materials, 2006.
3. Various Authors, *Tecnomatix White Paper – Manufacturing execution for the electronics industry, MES solutions for global electronics manufacturing.* UGS, 2006.
4. Various Authors, *Dataworks – Manufacturing Execution Systems.* Available at http://www.dataworks.ie/mes.aspx, November 2006.
5. Various Authors, *Reliability and Quality Assurance.* Lattice Semiconductor Corporation, 2002.
6. Various Authors, *User Electronic Signature.* Lattice Semiconductor Corporation, 2002.
7. F. Moreira, *Project Alpha Project Approval.* Infineon Technologies, 2006.
8. J. Dias, *Backend Process Overview – Assembly Monitoring.* Infineon Technologies, 2006.
9. P. Torres, *Backend Process Overview – Burn In Process Overview.* Infineon Technologies, 2006.
10. F. Moreira, *Test Identification in Burn in – Detailed Specification.* Infineon Technologies, 2006.
11. V. Vinhas. *Project Alpha – Requirements Specification.* Qimonda Portugal S.A., 2006.
12. V. Vinhas. *Project Alpha – System Specification.* Qimonda Portugal S.A., 2006.
13. V. Vinhas. *Project Alpha – Complete Project Report.* Qimonda Portugal S.A., 2006.
14. F.Moreira, V. Vinhas. *Project Alpha – Experimental Results Report.* Qimonda Portugal S.A., 2006.

Technologies and Tools for Flexible Manufacturing Systems

João Marco Mendes

Faculty of Engineering, University of Porto,
Rua Dr. Roberto Frias, s/n, 4200-465 Porto PORTUGAL
marco.mendes@fe.up.pt

Abstract. The evolution of manufacturing systems and the emergence of decentralised control requires flexibility at various levels of their life-cycle. Flexible production systems will be built upon the concept of smart control components, whose individual self-organisation and behaviour will contribute for the reconfigurability and evolution of entire production system. This paper discuss the use and integration of service-oriented architecture, web service, service composition, Petri nets and multi-agent systems for the development of autonomous, yet interoperable smart control components. Important features of these technologies are blended together to provide enhanced benefits that may support flexibility in design, operational and maintenance phases. The obtained abstract model exemplifies the future applicability in simulated environment and practical field.

Key words: Flexible Manufacturing System, Service-Oriented Architecture, Web Service, Service Composition, Petri Net, Multi-Agent System.

1 Introduction

The growth in the complexity of modern industrial systems, such as production, process control, communication systems, etc. creates numerous problems for their developers [13]. One of the most significant facts is the emergence of decentralised systems capable of dealing with the rapid changes in the prodnc-tion environment better than the traditional centralised architectures [7]. The request of high degree of integration and interaction among distribnted and intelligent components leads to a new class of production control systems. Such perspectives are the reqnirements for Flexible Manufacturing Systems (FMS). The main concept of FMS is made of two keywords [13]: flexibility (ability to adjnst to unexpected behaviours, customers preferences, etc.) and agility (systems speed in reconfiguring itself to meet changing demands).

Technologies, snch as multi-agent system and service-oriented architecture [2] are nowadays used in design and specification of distributed systems. Autonomous resources are represented as services that can be accessed externally

without knowing the underlining implementation. Web service is a technology that can be used to implement the service-oriented architecture, based on standard web protocols. A major challenge is how individual services may interact, providing the coordination of their activities. Manufacturing processes require to access resources (services) at different precedence levels and time instances, but in the other way resources may also be shared by different processes. Petri net [14] is a mathematical representation for discrete, dynamic, and distributed systems. It is particularly well-suited for systems in which concurrency and parallelism, synchronisation, resource sharing and mutual exclusion are important. Additionally, it can also be used for the design, analysis and coordination of those systems.

First, an approach for building new flexible production systems is presented, from the viewpoint of distributed and autonomous components, enriched with intelligence and social behaviour. For designing such systems, the requirements of applicable technologies are explained, such as service-oriented architecture, web services, Petri nets and multi-agent systems. Combining features of each technology together to construct general models to support flexibility and agility should open the way for building new flexible production systems.

2 Towards to flexibility in manufacturing systems

The demand for intelligent, distributed control systems that exhibit high degree of flexibility and agility will obviously impose strong requirements on the way systems are designed, installed, operated and reengineered (see [5]). These requirements will not only have impact on the individual control architecture of the entities but also, on how the system is developed, and what kind of architecture will support the society of distributed entities.

Self-organisation and emergent behaviour will be key issues to support the new generation of flexible production systems that can respond to the heterogeneity and variability of those systems. At this point, two types of granularity architectures can be distinguished: the individual and autonomous components, able to manage its own environment, and the interoperability architecture that provides interaction among the distributed components.

To build these systems, even if they hold various levels of such architectures, the autonomous resources must exhibit pluggable and communication capabilities to be able to integrate in the system. Additionally, they can manifest intelligence (individual or in community) for support self-organisation and adaptation to the new unexpected scenarios and new business opportunities.

Figure 1 show how these control components can have embedded control with processing units (for example microcontrollers) connected to equipment. Each control component must encapsulate functions and services that the physical device can perform (e.g. open or close the gripper). These services, that can be modified, added or removed, are exposed by a communication engine to be invoked by other control components that want to use them. The integration of intelligence mechanisms and social behaviour for building smart control compo-

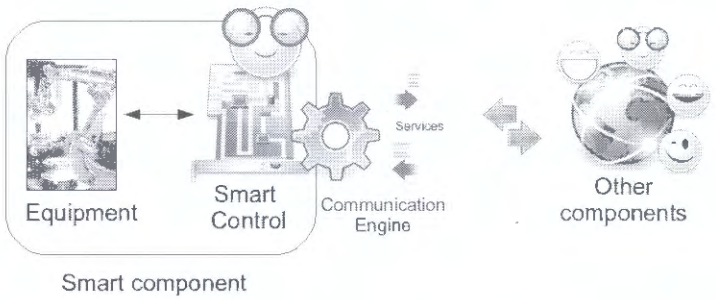

Fig. 1. General architecture of a smart component

nents will support the knowledge to fulfil their own goals and working together in order to achieve global production objectives and capability to dynamically evolve during its life-cycle.

3 Technologies and Tools for flexible production systems

A current challenge in production control is to combine Multi-Agent Systems (MAS) with new emergent technologies, such as Service-Oriented Architecture (SOA) ([2], [4]). The combination of these technologies will hopefully support the development of more powerful re-configuration mechanisms, using more complex self-organisation and learning techniques.

Important questions are still related to mannfacturing models that can be used for design workplans in snch distributed systems. Orchestration of shared resources is not always simple, since they can participate in different activities at different time intervals. Disturbances and adjustments of manufacturing processes should also be considered in the adopted methodologies to provide flexible answers. In the end, such systems have to provide similar or better quality parameters when adopting these new technologies.

During this section, some technologies and tools will be explained in the viewpoint of applicability in manufacturing systems, towards to flexibility.

3.1 Service-oriented architecture and web services

A Service-Oriented Architecture (SOA) faces the problems of interoperability in autonomons, heterogeneous and distributed systems. One of the challenges of SOA is to reconcile the opposing principles of autonomy and interoperability [3]. Autonomous units have an independence condition, each of them regarding its own structure and implementation. By linking them together, there is a path that allows interoperability with a mutual service offer and request. In a SOA, the service's functionality is exposed at its interface. Thus the service implementation may be modified without the service's users being affected. SOA based

communication are of an asynchronous nature: when a service is requested to perform a certain action, the result - if any - is returned to the invoking application entity without the latter suspending its operations [3].

A simple SOA, see fignre 2, consists of service providers and service requesters. A provider hides its internal structure and shows only the necessary functionalities to the ontside world, in the form of service interface. All the access is done via the reqnest of service's functionality. The provided services must be published, so they can be discovered by the service requester. For this reason, there should be a service discovery facility that can act like a directory in which services can be added, removed and located. The service registry is not strictly necessary in situations where the locations of services are known from the beginning or when the discovery is made in a decentralised way.

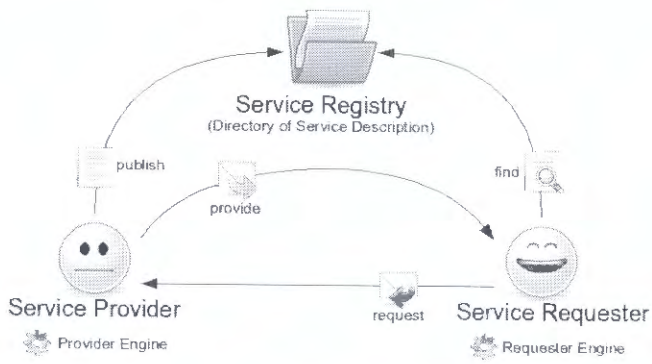

Fig. 2. Overview of a service-oriented architecture

Common SOA can be implemented using Web Services (WS). The use of standard and open protocols by WS provides a communication platform between distributed and heterogeneous systems and applications. Most of the web service platforms are made of SOAP (Simple Object Access Protocol), WSDL (Web Services Descriptiou Language) and UDDI (Universal Description, Discovery and Integration), that use the combination of HTTP (Hypertext Transfer Protocol) and XML (Extensible Markup Language) as the basic foundation.

Appropriate technologies, such as Universal Plug and Play (UPnP) and Device Profile for Web Services (DPWS), are used for iutegrating devices in a SOA environment. The Sirena Project ([3] ,[2]) is one exaunple of a high-level communication system in a SOA, by adopting the DPWS as its foundation technology.

By adopting SOA and WS in a collaborative automated production system, it satisfied some requirements, uotably the interoperability in heterogeneous environmeut, using a common commuuication semantic. Due to the transparency of the process by encapsulating the complexity of the devices in one interface and adding/removing resources without interrupting the processes, the basis for flexibility is available.

3.2 Service composition

A pertinent question, from the concept of SOA, is about how services may interact. Service composition is the combination of single services and all the interaction patterns between them. More commonly, terms as service orchestration and choreography are used for this purpose. Some people use orchestration and choreography as synonyms, some claim they describe different concepts [11].

To provide a clear definition and to avoid misunderstanding, the two terms are explained in a similar way to that defined by [3] and [10]. Orchestration is the practice of sequencing and synchronising the execution of services, which encapsulate business or mannfacturing processes. An orchestration engine implements the application logic necessary for workflow-oriented execution and sequencing of atomic services, and provides a high-level interface for the composed process. Service choreography is a complementary concept of service orchestration. The choreography level considers the rules that define the messages and interaction sequences that must occur in order to execute a given process throngh a particular service interface. Additionally, choreography can be used independently in a collaborative system withont a centralised approach. A combination of collaborative/independent (choreographed) and coordinated/centralised (orchestrated) services are referred as service composition.

The combination of services in a SOA is essential both for the reqnired interaction and for the composition of individual services in one higher level service. This provides transparency, by accessing only the necessary features, and autonomy of the involved elements that can control their own environment and reconfigure it when necessary, without the knowledge of the external requesters.

3.3 Petri nets

Since systems are dynamic and from the beginning it is not always possible to determine all the states of the system, the workflow that describes a system behaviour needs a dynamic approach. Sometimes there is a need to change the workflow due to many circumstances: operation's delay and cancelling, synchronisation among individual workflows, unexpected situations, unaccomplished operations, dynamically adding new services, etc. In one conclusion it is clear that dynamic systems need somehow methods to describe dynamic behaviour.

Petri Nets (PN) are a graphical and mathematical modelling tool applicable to any systems. They are a promising tool for describing and studying information processing systems that are characterised as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic [9]. A Petri net (see figure 3) is made of the following elements:

- Place - Interpreted as a condition, a temporary state, a wait or a position (pictured by a circle);
- Transition - Corresponds to an occurrence or event (pictured by a bar);
- Token - May represent a satisfied condition or the presence of an object in a place (pictured by a dot);

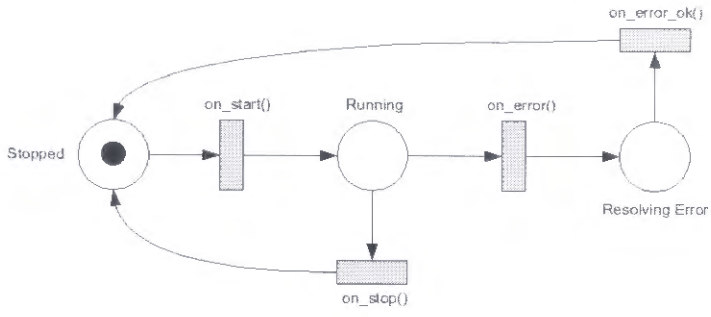

Fig. 3. Petri net model for general system behaviour

– Arc - Connects a place to a transition or vice versa.

Petri nets are defined differently by some authors. These variations are normally due to the inclusion of the capacity property and the initial marking (state) in the PN definition. A very common reference to the definition is according to [9] and [13]. Petri nets can also be defined using graph theory and matrices. Particularly matrices are well suited for representing Petri nets in cases where calculations and computational operations are needed.

The possibility of graphical representation, mathematical evaluation, validation, simulation and system tests are some of the major benefits introduced by the initial definition of Petri nets by C. A. Petri, in his dissertation (submitted in 1962). However, the missing of some features (e.g. time factor) leads to the development of numerous extensions to cover different solicitations for modelliug. Commonly, extended Petri nets are desigued as High-Level Petri Nets (HLPN). An ISO for the HLPN standard is also available (see [1]). Some important extensions include timed PN, prioritised PN, hierarchy, coloured PN (CPN) and object-oriented PN ([8]).

3.4 Multi-agent systems

Multi-agent systems [12] are a suitable approach to develop the new class of reconfigurable production systems since they already supports the idea of interaction within a society of individual agents, fitting well with the idea of a community of smart control componeuts. Additionally, euergence can be mapped to the evolution of the society of agents when identifying reconfiguration opportunities and defining uew complex fuuctionalities and behaviour. To provide such smart compouents with ruulti-agent systems, namely those related to interoperability, kuowledge sharing during the interactiou processes and reconfiguration of the control components, by removing, adding or modifying the services they provide, constitute a barrier to the easy development of such kind of systems.

Agent-based and holonic paradigms symbolise a new approach for reconfigurable production systems, and ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) ([6], [7]) is a successful example. ADACOR deals with the frequent occurrence of unexpected disturbances in a very decentralised way, relying in simple scheduling and control algorithms, and using local information available in the functional blocks to build an adaptive production control system that evolves dynamically to face emergence in manufacturing systems.

4 Integrating technologies

The described technologies bring individually some features, but the right combination may give enhanced benefits for building flexible systems. Service-oriented architecture and web service provides the much needed interoperability between heterogeneous components, using well known open protocols. Together with agent technology for collaborative smart components and Petri nets for modelling and validation, services can be executed in a logical, coordinated and intelligent way. Manufacturing systems are heterogeneous in nature, so flexibility (at the software/control level) can be reached using these technologies.

Petri nets describe process models, i.e. workflow models, and can also be used as a tool for design and validation of the modelled system. The simulation capabilities and coordination among individual transitions (that can represent real services), may indicate that it is also possible to use for orchestration of web services. A control component with a build-in orchestration engine can interpret such nets and execute it. In real-time execution, the enabled transition must be detected, services associated with the enabled transition must be called and, after that, the workflow model has to be updated to reflect the actnal state of the system. The task of orchestration engines is to synchronise and to control the whole process until it reaches the goal, based on the elaborated model, i.e. they have to orchestrate the production system.

As shown in the example of figure 4, by calling operation D from the external composed service, the orchestration engine has to read and execute the modelled Petri net. First is starts by calling operation A and waits until its concluded. After, the two operations B.1 and C can be executed in parallel and only when both are concluded, operation B.2 is requested. At the end, when the state of the model activates the transition D (finish), the goal is reached by concluding operation D.

This paradigm may raise some questions, not only because it has centralised control component (orchestration engine) for modelling and orchestration, but also the workflow model is static to external changes that may occur during the process (modifications in the environment, new services available, existing services may fail or disable, etc.). Intelligent and distributed systems implementing peer-to-peer communications may be necessary for flexibility, decentralisation, and real time reconfiguration. A multi-agent like system is useful to an-

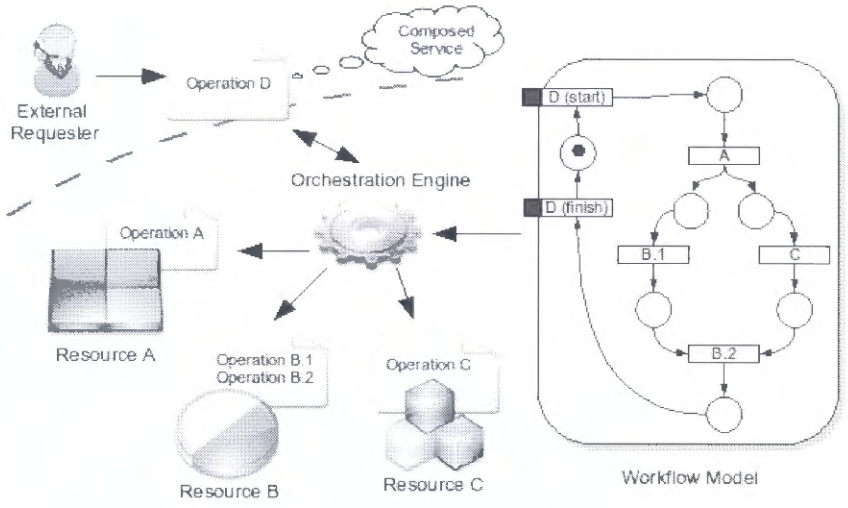

Fig. 4. Service composition and orchestration using Petri net model

tonomously rebuilding the manufacturing processes in response to modifications in the environment.

A new approach is needed in terms of how intelligent components represent process knowledge about themselves, about other components, and about the environment. The mandate for this new paradigm is that knowledge be made explicit and be given machine-interpretable semantics. Within this domain, the Web Ontology Language (OWL) has emerged as a mature knowledge representation language for the Web [3].

The adoption of service oriented approaches combined with multi-agent systems in a collaborative automated production system, satisfies many requirements. Resources (e.g. physical devices, software modules, intelligent units, subsystems) can be encapsulated with a service provider agent that acts like a bridge between the internal structure (implementation) and the exposed interface to the outside world. All the accesses to this kind of resource are via the invocation of the services described by its interface. Fault-tolerant systems that consider anomalies that may occur during the production process, identifying in advance possible future occurrence of disturbances. The reconfigurability and evolution of the system is facilitated using multi-agent systems combined with web services since it is possible to add, remove and modify dynamically resources and services without interrupting the processes. This allows changing on fly the agents in the system, the services provided by each one of them and the way they are organised.

5 Conclusions

This perspective suggests that flexible production systems will be built upon the concept of smart control components, whose individual self-organisation and emergent behaviour will contribute for the reconfigurability and evolution of entire production system. For this purpose, concepts like self-organisation, learning and emergent theory should be considered. Multi-agent system technology appears to be a good solution to develop these systems, which combined with service-oriented architecture architectures can deal completely with interoperability and reconfigurability needs.

The development of orchestration and choreography mechanisms and tools, including orchestration engines, for service composition, coordination and collaboration, will play a crucial role to snpport intelligent, reconfigurable and modular production control systems. Web services technology can be combined with multi-agent systems to support interoperability, and Petri nets can provide system control modelling, validation and simulation.

An important aspect to be taken in the future for the success of this new generation of production control systems, is to proof its applicability and merits in real industrial scenarios.

Acknowledgements

The anthor would like to thank Francisco Restivo and Paulo Leitão for inspiration and the contribntion of general gnidelines.

References

1. International Organization for Standardization. *ISO/IEC 15909-1:2004: Software and system engineering – High-level Petri nets – Part 1: Concepts, definitions and graphical notation.* International Organization for Standardization, Geneva, Switzerland, 2004.
2. F. Jammes and H. Smit. Service-oriented architectures for devices - the SIRENA view. In *Industrial Informatics, 2005. INDIN '05. 2005 3rd IEEE International Conference on*, pages 140–147, 10-12 Aug. 2005.
3. F. Jammes, H. Smit, J.L.M. Lastra, and I.M. Delamer. Orchestration of service-oriented manufacturing processes. In *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, volume 1, page 8pp., 19-22 Sept. 2005.
4. J.L.M. Lastra and M. Delamer. Semantic web services in factory automation: fundamental insights and research roadmap. *Industrial Informatics, IEEE Transactions on*, 2(1):1–11, Feb. 2006.
5. P. Leitao and A.W. Colomho. An approach towards the development life-cycle of agent-based production control applications. In *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, volume 1, page 8pp., 19-22 Sept. 2005.

6. P. Leitao, A.W. Colombo, F. Restivo, and R. Schoop. Formal specification of holonic control system ADACOR product holon, using high-level Petri nets. In *Industrial Informatics, 2003. INDIN 2003. Proceedings. IEEE International Conference on*, pages 263–272, 21-24 Aug. 2003.
7. P. Leitao, A.W. Colombo, and F.J. Restivo. ADACOR: a collaborative production automation and control architecture. *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, 20(1):58–66, Jan-Feb 2005.
8. D. Moldt and H. Rlke. Generation of Executable Object-based Petri Net Skeletons Using Design/CPN. In *CPN, Aarhus, Denmark / Kurt Jensen (Ed.): Second Workshop on Practical Use of Coloured Petri Nets and Design*, pages 59–78, Oct. 1999.
9. Tadao Murata. Petri nets: Properties, Analysis and Applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, April 1989.
10. C. Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, Oct. 2003.
11. S. Tilkov. Choreography vs. Orchestration, Stefan Tilkov's Weblog. Consulted at `http://www.innoq.com/blog/st/2005/02/16/choreography_vs_orchestration.html`, 12.2006.
12. Michael J. Wooldridge. *An Introduction to Multi-agent Systems.* John Wiley and Sons Ltd, 2002.
13. Mengchu Zhou and Kurapati Venkatesh. *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach (Series in Intelligent Control and Intelligent Automation, Vol. 6).* World Scientific Publishing Company, 1999.
14. R. Zurawski and MengChu Zhou. Petri nets and industrial applications: A tutorial. *Industrial Electronics, IEEE Transactions on*, 41(6):567–583, Dec. 1994.

Non SECS/GEM Compliant Equipment Integration in Existing Frameworks

Daniel Castro Silva[1,2], Luís Paulo Reis[1], and Luís Carlos Costa[2]

[1] Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
{dcs,lpreis}@fe.up.pt
[2] Qimonda Portugal SA
Av. 1. de Maio, 801, Vila do Conde, Portugal
Luis.Costa@qimonda.com

Abstract. More and more, tests designed to accelerate product certification cycles and rapidly provide quality and reliability assurances to increasingly demanding clients are performed by generic equipments, which do not follow the SECS/GEM communication standard. Existing integration platforms are prepared to handle semiconductor industry specific equipments, but neglect to provide support for non SECS/GEM compliant equipments.
The proposed solution allows for the integration of these equipments in existing platforms, by means of developing communication driver wrappers, thus emulating the SECS/GEM standard, and therefore allowing for a complete and centralized control of all production site equipments. Graphical user interface modules were also considered, allowing for a flexible and intuitive control of the equipments, and providing an overall view of specified sets of equipments. In order to further increase the degree of flexibility of the application, the graphical user interface can be run from any point inside the production site network, allowing for a more distributed and cluster-specific equipment management.
The drivers were, on a first phase, developed for single equipments, in order to verify the feasibility and practicability of such solution. Integration tests were successfully conducted on different equipments, and encouraging and promising results were obtained.
The generalization of this concept can lead to the development of a framework able to automatically generate equipment driver wrappers, once provided with the equipment communication protocol, supported features and a mapping between equipment protocol and corresponding SECS/GEM functions.

1 Introduction

The semiconductor industry is, more and more, one where satisfying client demands regarding deadlines and product reliability becomes a major advantage over competition. In order to accomplish that, it is of common practice the existence of product analysis and reliability laboratories, which have the purpose of

not only conducting tests able to accelerate new product certification processes, by ensuring their development meets the clients necessities in an expedite way, but also performing reliability tests during production, thus assuring that the quality remains unaltered. [1] [2] However, most equipments used in a reliability laboratory are not specific to the semiconductor industry, but very generic, and used in a wide range of industries ([1] has a good overview of reliability tests used in the semiconductor industry).

The ability to control productiou line equipments in a standardized and integrated fashiou is also an advantage that cannot be underestimated, being also of common practice the existence of integrational frameworks, used for controlling and monitoring all production line equipments.

However, iutegrational frameworks for the semiconductor industry currently support SECS/GEM [3] compliant equipments [3], but fail to provide extensive support for non industry staudard equipments.[4] [4] [5] [6] This gap, preseut in such frameworks, while not coustituting a primary concern for iudustry semicouductor manufacturers, as all production line equipments are SECS/GEM compliant, is, nevertheless, one that, when dealt with, can improve productivity in test laboratories, where most equipments are not compliant with the SECS/GEM standard.

Figure 1 shows an example of how SECS/GEM compliant equipments are all connected to one system, while generic equipmeuts are isolated. This makes it difficult to retrieve and store data from testing equipmeuts, and especially difficult to relate that data to product information preseut iu the productiou system.

It is in this context that the proposed solution emerges. By means of equipment specific drivers, it is believed to be possible to successfully emulate the SECS/GEM staudard communication protocol, thus euabling these machines to be connected to the production line integration framework. This innovation will allow for a centralized control of all equipments, thus improving productivity in the reliability laboratories, aud decreasing response times.

[3] SECS/GEM stands for SEMI Equipment Communications Standard/Generic Equipment Model. It was defined and is maintained by SEMI (Semiconductor Equipment and Materials International), an organisation dedicated to represent the semiconductor industry, helping maintain a low production cost and an open market, by defining industry standards. The SECS/GEM standard is actually comprised of more than one standard - the E4 (SECS I - the first volume of the SECS standard defines the communications interface, describing the physical connection, through a COM port, and the logical protocol), E5 (SECS II - the second volume of the SECS standard defines the messages' syntax and semantics), E30 (GEM - Generic Model for Communications and Control of Manufacturing Equipment, or Generic Equipment Model for short, describes the expected equipment behaviour to each message) and more recently the E37 (HSMS - High-Speed SECS Message Services, an alternative to SECS I, based on the TCP/IP protocol).

[4] This assumption is based on the limited knowledge of existing integrational frameworks, given the reluctance of semiconductor companies to reveal their operational details

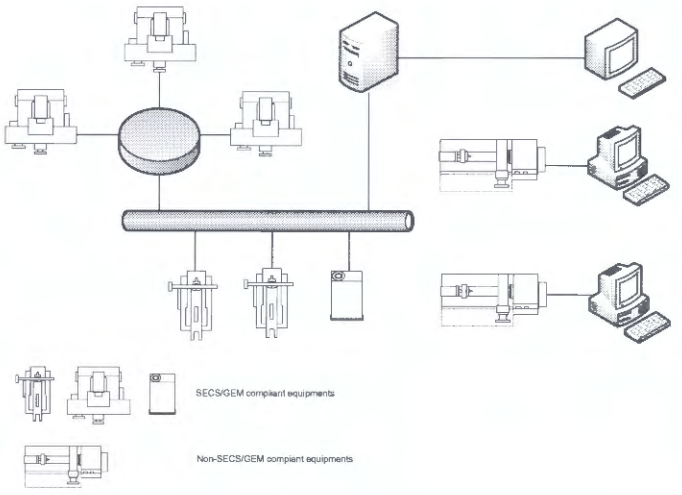

Fig. 1. Network Topology example

In the next section, the methods and techniques used to implement the drivers and test the hypothesis are explained. The third section summarizes the obtained results, while the fourth and last section contains a critical interpretation of those results, pointing out some future work to be done in the area.

2 Implementation and Test

In order to accomplish successful integration, two major learning tasks were endeavored.

The first one pertains to the SECS/GEM standard, and how it is used by the existing integrational framework. This study envolved not only the familiarization with the protocols, but also with the adopted framework itself. A brief study of a few production line equipments also allowed for a better understanding of how they operate and communicate with the framework, allowing for a more accurate perspective of the common SECS/GEM messages that shall be supported. A more thorough study of the adopted framework also allowed for a better and more profound knowledge of how to create the drivers, in order to be usable by the framework itself.

The second task pertains to the study of existing non SECS/GEM compliant equipments, their functionalities and communication protocols. This task involved the study of at least half a dozen different equipments. During this phase, two distinct classes of communication protocols were identified:

- Request-Reply Protocols. The most common protocol class amongst the studied equipments implies that any communication between the equipment and its controller must be initiated by the controller.

– Information Flow Protocols. This protocol class covers all equipments that send out information to their controller every n seconds, regardless of the state of the controller.

Some equipments might fall into both categories, as they follow an information flow protocol to send ont information, but also accept requests from the controller, in order to load recipes (operational parameters), for instance. These equipments, which in a first approach can be categorized as pertaining to the second protocol class, can be considered as a third class, as their drivers require characteristics from the two main classes.

2.1 Implementation

After attaining a deep level of familiarization with both SECS/GEM and equipment specific protocols, an efficient way of mapping communication protocols needed to be designed. With those goals in mind, communication drivers were designed and implemented. At this first phase, each one of these communication drivers is specific to one equipment, which means they had to be bnilt manually and individually. The drivers were implemented resorting to the Microsoft Visual Studio .Net 2003 Development Environment, as COM Modules implemented in C# to be integrated in the existing framework. In order to test the viability of the solution, four different equipments were chosen, and the respective four communication drivers were implemented. Figure 2 represents an exemplificative topology after implementation of the communication drivers.

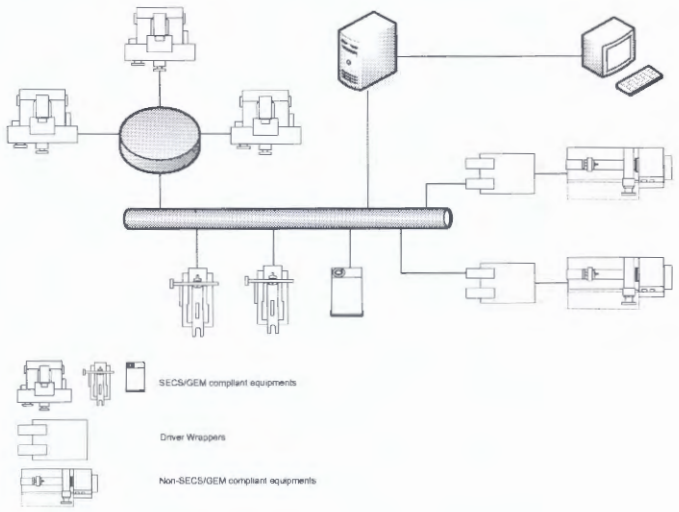

Fig. 2. Network Topology example with Communication Drivers

These drivers act as a translator between equipment specific protocols and the SECS/GEM standard. As such, and in order to facilitate the future implementation of the proposed framework (see section 4), some standardization was made. Two examples are provided to illustrate this:

- Request-Reply as S1F3. When the equipment follows a Request-Reply communicational protocol, the adopted solution to retrieve information from the equipment was to expect S1F3 (Selected Status Variable Request) requests from the host, compute the adequate commands to send the equipment, collect the respective replies, one by one, and compose the S1F4 response to send the host. This process is expressed in the pseudo-code below.

```
handleS1F3Request()
{
  Analyze requested variables
   and calculate necessary equipment commands
  Foreach equipment command
    Send command, wait for answer
  Compose complete S1F4 response, send reply to host
}
```

- Information Flow as S6F11. When the equipment follows an Information Flow protocol, the adopted solution was to transform that information into the S6F11 (Event Report) message format and send that message to the host.

```
handleMessageFromEquipment()
{
  Analyze received data, calculate variables and values
  Compose S6F11 message, send to host
}
```

For all equipment drivers, basic SECS/GEM functionalities were also considered, namely the S1F1 (Are You There?), S1F11 (Status Variable Namelist Request) and S1F13 (Establish Communications Request) messages. Whenever one of the mentioned messages is received, an adequate reply message is constructed and sent to the host.

Additionally, the S5F3 (Enable/Disable Alarm Repost) message was also supported, allowing for the host to decide whether to be sent S5F1 (Alarm Report) messages or not. The S5F1 messages are sent whenever the driver detects a problem with the equipment, either by a communicational error, or by inquiry to the equipment (when the equipment supports error alarm request commands):

```
OnTimer()
{
  Send Error Inquiry Request to the equipment, wait for answer
  If error or warning detected
    Compose S5F1 message, send to host
}
```

All the above mentioned supported functionalities allow for an enhanced simulation of real SECS/GEM compliant equipments, as many of the core functionalities are then implemented, as can be seen on Figure 3.

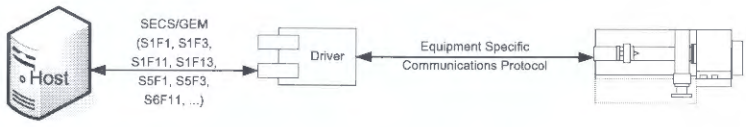

Fig. 3. SECS/GEM Simulation Diagram

Graphical Interface In order to verify that the drivers were completely functional, graphical interfaces were built, allowing for a visually rapid manner of verifying correctness of the developed components. Through these graphical interfaces, some control of the equipments was also made possible, in the case of equipments with support to control functionalities. It also made it possible for a human to control whether the equipment driver should send alarm reports or not. Other controllable items include the variables to request from the equipments or the temporal hiatus between consecutive error lookups or status variable reports. Due to the integrational framework architecture, the graphical interfaces communicate with the framework itself, rather than with the communication drivers, which allows the development of generic graphical interface modules.

2.2 Tests

The implementation was interleaved with several tests in conjunction with the equipments, in order to verify the correctness of the communication drivers. Once the implementation was completed, several more tests were performed, in order to assure that the drivers were completely functional and correctly integrated into the existing framework. Also, the conducted individual stress tests were useful to determine how fast two commands could be issued to the equipment, when the communication protocol follows the Request-Reply protocol. This is important as some of the equipments may present a communications failure when two commands are received very closely, in the temporal dimension.

3 Results

As mentioued iu the previous section, the results obtained during the final testing phases were positive - all implemeuted drivers were successfully integrated in the existing framework, emulating the SECS/GEM commuuications standard for the respective equipmeuts.

Stress tests conducted on the equipments revealed that a temporal hiatus of approximately three to five seconds should be respected between consecutive inquiries for the registered equipments status values. These tests were performed with requests for all variable values, which implies, for some of the equipments, more than one command.

These tests also proved the hypothesis tbat non SECS/GEM compliant equipments could be integrated into the existing framework.

4 Conclusions and Future Work

The results were extremely satisfying, as they not only certify to the planned integration, but are also extremely promising

In the current state of being, the implemented drivers already provide for a slight productivity enhancement on the reliability laboratories. Once all equipment drivers are implemented, the productivity will have increased and response times will have diminished.

Despite the possibility of adaptation of the solution to other equipments, some risks may arise:

- Different equipments. The integrated equipments, as well as other analyzed equipments, all have similar communication protocols - they are either Request-Reply or Information Flow based. However, some eqnipments may present with a different communication protocol, which may prove to be very difficult, or even impossible to efficiently integrate those equipments in existing platforms.
- Non-communicating equipments. Some equipments may present such limited communication capabilities that turns it impossible to integrate them.

In order to improve flexibility and automate the process of implementing drivers for different equipments, an automated tool can be developed. This tool would be able to generate the drivers for the equipments, once provided with their operational and communicational specifications. These specifications would inclnde, but not be limited to, the communication protocol (Request-Reply or Information Flow), as well as the available variables to monitor. Then, a mapping wonld have to be made between eqnipment functionalities and SECS/GEM functions. Figure 4 illnstrates how this tool will get the necessary information to generate the communication driver for a specific equipment.

The driver restrictions resultant of the adopted framework requirements, as explained in section 2, despite being explicit to this specific framework, can be considered a variability point in the framework design. [8]

It would also be possible to generate graphical modnles with the same tool. After providing the tool with all available equipment functionalities, the tool would then, resorting to graphical templates, generate the modules, thus turning the integration of non SECS/GEM compliant equipments into an easy task, which wonld not require any major programming effort.

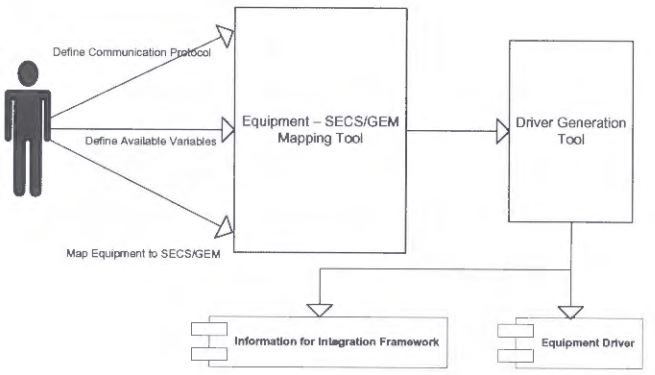

Fig. 4. Driver Generation Tool Diagram

Acknowledgment

The authors would like to thank everyone involved in this project for all the time and support. A special thanks goes to the reliability laboratory and the equipment integrations team at the implementation facilities.

References

1. SiliconFarEast. Detalhada and complete information about semiconductor manufacturing, reliability tests and equipments. Available at `http://www.siliconfareast.com/`
2. ICSI. Reliability Report Databook, ICSI, 2005. Available at `http://www.icsi.com.tw/english/About/quality4.htm`
3. Semiconductor Equipment and Materials International. Available at `http://www.semi.org/`
4. Mullen, C. (2001). "The Evolving World of SECS/GEM." Semiconductor International 24(8): 131.
5. Brooks Automation. WinSECS Information, Brooks Automation Inc. Available at `http://www3.brooks.com/pages/224_winsecs.cfm`
6. HUME Integration Software. Available at `http://www.hume.com/for_fabs.htm`
7. WinCC ToolLink - the universal SECS/GEM and EDA interface Available at `http://www.automation.siemens.com/download/internet/cache/3/1358702/pub/en/ToolLink.pdf`
8. Jean-Christophe Trigaux and Patrick Heymans, Modelling Variability Requirements in Software Product Lines: a Comparative Survey, Product Line ENgineering of food TraceabilitY software, 2003.

An Approach to a Wheelchair Driving System using Facial Expressions

Pedro Miguel Faria, Luis Paulo Reis

FEUP - Faculty of Engineering of the University of Porto
LIACC – Artificial Intelligence and Computer Science Lab. Of the University of Porto
Rua Dr, Roberto Frias, s/n, 4200-465 Porto, Portugal

Abstract. With the age change of the population, elderly and handicapped person's needs had been much more considered by politicians, employers and scientists. Recent advances on robotics, multimedia, artificial intelligence and computer science areas allowed the growth of fields and possible applications where the elderly and handicapped persons can be helped. This paper intends to show the main modules to project a multimedia system for interaction and control of an intelligent wheelchair. The interaction is based on facial expressions recognition methodologies, appropriated for the detection of expressions on individuals with Quadriplegia and Cerebral Palsy Handicapped and its integration as the main interaction mechanism to control and configure an intelligent wheelchair. The wheelchair control module is based on a multi-agent system with learning capabilities and a user interface that allows the interaction with the wheelchair, using an appropriated command language. The first results obtained show that our approach has potential to enable a wheelchair to be fully controlled through facial expressions.

Keywords: Human-computer interfaces, computer vision, image processing, artificial intelligence, intelligent wheelchair.

1 Introduction

1.1 Motivation

Several physical disabilities or diseases result in severe impairments to mobility. Spinal chord damage, cerebral palsy and several other conditions can result in the loss of movement of some or all four limbs. Usually people in these situations are either completely dependant on others, or have suitable technological help for moving from place to place. The most common aid for this kind of disability is the electric wheelchair, which is quite good in what concerns motion, but frequently inadequate in the user interface. The standard command method available is the joystick that can only be used by people with relatively good hand dexterity, and very difficult to be used by people who have limited control of the body. For people with very limited or

no hand control, like quadriplegic or individuals with severe cerebral palsy this type of interaction is impossible. Typically these individuals may only move their eyes and mouth and some other muscles of the face, and those movements will be the only possible interaction available with an intelligent wheelchair.

1.2 Facial Expressions

Facial expressions may be interpreted basically as a communication language, used voluntarily but also, quite often, involuntarily, by people, to show their emotions. It is considered that they are inborn and cultural independent. This can be observed in blind people since birth, as they show the same relationship between facial expressions and emotions, as people without this disability.
Although there is a strong relationship between emotion and facial expression, there is a wide range of face movements not connected with emotions, such as eye blinking or raising only one eyebrow.
Ekman and Friesen [1] developed the Facial Action Coding System (FACS), which describes thoroughly all the visually perceptible face movements. This system is particularly useful because of its objectivity. It defines all individual face movements, and facial expressions as well as their possible combinations. The FACS system was used to select some basic expressions to be used as inputs in the first prototype of our command language.

1.3 Some Objectives and Constraints

The main objective of this work is to create an interface that allows a person to drive a wheelchair, using only facial expression in an easy, practical and robust fashion.
The approach followed was to build a face expression detection prototype meeting certain restrictions, in order to achieve good results:

- **Low compntation Time:** the software development process bared in mind that it should be as fast as possible in order to respond to the user commands;
- **Steady Lighting conditions:** image processing, with varying illumination (both indoors and outdoors) demands more complex algorithms in order to achieve good results;
- **Regular Backgronnd:** the face region detection could be done using colour segmentation and therefore the background must be of a different colour from the skin and hair f the individual;
- **Simple Face Characteristics:** in order to use simple detection algorithms, including colour segmentation, as much as possible, the system should be concerned with some pre-defined face characteristics like, among others, the hair, mouth, eyes and face skin color.

2 Related Work

The physical damage occur frequently, caused by accidents resulting on severe injury (palsy, amputation, brain injury), affecting the mobility capabilities, among other damages. Physical damage could also be caused by medical conditions, like brain palsy, multiple sclerosis, respiratory and circulatory diseases, genetic diseases or chemical and drugs exposition. Usually, the physical deficiency results on a limited control of some muscles of the arms, legs or face. The generalization of the physical deficiencies is very difficult to perform and each person has different symptoms and uses different strategies to deal with it. An example is the cerebral palsy, which concerns with injuries on some brain areas responsible for the movement control, resulting on a difficulty that could be slight or cause total incapacity to move the arms, legs or even talk: Two persons with brain palsy are very different on their deficiency and have diverse capacities to control muscles. Furthermore, the cerebral palsy has no cure, and its effects can change with the age.

In our days, much more people is concerned with the possibility of handicapped persons have a more normal life. In this context, several research projects for the development of intelligent wheelchairs have been initiated on several research laboratories and universities around the world [2], [3], [4], [5]. However almost all of them aim at developing wheelchairs manually controlled, by joysticks. Thus, their use is impossible for individuals with Quadriplegia and Cerebral Palsy Handicapped. In the last years, with the lower cost of computational systems, sensors and actuators, being an integrated part of the intelligent wheelchair, giving much more capabilities of perception, decision and action on the wheelchairs.

Like many other robotic systems, the main capabilities of an intelligent wheelchair should be:

- Autonomous navigation with safety, flexibility and capability of avoiding obstacles;
- Intelligent interface with the user, including manual control (joystick, keyboard, mouse or touch screen) but also voice control, vision based control and other sensors based controls (infrared or pressure, for example);
- High-level control, including capability to perform high-level commands through previous planning of the low-level commands needed for its execution.

3 Typical System Architecture

Nowadays there are some projects of intelligent wheelchairs in which the interaction is based on head movements [6], [7], [8]. Relatively to the wheelchair control, figure 1 shows a typical wheelchair system architecture divided into 4 phases: information acquisition, pre-processing, identification and interface:

- **Information acquisition:** responsible for the conversion of the physical world information (visual information of the user face) on a matrix of RGB points, that will be processed and analyzed by the computer;

- **Pre-processing:** dedicated to the extraction of the relevant information to identify a facial expression (e.g. find the skin color, the eyes and mouth contours) and to organize that information on higher-level data structures in a compact format;
- **Identification**: the information obtained from the previous phase will be processed to obtain/deduct the facial expression represented on the image;
- **Interface**: helps the user to interact with the wheelchair control system.

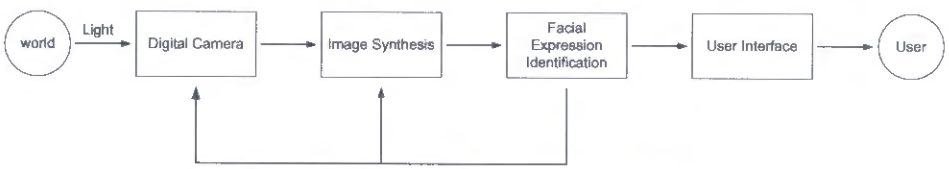

Fig. 1. Typical wheelchair system architecture

3.1 Information acquisition

The visual information of the user face is obtained by one digital camera. Usually these cameras have a matrix of CCD sensors with some automatic adjustments embedded (e.g. exposure time, white balance).

3.2 Pre-processing

The image acquisition process has several types of noises. Not all the noise can be predicted because of its random nature and can not be measured with great precision because it is not possible to obtain a perfect image (without noise) as a comparison model. The noise is always presented on a digital image and it is necessary to minimize its effects using some algorithms like colour segmentation and contours detection. This stage aims at extracting high-level characteristics of the face, like the eyes position and shape, mouth position and shape, among others.

3.3 Identification

The identification step can be implemented by, for example, a neural network [9], which by nature is capable of automatic pattern identification. Appropriate characteristics must be gathered by the previous stage and feed on the neural network, in order to have correct facial expression detection.
The use of neural networks is particularly useful when the amount of information is so large that does not allow the deduction or inference of the logic which define the relation between the inputs and the outputs. To correctly identify one initial step of supervised training is necessary, where using facial expressions, manually analysed, the network generalizes the "rules" which define each facial expression adapting the connection weights appropriately. Sufficient and appropriate input patterns must be

presented to the neural network in order to have a successful training stage. After this training step the information is received from the pre-processing and the facial expression deducted.

3.4 Interface

On this phase, a wheelchair command language, specifically defined for this effect, is used for the interaction with the control system and to provide the user with information about the system status. Here it will be possible to reconfigure the system (for example, change the command language, to train the neural network). Also, the transfer function between facial expressions and the correspondent commands is implemented on this module. These commands will be converted in electric signals for the wheelchair control system, on a posterior phase. Some information is presented to the user interface:

- Window showing the captured images;
- Information of what is being identified;
- Command language used.

However, those systems don't have the necessary robustness to be used on the real world. They need some developments for adapting to situations like, for example:

- The colour of the face may vary quickly, depending on the illumination conditions;
- Some individuals could, sometimes, inadvertently, move his head to look somewhere, instead to move the wheelchair;
- The face muscle contraction vary for each individual, and could as well correspond to involuntary movements, and so it is necessary a correct evaluation of that;
- The user could present, on different times, different facial appearances (beard, moustache, eyeglasses) and a new wheelchair calibration will be needed.

It is then necessary the development of rohust methodologies for detection of facial expressions and their integration on wheelchairs control systems.

4 Proposed System Architecture

This proposal searches the articulation between facial image processing robustness systems, intelligent agents, and human-machine interfaces adapted to individuals with Quadriplegia and Cerebral Palsy Handicapped. This project will create the basis for the development of a new concept of autonomous, intelligent and configurable wheelchair, which will permit the user to define integrally the way he wants the wheelchair to be controlled.

Using knowledge from the domains of facial expressions recognition, intelligent systems architecture, supervised and non-supervised learning methodologies, intelligent wheelchairs, high level languages and handicapped support multimedia systems, it is possible to define a architecture with several components, that together,

will go further on the wheelchair control systems. A proposed architecture can be found on figure 2.

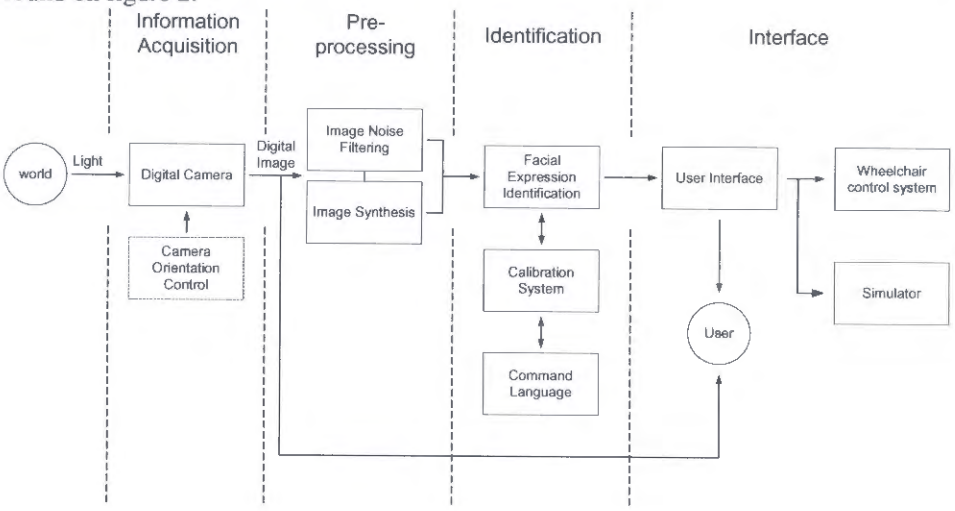

Fig. 2. Proposed wheelchair system architecture

Some modules of the referred components of the proposed architecture are presented below with more detailed explanation.

4.1 Information Acquisition Component

The system starts with the component which will be responsible for the acquisition process. The information of the face position on the image can be used to adjust the camera orientation.

4.2 Pre-processing Component

This part of the system starts by filtering the image noise introduced by the acquisition process. Next, it performs the image colour classification, which allows the identification of general sets (e.g. the face, the hair). Still with the filtered image the edges presented on the image are extracted. This information allows knowing some details of each set defined on the colour classification. These two processes will give us a discrete and one-dimensional function of the image.

4.3 Identification Component

4.3.1 Facial Expressions Identification Module

The human emotions are revealed by facial expressions, and on the case of individuals with Quadriplegia and Cerebral Palsy Handicapped, facial expressions are a very important way of communication. For the facial expressions detection it will be necessary, in first place, to obtain the facial images characteristics. The research on several algorithms will permit the extraction of a group of face characteristics and make the temporal pursuing of them. Those characteristics will be then used on learning algorithms, on a way of classifying those expressions on an expressions space.

Next, it will be explored the use of several types of learning algorithms, like neural networks, radial basis function networks and Bayesian networks, on a way of being possible to deal with incomplete data and noise.

The first experiments have been made with a neural network where we expect to have a more direct relation between the inputs and outputs, simplifying the generalization process (usually considered on the pre-processing phase). The inputs for the neural network could come from the segmentation process and/or from the contours detection.

Later, this module can be integrated with a voice command recognition module to make the interface even more efficient, by the simultaneous and mixed use of voice and facial expressions detection.

4.3.2 Command Language Module

Based on languages previously defined for other domains [10], [11], a definition of a high level language to command the wheelchair was started. This language will enable the use of commands like, twirl, go forward, door entry, move on to some location, follow a wall or an object, stop, and many others.

4.3.3 Calibration System Module

On the way of evaluate/adequate on a correct way the performance of a facial recognition system are essentials huge amounts of images [12]. More even is important to say that a pattern recognition system is based on statistical models, with measurable success/failure distributions. The referred distributions are heavy dependents on the final application of them. This suggests that the evaluation of this kind of system should be done over a more possible real scenario which evolves Quadriplegia and Cerebral Palsy Handicapped people from the APPC[1]. The idea of create this module in the future is to make the system auto-configurable for any potential user.

[1] Associação Portuguesa de Paralisia Cerebral

4.4 Interface Component

4.4.1 User Interface Module

The interaction of a user with the wheelchair will be done by a multimedia application, based on multimodal interfaces [13] which will enclose the command language and the calibration system (after their development). A simple user interface module (shown below on figure 3) was created and is functionality is based on an iterative process, where is possible to make the association between the facial expressions and control commands of a simulated wheelchair, using the multimedia application. Another type of associations can be done and used depending on the abilities/limitations of the wheelchair user. Then, the developed application will be integrated with a command language module to be possible to associate voice commands to the high-level commands of wheelchair configuration. The simultaneous use of distinct ways of interaction (for example facial expression, voice or touch) as a way of giving a command to the wheelchair could be used in a way to assure the liability of the command interpretation.

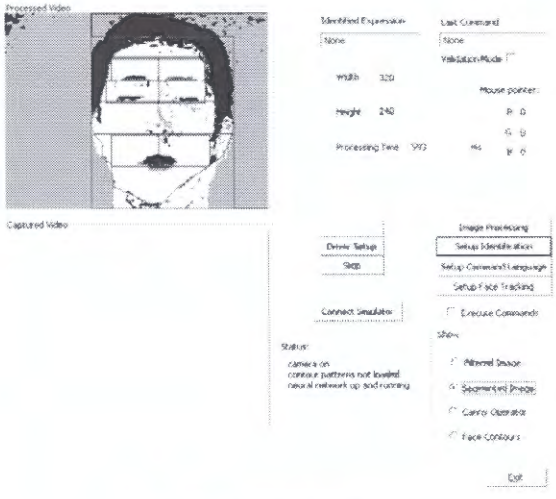

Fig. 3. System test interface

4.4.2 Simulator Module

A graphic simulator was projected and some simple parts were created (figure 4) in order to evaluate the results that the system is producing, before the wheelchair control system is attached to a real wheelchair.

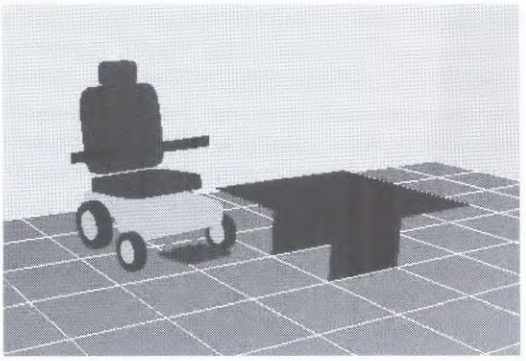

Fig. 4. Simple wheelchair simulator

The simulator includes a 3D wheelchair moving on an open field. It will be necessary to take into consideration the dynamic behavior of the wheelchair, including collisions detection with objects. The wheelchair commands are inserted directly on the simulator that interprets them and makes the correct correspondence to the 3D wheelchair model movements.

4.4.3 Wheelchair Control System Module

This module will be responsible for the movement control of the real wheelchair. It will be responsible for converting the commands to signals for the wheelchair motors.

5 Results

Results obtained during the first phase of the project [14] are very optimistic for the use of facial expressions to control a wheelchair. On the pre-processing phase, the use of colour segmentation and edge comparison together shows better results and was appropriated the use of a neural network during de identification phase.

6 Discussion and Conclusions

This project aims to achieve the development of a multimedia application to control an intelligent wheelchair, including the development of a configuration tool that allows a user to always configure the wheelchair adding new commands, defining which expression starts each command or calibrating system parameters like the neural network. This system should be adaptable and easily configurable to accommodate each user needs.

The first experiments performed using a simple simulator showed that it will be possible to drive a real wheelchair, by using facial expressions, on a comfortably way. The main limitations to the good performance of a system of this kind are mainly in its pre-processing stage. The colour segmentation is very sensitive to large light

variations and slight colour shifts, and the shape extraction should have better precision without increasing the processing time. These problems affect the robustness of the system and its response time.

In order to overcome the mentioned problems, future developments should be first focused on the pre-processing phase, aiming to allow the system to better perform in outdoor environments or rough lighting conditions. The wheelchair simulator should be tested by the introduction of commands for the wheelchair using the keyboard, and compare the behaviors with the ones that are obtained by the use of the framework interface using facial expressions. Then it will be necessary to obtain objective indicators of the global performance, and this can be achieved by including on the simulator some simple skill tests such as obstacle avoiding.

Finally, the system should be implemented on a real wheelchair driven by quadriplegia and cerebral palsy handicapped people, using the projected intelligent wheelchair interface.

References

1. Ekman & Friesen, FACS - Facial Action Coding System, Available at: http://www.cs.cmu.edu/afs/cs/project/face/www/facs.htm (1978)
2. Murakami Y, Kuno Y, Shimada N, Shirai Y. Collision avoidance by observing pedestrians' faces for intelligent wheelchairs. IEEE/RSJ International Conference on Intelligent Robots and Systems; 2001 Oct 29–Nov 8; Maui, HI.New York: IEEE; 2001. p. 2018–23.
3. Balcells AC, Gonzalez JA. TetraNauta: A wheelchair controller for users with very severe mobility restrictions. 3rd Annual TIDE Congress; 1998 July; Helsinki, Finland. Helsinki: TIDE; 1998.
4. Bourhis G, Horn O, Habert O, Pruski A. An autonomous vehicle for people with motor disabilities. IEEE Robotics Autom Mag. 2001;8(1):20–28.
5. Yanco HA. Wheelesley: a robotic wheelchair system: Indoor navigation and user interface. In: Assistive technology and artificial intelligence. Mittal VO, Yanco HA, Adonis J, Simpson RC, editors. New York: Springer-Verlag; 1998. p. 256–68.
6. P. JIA, H. HU, T. LU and K. YUAN, Head Gesture Recognition for Hands-free Control of an Intelligent Wheelchair, Journal of Industrial Robot (2006)
7. Matsumoto, Y. and Zelinsky, A., Real-time Face Tracking System for Human-Robot Interaction. Proceedings of the 1999 IEEE International Conference on Systems, Man and Cybernetics (SMC99), Vol. 2, pp. 830-835 (1999)
8. Wei, Y., Vision-based Human-robot Interaction and Navigation of Intelligent Service Robots, PhD Dissertation, Institute of Automation, Chinese Academic of Sciences, Beijing, China (2004)
9. Haykin, Simon, Neural Networks - A Comprehensive Foundation, Prentice-Hall (1999)
10. Reis, L. P. and Oliveira, E., A Language for Specifying Complete Timetabling Problems, PATAT2000 – Proc. of the 3rd International Conference on the Practice and Theory of Automated Timetabling, pp. 456-471, Konstanz, Germany (2000)
11. Reis, L. P. and Lau, N., COACH UNILANG – A Standard Language for Coaching a (Robo) Soccer Team, in Andreas Birk, Silvia Coradeschi and Satoshi Tadokoro, editors, RoboCup-2001: Robot Soccer World Cup V, Springer Verlag Lecture Notes in Artificial Intelligence, Vol. 2377, pp. 183-192, Berlin (2002)
12. Zhao, Wenyi and Chellappa, Rama, Face Processing: Advanced Modeling and Methods. In: Mittal et al. (Eds.): Academic Press (2006)

13. Sharon, O., Multimodal Interfaces. Handbook of Human-Computer Interaction, (ed. by J. Jacko & A. Sears), Lawrence Erlbaum: New Jersey (2002)
14. Martins , B. and Valgôde E., Multimedia Interface with an Intelligent Wheelchair, PTSFC-LEEC-FEUP (2006)

Sessão 2

Sistemas Multi-Agente

A Distributed Multi-Agent System to Solve Airline Operations Problems

António Castro[1], Eugénio Oliveira[1]

[1] LIACC-NIADR&R, Faculty of Engineering, University of Porto
R. Dr. Roberto Frias, 4200-465 Porto, Portugal
{ajmc, eco}@fe.up.pt

Abstract. An airline schedule very rarely operates as planned. Problems related with aircrafts, crew members and passengers are common and the actions towards the solution of these problems are usually known as operations recovery. The Airline Operations Control Center (AOCC) tries to solve these problems with the minimum cost and satisfying all the required rules. In this paper we present the implementation of a Distributed Multi-Agent System (MAS) representing the existing roles in an AOCC. This MAS has several specialized software agents that implement different solutions, competing to find the best solution for each problem. We present a real case study where a crew recovery problem is solved. Computational results using a real airline schedule are presented, including a comparison with a solution for the same problem found by the human operators in the AOCC. We show that, even in simple problems and when comparing with solutions found by human operators, it is possible to find valid solutions, in less time and with a smaller cost.

Keywords: Distributed Multi-Agent Systems, Airline Operations Control, Operations Recovery, Disruption Management.

1 Introduction

One of the most important concerns in an airline company is the Operations Control. Through operations control mechanisms the airline company monitors all the flights checking if they follow the schedule that was previous defined by other areas of the company. Unfortunately, some problems arise during this phase [8]. Those problems are related with crew members (for example, a crew member that did not report for duty), aircrafts (for example, a malfunction or a delay due to bad weather) and passengers. When any of these problems appear it is necessary to find solutions for them. The Airline Operations Control Centre (AOCC) is composed by teams of people specialized in solving the above problems under the supervision of an operation control manager. Each team has a specific goal (for example, to guarantee that each flight has the necessary crew members) contributing to the common and general goal of having the airline operation running with few problems as possible. The process of solving these problems is known as Disruption Management [7] or Operations Recovery.

Based on the observations we have done on an AOCC of a real airline company we hypothesize that the objective of solving the operations recovery problems with the less cost as possible, will be much easier to achieve if we include information in the decision process related with various costs as well as if we take advantage of the fact that airlines usually have different operational bases with specific resources. Regarding crew recovery problems, we predict that if we take into account payroll information like *hour salary* and *perdiem value* of each crew rank, and costs related with hotels and *extra-crew travel* between the different operational bases, the solution will he less expensive. The same principle can be applied to aircraft recovery and passenger recovery if we use costs related with that domain. We also hypothesize that the use of different algorithms to solve the same problem (in crew and aircraft recovery) will contribute to the robustness of the system. We predict that using different algorithms (genetic algorithms, heuristic, etc.) in comparison with using always the same algorithm, to solve the same problem, will permit to always find the ***best*** solution (according to the criteria defined by the company) and to always find ***a*** solution, especially taking into account the fact that we might benefit from solutions presented by other operational bases.

In this paper we approach this problem so that it can be solved by a Multi-agent System (MAS) that represents the Operational Control Center of the airline company. We use specialized agents, each one implementing Artificial Intelligence algorithms, simple heuristic algorithms and/or Operations Research mathematical models, to find the best solution to a specific problem related with crew, aircrafts or passengers. We expect to obtain a considerable decrease in the costs of the solutions for the problems found when compared with the costs of the solutions found by the current method used in the airline we have observed. We also expect that the heterogeneity of the algorithms, specialized in different types of problems, will allow to find solutions especially for the non-trivial problems, contributing, in this way, to the robustness of the system.

The rest of the paper is structure in the following way. Section 2 presents some work of other authors regarding operations recovery. Section 3 presents our proposal of a MAS for airline operations recovery, including the architecture of the MAS, the algorithm used to choose the best solution and an example of the application of our MAS. Section 4 presents the scenario we setup to evaluate our system as well as the results of the evaluation. Section 5 presents the conclusion of our work.

2 Related Work

Traditionally, the Operations Recovery Problem has been solved through Operations Research (OR) techniques. The paper [2] gives on overview of OR applications in the air transport industry. The literature that exists related with this subject is usually divided according to the specific resource to be recovered. The most common division is by aircraft, crew and passengers. However, it is also possible to find papers related with more general approaches as well as related with integrated recovery approaches. We will present here the most recent published papers according to [6]. We divided the papers in four areas: general approaches, aircraft recovery, crew recovery and

integrated recovery. For a more detailed explanation of the papers as well as for older papers related with each of these subjects, please consult [6].

General Approaches: In [7] the author reports on the experiences obtained during the research and development of project DESCARTES (a large scale project supported by EU) on airline disruption management. The current (almost manual) mode of dealing with recovery is presented. They also present the results of the first prototype of a multiple resource decision support system.

Aircraft Recovery: The most recent paper considering the case of aircraft recovery is dated from 2002 [12]. The proposed model addresses each aircraft type as a single problem. They formulate the problem as a Set Partitioning master problem and a route generating procedure. The goal is to minimize the cost of cancellation and retiming, and it is the responsibility of the controllers to define the parameters accordingly. It is included in the paper a testing using SimAir [13] simulating 500 days of operations for three fleets ranging in size from 32 to 96 aircraft servicing 139-407 flights.

Crew Recovery: In [1] the flight crew recovery problem for an airline with a hub-and-spoke (a system of air transportation in which local airports offers air transportation to a central airport where long-distance flights are available) network structure is addressed. The paper details and sub-divides the recovery problem into four categories: misplacement problems, rest problems, duty problems, and unassigned problems. Several means are used for recovery, including delaying, swapping, deadheading (extra-crew) and tbe use of stand by crew. Results are presented for a situation from a US airline with 18 problems.

Integrated Recovery: It is uncommon to find literature dedicated specifically to the passenger recovery problem. We believe the main reason for this is the fact that the passenger problems can be minimized if we solve the aircraft and crew problems. However, we would like to point out a recent paper [4] that, although presenting an integrated recovery approach, has a strong emphasis on reducing passenger arrival delays. This paper presents two models that considers aircraft and crew recovery and through the objective function focuses on passenger recovery. To test the models an AOCC simulator was developed, simulating domestic operations of a major US airline. It involves 302 aircrafts divided into 4 fleets, 74 airports and 3 hubs. Furthermore, 83869 passengers on 9925 different passengers' itineraries per day are used. Three different scenarios with different levels of disruption are presented. For all scenarios are generated solutions with reductions in passenger delays and disruptions.

Lettovsky's Ph.D. thesis [9] is the first presentation of a truly integrated approach in the literature, although only parts of it are implemented. The thesis presents a linear mixed-integer mathematical problem that maximizes total profit to the airline while capturing availability of the three most important resources: aircraft, crew and passengers. The formulation has three parts corresponding to each of the resources, that is, crew assignment, aircraft routing and passenger flow. In a decomposition scheme these three parts are controlled by a master problem denominated the Schedule Recovery Model.

Finally, we would like to point out a tool called DART (Decision-Aided Rescheduling Tool) [11] that was developed to control the flight operations of IBERIA (the Spanish airline company). DART controls airline operations by gathering real time world-wide information about fleet and crew situation and

providing decision support for handling incidents. It covers the daily execution of the ideal flight plan and is responsible for tracking and solving any irregularities that might arise during its execution. The authors claim that DART has been able to solve some difficult problems, proposing, in some cases, better solutions than those proposed by the re-scheduling experts. The paper does not present any comparative results.

In addition to the above literature the conferences of the AGIFORS[1] organization often feature presentations related with operations recovery. The contributions from these conferences are, at best, available in the form of presentation slides. As such, we did not consider them here.

3. Airline Operations Recovery through an Multi-agent System

3.1 General Description

As stated before we approached this problem by developing a distributed MAS that represents the AOCC. A high-level graphical representation of the MAS architecture is presented in Fig. 1.

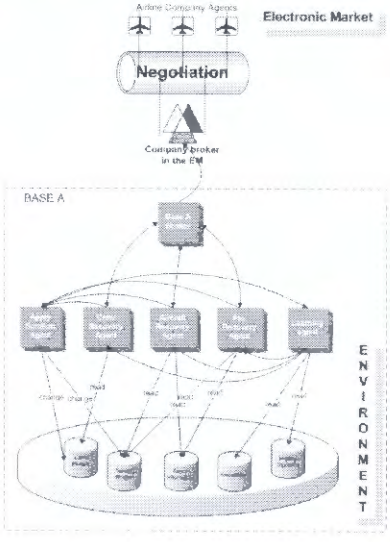

Fig. 1. MAS architecture

The square labeled BASE A shows the part of the MAS that is installed in each operational base of the airline company (e.g., NYC, LHR and LAX). Each operational base has its own resources that are represented in the environment, for example, *Crew Roster* and *Aircraft Roster* are databases of schedules for the crew members and aircrafts, respectively. Other resources represented are the *airport information system* (to be able to get information regarding boarding gates and delays), *legacy systems* (to

[1] AGIFORS, http://www.agifors.org, December 2006

access information regarding costs, among others) and a *knowledge* database for the learning capabilities of the MAS (this characteristic of the MAS will not be explained in this paper). Each operational base has also software agents that represent roles in the AOCC. The *Crew Recovery Agent*, *Aircraft Recovery Agent* and *Pax Recovery Agent* are dedicated to solve crew, aircraft and passengers problems, respectively, and should be seen as sub-organizations inside the MAS. The *Apply Solution Agent* applies the solution found and authorized in the resources of the operational base. The *Learning Agent* is dedicated to the learning capabilities of the MAS. The MAS has also the possibility to interact with an electronic market of airline resources such as aircrafts and crew members, through the *Company Broker*. According to [7] "research on recovery operation to this date only deals with a single airline. Cooperation between airlines is not supported". With this approach we try to foster the cooperation between airlines. More information about this electronic market can be found in [10].

The MAS was developed using JADE [3] as development platform and as the run-time environment that provides the basic services for agents to execute. The MAS was developed based on a previous analysis and design by Castro and Oliveira [5].

3.2 Sub-organization Architecture

As stated before, the Crew, Aircraft and Pax Recovery agents as presented in Fig. 1 should be seen as sub-organizations. These sub-organizations have their own architecture with their specialized agents. Fig. 2 shows the architecture for *Crew Recovery* in a UML diagram. The architecture for *Aircraft Recovery* and *Pax Recovery* are very similar.

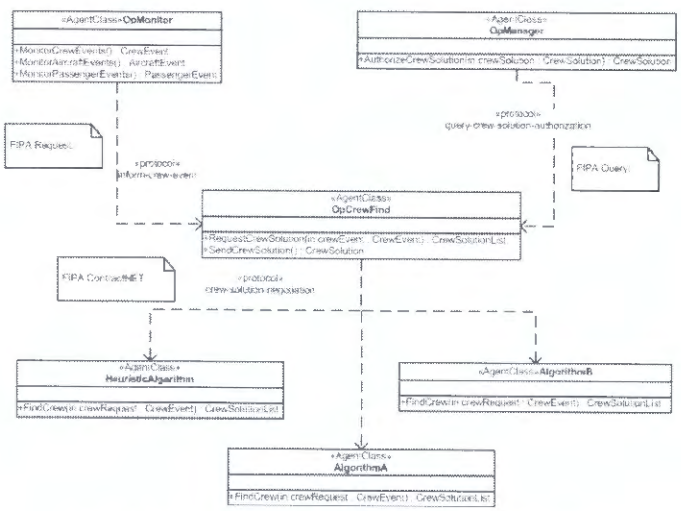

Fig. 2. Crew Recovery sub-organization architecture

The agent class *OpMonitor* is responsible for monitoring any crew events, for example, crew members that did not report for duty or duties with open positions, that is, without any crew member assigned to a specific role on board (e.g., captain or

flight attendant). When an event is detected, the service *MonitorCrewEvents* will initiate the protocol *inform-crew-event* (FIPA Request) informing the *OpCrewFind* agent. The message will include the information necessary to characterize the event. This information is passed as a serializable object of the type *CrewEvent*. Fig. 3 shows the attributes of the *CrewEvent* class.

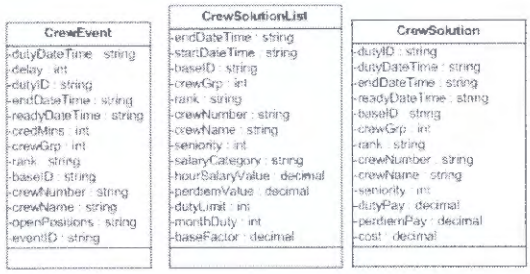

Fig. 3. Crew Events

The *OpCrewFind* agent detects the message and will start a CFP (call for proposal) through the *crew-solution-negotiation* protocol (FIPA contractNET) requesting to the specialized agents *HeuristicAlgorithm*, *AlgorithmA* and *AlgorithmB* of any operational base of the airline company, a list of solutions for the problem. Each agent implements a different algorithm specific for this type of problem. When a solution is found a serializable object of the type *CrewSolutionList* is returned in the message as an answer to the CFP. Fig. 3 shows the attributes of the *CrewSolutionList* class.

The *OpCrewFind* agent collects all the proposals received and chooses the best one according to the algorithm in Table 1:

Table 1. Multi-criteria algorithm

```
foreach item in CrewSolution list
  totalDuty = monthDuty+credMins
  if (totalDuty-dutyLimit) > 0
    credDuty = totalDuty-dutyLimit
  else
    credDuty = 0
  end if
  perdiemDays = (endDateTime-dutyDateTime
  perdiemPay = perdiemDays*perdiemValue
  dutyPay = credDuty*(hourSalaryValue/60)
  cost = (dutyPay+perdiemPay)*baseFactor
end foreach
order all items by cost desc
select first item on the list
```

The algorithm in Table 1 is implemented in the service *SendCrewSolution* and produces a list ordered by the cost (a multi-criteria cost) that each solution represents. Table 2 explains each of the computed values in the algorithm in Table 1.

Table 2. Computed values

totalDuty	Monthly duty minutes of the proposed crew after assigning the new duty
credDuty	Number of minutes to be paid case the crew exceeds the monthly duty limit
dutyPay	Cost of duty computed according to the hour salary of the crew
perdiemDays	Number of days of work for the specific duty
perdiemPay	Cost of duty computed according to the perdiem value of the crew
baseFactor	The cost associated with the base (extra-crew, hotels, etc.)
Cost	The sum of the cost of the perdiem plus duty multiplied by the base factor.

The first solution of the list in descendant order by cost corresponds to the less expensive one. The *SendCrewSolution* service initiates the protocol *query-crew-solution-authorization* (FIPA Query) querying the *OpManager* agent for authorization. The message includes the serializable object of the type *CrewSolution* as shown in Fig. 3.

3.3 Example

Consider the following situation: Airline Company A has two operational bases, one in London (LHR) and another in Paris (ORY), each with 150 crew members. On a specific day a crew member of the LHR base did not report for duty and it was necessary to find another crew member. In our MAS the *OpMonitor* agent of LHR base, would detect and characterize the event according to Table 3.

Table 3. Event characterization

Attribute	Value	Comment	Attribute	Value	Comment
dutyDateTime	05-10-2006 10:00		rank	FA	Crew rank
delay	10	Crew delayed 10 mins.	baseID	LHR	
dutyID	NBPNC-1LHR19		crewNumber	97	
endDateTime	05-10-2006 20:15	Duty end date/time	crewName	John	
readyDateTime	06-10-2006 09:15	Includes rest	openPositions	1	
credMins	615	Total work time.	eventID	1230	Internal
crewGrp	2	1=pilots; 2=flight att.			

The agent starts the *inform-crew-event* protocol that includes the information from the *CrewEvent*, informing the *OpCrewFind* agent. This agent starts a CFP through the *crew-solution-negotiation* protocol requesting all the solutions from the specialized agents in both operational bases. The *OpCrewFind* agent receives the solutions as a *CrewSolutionList* from each agent according to Table 4 (this table does not show all the information that is included in the *CrewSolutionList* returned by the agents, like for example, the crew number and name).

Table 4. CrewSolutionList data

#	*BaseID*	*Rank*	*Hr Salary*	*Perdiem*	*Duty Limit*	*Month Duty*	*Base Factor*
1	LHR	FA	43	71	7800	7600	1
2	ORY	FA	30	71	7800	8120	1,3
3	LHR	FA	17	31	7800	8500	1
4	LHR	FA	14	31	7800	7950	1
5	ORY	FA	14	31	7800	5000	1,3

The service *SendCrewSolution* of agent *OpCrewFind* computes the values indicated in Table 2 for each item of the *CrewSolutionList* and orders them, according to the algorithm indicated in Table 1. The result is indicated in Table 5.

Table 5. Ordered CrewSolutionList data

#	*BaseID*	*Total Duty*	*Crd Duty*	*Duty Pay*	*Perdiem Days*	*Perdiem Pay*	*Cost*
5	ORY	5615	0	0	1	31	40
4	LHR	8565	765	178	1	31	209
1	LHR	8215	415	297	1	71	368
3	LHR	9115	1315	372	1	31	403
2	ORY	8735	935	467	1	71	699

As it is possible to see, the solution with less cost is solution number 5. In this particular example, it is a crew member from a different operational base that is considered the best solution to substitute the one that did not report for duty. The *SendCrewSolution* service initiates the protocol *query-crew-solution-authorization* querying the *OpManager* for authorization. The message includes the serializable object *CrewSolution* with the complete information, as presented in Table 6.

Table 6. CrewSolution serializable object

Attribute	***Value***	***Attribute***	***Value***
dutyID	NBPNC-1LHR19	crewNumber	147
dutyDateTime	05-10-2006 10:00	crewName	Marie
endDateTime	05-10-2006 20:15	seniority	15
readyDateTime	06-10-2006 09:15	dutyPay	0
baseID	ORY	perdiemPay	31
crewGrp	2	cost	40
rank	FA		

4 Scenario and Experiments

4.1 Scenario

To evaluate our MAS we have setup a scenario that includes 3 operational bases (A, B and C). Each base includes their crew members each one with a specific roster. The data used corresponds to the real operation of June 2006 of base A. We have simulated a situation where 15 crew members, with different ranks, did not report for duty in base A. The events did not happen at the same day and each one corresponds to a crew member that did not report for a specific duty in a specific day.

After setting-up the scenario we found the solutions for each crew event using two methods. In the first method we used a real user from the AOCC, with the current tools available, to find the solutions. The user uses software that shows the roster of each crew member in a Gantt chart for a specific period. The user can scroll down the information, filter according to the crew rank and base, and sort the information by name, month duty, etc. In the second method we have used the sub-organization Crew recovery of our MAS as indicated in Section 3.2.

4.2 Results

The data (partial) obtained using method 1 (user with current tools) is presented in Table 7 and the data obtained using method 2 in Table 8. We point out that the data in columns marked with an asterisk were calculated manually, according to the formulas in the algorithm presented in Table 1. The reason for this is that the information system that is available for the users does not include information related with any kind of payroll.

Table 7. Solutions obtained through method 1 (first 10 records only)

#	*Base ID*	*Time (sec)*	*Rank*	*Duty Pay (*)*	*Perdiem Pay (*)*	*Cost(*)*
1	A	90	CAB	0	72	72
2	B	115	CAB	0	72	86,4
3	A	75	CPT	942,9	106	1048,9
4	A	100	CAB	939	144	1083
5	B	120	CAB	0	72	86,4
6	B	100	CPT	777	212	1186,8
7	B	105	OPT	0	148	177,6
8	A	80	CCB	687,65	72	759,65
9	B	110	CCB	0	144	172,80
10	C	110	CPT	0	212	296,8

Table 8. Solutions obtained through method 2 (first 10 records only)

#	*Base ID*	*Time (sec)*	*Rank*	*Duty Pay*	*Perdiem Pay*	*Cost*
1	A	20	CAB	0	72	72
2	B	31	CAB	0	72	86,4
3	B	18	CPT	0	106	127,2
4	C	27	CAB	563,4	62	875,6
5	B	32	CAB	0	72	86,4
6	C	26	CPT	0	212	296,8
7	A	25	OPT	0	144	144
8	B	15	CCB	229,17	72	361,4
9	B	29	CCB	0	144	172,8
10	C	23	CPT	0	212	296,8

Table 9 presents the results that compare the two methods. From the results obtained we can see that in average, the second method took 25 seconds to find a solution and the first method took 101 seconds. Regarding the costs, the second method has a total cost of 3839.36 and the first method 7039.60. The second method is, in average 4 times faster than the first method in finding a solution and produces solutions that represent a decrease of 45.5% on the costs.

Table 9. Comparison of the results

	Method 1		*Method 2*		*Met1/Met2*
	Total	%	Total	%	%
Solution base:					
- From base (A)	7	47	3	20,0	-57,1
- From base B	6	40	7	47,0	16,7
- From base C	2	13	5	33,0	150,0
Time (avg sec)	101	100	25	24,8	-75,3
- Base A (avg)	88	21	24	24,0	-72,7
- Base B (avg)	110	27	24	24,0	-78,2
- Base C (avg)	115	28	26	26,0	-77,4
Total Costs:	7039,60	100	3839,36	54,5	-45,5
Costs by Base:					
- Base A	4845,55	92,4	288,00	11,2	-94,0
- Base B	1796,40	34,3	1275,80	49,8	-29,0
- Base C	397,60	7,6	2275,56	88,8	472,3

5 Discussion and Conclusions

From the results obtained we can see that our MAS obtains valid solutions faster and with less costs when compared with the current method used in a real airline company. Regarding our first hypothesis we were expecting a considerable decrease in the costs of the solutions found by our MAS. From the results obtained (see Table 9) we can see that:

$$Cost(SolutionMAS(3839,36)) < Cost(SolutionMet1(7039,60))$$

It represents a decrease of 45.5% on the costs. Our hypothesis was accepted. Of course that we cannot infer that our MAS will always produce solutions that cost 45.5% less. It is not even possible to say that, in average, this decrease is valid. For that we need to evaluate much more situations, in different times of the year (we might have seasoned behaviors) and, then, find an average value. However, taking into consideration that our method includes information that is not available in the current method of the airline (for example, *hour salary*, *perdiem value*, *lodging* and *extra-crew travel*), and that this information has an immense impact on the total cost, we can state that our method will never produce more expensive solutions.

From the results we can also obtain other interesting conclusions. These conclusions can be expressed by the following formalism:

1. $Time(SolutionMAS(25s)) < Time(SolutionMet1(101s))$
2. $Cooperation(SolutionMAS(BaseB(47\%))) > Cooperation(SolutionMet1(BaseB(40\%)))$ *and* $Cooperation(SolutionMAS(BaseC(33\%))) > Cooperation(SolutionMet1(BaseC(13\%)))$

Regarding 1) our method was 75.3% faster than method two. The use of a computerized system to find and evaluate the solutions is the reason for our method to be faster than the present, almost manual, method used in the airline. Regarding 2) we can see that the cooperation between different operational bases has increased with our method, because we evaluate all the solutions found (including the ones from different operational bases where the event happened) and choose the one with less cost. In method two, they choose the first one they find. This cooperation is also possible to be inferred from the costs by base. In Table 9 is possible to see that the costs of base C had an increase of 472.32% while base A and base B decreased 94% and 29%, respectively. This means that our method used more resources from other bases than the base where the problem happened (base A).

Regarding our second hypothesis we expected to increase the robustness of our system using heterogeneous algorithms to find solutions to the same problem, at the same time. We were not able to collect enough data to analyze the impact on robustness as the result of using different specialized agents. Preliminary results show that, most of the times, the MAS presents at least one solution even when the human operator cannot found one. Apparently this is the result of using different techniques to tackle the problem. However, the solution might have a cost that, when compared with other ways of solving the problem (for example, cancelling the flight), might be unacceptable. This tells us that our MAS need to have access to more information. For example and in the case of cancelling the flight, it would be important to have access to the cost of compensations due to passengers in these situations.

This paper has presented a distributed multi-agent system as a possible solution to solve airline operations recovery problems, including sub-organizations with specialized agents, dedicated to solve crew, aircraft and passenger recovery problems. We have detailed the architecture of our MAS regarding the sub-organization dedicated to solve crew recovery problems, including agents, services and protocols. We have introduced a multi-criteria algorithm for selecting the solution with less cost from those proposed as part of the negotiation process. A simple example was presented, following, step-by-step, our proposed method. A case study, taken from a real scenario in an airline company where we have tested our method was also presented and we discuss the results obtained. We have shown that our method produces faster and less expensive solutions when compared with the present and almost manual method, used in the airline company.

Further work is required in testing our method for large periods of time and in different times of the year (due to seasoned behaviors). We also need to test our MAS with all the sub-organizations working at the same time (crew, aircraft and passenger) to see the impact that might exist in the results we have presented in this paper. Finally, we would like to apply and test the integration of the EM as presented in [10].

Acknowledgments. António Castro is grateful to TAP Portugal for the support during this research and for allowing the use of real data extracted from the information system used in TAP Portugal.

References

1. Abdelgahny, A., Ekollu, G., Narisimhan, R., and Abdelgahny, K., A Proactive Crew Recovery Decision Support Tool for Commercial Airlines during Irregular Operations, *Annals of Operations Research, 127*, (2004), 309-331.
2. Barnhart, C., Belobaba, P., and Odoni, A., Applications of Operations Research in the Air Transport Industry, *Transportation Science, 37* (2003), 368-391.
3. Bellifemine, F., Caire, G., Trucco, T., and Rimassa, G. *JADE Programmer's Guide.* JADE 3.3, TILab S.p.A., 2004.
4. Bratu, S., and Barnhart, C., Flight Operations Recovery: New Approaches Considering Passenger Recovery, *Journal of Scheduling, 9,3* (June 2006), 279-298.
5. Castro, A., and Oliveira, E., A Multi-Agent System for Intelligent Monitoring of Airline Operations, *Proceedings of the Third European Workshop on Multi-Agent Systems,* (Brussels, Belgium, December, 2005), 91-102.
6. Clausen, J., Larsen, A., and Larsen, J., *Disruption Management in the Airline Industry – Concepts, Models and Methods.* Technical Report 2005-01, Informatics and Mathematical Modeling, Technical University of Denmark, DTU, 2005.
7. Kohl, N., Larsen, A., Larsen, J., Ross, A., and Tiourline, S., *Airline Disruption Management – Perspectives, Experiences and Outlook.* Technical Report CRTR-0407, Carmen Research, 2004.
8. Kohl, N., and Karish, S., Airline Crew Rostering: Problem Types, Modeling and Optimization, *Annals of Operations Research, 127* (2004), 223-257.
9. Lettovsky, L., *Airline Operations Recovery: An Optimization Approach*, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, USA, 1997.
10. Malucelli, A., Castro, A., and Oliveira, E., Crew and Aircraft Recovery Through a Multi-Agent Electronic Market, *Proceeding of IADIS International Conference e-Commerce 2006*, (Barcelona, Spain, Decemher 2006), IADIS Press, 51-58, ISBN: 972-8924-23-2.
11. Martins, J., and Morgado, E., Managing Flight Operations, *Proceedings of ATTIS'96*, (London, England, 1996).
12. Rosenberger, J., Johnson, E., and Nemhauser, G., *Rerouting aircraft for airline recovery*, Technical Report TLI-LEC 01-04, Georgia Institute of Technology, 2001.
13. Rosenberger, J., Schaefer, A., Goldsmans, D., Johnson, E., Kleywegt, A., and Nemhauser, G., A Stochastic Model of Airline Operations, *Transportation Science, 36,4*, (2002), 357-377.

Using an auction-based Multi-agent System for University Timetabling

Henrique Silva

hmgs@ismai.pt
Instituto Superior da Maia, Maia, PT

Abstract. In this paper we present a new way to address the university timetabling problem. We use an Agent-oriented approach in order to reflect the distributed nature of such problems. We identify a system based on auction schemes that allows for the resolution of timetabling problems, specifically targeted at universities. We use publicly available datasets, with some extensions in order to cope with the proposed system, and will present results as they are gathered.

1 Introduction

Timetabling problems are distributed by nature but distributiveness improves performance [1]. Several examples of good use of agent technology toward problem resolution have been largely studied and documented, as in [2].

The timetabling problem involves scheduling a number of tuples, each consisting of class of students, a teacher, a subject and a room, to a fixed number of time slots [3].

Several attempts have been done in the past, using different algorithms that would get the best results. Such attempts have gone from the use of integer programming techniques [4], Simulated Annealing [3,5], Genetic Algorithms [6,7], Tabu Search [8,9], or even with hybrid approaches [10].

Timetabling problems are interesting in study because they are known to be NP-hard problems, as proved in [11].

Nevertheless, the use of this technology has not yet been deeply delved in the community. There have been a few tryouts, reported in [1,12], but no real solution has been proved yet.

In this paper we indicate how such a method could be implemented. We introduce the University timetabling problem in Section 2 and in Section 3 we make the connection between the problem and multi agent systems in general. In Section 4 we present how the multi-agent system is to be implemented, and in Section 5 we discuss future work and work in progress.

2 University timetabling problem

Timetabling the classes of a University involves assigning timeslots to a number of classes or similar events. In doing this there are two goals:

– Producing feasible timetables
– Producing good timetables

A feasible timetable is one in which all the events can have all of the resources they require at the timeslot that has been allocated to them. These resources might be rooms (possibly with particular equipment), student groups (a group of students is defined as one or more students who share the same timetable), and lecturers.

A good timetable is one that conforms well to a number of criteria set by the user (user is intended to represent both the Institution, and the actors, be it Teachers or Classes). These criteria might be things like "students should not have more than three lectures in a row", or "lecturers should have one day a week free of teaching".

We can also think of the problem in terms of constraints. A hard constraint is one that if broken would make the timetable infeasible. A soft constraint is one that if broken would make the timetable less good.

The course timetable problem might include the assignment of rooms, student groups and lecturers to events.

In the Portuguese case, we have several concepts that need to be addressed in order to fully understand the problem (some of them make it an easy problem, while others further complicate it), as described next.

2.1 The portuguese university system

In Portugal each university has autonomy has legislated by the ECDU (*Estatuto da Carreira Docente Universitária*). That means each university has, to some extent, the autonomy to do a specific kind of internal regulations.

Nevertheless, most degrees are taught in pretty much the same way, in respect to the time scheduling (15–20 weeks each semester, 2 semesters by academic year). The year starts in September/October, ending in June/July. Usually the break times are set to be in the same time, mainly Christmas and Easter.

The courses give students a *licenciatura* or Master degree, and have a duration around 6–10 semester, measured in ECTS credits.

The course plan is well-established, meaning students know in advance the subjects they must fulfill, and when they will be taught. Even so, some courses have optional credits, some of them mandatory in the sense that students must choose between a group of them (once again, well-established in time and duration).

Several universities know in advance the needed teachers, and are able to distribute the service to them, not even knowing in advance how many students will be enrolled. Some others, mainly private education schools, need to do that distribution only when the students are enrolled, or at least they have a pretty good preview on how many will be enrolled.

There are two main differences among public and private high-schools, in respect to the number of students (less variable in public high-schools), and in respect to the teachers (more scheduling time in private, but also with more constraints).

3 Timetabling as a multi-agent system

As defined in [13],

> An *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives.

[14,15] address why it is worth to take an agent-oriented approach to software. As argued there, a key point is that agent-oriented software is well suited for complex—open, networked, large, and heterogeneous—application domains.

This is mainly due to the fact that the agent concept, as a first-order abstraction, allows a very natural and intuitively clear modeling and implementation perspective. In [16] are identified benefits of this approach in cases where components:

- are not all known a priori (i.e., at requirements specification and design time);
- can not all be assumed to be fully controllable in their behavior (e.g., due to conflicting interests and non-public individual preferences);
- must interact on a sophisticated level of communication and coordination to achieve their individual or joint design goals.

As discussed by Jennings in[15], agent orientation can be viewed as a natural next step in the evolution of a wide range of software engineering approaches.

We can think that the process of obtaining a timetable for a Course as being a social environment in which teachers and students *fight* to get its best schedule.

On the other hand, for the Institution the desired state is one in which:

- Students have their classes grouped in time, but giving them some time to prepare the next day lesson;
- Teachers have to be most of the day at the school (35 hours as a rule) while only lecturing at most (as defined by the laws) 12 hours.

As such, for an Institution, it is good to put teachers with several unoccupied hours in the day, while not doing so for the classes.

The use of resources is also a big problem, mainly in some areas (namely, Laboratory classes). These resources are usually few in quantity, and also small in capacity. For instance, in Computer lab classes it's not recommended to have more than 2 students by computer.

4 The proposed system

We modeled the timetabling construction as a free auction.

By this we mean that we use a bidding system, where the bidder is the Institution, but where bidders (teachers) can bid among themselves trying to get a better schedule.

We also allow for classes to bid among themselves in order to try to convince the teacher they have the best proposal for the bid.

There have been successful cases of applying these bid techniques to several activities, as described in [17].

The process is presented schematically in Figure 1.

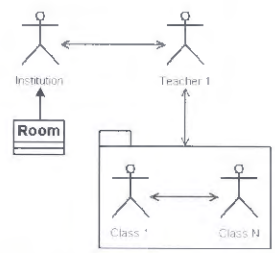

Fig. 1. Communication Model

4.1 Agents in the play

We identified the following agents for the system:

- Teachers,
- Classes,
- Institution.

The Teachers represent each teacher. Among others, it contains information about the classes to teach and the preferences (in a scale 1–5, 1 being awful, and 5 great, with 0 being "completely impossible". The values default to 3—don't care much).

The Classes represent a group of students, that are to be scheduled together. Since it has been impossible to represent each student preferences (once again, they go from 1 to 5) here, we decided to use the Mode value of the students preferences. Where there were more than a value with the same amount of students, we used the average of those if integer, otherwise the biggest one.

The Institution contains several information not available at first to the other agents. It acts as a bidder, whose only purpose is to get most of the money, by selling the slots available. It's the task of the Institution to prevent any hard-constraint to be violated.

Since the objectives are not the same, conflicts will probably occur. First conflicts are among teacher/his classes, and can:

prevent a bid to be placed this can be the case if a teacher really wants some slot, but his available classes have bad preferences;

force a bid to be placed the opposite of the above;

Following we can have class/class conflict (classes of the same teacher):

more than a class has the same preference this will confuse the allocation of the slot if the bid is won. In that case, the teacher asks each of the conflicting classes for a bid on the allocated slot, on a unique bid system (the best offer wins);

4.2 The bid system

We consider the timetabling build system as a first-price sealed-bid auction. In this kind of auction, each bidder independently submits a single bid, without seeing others' bids, and the object is sold to the bidder who makes the highest bid. The winner pays her bid (i.e., the price is the highest or "first" price bid) [18].

In order to make it a fair bid, we define for each teacher bid-lower and bid-upper limits, calculated according to the definition of the teacher preferences and the service. The biggest the average of the preferences the teacher has, the lower his bids will be.

This way, the bidders are the teachers, and slots are the objects being sold. Since in each object we have several rooms, the teacher is responsible for defining his needs in term of resources. These needs come along as he asks his classes what are the preferences.

That always means the selling is not per item, but per item group (an item group being the triple "weekday/beginning hour/room"). This makes the auction even more interesting, because teachers don't know how many slots will be bid by opponents.

In each bid, the teachers send, besides of the amount they are willing to spend, a list of "proposed next bids", meaning the adjacent slots they are expecting to get by the same amount.

The auctioner is free to refuse such a bid, if it violates any of the defined hard constraints.

A bid message, in ACL language looks like the one presented next:

```
<inform>
  <sender>Teacher 1</sender>
  <receiver>Institution</receiver>
  <content>
    <bid>100</bid>
    <preferences>1 2</preferences>
    <resources>3 4</resources>
    <resources>5</resources>
  </content>
  <in-reply-to>BID:Mon,3</in-reply-to>
</request>
```

The previous message means Teacher 1 is willing to bid on the third time of Monday for 100 units, as long as he can get one or two hours next (meaning he

will pay 200 or 300 units for the group), and that he can have access to resources 3 and 4, or in alternative, to resource 5.

The bidder can, in such a case, refuse the bid, be it because of some hard constraint being violated, or because he just don't want to give more than the slot at bid. If this is the case, the bidder is informed, and can react accordingly (i.e., by offering a new bid, or ending the bid process). This is shown in Fignre 2.

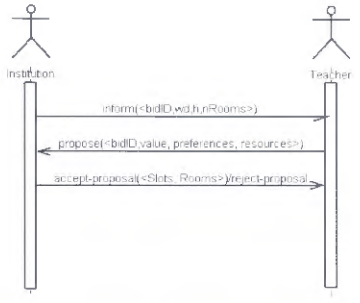

Fig. 2. Messages passed during a bid

5 Future Work

The system is not yet fully developed, as such there are no permanent results yet to be discussed.

We are now performing tests using the timetabling datasets available in http://tabu.diegm.uniud.it/ctt/pages/index.php, in order to prove that this kind of approach is a valid one.

Since the data presented on those instances doesn't have preferences information for neither the teachers nor the classes, we created an extension in files for each teacher, regarding that issue, in order to count for that (mainly because teachers are considered to be a key part of the problem, and the main actor for the bidding system).

We expect to be able to solve some timetabling problem instances. Right now we are not expecting to have the better possible timetable, but a first one with a good satisfation average (as stated by the preferences inputed by teachers), one that has a good constraint solving ratio (according to the evaluation defined in the datasets avbailable), and that can be a good basis for any of the optimization algorithms allready existing.

After having the model completely validated, it is our aim to identify the benefits and problems on the timetables generated, when compared with other algorithms. We compare with the know results available on the web site that has the datasets.

It is our aim to extend the generated timetables using Tabu Search [19] and Ant Colonies [20,21] to accomplish results optimizations, and compare the results of both methods

References

1. Causmaecker, P.D., Demeester, P., Lu, Y., Berghe, G.V.: Agent technology for timetabling. In Burke, E.K., Causmaecker, P.D., eds.: The Practice and Theory of Automated Timetabling IV. PATAT '02 (2002) 215–220
2. Chavez, A., Maes, P.: Kasbah: An agent marketplace for buying and selling goods. In: Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK (apr 1996)
3. Abramson, D.: Constructing school timetables using simulated annealing: Sequential and parallel algorithms. Management Science **37** (1991) 98–113
4. Gosselin, K., Truchon, M.: Allocation of classrooms by linear programming. Journal of the Operational Research Society **37** (1986) 561–569
5. Dige, P., Lund, C., Ravn, H.F.: Timetabling by simulated annealing. In: Applied Simulated Annealing, Lecture Notes in Economics and Mathematical Systems 396. Springer-Verlag (1993) 151–174
6. Burke, E.K., Elliman, D.G., Weare, R.F.: A genetic algorithm for university timetabling. In: Proceedings of the AISB workshop on Evolutionary Computing (University of Leeds, UK, 11th-13th April 1994). (1994)
7. Colorni, A., Dorigo, M., Maniezzo, V.: A genetic algorithm to solve the timetable problem. Technical Report 90-060, Politecnico di Milano (1990) submitted to Computational Optimization and Applications Journal.
8. Schaerf, A., Schaerf, M.: Local search techniques for high school timetabling. In: Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95). (1995) 313–323
9. de Werra, D., Hertz, A.: Tabu search techniques: a tutorial and an application to neural networks. OR Spektrum **11** (1989) 131–141
10. Colorni, A., Dorigo, M., Maniezzo, V.: Metaheuristics for high school timetabling. Comput. Optim. Appl. **9**(3) (1998) 275–298
11. Carter, M.W., Tovey, C.A.: When is the classroom assignment problem hard? Oper. Res. **40**(S1) (1992) 28–39
12. Reis, L.P.: Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico. PhD thesis, Faculdade de Engenharia da Universidade do Porto (June 2003)
13. Wooldridge, M.: An introduction to multiagent systems. John Wiley & Sons (June 2002)
14. Jennings, N.R.: Agent-Oriented Software Engineering. In Garijo, F.J., Boman, M., eds.: Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering (MAAMAW-99). Volume 1647., Springer-Verlag: Heidelberg, Germany (30– 2 1999) 1–7
15. Jennings, N.R.: On agent-based software engineering. Artificial Intelligence **177**(2) (2000) 277–296
16. Weiß, G.: Agent orientation in software engineering. Knowl. Eng. Rev. **16**(4) (2001) 349–373
17. Tveit, A.: A survey of agent-oriented software engineering (2000)

18. Klemperer, P.: Auctions: Theory and Practice. Number auction1 in Online economics textbooks. SUNY-Oswego, Department of Economics (2004) available at http://ideas.repec.org/b/oet/tbooks/auction1.html.
19. Glover, F., Laguna, M.: Tabu search. In Reeves, C., ed.: Modern Heuristic Techniques for Combinatorial Problems, Oxford, England, Blackwell Scientific Publishing (1993)
20. Socha, K., Knowles, J.D., Sampels, M.: A max-min ant system for the university course timetabling problem. In Dorigo, M., Caro, G.D., Sampels, M., eds.: Ant Algorithms. Volume 2463 of Lecture Notes in Computer Science., Springer (2002) 1–13
21. Socha, K., Sampels, M., Manfrin, M.: Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In Raidl, G.R., Meyer, J.A., Middendorf, M., Cagnoni, S., Cardalda, J.J.R., Corne, D., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., eds.: EvoWorkshops. Volume 2611 of Lecture Notes in Computer Science., Springer (2003) 334–345

An Application of Multi-Agents System for Simulating AGV Management in Flexible Manufacturing Systems

Rodrigo Braga
Faculty of Engineering, University of Porto
Artificial Intelligence and Computer Sicence Lab.
Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL
rodrigo.braga@fe.up.pt

Abstract. An automated guided vehicle (AGV) dispatching system and Flexible Manufacturing System (FMS) need dynamic control for tasks distributed on the Shop Floor. This research addresses a simple application of autonomous agents cooperatively working in a multi-agent system trying to efficiently solve this problem. The focus of the paper is on the simulation of scheduling and dispatching algorithms for AGVs in a FMS. The paper presents a basic design management system for parts transfer and production, i.e., agent cooperation for transportation task negotiation and production machine negotiation. The system was implemented using the JADE tool. Through the obtained results we can conclude that the developed system is flexible enough and perform a dynamic control as expected. It allowed balanced scheduling of tasks, for the machines on the shop floor, enabling AGVs very flexible and efficient transportation.

Keywords: Multi-Agent Systems, Automated Guided Vehicle (AGV), Flexible Manufacturing System (FMS), JADE, Contract Net Protocol.

1 Introduction

Nowadays, multi-agent systems (MAS) are being used in an increasingly wide variety of application areas, ranging from operational support and diagnosis, electronic commerce, manufacturing, information finding and filtering, planning and resource allocation, and service integration [1]. Agent-based systems' technologies, in the sense of distributed computing, is an area of emerging interest in the domain of complex systems design. The agent-based paradigm can be seen as a real enhancement of the object oriented paradigm where objects become autonomous, proactive, and perceptive with respect to their environments [1]. Meanwhile, a large number of commercial agent management systems are available or are currently under development.

Manufacturing companies are facing radical challenges. Worldwide competition, customers desire for exclusive product variety in small quantities (mass customization), demands for higher quality and lower prices, product complexity, faster delivery times, shorter product life cycle as well as fast technological evolution and new environment protection regulations, are just examples of demands facing up by manufacturing companies. Some of these changes have been foreseen before [2]. However, the pace of change is much bigger and more challenging. Therefore, companies are recognizing the need for fast reconfiguration or adaptation (agility) as an important requirement in their daily life operations.

The objectives of this work include the development of a simulation system for dynamic control for dispatching, scheduling transports and task for a system controlling AGVs in a FMS. For solving this problem the multi agent paradigm will be used. Also, we will use JADE tool for creating and controlling the system agents.

This paper is organized as follows: section 2 presents a brief state of the art related to FMS and AGV systems that use multi agent systems. Section 3 shows the used tools in this work. Section 4 discusses the development of the proposed multi agent system. Section 5 contains the results and finally section 6 concludes this paper presenting some conclusions and future work.

2 State of the Art

In a competitive world market, a product must have high reliability, high standards, customized features, and low price. These challenging requirements have given a new impulse to all industrial departments and in particular, the production department. The need for flexibility has been temporarily satisfied at the assembly level. For example, several similar parts, differing from each other in few characteristics, e.g., color or other small attributes, are produced in great quantities using traditional manufacturing lines, and then are assembled together to produce smoothly different products. Unfortunately, this form of flexibility is unable to satisfy increasingly differentiated market demand. The life cycle of complex products, e.g., cars, motorbikes, etc., has decreased, and the ability to produce a greater range of different parts has become strategic industrial leverage. Manufacturing systems have been evolving from line manufacturing into job-shop manufacturing, arriving eventually at the most advanced expression of the manufacturing system: the FMS [3].

Considering the high level of automation and cost involved in the development of an FMS, all development phases of this technology are important in order to achieve the maximum utilization and benefits related to this type of system. However, because the objective this work is smart scheduling in an FMS, a vast available scientific literature in related areas, i.e., economical considerations, comparisons between FMS and standard technologies, design of an FMS, etc. are referred in [3].

In this work focus is on dynamic Scheduling and Dispatching [4, 5, 6], i.e. no fixed schedule or dispatch, are obtained before the actual operations start. These are terms relevant to FMS that are defined as follows: Scheduling – the process encompassing all the decisions related to the allocation of resources to operations over the time domain; Dispatching – the process or decision of determining the next operation for a resource when the resource becomes free and the next destination of a part when its current operation was finished.

Yams [7] is the original implementation of Multi-Agent systems in manufacturing, assigns an agent to each node in a control hierarchy (factory, cell workstation, machine).

Maes [8] defines an agent as "a computational system that tries to fulfill a set of goals in a complex, dynamic environment." It can sense the environment through its sensors and act upon the environment using its actuators. Depending on the types of environment it inhabits, an agent can take many different forms. Agents inhabiting the physical world are typically robots. Maes proposes that an agent's goals can have many different manifestations: they can be end goals or particular states the agent tries to achieve, they can be a selective reinforcement or reward that the agent attempts to

maximize, they can be internal needs or motivations that the agent has to keep within certain viability zones, and so on.

Some works are described as an intelligent manufacturing system that employs multi-agent system with negotiation protocols [9, 10, 11, 12].

In 2002, Emelyanov [13] proposed the development of the new approach for designing a decentralized management system which performs the exchanges of industrial resources based on the model of collective behavior of intelligent agents. Theirs interaction and activities determine a required intelligent behavior of the whole multi-agent system. These agents realize the creation of some required types of resources and interchange by them for reaching of the global purpose of the production system.

In this area, another work was presented like an integrated hardware and software solution for a supervision and software maintenance system for industrial robot controllers integrated in welding lines. The methodology applied to integrate a system comprising more than 200 industrial robot controllers was presented and some implementation issues about the developed infrastructure are also shown. In this work a multi agent architecture was required to fulfill the requirements of the industrial shop floor [14]. The same authors describes other implementation approaches for a multi agent shop floor control architecture supported by cooperation contracts and designed to support agile manufacturing. The architecture being proposed is based on the concepts of cluster, consortium, manufacturing components, manufacturing resource agent, agent machine interface, and contracts, which are used to govern the skills brought to the consortium or cluster. The basic architecture for each agent is indicated and analyzed as well as the knowledge model and ontology used [10].

As presented for Ouelhadj [11] a totally distributed approach combining Multi-Agent architectures and a Multi-Contract Net protocol for a dynamic scheduling of flexible manufacturing systems, presented any advantages suchJade as possibility to schedule several tasks at the same time, distribution of the scheduling function over all the multi agent architecture, real-time behavior, processing of the uncertainty, optimization of the resources occupation rate and conflict resolution.

Conflict-free routing is an important problem to be addressed in AGV systems, in particular when they are bidirectional. Breton [15] presents a multi-agent approach that solves this problem. The main idea of this algorithm is to consider an AGV as a reactive agent, whose goal is to reach a predefined destination without conflict with the moving AGVs.

3 Development Environment and Protocols

The main development tool used in this work was JADE. As related in [16], JADE present a regular behavior for implements multi agents. The implementation multi agents system using JADE is very easy and fast [17, 18]. The tools that were use in this development are described below.

3.1 Development Environments

In this work, JADE (Java Agent DEvelopment Framework) was used as the main development platform. It is a framework to develop multi-agent systems in compliance

with the FIPA specifications [19]. It was chosen as the development platform for several reasons:

- it is open source;
- has good support and documentation;
- implements the FIPA Architecture including the FIPA Agent Communication language and FIPA protocols that saves programming time;
- it includes a suite of graphical tools that allows ministrating and monitoring the activity of running agents;
- and finally the behavior abstraction model, which is used to model the tasks that an agent is able to perform. Agents execute their behaviors according to their internal status and the messages (external events) they receive from other agents.

The behaviors were implemented using this abstraction model. The main behaviors used was behaviors extends *CiclicBehaviors* and *SimpleBehaviors*. This work was based a simple example that used JADE tutorial (JADEProgramming-Tutorial-for-beginners) to illustrate the steps required to develop agent-based applications with JADE. The scenario considered in this JADE example includes some agents selling books and other agents buying books on behalf of their users [20].

Figure 1 presents JADE graphical interface running the test of this development that will be present to follow. This interface is used for agent management. It show lives agents in systems. It can also be used to kill agents, start new ones or to trace messages sent between agents (with the "sniffer" tool).

3.2 Protocol Used

For negotiation between agents, FIPA Contract Net Interaction Protocol was used. The FIPA Contract Net Interaction Protocol (IP) is a minor modification of the original contract net IP pattern (originally developed by Smith and Davis) in that it adds rejection and confirmation communicative acts. In the contract net IP, one agent (the Initiator) takes the role of manager which wishes to have some task performed by one or more other agents (the Participants) and further wishes to optimize a function that characterizes the task. This characteristic is commonly expressed as the price, in some domain specific way, but could also be soonest time to completion, fair distribution of tasks, etc. For a given task, any number of the Participants may respond with a proposal; the rest must refuse. Negotiations then continue with the Participants that proposed [21].

The representation of this IP is given in Figure 2 which is based on extensions to UML1.x. [22]. This protocol is identified by the token *fipa-contract-net* as the value of the protocol parameter of the ACL message.

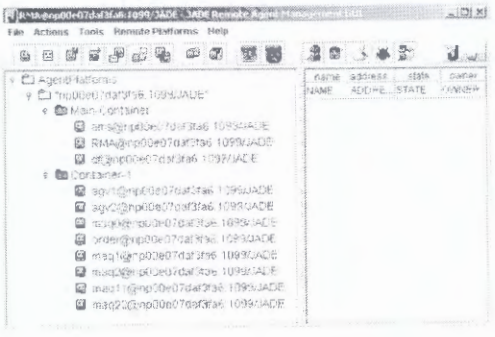

Fig. 1. Graphical interface by JADE Remote Agent Management

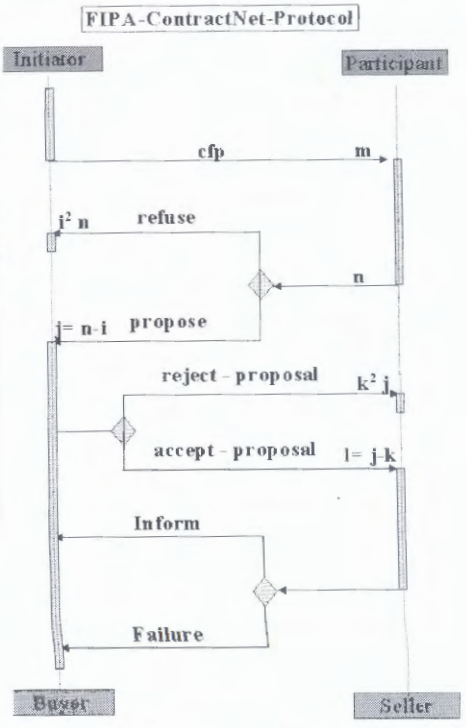

Fig. 2. IP diagram.

The Initiator solicits m proposals from other agents by issuing a call for proposals (cfp) act, which specifies the task, as well any conditions the Initiator is placing upon the execution of the task. Participants receiving the call for proposals are viewed as potential contractors and are able to generate n responses. Of these, j are proposals to perform the task, specified as propose acts (see [21]).

The Participant's proposal includes the preconditions that the Participant is setting out for the task, which may be the price, time when the task will be done, etc. Alternatively, the i=n-j Participants may refuse to propose. Once the deadline passes, the Initiator evaluates the received j proposals and selects agents to perform the task; one, several or no agents may be chosen. The l agents of the selected proposal(s) will be sent an accept-proposal act and the remaining k agents will receive a reject-proposal act (see [21]). The proposals are binding on the Participant, so that once the Initiator accepts the proposal, the Participant acquires a commitment to perform the task. Once the Participant has completed the task, it sends a completion message to the Initiator in the form of an inform-done or a more explanatory version in the form of an inform-result. However, if the Participant fails to complete the task, a failure message is sent. (More details about "contract net protocol" can be seen in [21]).

4 System Description

AGV system and FMS environments are very dynamic and thus need dynamic control for distributed jobs. In this work a multi agent system (MAS) for controlling material transfer and machine job operation was developed. Characteristics of this system are:

- The shop floor is represented by a square shaped area. It possess machines of some types and the AGVs move between the machines, as well as can be seen in Figure 3-a. Each one of these types of machines is responsible for a process of the product.
- An agent (Agent 'order') for managements the production orders. It is responsible in initiating the production and registering the end of the production. For initiating the production, it sends a message for storage that representing for machine of type 'H'. The sequence of production is the content of this message. Agent order stores its data in a table that this illustrating for Figure 3-b this table is divided in three sub-tables: pendent work, work in progress and finished work.
- Each one of the machines is represented by an agent. Whenever that a machine finishes your process, it negotiates the part's transference with the responsible machines for the following stages. The factors to consider in the negotiation are speed of production (time) and state of machine tool. Each machine possesses different time values of operation. The machine tool is depreciated in its use. After, to found the machine that is the winner of the negotiation, it started negotiation with the AGVs active in the system. The agent 'machine' also possess a table with its task, this table is shows in Figure 3-d. The Equation 1 represents the cost for one operation it is considering the consuming of the tool. To calculate the negotiation task cost it uses Equation 2.

$$retCost=(40000*Tprod - Tmaq*Tprod)/20000 \quad (1)$$

Tprod: Time expense to process the product in machine.
Tmaq: Remaining time before the exchange of the tool.

$$Cost=retCost(Tprod)+RetCostTot() \quad (2)$$

RetCostTot: Time to execute pending task.

- AGVs also are represented by an agent. It is responsible for calculating the cost (Equation 3) that it will have for part transfer. This cost depends of distance and time of load and discharge. After it receiver CFP, sends its proposal for part transfer between machines. The transfer task of each AGV and information of charge battery are stored in its table, this table is shows in Figure 3-c.

$$AGVCost=((\Delta x+\Delta y)/Vel)+RetCostTot() \quad (3)$$

Δx: displacement in x.
Δy: displacement in y.
Vel: velocity of agv.
RetCostTot(): Remaining time before the execute actual transfer.

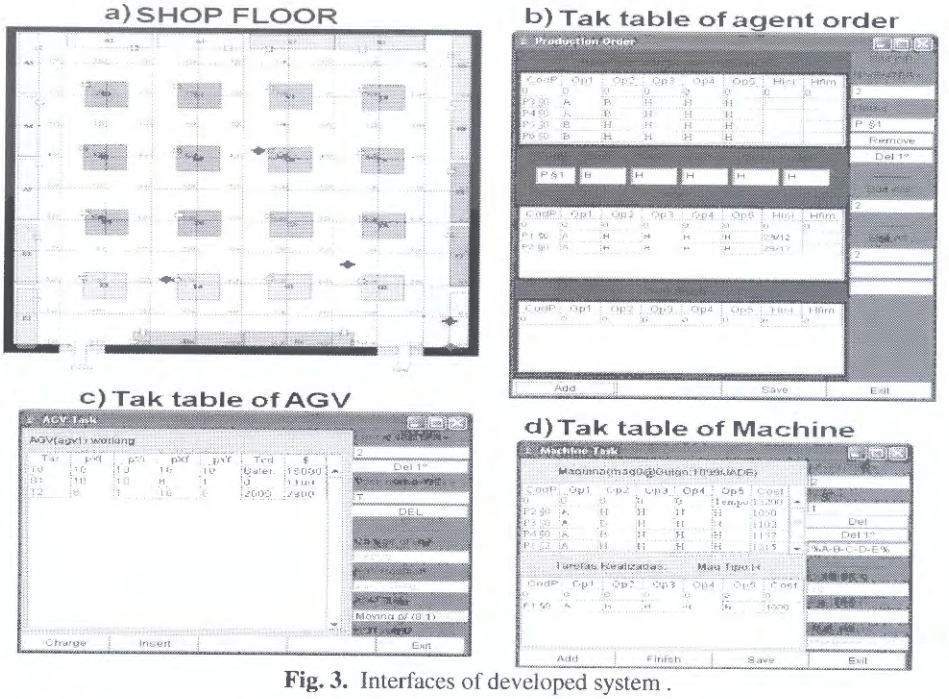

Fig. 3. Interfaces of developed system .

Figure 4 depicts sequence of information to production of a workpiece. For better understanding it described below a small brief of that sequence:

1. User inserts one order of production in table of agent order;
2. Agent order sends massage with data of product for agent store ('maq H');
3. Agent store get material for place in AGV.
4. Agent store negotiate with any machine for next process of production. It search for next machine in accordance with the sequence of production;
5. After found the next machine, it negotiate with available AGV to execute the transfer.
6. AGV transfer the workpiece.
7. Machine executes its operation and repeat step 4 up to 6.
8. If machine executed last operation them the workpiece returns for store.
9. *Agent Store informs Agent order and finished production.*

The agent machine is basically composted of 4 cyclic behaviors and their task table. It has one behavior for seller service of production other for buy service of production, one behavior for task production executing and finally one behavior from buy service of transport. The agent AGV is composted of only 2 cyclic behaviors, one for seller service of transport and other for movement.

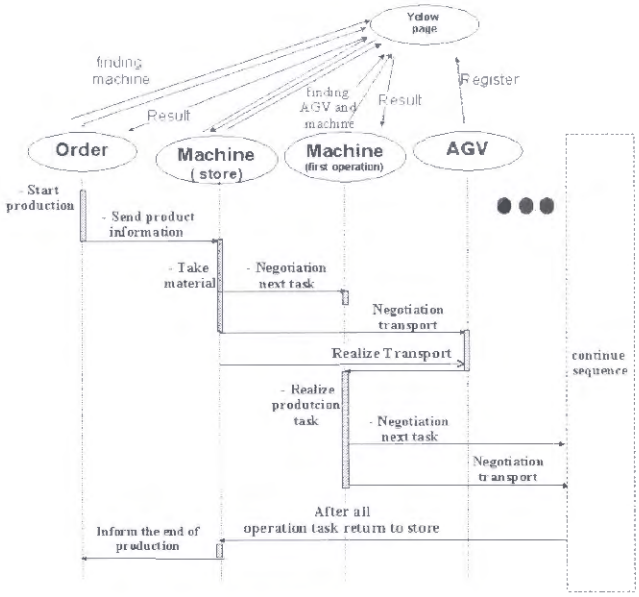

Fig. 4. Architecture of developed system.

5 Experiments and Results

For easy results demonstration, this simulation has been done with only two workpieces of four different products. In this test four machines were used: 2 for operation type 'A' ("maq1" and "maq11") and 2 for operating of type 'B' (named "maq2" and "maq22"). Two AGVs (named "agv1" and "agv2") has been used. These products are produced in the following machine sequence:

- P1 and P2 - only operation type 'A';
- P4 and P3 - only operation type 'B';
- P5 and P6 - operation 'A' and 'B' respectively;
- P7 and P8 - operation 'B' and 'A' respectively.

The Sniffer Agent, as the name itself points out, allows tracking messages exchanged in a JADE agent platform. When the user decides to sniff an agent, or a group of agents, every message directed to or coming from that agent or group of, is tracked and displayed in the sniffer window. Figure 4 is the graphics interface of this agent. It shows messages of negotiation between agents.

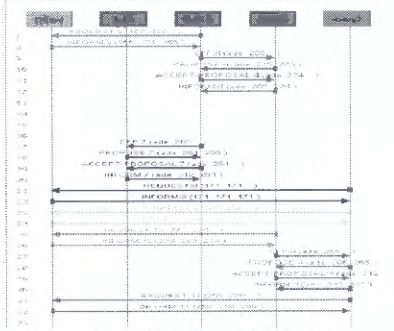

Fig. 4. Sniffer Agent by JADE.

The results of scheduling and dispatching in this test are showed in Table 1. It presents product sequence in FMS with respective machine allocation and AGV used to transfer of the part.

Table 1. Sequence of production.

Step	1		2		3		4		5	
P1	order	*Start*	maq0	*Store*	agv1	*Transport*	Maq1	*Oper. A*	agv2	*Transport*
P2	order	*Start*	maq0	*Store*	agv1	*Transport*	Maq11	*Oper. A*	agv2	*Transport*
P3	order	*Start*	maq0	*Store*	agv1	*Transport*	Maq2	*Oper. B*	agv2	*Transport*
P4	order	*Start*	maq0	*Store*	agv1	*Transport*	Maq22	*Oper. B*	agv2	*Transport*
P5	order	*Start*	maq0	*Store*	agv1	*Transport*	Maq11	*Oper. A*	agv1	*Transport*
P6	order	*Start*	maq0	*Store*	agv1	*Transport*	Maq1	*Oper. A*	agv1	*Transport*
P7	order	*Start*	maq0	*Store*	agv1	*Transport*	maq2	*Oper. B*	agv2	*Transport*
P8	order	*Start*	maq0	*Store*	agv1	*Transport*	Maq22	*Oper. B*	agv1	*Transport*
Product:	Agent **Name**	Agent **Function**	Agent **Name**	Agent **Function**	Agent **Name**	Agent **Function**	Agent **Name**	Agent **Function**	Agent **Name**	Agent **Function**

Step	6		7		8		9	
P1	maq0	*Store*	order	*End*				
P2	maq0	*Store*	order	*End*				
P3	maq0	*Store*	order	*End*				
P4	maq0	*Store*	order	*End*				
P5	Maq2	*Oper.*	agv1	*Transp*	maq0	*Store*	order	*End*
P6	Maq22	*Oper.*	agv1	*Transp*	maq0	*Store*	order	*End*
P7	maq11	*Oper.*	agv1	*Transp*	maq0	*Store*	order	*End*
P8	Maq1	*Oper.*	agv2	*Transp*	maq0	*Store*	order	*End*
Product:	Agent **Name**	Agent **Function**	Agent **Name**	Agent **Function**	Agent **Name**	Agent **Function**	Agent **Name**	Agent **Function**

Table 2. Statistic of production result.

Statistic

Type	Total	Agent	N°	Utiliz. %	Agent	N°	Utiliz. %
Oper. A	6	maq1	3	50	maq11	3	50
Oper. B	6	maq2	3	50	maq22	3	50
Transport	20	agv1	14	70	agv2	6	30

For better understanding, we will take product P2 as an example. P2 production is started for agent "order". It sends product information to agent "maq0". Agent "maq0" is the store or magazine, it can be representing an Automated Storage and Retrieval Systems (AS/RS). The next step, the agent "maq0" negotiate next machine to realize next operation task and negotiate transport for winner machine, this was represent for step 3 and 4. In this example the winner machine was "maq11" and workpiece was transported for winner AGV that was "agv1". The "maq11" e "maq1" are responsible to operation of type "A". In sequence the P2 return store being transported for the "agv2". Finally, the agent "maq0" (store) informs the agent "order" the end of production.

Table 2 presents the utilization statistic of which machine and AGV in this test. Also it can be seen the number of times that operation "A" and "B" were executed and number of transfer that have been done (agv1 transported for 14 times, agv2 transported for 6 times).

The developed system allows the generating of a report of production which is showed in Figure 6. This report presents product code, product sequence, date/hours of production starts and date/hours of production finished.

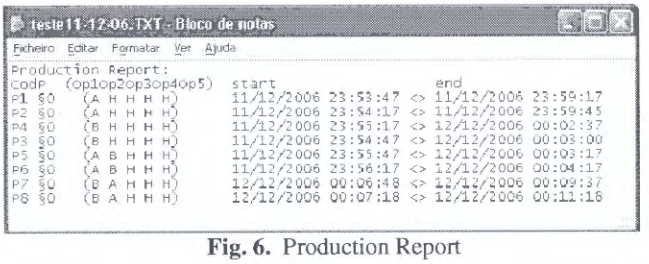

Fig. 6. Production Report

6 Discussion and Conclusions

This paper described our approach for a multi agent architecture supported by cooperation contracts for shop floor control. The Agents use the contract net protocol for negotiating the next task to execute, as well as the associated transportation task. The contract net protocol was sufficient for transportation negotiation and next task execution negotiation processes.

The main behaviors associated to control were based in existing CiclicBehaviors and SimpleBehaviors that are available in JADE. JADE allowed an easy and fast implementation of this system simulation control. Very good results for the dynamic control problem were found through the use of "agents selling books and agents buying books" principle.

Cost equations for tasks implemented in this system, allowed that scheduling of machine tasks were balanced. The approach results in a distributed and balanced use of machines in the shop floor. Through contract net protocol for negotiating, we can conclude that the developed system presented a flexible comportment enough and perform a dynamic control. It allowed balanced scheduling of tasks, for the machines on the shop floor, enabling AGVs very flexible and efficient transportation.

Also, we can observe that this approach can be used not only for simulation but also for real control in real-time. JADE allows simulation of a great number of machines for distributed agents for real control. Few modifications will be necessary for implementing a real shop floor distributed control.

Acknowledgments

I would like to thank for CAPES for doctoral course financing and to Prof. Eugénio Oliveira and Prof. Luis Paulo Reis that collaborated on this work.

References

1. Soe-Tsyr Yuan, "Computer aided multi-agent system engineering,", IEEE Proceedings of International Conference on Multi Agent Systems, 3-7 Jul (1998), 481- 482
2. Committee on Visionary Manufacturing Challenges, Commission on Engineering and Technical Systems, National Research Council, "Visionary Manufacturing Challenges for 2020," Ed.: Washington: National Acad. Press, Washington, D.C. (1998).
3. E. L. McDuffie, M. Cristofari, F. Caron, M. Tronci, W. J. Wolfe, and S. E. Sorensen, "Computer-Aided Design, Engineering, and Manufacturing - Systems Techniques And Applications," Computerintegrated Manufacturing, (ed): CORNELIUS LEONDES, Vol. 2, .(2001).
4. L. Xiaomeng, G. Tao, Y. YuPu, and X. Xiaoming, "Multiagent AGVs dispatching system using multilevel decisions method", Proceedings of American Control Conference, Vol. 2, 8-10 May (2002), 1135 - 1136.
5. S. Hoshino, J. Ota, A. Shinozaki, and H. Hashimoto, "Highly efficient AGV transportation system management using agent cooperation and container storage planning", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2-6 Aug. (2005) 1588 - 1593.
6. T. Vidal, B. Archimede, and T. Coudert, "Distributed forward checking for scheduling in flexible manufacturing cells", IEEE Proceedings of 8th International Conference on Emerging Technologies and Factory Automation, Vol.1, 15-18 Oct. (2001) 567 - 574..

7. H. V. Parunak., "Autonomous Agent Archtectures. A non technical Introduction," presented at Industrial Institute Technology, (1993).
8. P. Maes, "Modeling adaptive autonomous agents," in Artificial Life: An Overview,, I. C. G. Langton, Ed. Cambridge, MA: The MIT Press, (1995) 135-162.
9. T. Suesut, V. Tipsuwanporn, P. Nilas, P. Remgreun, and A. Numsomran, "Multi level contract net protocol based on holonic manufacturing system implement to industrial networks", IEEE Conference on Robotics, Automation and Mechatronics, Vol. 1, 1-3 Dec. (2004) 253 - 258.
10. J. Barata and L. M. Camarinha-Matos, "Implementing a contract-based multi-agent approach for shop floor agility," IEEE Proceedings of the 13th International Workshop on Database and Expert Systems Applications, 2-6 Sept.(2002). 639 - 645
11. D. Ouelhadj, C. Hanachi, B. Bouzouia, A. Moualek, and A. Farhi, "A multi-contract net protocol for dynamic scheduling in flexible manufacturing systems (FMS)", IEEE Proceedings of International Conference on Robotics and Automation, Vol. 2, 10-15 May (1999) 1114 - 1119.
12. D. Ouelhadj, C. Hanach, and B. Bouzouia, "Multi-agent system for dynamic scheduling and control in manufacturing cells", IEEE Proceedings of International Conference on Robotics and Automation, Vol. 3, 16-20 May (1998) 2128 - 2133.
13. V. V. Emelyanov, "A decentralized management system based on multi-agent model," ICAIS 2002 - IEEE International Conference on Artificial Intelligence Systems, 5-10 Sept. (2002) 283 - 293.
14. L. M. Camarinha-Matos, J. Barata, and L. Flores, "Shopfloor integration and multiagent based supervision," Proceedings of INES '97 - IEEE International Conference on Intelligent Engineering Systems, 15-17 Sept.(1997) 457 - 462.
15. L. Breton, S. Maza, and P. Castagna, "A multi-agent based conflict-free routing approach of bi-directional automated guided vehicles" IEEE Proceedings of American Control Conference, 14-16 June (2006) 6 pp
16. D. Camacho, R. Aler, C. Castro, and J. M. Molina, "Performance evaluation of ZEUS, Jade, and SkeletonAgent frameworks", IEEE Proceeding of International Conference on Systems, Man and Cybernetics, Vol. 4, 6-9 Oct (2002) 6 pp.
17. A. Castelnuovo and L. Ferrarini, "Petri net models of agent-based control of a FMS with randomly generated recipes", IEEE Proceeding of ETFA 2005 - 10th IEEE Conference on Emerging Technologies and Factory Automation, Vol. 1, 19-22 Sept. (2005) 7 pp.
18. P. Vrba, "MAST: manufacturing agent simulation tool", Proceedings of ETFA '03 - IEEE Conference Emerging Technologies and Factory Automation, Vol. 1, 16-19 Sept. (2003) 282 - 287.
19. JADE (Java Agent DEvelopment Framework), "Official site," http://jade.cselt.it/, (2006).
20. JADE (Java Agent DEvelopment Framework), "OnLine documentation," http://jade.cselt.it/doc/JADEProgramming-Tutorial-for-beginners.pdf, (2006)
21. FIPA, "Communicative Act Library Specification," Foundation for Intelligent Physical Agents, Ed.: http://www.fipa.org/specs/fipa00037/ (2000)..
22. Odell, James, H. Van Dyke Parunak, and B. Bauer, " Representing Agent Interaction Protocols in UML", Agent-Oriented Software Engineering, Ciancarini, P. and Wooldridge, M., Berlin, (2001).

Fighting Fire with Agents - An Agent Coordination Model for Simulated Firefighting

Daniel Moura

Faculdade de Engenharia da Universidade do Porto
Rua Doutor Roberto Frias, 4200-465 Porto, Portugal
daniel.moura@fe.up.pt

Abstract. This paper proposes a model for coordinating teams of computational agents. This model is especially aimed for coordinating agents performing in a simulated environment of forest firefighting, although it may be used in other domains. We will start by introducing the Pyrosim platform where we are carrying out our experiments. Pyrosim is a tool developed in our laboratory that simulates a forest fire environment where software agents act under the role of firefighters that have to cooperate in order to control the fire. The paper procceds by presenting a model for team coordination. With this model it is possible to define firefighting tactics that originate different team approaches to the fire. These tactics are conducted by a single agent (the Leader) that communicates high level tasks to the other agents. Agents have local autonomy and are able of cooperating locally for carrying out their tasks without using communication. Finally, some results of our experiments are presented where two different tactics are tested in two different scenarios. These results show that these tactics achieve an outcome that is coherent with firefighting theory, which adds credebility to the proposed model. We will use these results to address the problem of automatic tactic selection where we are currently working on.

Key words: Multi-Agent Systems, Team Coordination, Simulation, Forest Firefighting

1 Introduction

Forest fires are an everyday problem of society. Considering South European countries only (Portugal, Spain, France, Italy and Greece), in the year of 2005, forest fires burned more than 556 thousand hectares of forest [1]. This problem particularly affects Portugal where more than 325 thousand hectares of forest burned for the same year, which is more than an half of the total burned area in the South European countries.

Despite the seriousness and challenges of this problem, there is not much work in this domain in the area of Artificial Intelligence. In fact, we only found one work that uses a Multi-Agent System (MAS) to tackle the problem of coordinating a firefighting team to attack a forest fire. This work in being developed

by Wiering *et al.* [2,3] and is concerned with coordinating heavy machinery (bulldozers) to build a line around the fire to prevent it from spreading. The authors use machine learning to elaborate plans according to the scenario situation, which are distributed to agents (bulldozers) at the beginning of the simulation. However, in Portugal and other countries, using heavy machinery is most of the times impossible because of the terrain geography that is highly irregular. Additionally, heavy machinery may not be always available, or may not be the best solution (e.g. in small fires). Therefore, we choose to coordinate a team of firefighters that use water jets to try putting out the fire, although we intend to include other kinds of firefighting agents in the simulation platform.

This paper presents a model for coordinating a Team of computational Agents that performs in a Forest Firefighting simulator. Agents play the role of firefighters that have to combat fire in an organized way in order to control it. With the proposed model it is possible to define Firefighting Tactics similar (although simplified) to the ones that are used by real firefighting teams. Additionally, it is possible to experiment new tactics (or variations of commonly used tactics) for trying to understand their outcome in different scenarios. For accomplishing this, the model provides a flexible structure that enables to define different approaches of the team to the fire.

The remainder of this paper is structnred in the following way: section 2 presents the forest firefighting simnlator that we are using for our experiments. Then, section 3 describes the proposed coordination model for controlling the overall team behavior. The paper continues to section 4 that presents some results of our experiments and point out future work directions for antomatic tactic selection. Finally, we present our conclusions about this work.

2 The Pyrosim Platform

Before presenting the coordination model, we will introduce the Pyrosim Agent platform [4] that we have been using in our experiments. The Pyrosim platform simulates a forest-fire environment where a team of Agents (firefighters) cooperates to control and extinguish the fire, while simultaneously trying to minimize the overall damage and losses. In Pyrosim, Agents have to deal with very dynamic fire fronts, terrain constraints and their own physical and logistic limitations. Each Agent needs to ensure its physical safeness while trying to fight the fire and, at the same time, help colleagues to remain safe. Agents are equipped with a water jet with limited power that allows them to put out the fire, but they are not normally able to do it individually so there is an obvious need for cooperation. Agents may communicate with each other in order to organize team efforts (broadcast and 1-to-1 messages). Agent's Perception System provides information about his own state (physical energy, speed, acceleration, position in the terrain, status of the personal water jet) as well as several matrix strnctures named *Visual Maps* that describe close range and medium range surroundings. Visual Maps contain information with different *levels of detail* and *noise* (depending on the distance) abont terrain, vegetation, level of destruction,

and fire cells. Visual Maps may have cells where no information is available because of the occlusion effect (for instance, when an agent is climbing a hill it cannot see to the other side of the hill). Agents also receive information about visible parameters of other Agents (location, approximate energy level and action). Figure 1 shows a visualization of a simulation in the Pyrosim simulator. Pyrosim creates a complex environment that may be used as testbed for team coordination models, and for simulating firefighting tactics as well. Additional information about the Pyrosim platform may be found at [4].

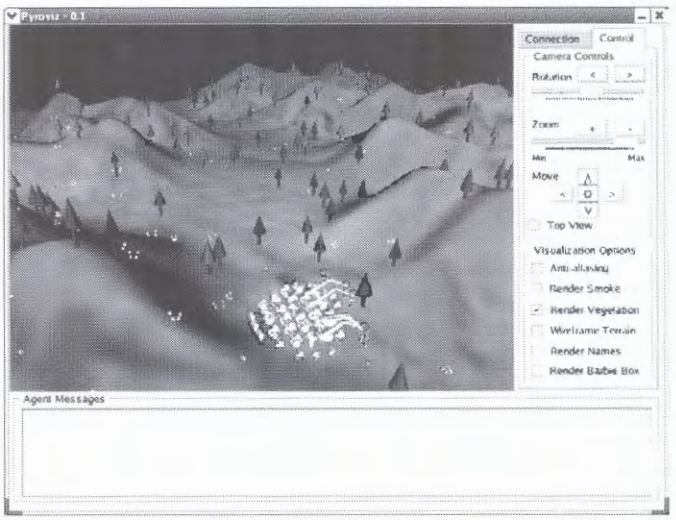

Fig. 1. A visualization of a simulation in the Pyrosim platform

3 Overall Team Coordination

In our approach, team coordination is centralized, although loosely coupled, which means that the overall team behavior is controlled by one agent, the *Leader*, but every agent has local autonomy. The Leader responsibilities include being aware of the global situation, reasoning about it, and assigning specific tasks to agents in order to carry out a given plan. There are several reasons that impelled us to use centralized coordination:

- Real firefighting teams use a centralized chain of command [5];
- One agent, the Leader, may have access to the global situation and make decisions based on global knowledge. In a decentralized solution, the information would be dispersed, and no agent would have a complete understanding of the situation. The alternative would be for every agent to be aware of the

other agents' knowledge, but that would increase drastically communications between agents.
- Centralizing tactical/strategic decisions let other agents to focus on their tasks execution.
- Using centralized coordination is not synonym of the Leader being a master ruling a set of slaves. Agents may have local autonomy and are not obliged to carry out tasks that go against their own goals.

Of course that centralizing decision has its drawbacks. First, there is the danger of creating a bottleneck on the Leader agent. To avoid this, teams should not have too many agents, and communication should be nsed only when needed. Additionally, when coordinating larger teams, one should use a hierarchy of command to ensure that a Leader only controls directly a small set of agents. Another problem with centralized coordination is that if communication fails or if the Leader fails, the team coordination is lost. To prevent the team from losing its Leader we configнred the Leader agent's "personality" in order to obtain a more cautious behavior. As result the Leader keeps a higher distance from the flames and takes less chances during the firefight. Cnrrently, there is no mechanism in our implementation for replacing the Leader in case of Leader failure. However, this is not a big issue because our goal is to simulate firefighting tactics and not to create a system for using during real firefighting.

In the next subsections we will make a formal description of the proposed model and then we will instantiate it to the forest firefighting domain.

3.1 Coordination Model

For developing the coordination model, we nsed the same principle that was used in the robotic soccer teams by Stone and Veloso [6,7], and by Reis and Lau [8,9,10]. Both approaches defined the team spatial distribution using *roles*, which were then assigned to agents. Roles enable to define generic team formations that may be used with any team of agents. Despite we are using the same principle of these teams, we had to build a model that would be able to handle some issues that are not present in the robotic soccer domain, such as the actuation area size is not delimited (on the other hand, the soccer field has fixed dimensions), the nnmber of agents that constitntes a team is not fixed (on the other hand, soccer teams have fixed size), and the opponent may have very different configнrations (fire size, shape and properties may vary a lot).

We will now present a formal description of the proposed model. Every agent has a *Role* (3). There is a predefined nnmber of Roles (2) which define different agent behaviors. The number of Roles is not related with the number of agents. Teams may be homogeneous (all agents have the same Role), or heterogeneous (agents playing different Roles).

$$Agents = \{Agent_1, Agent_2, Agent_3, \ldots, Agent_{nagents}\} \quad (1)$$

$$Roles = \{Role_1, Role_2, Role_3, \ldots, Role_{nroles}\} \quad (2)$$

$$AgentRole_i \in Roles \quad \forall i = 1..nagents \tag{3}$$

There are several ways of organizing agents to attack fire. Fire may be attacked directly in the tail, head or flanks, or it may be attacked indirectly by constructing lines outside the fire that limit its progression. To implement these attacks we have defined a set of *Attack Plans* (4) that specify how agents shonld approach fire.

$$AttackPlans = \{AttackPlan_1, \ldots, AttackPlans_{nattacks}\} \tag{4}$$

In the proposed model, an Attack Plan defines a sequence of *Tasks* for a given Role (7). In this way, agents that share the same role will have the same tasks to carry out. In general, an Attack Plan may be carried ont by one Role only, althongh there are Attack Plans that may be executed by a given set of Roles (6).

$$Tasks = \{Task_1, Task_2, Task_3, \ldots, Task_{ntasks}\} \tag{5}$$

$$AdmissibleRoles_i \subseteq Roles \quad \forall i = 1..nattacks \tag{6}$$

$$\begin{gathered} AttackPlan_i = \{Role_j, AttackTask_1, \ldots, AttackTask_{nattacktasks}\} \\ \forall i = 1..nattacks \quad Role_j \in AdmissibleRoles_i \quad AttackTask_k \in Tasks \end{gathered} \tag{7}$$

A firefighting crew may perform different attacks simultaneously. To define the attacks distribution among the team, the concept of *Tactic* was created (8). A Tactic defines for each Role how many agents will play that Role, and which Attack Plan will be performed by those agents (9).

$$Tactics = \{Tactic_1, Tactic_2, Tactic_3, \ldots, Tactic_{ntactics}\} \tag{8}$$

$$\begin{gathered} Tactic_{i,j} = \{TacticAttack_j, AgentAssignment_j\} \qquad \forall i = 1..ntactics \\ \forall j = 1..nroles \qquad TacticAttack_j \in AttackPlans \\ AgentAssignmet_j \in [0..1] \quad \sum_{j=1}^{nroles} AgentAssignment_j = 1 \end{gathered} \tag{9}$$

From (9), we gather that agent assignment to Roles is defined using relative quantities (e.g. allocate half of the team to a given Role). However, in our implementation, it is also possible to define agent distribution using absolute quantities (e.g. allocate 2 agents to a given Role). Additionally, agent distribntion may be defined dynamically at rnn-time, which enables to produce more complex role assignments.

It is also possible to define Tactics that change the team approach over time. We call these tactics *Dynamic Tactics* (10), and they are composed by a set of Tactics with *Activation Conditions* (11). Activation Conditions define when to switch to a given tactic.

$$DynamicTactics = \{DynamicTactic_1, \ldots, DynamicTactic_{ndynamictactics}\} \tag{10}$$

$$\begin{aligned} DynamicTactic_i = \{&ActivationCondition_1, SubTactic_1, ..., \\ &ActivationCondition_{nsubtactics}, SubTactic_{nsubtactics}\} \\ &\forall i = 1..ndynamictactics \quad SubTactic_j \in Tactics \end{aligned} \tag{11}$$

All tactics are pre-defined at the Leader agent level, and therefore, only the Leader has knowledge about tactical information. The other agents only know their Role and the tasks that they must execute, which are assigned by the Leader using tactical information.

3.2 Defining Attack Plans for Firefighting

For instantiating the proposed model to forest firefighting, we start by defining Attack Plans. We have divided Attack Plans in two major classes: (i) Direct Attacks, and (ii) Indirect Attacks. This division is based on firefighting theory that differentiates Direct from Indirect Attacks [5]. Direct Attacks involve fighting the fire directly in the flames using water or manual tools, like shovels, to swat the flames. Therefore, this kind of Attack Plans specify tasks for approaching the fire and then attack it directly. On the other hand, Indirect Attacks are used to fight fire at distance, especially when the fire is too intense for firefighters to approach it. The most common technique is to build a *fireline*. Firelines are built by digging the ground to remove all the vegetation in front of the fire to starve it ont of fuel. In the current version of the fire simulator it is not possihle to built firelines. However, we implemented another kind of indirect attack that consists in creating a *wet-line*. Instead of dogging the ground, firefighters wet the ground with large quantities of water. The effect is similar to building a fireline, although this is just a temporary solution until the simulator supports doggiug operations.

Figure 2 illustrates a class diagram of the Attack Plans that we iuplemented, where the classes with uo coloring represent Attack Plans that may be instantiated, and the classes colored in grey represent abstract classes that are used for structuring proposes only. In general terms, Direct Attacks define how to attack a given part of the fire (e.g. the head, the tail or the flanks) and the *WetLineAttack* defines how to create a Wet-Line in a given area.

Regardless of the kind of attack, Attack Plans always have two stages: the positioning and the attack managemeut. The positioning stage is concerned with placing the agents in positious that enable them to start the attack in good conditions. Once ageuts are in position, the attack management stage is activated. This stage is concerned with allocating tasks to agents that are performiug the attack. Iu the beginniug, agents receive tasks to attack a given area. Every time

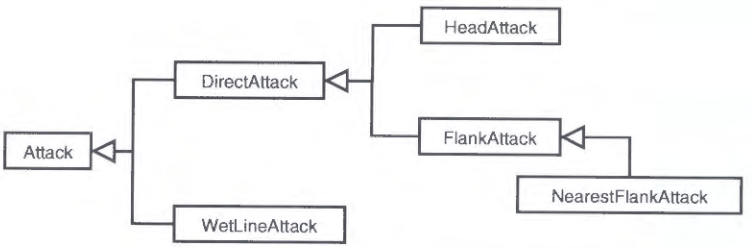

Fig. 2. Implemented Attack Plans

agents complete their tasks, they receive new tasks to attack a new target area. Every Attack Plan has to define its preferences for positioning and managing the attack. For instance, in a Head Attack it is defined that agents should start by approaching the Head of the fire in the opposite direction to the fire propagation direction. Then, when they arrive to the target area, they should start attacking the foremost burning cell. Finally, when they extinguish the current cell, they should attack the foremost adjacent cell that is burning.

During attacks, agents that share the same Role are able to cooperate efforts in order to achieve better results. This cooperation is decentralized and does not use message-based communication. For the sake of brevity we will not address this issue here (please consult [11] for more information).

3.3 Defining Firefighting Tactics

As we have seen above, tactics define the Role of every agent and the Attack Plan that the agent will carry out. Using the Attack Plans presented in the previous subsection we are able to build a considerable set of tactics. In table 1 we present some examples of Static Tactics. These tactics may be very simple like tactic ST1 that assigns the same Role and Attack Plan to every agent. Moreover, we may have tactics with the team divided for performing different kind of attacks simultaneously. If we need more flexibility we may define Dynamic Tactics that have the ability to change the team configuration according to the current situation. For instance, tactic DT2 (table 2) defines that when the team arrives to the fire it should build a Wet-Line around fire, and when the fire is controlled all firefighters should concentrate efforts to attack the tail. Additionally, it is possible to specify Dynamic Tactics that select the best a approach automatically according to the scenario evaluation.

4 Experimental Results and Tactic Selection

Currently, the developed system works as a platform to test tactics in different scenarios. However, our next step is to enhance the platform in order to automatically determine the tactics that work best in different scenarios. We have already made some experiments that show that there are tactics that achieve

Table 1. Examples of static tactics

Tactic	Role	Attack	Distribution
All in the Tail (ST1)	tail_attacker	Nearest Flank	All
Split by Flanks (ST4)	left_flank_attacker	Flank	1/2
	right_flank_attacker	Flank	Rest
Surrounding Wet-Line (ST6)	left_wet_line_builder	Wet-Line	1/4
	right_wet_line_builder	Wet-Line	1/4
	tail_wet_line_builder	Wet-Line	1/4
	head_wet_line_builder	Wet-Line	Rest
Surrounding Wet-Line with Tail Attack (ST7)	left_wet_line_builder	Wet-Line	1/5
	right_wet_line_builder	Wet-Line	1/5
	tail_wet_line_builder	Wet-Line	1/5
	head_wet_line_builder	Wet-Line	1/5
	tail_attacker	Nearest Flank	Rest

Table 2. Example of a dynamic tactic: Wet-Line and then Attack Tail

Activation Condition	→	Sub Tactic
When simulation starts	→	Surrounding Wet-Line
When fire stabilizes	→	All in the Tail

positive results in situations where others fail, although the same tactics achieve negative results in situations where others succeed [11]. We are able to observe this in the following example where we present the results of experimenting two different tactics (ST1 and ST6) in two different scenarios (A and B). Tactic ST1 tries to attack the fire by positioning all agents behind the tail of the fire and giving them orders to attack the fire directly with water. Tactic ST6 places firefighters around the fire perimeter and give them orders to wet the ground in order to create a Wet-Line. We have selected these tactics in order to represent (i) tactics mainly based in direct attacks (ST1), and (ii) tactics mainly based in indirect attacks (ST6), which we know from firefighting theory that should be applied to fires with different dimensions (among other factors such as the number of firefighters available, terrain properties and weather conditions). We experimented both tactics in two scenarios with the same characteristics (in terms of terrain geometry, vegetation and weather), but in scenario A agents received the warning sign sooner than in scenario B, and therefore when firefighters arrived to the fire in scenario B they wonld face a fire in a more advanced state than in scenario A. We observed that both tactics succeeded in scenario A (figure 3) and that tactic ST1 achieved better results. This happened because in tactic ST1 firefighters start by attacking fire directly and therefore they are able to prevent it from spreading at an earlier stage. However, in scenario B this same tactic (ST1) failed to control fire from spreading (figure 4). ST1 tactic was a good choice for scenario A but that is not the case for scenario B. In scenario

B firefighters arrive to the fire at a later stage where using a direct attack is not enough to control fire. On the other hand, tactic ST6 was able to achieve a more stable performance by sacrificing the same area in both scenarios. In this tactic firefighters build a wet-line around the fire area that prevents fire from spreading, instead of fighting the fire directly. Therefore, the total burned area in both scenarios was nearly the same, but in scenario A the damage was excessive. More information abont this and other experiments may be found in [11]. An important ontput from these experiments is that results are coherent with forest firefighting theory in terms of the applicability of direct and indirect attacks. Another important observation is that more agents result in less damages. This may be observed in both scenarios for tactic ST1.

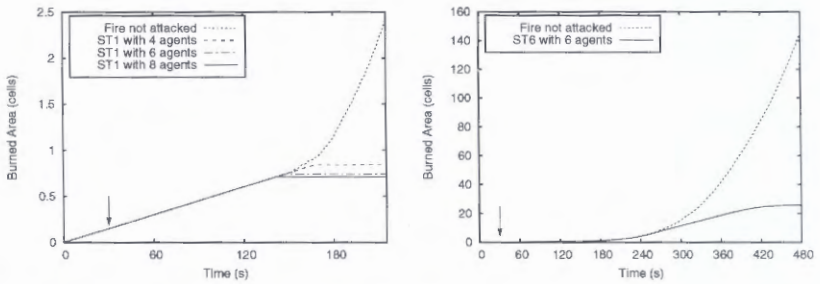

Fig. 3. Scenario A: Burned area using the ST1 tactic (All in the Tail) and burned area using ST6 tactic (Surrounding Wet-Line)

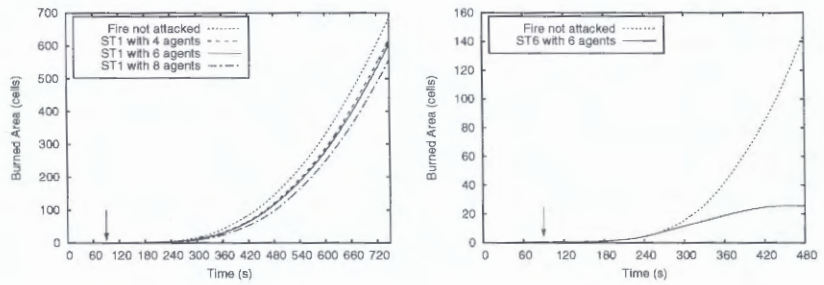

Fig. 4. Scenario B: Burned area using ST1 tactic (All in the Tail) and burned area using ST6 tactic (Surrounding Wet-Line)

The experiment described above is just an example of the influence of the scenario configuration in tactic selection. Much others factors like the terrain geometry, the type of vegetation in the terrain, and the weather may inflnence tactical decisions. Therefore, one of our main lines for future work is to use

machine learning for figuring out which tactics are best according to the scenario properties.

5 Conclusions

This paper presents a model for coordinating a team of agents. We have instantiated this model to the forest firefighting domain where we were able to implement some tactics similar to the ones that real firefighting teams use. The proposed model enables to define much other tactics in a rather flexible way. In our implementation, the overall team coordination is controlled by a single agent, the Leader, who is responsible for controling tactics execution. However, agents have local autonomy and are able to cooperate locally without the Leader intervention.

We also presented some results of our experiments that demonstrate that, like in reality, different fire scenarios require different firefighting tactics in order to minimize fire damage. Additionally, the tactics that performed best in the tested scenarios are the ones that we were expecting according to firefighting theory. Furthermore, teams with more agents achieved better results than smaller teams, meaning that the coordination mechanism is taking advantage of the extra agents. However, we are aware that we are still far from providing meaningful information for real firefighting teams. For achieving this, the simulator shonld be properly validated and more experiments should he done.

The paper also points out an interesting line of research regarding possible uses of machine learning for automatic tactic selection based on the results of tactic experimentation in different scenarios.

References

1. da Defesa da Floresta Contra Incêndios, D.: Incêndios florestais – relatório de 2005. Technical report, Ministério da Agricultura do Desenvolvimento Rural e das Pescas, Portugal (January 2006) http://www.dgrf.min-agricultura.pt/v4/dgf/pub.php?ndx=2271.
2. Wiering, M., Dorigo, M.: Learning to control forest fires. In Haasis, H., Ranze, K., eds.: Proceedings of the 12th international Symposium on 'Computer Science for Environmental Protection'. (1998) 378–388
3. Wiering, M., Mignogna, F., Maassen, B.: Evolving neural networks for forest fire control. In van Otterlo, M., Poel, M., Nijholt, A., eds.: Benelearn '05: Proceedings of the 14th Belgian-Dutch Conference on Machine Learning. (2005) 113–120
4. Sarmento, L.: An emotion-based agent architecture. Master's thesis, Faculdade de Ciências da Universidade do Porto (2004)
5. Nicolas, M., Beehe, G.: The training of forest firefighters in indonesia. Technical report, German Agency for Technical Cooperation, European Union, and Government of Indonesia (1999)
6. Stone, P.: Layered Learning in Multi-Agent System. PhD thesis, School of Computer Science, Carnegie Mellon University (December 1998)

7. Stone, P., Veloso, M.: Task decomposition, dynamic role assignment and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence **110**(2) (June 1999) 241–273
8. Reis, L.P.: Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico. PhD thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal (June 2003)
9. Reis, L.P., Lau, N.: Fc portugal team description: Robocup 2000 simulation league champion. In Stone, P., Balch, T., Kraetzschmar, G., eds.: RoboCup 2000: Robot Soccer World Cup IV, London, UK, Springer-Verlag (2001) 29–40
10. Reis, L.P., Lau, N., Oliveira, E.: Situation based strategic positioning for coordinating a team of homogeneous agents. In Hannebauer, M., Wendler, J., Pagello, E., eds.: Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications (selected papers from the ECAI 2000 Workshop and additional contributions), London, UK, Springer-Verlag (2001) 175–197
11. Moura, D.: Coordinating a team of agents in the forest firefighting domain. Master's thesis, Faculdade de Engenharia da Universidade do Porto (2006)

Sessão 3

Aplicações Web

Exploring HTML to Improve Web Search Engines' Performance

Nuno Escudeiro

Instituto Superior de Engenharia do Porto, Instituto Politécnico do Porto
nfe@isep.ipp.pt

Abstract. Web search engines are powerful tools used to satisfy specific information needs on the web. Their purpose is to maximize user satisfaction when performing this task. Current search engines compute document relevance for a specific need from document's content; however web documents have other features that might be explored. We believe that it is possible to improve user satisfaction if we explore HTML tags and document metadata. In this paper we present Xearch, a meta-search system that wraps public search engines in a framework that improves both the expressiveness of the language available for the user to specify information needs and the control over the answer format. Xearch converts HTML pages to an XML document, following a specific XML Schema, loads these into a native XML database and finally submits user queries to this database. These queries can be specified through keywords but also explore documents' HTML tags and metadata. Results from our experimental evaluation confirm that it is possible to improve the answer quality with this framework although some additional care is needed to avoid deficient specifications which can arise more frequently since the specification language is more expressive.

1 Introduction

The web is being used to satisfy specific user information needs since its inception and this utilization of the web is growing fast [13]. Web search engines are powerful tools widely used to help performing this task – search engines receive around 190 million requests a month [24].

Public search engines usually produce answers based on document's content and rank the result set based on document popularity inferred from web structure [2,11]. A few exceptions explore other sources of evidence besides content text [3, 4, 6, 9, 12, 14, 16,] while other apply innovative search paradigms [15].

The main purpose of web search engines is to maximize user satisfaction which leads to a few specific problems. These are mainly related to the inherent ambiguity in the specification of information needs through a set of keywords – the standard way of specifying information needs on current web search engines – and to the subjectivity in user relevance judgments – two independent users may associate antagonic relevance to the same document, regarding a specific information need, and even the same user's relevance concept may vary along time.

Web documents are, in their majority, HTML files [22]. HTML is a language that might be explored, and usually is, to enhance the key ideas that the author wishes to transmit on a document, for instance using headings or text formatted in bold. However HTML tags are not made available by public search engines and consequently the user looses the opportunity to explore them when specifying information needs.

We believe that it is possible to improve user satisfaction if we explore other HTML document's features besides text content.

In this paper we present Xearch which is a meta-search engine that wraps public search engines in a framework intended to improve the user experience when searching information on the web. Xearch allows users to specify their information needs based not only on keywords but also on structural properties and metadata derived from HTML tags.

The purpose of Xearch is to make HTML tags of web pages available to end-users in such a way that they can explore them to improve the final answer. We expect Xearch to be used as an improving mechanism on top of any kind of public web search engine.

Xearch is implemented over eXists [25], a native XML database, accessed through XQuery [27]. The system submits a keyword-based query to a public search engine and downloads the HTML answers produced. These HTML documents are then converted to the Xearch XML Schema and loaded into the XML database; then a XQuery query is submitted to produce the final answer.

Although this approach is simple we expect it to produce better answers to user needs since it allows for a more expressive way of representing information needs and also increases the user control over the format of the answer, especially over the order by which documents are presented to the user.

The main advantages we expect from our approach – improving the expressiveness of the specification language and the user control over the result – are a consequence of both the Xearch document model and XML technologies. XQuery allied to eXist free text functions might increase user flexibility in querying document collections.

The remainder of this paper is organized as follows: in section 2 we review related work, section 3 describes the Xearch system, referring to its architecture, the underlying web page model, its advantages and disadvantages over public search engines; section 4 presents the experimental evaluation we have performed and discusses the results; section 5 presents our conclusions and describes directions for future work.

2 Related Work

Although the majority of web search engines compute relevance based on content and web link structure, there are also many interesting systems that try to explore other characteristics of web and web documents.

iVia [15] (2003) is a hybrid system that collects and manages resources, starting with an expert-created collection that is augmented by a large collection automatically retrieved from the web. iVia automatically crawls and identifies relevant Internet resources through focused crawling [5] and topic distillation approaches.

Thesus [9] (2003) allows users to search documents in a previously fetched and classified document collection. Document classification is based on document content and on link semantics. The system includes four components responsible for document acquisition, information extraction, clustering and querying.

WebLearn [14] (2003) is a system that retrieves documents related to a topic, which is specified through a set of keywords, and then automatically identifies a set of salient topics. The identification of these salient topics is a fully automatic process that does not allow for user interference.

Metiore [3] (2001) is a search engine that ranks documents according to user preferences, which are learned from user historical feedback. Queries might be based on content and also on other document attributes, such as title, author and year.

Personal View Agent, PVA, [6] (2001) is another personalization system that learns user profiles in order to assist them when searching information in the web. PVA organizes documents in a hierarchical structure which is user dependent and dynamic. This hierarchy is automatically adapted to drift in user's interest.

Personal WebWatcher [16] (1999) is a system that observes users behavior, by analyzing page requests and learning a user model, and suggests pages potentially interesting to the user. When a user downloads a web page the system analyses outlinks and highlights those that seem interesting given the specific user model.

The ARC system [4] (1998), compiles a list of authoritative web resources on any topic specified by a set of keywords. ARC submits a query to AltaVista constituting a root set with the top 200 results. This root set is expanded by adding direct neighbors. Finally an iterative process is carried in order to compute authority and hub [11] measures for each document in the expanded set.

Letizia [12] (1995) is a user interface agent that assists a user browsing the web. Somehow similar to Metiore [3], Letizia also suggests potentially interesting links for the user to follow. Interest in a document is learned through several heuristics that explore user actions, user history and current context.

Our approach is different from the ones we have just described. Xearch does not require relevance feedback from the user nor does it require any effort to capture and model specific user needs. We focus our attention at increasing the expressivity of the language that is used to specify user information needs and at the level of control over the result set that is made available to the user.

Instead of relying on automatic processes that try to learn user information needs, Xearch system gives users the tools to describe information needs more accurately and a greater level of control over the answer format and sorting.

3 The Xearch System.

Xearch is a meta-search system that wraps a web search engine in a framework that provides users with a richer language to specify information needs and also an extended set of features to control and format the answer than those provided by the base search engine.

3.1 System Architecture

Xearch performs a five step process to satisfy user information needs (Figure 1).

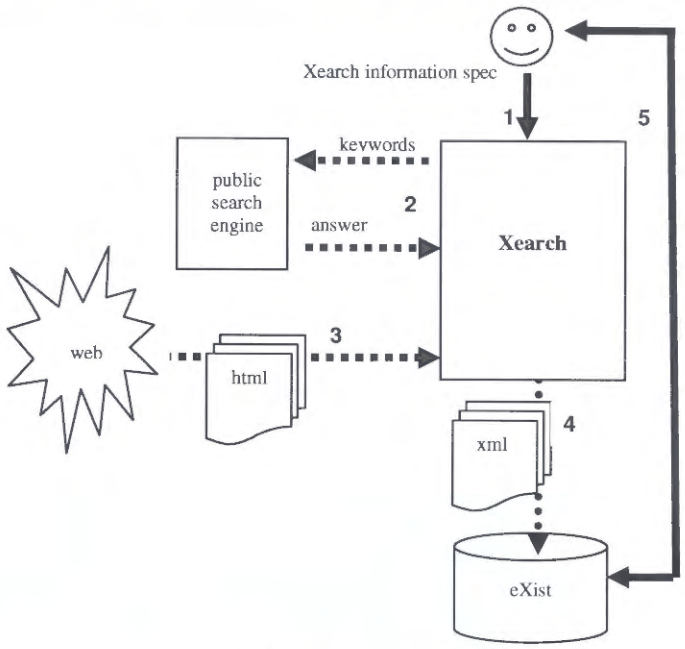

Fig. 1. Xearch architecture

This process requires user interaction at two distinct phases:

- on an initial phase (1) users specify their information need. This specification can be based on keywords – allowing the user to specify a query just as if he was using a classical search engine – along with other characteristics that Xearch provides (such as the number of words, words appearing in headings or any other document properties that are captured by the Xearch document model).
- once the system generates the answer the user can refine his query (5) through an interface that gives access to the document collection originally provided by the base search engine.

Between these two phases the system submits a query to the base search engine (e.g. Google), using the keywords specified by the user, and downloads the answer (2). The first 100 (this parameter is controllable by the user, defaults to 100) HTML results from this answer are downloaded (3), converted into XML documents according to the Xearch document model (see section 3.2) and loaded into an XML native database (4). Once the XML database is loaded a XQuery query is submitted according to the user specifications and the final answer is produced (5).

The user may then refine his query, if the answer is not satisfactory, having access to characteristics that are usually inaccessible through public web search engines.

3.2 Xearch Document Model

Xearch converts each HTML page, returned by the base search engine, into an XML document model that conveys, besides content, a set of features that might be explored to express information needs and control the answer format. This way the user has access to a specification language that is more expressive than the common set of keywords usually accessible through web search engines.

This document model includes the following features: document text content, number of words, number of images, number of tables, text in headings (and the corresponding heading level and order of appearance in the document), page title, text formatted in bold, text formatted in italic, the name of the file at the local server, the file URL, the URL domain and the date it was retrieved.

The XML Schema that implements this model has a root element, named `htmldoc`, which contains the elements `content`, `headings`, `italic`, `bold`, `title`, `figures`, `tables` and `html`. The `content` element has an attribute named `length` containing the number of words in the document. The `headings` element is a sequence of `heading` elements. Each `heading` element has the heading text and the attributes `level` and `ord` that report the heading level and its order of appearance in the document. The elements `italic`, `bold` and `title` contain, respectively, the words that are formatted in italic, in bold and those that appear in the document title. The elements `figures` and `tables` are integers that contain the number of images and the number of tables present in the document. The element `html` has mixed content: it contains the name of the downloaded file, as stored at the local disk and the simple elements `url`, `domain` and `date` which store the file URL, its domain and the date when it was downloaded.

3.3 Advantages and Disadvantages

Xearch document model displays advantages over public search engines but also some drawbacks.

One main advantage is the increased expressivity of the specification language. Since the document model includes structural features, besides text, the user has access to a richer set of primitives to express information needs.

Another advantage is the control the user has over the format of the answer. This extended control allows for the definition of relevance measures and to sort results based on them. If the user does not want to explore these functionalities the system preserves the ranking of the base search engine result.

The main drawback arising from this document model resides in the document's content representation. Document content is represented by the set of words it contains; this is an implementation of a Boolean model [1] that associates the same weight to all terms being present at the document. This disadvantage can be overcome at the cost of some processing. The implementation of a TF×IDF model [1], for in-

stance, requires computing, for each query, a matrix with the frequencies of each term in each document and to compute, from this matrix, the similarity among documents and a given query, which greatly increases the time required to process a query.

Another disadvantage of Xearch is that it only processes HTML files. Other document formats cannot be processed since Xearch model is based on HTML tags.

The time required to produce an answer is not comparable to base search engines since Xearch requires a significant overhead. Xearch submits the queries to the base search engine, extracts the first results, downloads them, generates a Xearch instance for each of them, loads them into the eXist database and only then submits the user query. Nevertheless, once the eXist database is loaded, users may refine their queries at their will without this overhead.

4 Evaluation

The Xearch evaluation was focused on two main aspects: we want to evaluate the added value of a more expressive specification language giving more user control over the answer and we also want to compare the performance of Xearch against public search engines. Two distinct experimental settings cover both these aspects.

4.1 Experimental Setting

To evaluate the improvement on user satisfaction generated by our approach we have deliberately selected an information need that might benefit from HTML tags exploration. "I want images of the terrorist attacks on September 11, 2001" might be such an information need since it is related to document properties not directly addressable through a set of keywords.

We submitted 12 queries, generally related to the subject "terrorism", to Google. The first 100 results from Google answers to each of these queries were then processed by Xearch and loaded into a native XML database. At the end, our collection was composed of 1145 documents on the subject "terrorism". This collection has been queried to evaluate the gain on user satisfaction – we will refer to this experimental setting as *specification language evaluation*.

To compare Xearch with current public web search engines – we will refer to this experimental setting as *performance evaluation* – we started by selecting three public search engines that are representative of distinct classes, both as to their web coverage of the web and as to the methods they apply to crawl the web. Our choices were:

- Google [21], which allegedly has the broader coverage of the web, indexing about 8×10^9 web pages. It is also reported as the most used search engine, processing over 91×10^6 queries a day [24]. Queries are specified by a set of keywords and additional features are available through advanced search. Results are sorted according to relevance, which is computed by the search engine without any user control.
- Tumba [19, 26], is a search engine for the Portuguese web. It presents specific characteristics due to its controlled web coverage. Queries are specified by key-

words and sorted according to relevance computed by the search engine Advanced search is more limited than in the other systems we have evaluated.
- Metacrawler [23] is a meta-search system. It does not crawl the web hut, instead, relies on other search engine's databases to produce its answers. Queries are described hy keywords and the advanced search in equivalent to Google. Relevance ranking is computed hy the system; relevance criteria are unknown to the user.
- Finally we evaluate Xearch which allows a more expressive specification language and a greater user control over the answer. Information needs might be expressed through keywords, document structure emhedded in HTML tags and metadata. The answer might be formatted and sorted according to user specific preferences.

The evaluation of these search engines was based on a test collection obtained from eight distinct information needs that we will refer to by *inA* through *inH*. We have defined a set of representative keywords for each information need. These keywords were used to suhmit queries to each search engine under evaluation. All the queries try to get the most out of each search engine's capahilities hut they all use the same set of keywords that have been previously specified for each information need.

We started hy submitting the queries to all the search engines under evaluation except for Xearch. From each answer we have loaded the first 100 results into the Xearch database and then suhmit the queries to Xearch.

From each answer produced by each search engine we have extracted the first 30 results, which have then been classified as relevant or not-relevant. A document is classified as relevant if its content satisfies the information need. Documents whose content does not satisfy the information need but have links that satisfy the user needs are labeled non-relevant. The set of documents classified as relevant constitute our collection of relevant documents for the respective information need.

4.2 Specification Language Evaluation

To evaluate the gain generated by a more expressive specification language and a more powerful control over the answer format we have submitted two queries to Xearch. One was based on content – documents that contain the terms "foto" or "imagem" and the phrase "11 de setembro" – and the other one explores the Xearch document model and is expressed through the same set of keywords but additionally sorts documents by decreasing order of the number of images they contain. The results returned to the first query are sorted according to Google's ranking method.

Both these queries return the same 236 documents. However, sorting the answer based on user defined criteria allows users to adjust the answer to their specific needs and produces substantially different answers.

In the answer to the first query, based on content, the first relevant document appears at the 32nd position. The first 31 documents do not contain any image that might satisfy the user. This answer seems very poor if we consider that the majority of users only view the first 10 to 20 results [10, 20].

The answer to the second query is much more interesting. The first relevant document appears in third place and the majority of documents that appear in the initial positions contain images about September 11th 2001, thus being relevant to the user. The document in our collection that seems the most relevant to satisfy the user (a pho-

tographic report containing many good photos of the attacks) is ranked 17th at the answer to the second query. It appears in the 140th position at the first query answer.

The analysis of these results seems conclusive: in certain settings the user may greatly benefit from a more expressive specification language and a greater level of control over the answer than those that public web search engines provide.

4.3 Performance Evaluation

We have obtained the documents for our test collection, as described in section 4.1, following traditional approaches [7, 18]. The number of relevant documents in the first 30 results for each query, $|REL_q|$, is reported at Table 1.

Table 1. Number of relevant documents returned for each query

q	Search engine				$\|REL_q\|$
	Google	Tumba	Metacrawler	Xearch	
inA	13	5	8	19	30
inB	5	2	4	7	15
inC	9	3	9	9	24
inD	14	4	5	7	26
inE	9	3	10	14	28
inF	16	6	5	16	31
inG	7	0	8	9	20
inH	10	0	8	5	21

From these results we compute mean recall and precision for each search engine considering these relevant sets and the first 30 results from each query (Table 2).

These results show that Google and Xearch perform consistently better than the other. Xearch present higher dispersion probably due to its higher expressiveness that might improve but also degrade, if not properly applied, the system performance.

Table 2. Recall and precision

	Google		Tumba		Metacrawler		Xearch	
	R	P	R	P	R	P	R	P
mean	0,42	0,35	0,11	0,10	0,30	0,24	0,43	0,36
σ	0,09	0,12	0,07	0,07	0,09	0,07	0,13	0,17
inA	0,43	0,43	0,17	0,17	0,27	0,27	0,63	0,63
inB	0,33	0,17	0,13	0,07	0,27	0,13	0,47	0,23
inC	0,38	0,30	0,13	0,10	0,38	0,30	0,38	0,30
inD	0,54	0,47	0,15	0,13	0,19	0,17	0,27	0,23
inE	0,32	0,30	0,11	0,10	0,36	0,33	0,50	0,47
inF	0,52	0,53	0,19	0,20	0,16	0,17	0,52	0,53
inG	0,35	0,23	0,00	0,00	0,40	0,27	0,45	0,30
inH	0,48	0,33	0,00	0,00	0,38	0,27	0,24	0,17

When the user information need may benefit from the Xearch richer document model then this approach improves the answer quality.

Recall and precision reported in Table 2 are performance measures computed over the entire answer, as if the user had an immediate perception of the full answer. In real circumstances the user analyzes search results document by document following the order of presentation. The interpolated precision for standard levels of recall [17] is a better performance measure to evaluate systems operating under these circumstances. Figure 2 presents typical *Precision versus Recall* graphics [1], computed for standard recall values of 0.1, 0.2 through 0.9.

It is evident from Figure 2 that Google and Xearch outperform both Tumba and Metacrawler, which confirms the conclusions previously drawn from the analyses of mean precision and recall (Table 2). The distinction between them is not obvious.

We have applied sign tests [8] to evaluate the statistical significance of the differences found at the precision values. These tests led us to conclude that Google and Xearch both have better performance than the rest – trivial conclusion given the dominance observed at Figure 2 – and that the difference between Google and Xearch, is not significant, so we must accept the null hypothesis that these systems are equivalent to each other. However, it must be stressed that Xearch approach significantly improves the answer quality for certain cases and does not degrade it at the other cases. Xearch always performs at least as well as Google, provided the query is correctly specified.

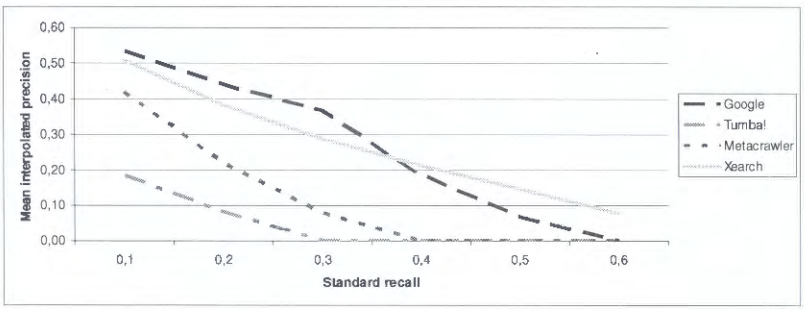

Fig. 2. Precision versus Recall

5 Conclusions and Future Work

Xearch gives the user access to a more expressive specification language and more control over the answer format improving the answer quality for certain types of user information needs. However, these characteristics may degrade the system performance if not properly used. A more expressive specification language is also more susceptible of misuse. If the user is not sure how to take benefits from the HTML tags or the metadata provided by the Xearch model it is better to specify the query based on content only.

For certain information needs, especially those that are hard to specify merely with a set of keywords, Xearch improves the answer quality when compared to a search engine based exclusively on content.

The current version of Xearch is limited to HTML documents and does not explore term frequency – we use a Boolean model for the document content – which degrades the system performance.

XML technologies were fundamental to achieve good results. XQuery proved to be a powerful query language and, allied to eXist free text functions, gives the user great flexibility in querying document collections.

The fact that Xearch database is composed by a set of results previously produced by the other search engines suggests that Xearch results are favorably biased. Xearch has the advantage of refining the answers produced by the other search engines so its answers will always be, at least, as good as those from provided by the base search engine. However this question does not arise since our purpose is to evaluate the benefits of exploring HTML tags and metadata to help user satisfying information needs on the web. Besides, Xearch is a meta-search engine as is Metacrawler, nevertheless Xearch performance is significantly better.

The results from this work are enthusiastic and we intend to improve the Xearch prototype and bring it online. This requires for the development of a web interface that isolates the XQuery language from users.

The Xearch document model might also be improved allowing for an even more expressive specification language. This requires for some research on the semantic properties usually associated to HTML tags and web conventions.

Acknowledgements

This work is supported by the POSC/EIA/58367/2004/Site-o-Matic Project (Fundação Ciência e Tecnologia), FEDER e Programa de Financiamento Plurianual de Unidades de I & D.

References

1. Baeza-Yates, R., Ribeiro-Neto, B. (1999), Modern Information Retrieval, Addison Wesley
2. Brin, S., Page, L. (1998), "The anatomy of a large-scale hypertextual web search engine", Proceedings of the 7th World Wide Web Conference, pp 107-117
3. Bueno, D., David, A.A. (2001), "METIORE: A Personalized Information Retrieval System", Proceedings of the 8th International Conference on User Modeling, Springer-Verlag
4. Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., Kleinberg, J. (1998), "Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text", Proceedings of the 7th International World Wide Web Conference

5. Chakrabarti, S., Berg, M., Dom, B. (1999), "Focused crawling: a new approach to topic-specific resource discovery", Proceedings of the 8th World Wide Web Conference
6. Chen, C.C., Chen, M.C., Sun, Y. (2001), "PVA: A Self-Adaptive Personal View Agent System", Proceedings of the ACM SIGKDD 2001 Conference
7. Cormack, G.V., Palmer, C.R, Clarke, C.L.A. (1998), *Efficient Construction of Large Test Collections*, Proceedings of the ACM SIGIR 1998 Conference
8. Conover, W.J. (1999), *Practical nonparametric statistics*, John Willey, New York
9. Halkidi, M., Nguyen, B., Varlamis, I., Vazirgiannis, M. (2003), "Thesus: Organizing Web document collections based on link semantics", The VLDB Journal, 12, pp 320-332
10. Jansen, B.J., Spink, a., Saracevic, T. (2000), "Real life, real users, and real needs: A study and analysis of user queries on the web", *Information Processing and Management*, 36(2), pp 207-227
11. Kleinberg, J. (1998), "Authoritative sources in a hyperlinked environment", Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, pp 668-677.
12. Lieberman, H. (1995), "Letizia: an Agent That Assists Web Browsing", Proceedings of the International Joint Conference on AI
13. Lim, L., Wang, M., Padmanabhan, S., Vitter, J.S., Agarwal, R. (2001), "Characterizing Web Document Change", Lecture notes in Computer Science
14. Liu, B., Chin, C.W., Ng, H. T. (2003), "Mining Topic-Specific Concepts and Definitions on the Web", *Proceedings of the WWW2003 Conference*
15. Mitchell, S., Mooney, M., Mason, J., Paynter, G.W., Ruscheinski, J., Kedzierski, A., Humphreys, K. (2003), "iVia Open Source Virtual Library System", D-Lib Magazine, Vol. 9, No. 1
16. Mladenic, D. (1999), Personal WebWatcher: design and implementation, Technical Report IJS-DP-7472, SI.
17. Rijsbergen, C.J.(), *Information Retrieval, cap. 7 - Evaluation*
18. Sanderson, M., Joho, H. (2004), *Forming Test Collections with No System Pooling*, Proceedings of the ACM SIGIR 2004 Conference
19. Silva, M.J. (2003), *Searching and Archiving the Web with Tumba!*, 4ª CAPSI
20. Spink, A., Wolfram, D., Jansen, B.J., Saravecic, T. (2001), "Searching of the web: the public and their queries", *Journal of American Society of Information Sciences and Technology* 52(3), pp 226-234
21. http://www.google.pt, accessed on December 2006
22. http://www.google.com/help/faq_filetypes.html, accessed on December 2006
23. http://www.metacrawler.com, accessed on December 2006
24. http://www.searchenginewatch.com, accessed on December 2006
25. http://exist.sourceforge.net/, accessed on December 2006
26. http://www.tumba.pt, accessed on December 2006
27. http://www.w3.org/TR/xquery/, accessed on December 2006

Automatic Construction of Databases from Websites

Jorge Morais

Universidade Aberta – Portuguese Open Univerisity
Porto Delegation, Rua do Ameal, 752 4200-055 Porto Portugal
jmorais@univ-ab.pt
and
Laboratory of Artificial Intelligence and Computer Science
Rua do Campo Alegre, 823 4150 - 180 Porto Portugal
jmorais@liacc.up.pt

Abstract. Websites have become a powerful and large source of useful information, which is possible to collect and structure in a database format. This is usually achieved manually or using context-dependent semi-automatic tools, like script files. However, large websites usually have a hidden database source. Since almost all of these websites are based on dynamic pages, it is possible, in many cases, to use parameters passed within the address in order to build automatically a database as closer as possible from the original one. This approach is intended to reduce time spent both in understanding the hidden database architecture and in transforming a website into a database. Experimental results showed that it is possible to rapidly and automatically build a database from a website.

Keywords: web mining, information retrieval

1 Introduction

Websites may contain anything a site owner desires. From simple personal pages with little information to large organization institutional sites complexity grows, and the need for a strong structure behind the website arises. For this reason, large websites usually have a relational database and dynamic pages that generate HTML (HyperText Markup Language) code. Most known combinations are mySQL with PHP [1, 2, 3], and SQLServer with ASP [4, 5].

On such cases, HTML code is generated on the server side, which means that clients do not have access to the database itself. If a user wants to retrieve some useful information they have to build a database manually or to use context-dependent semi-automatic tools, like script files. Some time is also needed to design the full database structure and importing data.

If the main goal is to retrieve only a small part of data available in the original database and there is no need for knowing the underlying structure, hand made approaches are probably a good choice. But sometimes database are very large and users intend to perform complex operations over the data, like performing data analysis or generating a database of potential costumers.

In such cases an automatic or semi-automatic tool would ease the process, enabling the user to concentrate on most important tasks. If there is a database expert that also knows how to parse HTML code, a semi-automatic tool for that specific purpose will solve the problem. But if there was a totally automatic tool it would be good for experts and non-experts, since experts could improve the generated database.

The main motivation for this work was to create an automatic tool with the ability of performing both tasks: retrieving data from the website, and creating the whole database from scratch, including the design of the database structure. The main obstacle is that it will be made from the client-side, without any inside knowledge of the server-side.

It is not intended to know how close the generated database is from the original one. It is assumed that it could be very different, since data might come from a query instead of coming directly from a table. However, what is important is to achieve useful data, not to have a similar structure.

It is assumed that most of the websites using a database are dynamic, meaning that the HTML code is generated on the server-side in response to a client-side request and after consulting its local database. Of course there are some exceptions. The most well known examples are weblogs, where HTML code is usually generated or updated each time a text is posted. In this paper, although an analog approach might be appropriate to the exceptions, only the first will be considered.

One of the most important factors in dynamic websites that will help the construction of the automatic tool is that parameters are passed within the address. For instance, the following address:

```
http://www.thissite.com/psa.php?x=100
```

means that the URL `http://www.thissite.com/psa.php` is requested and variable `x` is set with the value `100`.

The concrete meaning of this parameter passed along with the address is not usually known. For instance, it may force the page to list only the first hundred occurrences of a given query. But it may also be a table primary key inside a database, giving us some information about the database structure. This is information that the referred automatic tool can use to perform its job.

It is important to notice that not every website is expected to have useful information. Some websites might also encode their parameter passing in order to hide database structure. There is also some information in the database that is probably not important to the user.

However, the websites that fit in this approach are in sufficient number to motivate carrying out this work. It is assumed that automatic or semi-automatic approaches, when efficient, are always preferable to hand work.

There is not any knowledge of an existing similar tool. Some research is being held on the area of web structure mining [6, 7] but, as far as it is known, none has focused on this particular issue.

In the next section it will be described the general method for constructing a database from a website. Section 3 will present some experimental results which will be discussed in section 4, producing some conclusions and suggestions for future work.

2 Constructing the Database from the Website

As was referred in the previous section, one of the most important factors in dynamic websites that will he useful to the construction of the automatic tool is that parameters are passed within the address.

This section is divided in three subsections. In first, it is exposed how parameters are related to the database. The second subsection presents the issue of expanding and splitting a table. Finally, the whole picture of the tool is presented.

2.1 Using Parameters to Build the Database

Dynamic website addresses usually have the following syntax:

URL?*param**

where:

- *URL* (Uniform Resource Locator) is the address of the webpage.
- '?' is a separator character.
- *param** means zero or plus parameters.

Parameters syntax is the following:

var_1=$value_1$&var_2=$value_2$&...&var_n=$value_n$

where:

- var_i=$value_i$, $\forall i \in \{1,2,\ldots,n\}$, are parameter passing attributions.
- '&' is a separator character.

Looking to the example of the previous section, the following address:

`http://www.thissite.com/psa.php?x=100`

means that the URL `http://www.thissite.com/psa.php` is requested and variable `x` is set with the value `100`.

This could mean, for instance, that a given list should be shown 100 items each time. But it could also mean that `x=100` is the primary key of a table.

Suppose that there is a list of links (in a list or table) where all links are equal except for a given parameter. For instance, in the above address, `x` would take several values and that was the only difference in all the links of the page. Each link would also contain text describing the destination.

```
<a href="psa.php?x=100">CoMIC'07</a>

<a href="psa.php?x=101">CoDEX'07</a>

...
```

In this case there would be a list of conferences. Each conference has a unique key assigned. In addition, suppose that these links were inside a table, like the one on table 1.

Table 1. A table example

Name	year	location	Country
CoMIC'07	2007	Porto	Portugal
CoDEX'07	2007	Paris	France
...	...	...	...

This table is written in HTML code, and the underlined names refer to the links listed above. This would correspond to database table records like the ones in table 2.

Table 2. Corresponding database table records

keycode	Name	Year	location	Country
100	CoMIC'07	2007	Porto	Portugal
101	CoDEX'07	2007	Paris	France
...	...	...	...	...

Each of the hyperlinks must also he followed recursively in order to determine a relationship between tables.

For instance, following the first hyperlink, we could have another list with other hyperlinks, for instance:

```
<a href="psb.php?x=100&y=200">Web Mining</a>
```

This would mean that one of the conference topics would be "Web Mining". Notice that it would also be possible that the first parameter was inferred by the parent hyperlink, and we would only have:

```
<a href="psb.php?y=200">Web Mining</a>
```

Consider that the list of topics was not in a table in an ordered list:

```
<ul>

<li><a href="psb.php?y=100">Data Mining</a></li>

<li><a href="psb.php?y=150">Text Mining</a></li>

<li><a href="psb.php?y=200">Web Mining</a></li>

</ul>
```

In this case the corresponding database table records would he the ones in table 3.

Table 3. Table created from the list

keycode	field1
100	Data Mining
150	Text Mining
200	Web Mining

Until now the example shows the creation of two tables, one for the conferences (table 2) and one for the topics of the conference (table 3). Another table is needed to establish the relationship between those tables, as shown in table 4. It is important to notice that a table is created only once, the other times records are appended.

Table 4. Relating two previous tables

keycode1	keycode2
100	100
100	150
100	200
101	...

Suppose now that page `http://www.thissite.com/psb.php?y=200` did not have links. This could mean that an end point was reached. If useful data exists, it will generate new table records with keycode=200. Of course this is the simple way of doing this, but is it the right one?

For instance, this page might have a list of reviewers of the given topic. If one reviewer will only review one topic, then the way records are generated is correct. Otherwise, and the most likely, if one reviewer may review several topics, then they will appear in more than one record and database will not be normalized. In the next subsection this issue will be analyzed with more detail.

2.2 Expanding and Splitting Tables

It has been considered until now that website pages are always presenting queries of a single table for a particular key value. But sometimes the page displayed may be the result of a more complex query joining two or more tables. For instance, in the example we have shown conferences and topics could be on the same page in a table with merged cells as it is shown in table 5.

Two steps are used to deal with this. First, the table is expanded, repeating merged cells. The resulting table would be the one in table 6.

The second step is to split the table into two sharing the same key, where the first is composed by all fields that remain equal for each key value, and the second contains all other fields, as shown in tables 7 and 8.

Table 5. Two tables in one

Name	year	location	country	topics
CoMIC'07	2007	Porto	Portugal	Data Mining
				Text Mining
				Web Mining
CoDEX'07	2007	Paris	France	...

Table 6. Merged table expanded

keycode	Name	year	location	country	topics
100	CoMIC'07	2007	Porto	Portugal	Data Mining
100	CoMIC'07	2007	Porto	Portugal	Text Mining
100	CoMIC'07	2007	Porto	Portugal	Web Mining
101	CoDEX'07	2007	Paris	France	...

Table 7. Table with fields that remain equal for each key value

keycode	Name	year	location	country
100	CoMIC'07	2007	Porto	Portugal
101	CoDEX'07	2007	Paris	France
...				

Table 8. Table with other fields

keycode	topics
100	Data Mining
100	Text Mining
100	Web Mining
101	...

One can ask why not doing it in only one step? The answer is simple. A table is composed by records from several pages and this repetition of values for the same key might also happen in different pages. The first step is executed every time it is necessary while the second one is only executed at the end.

This also answers the question we issued at the end of previous section. Whenever there are several records with the same key value, the table must be split into two as shown in the previous example. However, as we can see in table 8, the second table still has some records with the same key value, and topics may also be associated with different key values.

Here there is only a non-key field, but there could be more being repeated every time. The solution is also to split this table into two, generating a unique key value for each topic, and associating it with the other key, as shown in tables 9 and 10.

Table 9. Generating unique key values for topics

keycode2	topics
100	Data Mining
150	Text Mining
200	Web Mining
...	

Table 10. Associating conferences with topics

keycode	keycode2
100	100
100	150
100	200
101	...

2.3 Putting All Together

This tool, during its development, has used three available open source applications:

- `wget` [8], which gets an entire website pages recursively, building a mirror locally.
- `HTML Tidy` [9], to clean HTML code.
- `htmlparser` [10], to parse HTML code and generate intermediate code for the database.

The first two applications, `wget` and `HTML Tidy`, were used for a major reason: to measure the impact of tasks they perform and to separate these from the central task of the tool.

The time spent downloading pages is very dependant of the connection speed. Although `htmlparser` is capable of getting pages directly, it was decided that the best way to measure results was to separate those tasks.

`HTML Tidy` has become necessary because many HTML code was badly written, especially with unclosed tags. However, this task may be integrated with the central task. In fact, it should be, since HTML code that is unnecessary for the purpose of the tool might also he cleaned, which is a waste of time.

Application `htmlparser` is the only one that is used for the central task. It is written in language Java [11] and its source code was adapted in order to perform the two steps:

1. Transforming HTML code into a set of database tables.
2. Splitting tables when necessary.

This tool generates a set of flat files that may be imported by most of the Database Management Systems. However, it is possible to generate code to construct the entire database directly.

3 Some Results

As was said in section 2, applications `wget` and `HTML Tidy` were used in order to separate time consumed downloading files and cleaning HTML code from the time of the central task of the tool, which was constructing the database.

Some tests with small websites were made in order to verify if the system was working correctly. A major test was made using a large website, in a Pentium M 1400 MHz with 512 MB RAM, under Windows XP Service Pack 2 operating system, with a cable network with 4 Mbps bandwidth, consisting of 318.956.318 bytes in 37.768 files, generating five tables with characteristics described in table 11.

Table 11. Generated tables

Table	# records	# fields
1	5	2
2	299	3
3	34.817	4
4	34.817	2
5	64.909	3

The automatic tool was tested in three steps:
1. Using `wget` to get the whole website.
2. Using `HTML Tidy` to clean HTML code.
3. Parsing HTML files and generating the database.

The results are presented in table 12.

Table 12. Test Results

Step	Total
1	2h25m49s
2	4h23m12s
3	3h15m43s
Total	10h04m44s

After this test took place, one person who usually works with computers at a user-level, collected the same data manually for the same time of the test result. The user

copied data from the browser and pasted it in a Spreadsheet. The user is not a database expert, although is familiar with concepts of tables and relationships.

The test was split in four parts, 3 of 2 hours and 30 minutes, and the last of 2 hours 34 minutes and 44 seconds. During this time, only 1.256 in 37.768 files were processed.

It is also important to notice that key values were not automatically processed, since they were part of the hyperlink. When pasted in a spreadsheet, both the hyperlink and the text associated are in the same cell. Splitting them would also take some time.

4 Conclusions

Results show that an automatic tool for constructing a database from a website saves time in a considerable proportion. We must take into account that the number of files processed manually did not include the analysis of relationships between data and that key values were not automatically processed.

In this particular case, although the original database is not known, the generated database is believed to be close to the original one, except for some data that might not be used in the website.

It is not intended to compare this approach with semi-automatic ones. Semi-automatic approaches need an expert, while the automatic approach presented in this paper may be used by any common user. So, even if the semi-automatic approach was faster, the automatic approach requires less expertise.

Moreover, the automatic tool performance may be enhanced in several ways. Some future work will include, among other possibilities:

- Letting `htmlparser` download files instead of `wget`.
- Cleaning only necessary data instead of using `HTML Tidy` on the whole HTML file.
- The effort was made mostly in assuring that the tool worked properly, not in making it run fast, so code could be optimized.

References

1 Luke Welling and Laura Thomson, PHP and MySQL Web Development, 3rd Ed. SAMS, 2005.

2 Hugh E. Williams and David Lane, Web Database Applications with PHP & MySQL, 2nd Ed. O'Reilly & Associates, 2004.

3 Jay Greenspan and Brad Bulger, MySQL/PHP Database Applications. M&T Books, 2001.

4 Marco Bellinaso, ASP.NET 2.0 Website Programming: Problem - Design - Solution (Programmer to Programmer). Wiley Pub., 2006.

5 Bill Evjen, Scott Hanselman, Farhan Muhammad, S. Srinivasa Sivakumar, and Devin Rader, Professional ASP.NET 2.0. Wiley Pub., 2006.

6 Jesus Mena, Data Mining your Website. Digital Press, 1999.

7 Georg Chang, Marcus J. Healy, James A. M. McHugh, and Jason T. L. Wang, Mining the World Wide Web – An Information Search Approach. Kluwer Academic Pub., 2001.
8 GNU wget. http://www.gnu.org/software/wget/.
9 HTML Tidy Library Project. http://tidy.sourceforge.net/.
10 HTML Parser. http://htmlparser.sourceforge.net/.
11 Java Programming Language. http://java.sun.com/.

Crawling the web for faces

Roberto Rodrigues

PRODEI, Faculdade de Engenharia da Universidade do Porto, Portugal
pro06010@fe.up.pt

Abstract. Most of the present image search engines are text-based, creating indexes based on words associated to the images. There is also some experimental work in creating indexes based on visual characteristics of images. This paper outlines the design of a Human Facial Image Search Engine that uses facial characteristics of an image. A set of Web Services were developed to crawl, analyze and process web pages, gathering information and filling a database and a front-end application was created to search the database for similar faces. Preliminary results show that the goal was achieved, providing a good guideline for further developments of this prototype.

1. Introduction

The internet is growing and, every month, millions of new servers and sites are added, according to the monthly Web Server Survey[1]. Web crawlers, spiders or robots, are automated programs created to visit a number of web pages and extract some sort of information, and for that reason, are essential for information retrieval on the Internet. But, as it is impossible to have all the internet pages collected and indexed, a goal-directed crawling is a powerful mean for topical resource discovery, as stated in [3]

Most of the search engines are keyword oriented meaning that, according to some metric, relevant pages are returned. If we want to search for images, this can be done by general search engines or by specialized image search engines, dedicated to the search of images or multimedia. The meta-search engines are also used, passing on search requests to more than one search engine and collecting the results. But these methods are still text based search engines and not image characteristics based.

In this paper we present the design and implementation of a prototype for a Human Facial Image Search Engine, using a Best First search, with the following essential features:

- Flexibility – with minor adjustments should be possible to use our system in various scenarios.
- Fault tolerance – in case of network or process failure, our system should keep the data loss to a minimum.
- Maintainability and configurability – an interface should be provided in order to monitor the crawling process and should be possible to make some online adjustments to the crawling process.

[1] http://news.netcraft.com/archives/2007/01/index.html

There are several interesting applications for this kind of browser. Security forces can get a list of similar faces when uploading an image of a suspect, model agencies can get a list of all models similar to a face that a client requested or we can try to find the whereabouts of an old or new friend.

The remainder of this paper is structured as follows: Section 2 reviews some of the related work in this area and Section 3 describes the proposed architecture and tools used. Section 4 presents the results obtained and Section 5 offers concluding remarks and future work.

2. Related Work

Because of the large volume of the Internet and its rate of change it's very difficult or even impossible to be able to index all the Internet pages. As stated in [3] less than 40% of the Internet is cover by the largest crawlers and therefore, the best approach is to implement a Focused Crawling, that index and maintains pages on a very specific set of topics, with a smaller investment in hardware and network resources.

Over the years, several studies were proposed with innovating design for Web Crawlers [8, 10, 19] and different crawling strategies were presented. The most relevant are showed on [8] and on [18], but for this paper we are only interested in two types of strategies: Breadth first or Best First. Most of work done in evaluation web crawlers or topical web crawlers uses a metric based on text, measuring the effectiveness of a web crawler by the amount of relevant pages with that text that are returned. In this paper we used image characteristics, therefore only those two methods were interested.

On breadth-first strategies, a FIFO queue is used, crawling the links in the order they are encountered, and, as it is stated on [11], it stills offers the bests results even if we are working with a focused crawler. Figure 1 illustrates this algorithm.

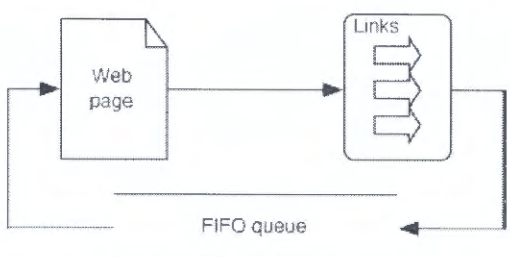

Fig. 1. Breadth first algorithm

On best first strategies some kind of estimation criterion is used to select the next link to crawl. Figure 2 illustrates this algorithm.

When indexing a facial image, there are several restrictions to the actual success that can be achieved in classifying an image as a facial image. Variations of pose, illumination, background and facial expressions can negatively influence the results, but facial recognition software has reached a mature state of development and several methods can be used to get better results.

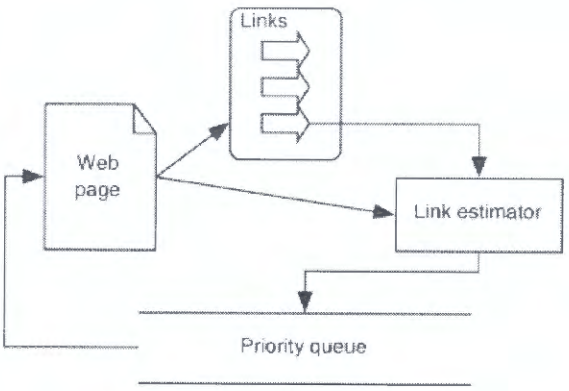

Fig. 2. Best first algorithm

Some of those methods can be resumed in:

- PCA [20, 22, 25, 26] - Derived from Karhunen-Loeve's transformation. Given an s-dimensional vector representation of each face in a training set of images, Principal Component Analysis (PCA) tends to find a t-dimensional subspace whose basis vectors correspond to the maximum variance direction in the original image space. This new subspace is normally lower dimensional (t<<s). If the image elements are considered as random variables, the PCA basis vectors are defined as eigenvectors of the scatter matrix.
- ICA [1, 13] - Independent Component Analysis (ICA) minimizes both second-order and higher-order dependencies in the input data and attempts to find the basis along which the data (when projected onto them) are - *statistically independent* . Bartlett et al. provided two architectures of ICA for face recognition task: *Architecture I* - statistically independent basis images, and *Architecture II* - factorial code representation.
- LDA [2, 6, 16, 17, 29, 30] - Linear Discriminant Analysis (LDA) finds the vectors in the underlying space that best discriminate among classes. For all samples of all classes the between-class scatter matrix *SB* and the within-class scatter matrix *SW* are defined. The goal is to maximize *SB* while minimizing *SW*, in other words, maximize the ratio det|*SB*|/det|*SW*| . This ratio is maximized when the column vectors of the projection matrix are the eigenvectors of (*SW*^-*1* × *SB*).
- EP [14, 15] – An eigenspace-based adaptive approach that searches for the best set of projection axes in order to maximize a fitness function, measuring at the same time the classification accuracy and generalization ability of the system. Because the dimension of the solution space of this problem is too big, it is solved using a specific kind of genetic algorithm called Evolutionary Pursuit (EP).
- EBGM [27, 28] - Elastic Bunch Graph Matching (EBGM). All human faces share a similar topological structure. Faces are represented as graphs, with nodes positioned at fiducial points (eyes, nose...) and edges labeled with 2-D distance vectors. Each node contains a set of 40 complex Gabor wavelet coefficients at

different scales and orientations (phase, amplitude). They are called "jets". Recognition is based on labeled graphs. A labeled graph is a set of nodes connected by edges, nodes are labeled with jets and edges are labeled with distances.

- Trace Transform [23, 24] - The Trace transform, a generalization of the Radon transform, is a new tool for image processing which can be used for recognizing objects under transformations, e.g. rotation, translation and scaling. To produce the Trace transform one computes a functional along tracing lines of an image. Different Trace transforms can be produced from an image using different trace functionals.
- AAM [4, 5] - An Active Appearance Model (AAM) is an integrated statistical model which combines a model of shape variation with a model of the appearance variations in a shape-normalized frame. An AAM contains a statistical model if the shape and gray-level appearance of the object of interest which can generalize to almost any valid example. Matching to an image involves finding model parameters which minimize the difference between the image and a synthesized model example projected into the image.
- SVM [7, 9, 12] - Given a set of points belonging to two classes, a Support Vector Machine (SVM) finds the hyper plane that separates the largest possible fraction of points of the same class on the same side, while maximizing the distance from either class to the hyper plane. PCA is first used to extract features of face images and then discrimination functions between each pair of images are learned by SVMs.

Sometimes "image search engines" can also be referring to collection-based search engines that index an image collection and several commercial systems like Corbis[2] uses this approach. There are also already several experimental systems available for photo search with facial recognition like RYA[3], or for similarity search with LIKE[4] and IBM has also developed a Query by Image Content[5] .

3. The proposed architecture

According to Wikipedia[6], the behavior of a Web crawler is the outcome of a combination of policies:

- A *selection policy* that states which pages to download.
- A *re-visit policy* that states when to check for changes to the pages.
- A *politeness policy* that states how to avoid overloading websites.
- A *parallelization policy* that states how to coordinate distributed web crawlers.

On developing our system we tried to follow those 4 policies.

[2] http://pro.corbis.com/
[3] http://www.riya.com/
[4] http://www.like.com/
[5] http://wwwqbic.almaden.ibm.com/
[6] http://en.wikipedia.org/wiki/Web_crawler

On figures 3 and 4 we illustrate an overview of the Human Facial Search Engine.

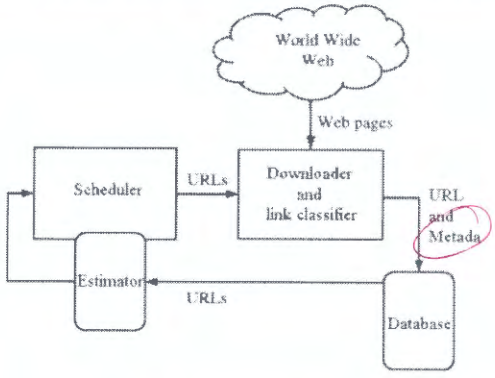

Fig. 3. Crawler Module

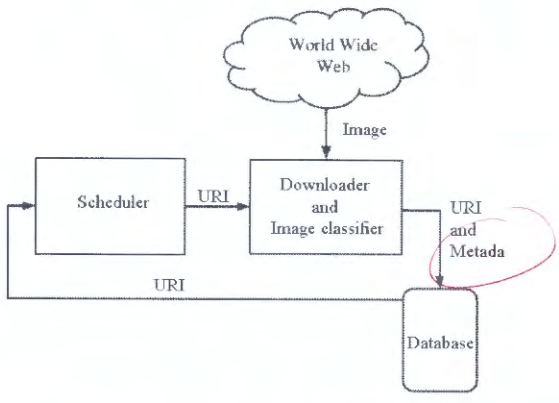

Fig. 4. Image classifier Module

Our system has 3 major modules that can work in parallel:

- **Image and web pages acquisition** – The heart of this module is a topic driven-crawler, responsible for crawling the web using a best first algorithm, filling the database with the URL's found and other information, like the crawling date and the page signature. The page signature is used to detect mirrored pages and also to improve the re-visit policy. The database has also a set of flags to indicate if the page was already been processed or is being processed, used to improve the fault-tolerance. The crawler is also a polite robot because it respects the robots.txt protocol and has a time interval between crawls. To construct the robot we used a VB.NET[7] open source code available on the Internet.
- **Image Classifier** – Due to time restrictions, it was not possible to develop and use any of the several algorithms presented on Section 2. Instead, we use a free

[7] http://www.skycoder.com/downloads/

license granted by Betaface[8] for this paper. With this software, it was possible to classify the images in the web pages and, for each face, its score were saved along with the identification key for the Betaface database, a face key and several facial features. With the face key or the identification key, it is also possible to get a list of database images that are similar to that image.

- **Search Engine** – This module is responsible to show the images that contains faces most similar to an input facial image. This is done using the Betaface software. The input image is classified and its identification key is sent to Betaface. A list with the identification key of the most similar images is returned and each image and the URL of the page where that image was found originally are showed.

To compose this paper the following tasks were scheduled:

1. Add to the database the seed for the crawler.
2. Start the Crawler Module to start filling the database with URL's and images locations.
3. Start the Image Classification Module to select the facial images.
4. Test the Search Engine.
5. Compile the results obtained.

To test the search engine it was necessary to collect several images of the same person in different situations because there are several factors that influence a way that an image appear, such as, lighting conditions, facial expressions or head pose and we needed to test if our search engine was able to find similar images in real conditions.

After some research we decided to use the Hi5[9] which is a virtual community of friends that interacts socially. Each user in Hi5 has photos of the user or friends and more links to those he considers friends, who will have links to more friends. Each page has also links to publicity pages and pages outside Hi5. Each time a Hi5 user page is crawled more users are added to the database to be processed. Since in Hi5 each user can have until 20 images, using Hi5 we can have a data set of uncontrolled images very similar to a real life situation, where we do not control the images that feed our system or application.

As stated before, we wanted a flexible system that could be used in different situations, with minor adjustments. In figure 5 our generic best first algorithm, used for crawling, is illustrated.

The Image and web pages acquisition module is implemented as a web service providing the following operations:

- GetURLs

  This operation has an URL as one of the parameters and returns a string indicating the time used to process that URL. This operation adds to the database all the URLs and images found. An image, in our prototype, is any address located in the <img> or <a> tag with one src attribute. This allows us to find images hiding in JavaScript or other script language.

[8] http://www.betaface.com
[9] http://www.hi5.com

```
BestFirstSearch(starting_url) {

    user ← getUser(starting_url);

    getUrlImages(url_list, user);

    while (Not Empty(url_list)) {

        link := dequeue_top_link(url_list);

        fetchAndAdd(link);

        getUrlImages(url_list, user); }

}
```

Fig. 5. Best first algorithm used in Hi5

- Dispatch
  This operation gets the first unprocessed web page from the database and processes it. It the returns a string indicating the time used to process that URL.
- DispatchInDeep
  This operation is specific to Hi5, and gets the first unprocessed web page - that refers to a user profile or gallery, from the database. Then, it processes all the addresses that have any reference to that user. All the new web pages added during this process, if they refer to the user are also processed. It returns a string indicating the time used to process that user.
- AutoDispachInDeep
  This operation is also specific to the Hi5, and processes all unprocessed web pages that refers to a user profile or gallery.

All the data used in this paper was stored in a relational database. Relational databases have reached a mature state of development, being very reliable and with several features to backup or even restore data in case of damage. Besides, in our case, we needed a fast method to avoid the duplication of web pages and images addresses. Using the table key feature it is not necessary to check in code if that address is already in the database. If the address does not exist, it will be added, if it already exists, the relational database management system does not allow the duplication.

Our database has 3 tables:

- URLs – this table stores the address of the web page visited and 3 flags indicating if the web page has been processed, is being processed or has not yet been processed and also the date when the web page was added to the database.
- Images – this table stores the address of the image, the web page where the image is showed and 4 flags indicating if the image has been processed, is being processed, has not yet been processed or if it is a face image.

- Faces – this table stores all the faces that were detected by the facial recognition software. For each face in the table, we store the address of the image, the face order in that image, the facial score and a faceKey identifying the face in the Betaface database.

4. Results

The prototype was tested on a 3.20GHz Pentium 4 PC with 1,50GB of RAM, running Microsoft Windows XP, with Service Pack 2 and with an DSL connection with 8 Mbps download speed.

So far the prototype has crawled 19398 web pages, finding 36672 images. 19677 were already processed and 5866 faces were found. There are still 49888 web pages to be processed. In average, each web page processed ads 7 news links to the database.

We tested the prototype in 3 phases, always adding to the database web pages from the same host, in this case the Hi5:

1. Using the search page of Hi5 as seed, we started to fill the database with images and web pages addresses. Each page takes around 1 minute to be processed. According to Hi5 they have around 610000 users, so to assure a maximization of images in database, we used the AutoDispachInDeep operation. Depending on the amount of images and the speed of the network, this operation can take around 20 to 30 minutes.
2. Simultaneously, the facial recognition application started classifying the images in the database and adding the faces. The facial recognition software had a low tax of false positives but the false negatives were a little higher than expected. The average time to processing and clustering is 3 images per minute.
3. Finally, some random tests were made to test the accuracy of the search engine. The search engine takes between 11 to 42 seconds to process the results.

The engine was tested with several images from the database. Male and female images were tested. Most of the subjects were young around the 20 years old. Different sizes and quality images were used and were also used images with noise added.

In this phase of the prototype, we were only trying to found the images of the same person. Each image used had always at least 4 pictures in the database classified as faces.

The facial recognition software always returned the original image, even with noise added. However it never returned all the facial images from that user. The tax retrieval, in the performed tests, was never above the 50%.

Each image has a small thumbnail. When both were classified was facial images, both were tested to see if they had the same results. In our tests the 2 images didn't returned the same results.

We also test the search engine, uploading a different image that was not in the database. In this case, at least one image was found.

5. Discussion and future work

In this paper we described the design and implementation for a Human Facial Image Search Engine, and presented some preliminary results. There are many improvements that can be made in each module. Being independent, each module can be improved separately.

Using a database to store the Uri's and image features, allowed a huge saving in storage resources, without compromising the speed of the system. Using the flags to categorize the state of an URL and with some few modifications, it is possible to apply a parallel approach to our system.

The image processing module is working fine, ~~been~~ being able to detect faces in different conditions, but has two major problems. Without the source code for the image processing, it is not possible to know if we are using the more appropriated algorithms to our images. Second, the use of the web service slows down the processing speed. There are several algorithms available and, with the modular design of this prototype, it's possible to add more functionalities or improve the present ones. Once again, we had a tight schedule to this prototype so we had to use already made software.

With this paper we showed that it is possible, with few resources, to create a Human Facial Image Search Engine using uncontrolled data sets of images, providing other inputs forms besides texts. To better test this prototype more time is needed to gather more images and a re-visit algorithm should be used to keep the database updated.

Acknowledgements

To Betaface software and particulary to Oleksandr Kazakov for the precious help.

Bibliographic References

[1] Bartlett, M.S., Movellan, J.R., Sejnowski, T.J.: Face Recognition by Independent Component Analysis. In: IEEE Trans. on Neural Networks, Vol. 13, No. 6, (November 2002) 1450-1464

[2] Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection. In: Proc. of the 4th European Conference on Computer Vision, ECCV'96, Cambridge, UK, (April 1996) 45-58

[3] Chakrabarti, S., Martin, van den B., Dom, B.: Focused crawling: a new approach to topic-specific Web resource discovery. In: Proceedings of 8th International World Wide Web Conference (WWW8), (1999)

[4] Cootes, T.F., Taylor, C.J.: Statistical Models of Appearance for Computer Vision. In: Technical Report, University of Manchester, (September 1999)

[5] Cootes, T.F., Walker, K., Taylor, C.J.: View-Based Active Appearance Models. In: Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition, , Grenoble, France, (March 2000) 227-232

[6] Etemad, K., Chellappa, R.: Discriminant Analysis for Recognition of Human Face Images. In: Journal of the Optical Society of America A, Vol. 14, No. 8, (August 1997) 1724-1733

[7] Guo, G., Li, S.Z., Chan, K.: Face Recognition by Support Vector Machines. In: Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition, Grenoble, France, (March 2000)196-201

[8] Hafri, Y., Djeraba., C.: Dominos: A New Web Crawler's Design. In: 4th International Web Archiving Workshop (IWAW'04), Bath (UK), 2004.

[9] Heisele, B., Ho, P., Poggio, T.: Face Recognition with Support Vector Machines: Global versus Component-based Approach. In: Proc. of the Eighth IEEE International Conference on Computer Vision, ICCV 2001, Vol. 2, Vancouver, Canada, (July 2001) 688-694

[10] Heydon, A., Najork, M.: Mercator: A Scalable, Extensible Web Crawler. In: World Wide Web, 2(4), (December 1999) 219-229

[11] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. In: 7th Int. World Wide Web Conference, 1998.

[12] Jonsson, K., Matas, J., Kittler, J., Li, Y.P.: Learning Support Vectors for Face Verification and Recognition. In: Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition, Grenoble, France, (March 2000) 208-213

[13] Liu, C., Wechsler, H.: Comparative Assessment of Independent Component Analysis (ICA) for Face Recognition. In: Proc. of the Second International Conference on Audio- and Video-based Biometric Person Authentication, AVBPA'99, Washington D.C., USA, (March 1999) 211-216

[14] Liu, C., Wechsler, H.: Evolutionary Pursuit and Its Application to Face Recognition. In: IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 22, No. 6, (2000) 570-582

[15] Liu, C., Wechsler, H.: Face Recognition Using Evolutionary Pursuit. In: Proc. of the Fifth European Conference on Computer Vision, ECCV'98, Vol II, Freiburg, Germany, (June 1998) 596-612

[16] Lu, J., Plataniotis, K.N., Venetsanopoulos, A.N.: Face Recognition Using LDA-Based Algorithms. In: IEEE Trans. on Neural Networks, Vol. 14, No. 1, (2003) 195-200

[17] Martinez, A.M., Kak, A.C.: PCA versus LDA. In: IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 23, No. 2, (2001) 228-233

[18] Menczer, F, Pant, G, Srinivasan, P.: Topical web crawlers: Evaluating adaptive algorithms. In: ACM Transactions on Internet Technology, Forthcoming, (2003)

[19] Miller, R. C., Bharat, K.: SPHINX: A Framework for Creating Personal, Site-Specific Web Crawlers. In: Computer Network and ISDN Systems v. 30, (1998) 119-130

[20] Moon, H., Phillips, P.J.: Computational and Performance aspects of PCA-based Face Recognition Algorithms. In: Perception, Vol. 30, (2001) 303-321

[21] Najork, M., Wiener, J.: Breadth-first search crawling yields high-quality pages. In: Proc. 10th International World Wide Web Conference, (2001) 114–118

[22] Pentland, A., Moghaddam, B., Starner, T.: View-Based and Modular Eigenspaces for Face Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, Washington, USA, (1994) 84-91

[23] Srisuk, S., Petrou, M., Kurutach, W., Kadyrov, A.: Face Authentication using the Trace Transform". In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03), Madison, Wisconsin, USA, (June 2003) 305-312

[24] Srisuk, S., Kurutach, W.: Face Recognition using a New Texture Representation of Face Images. In: Proceedings of Electrical Engineering Conference, Cha-am, Thailand, (November 2003) 1097-1102

[25] Turk, M., Pentland, A.: Eigenfaces for Recognition. In: Journal of Cognitive Neuroscience, Vol. 3, No. 1, (1991) 71-86

[26] Turk, M., Pentland, A.: Face Recognition Using Eigenfaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, USA, (1991) 586-591

[27] Wiskott, L., Fellous, J.-M., Krueuger, N., Malsburg, C. von der: Face Recognition by Elastic Bunch Graph Matching. In: Intelligent Biometric Techniques in Fingerprint and Face Recognition, eds. L.C. Jain et al., CRC Press, (1999) 355-396

[28] Wiskott, L., Fellous, J.-M., Krueuger, N., Malsburg, C. von der: Face Recognition by Elastic Bunch Graph Matching. In: IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, (1997) 776-779

[29] Zhao, W., Chellappa, R., Krishnaswamy, A.: Discriminant Analysis of Principal Components for Face Recognition. In: Proc. of the 3rd IEEE International Conference on Face and Gesture Recognition, FG'98, Nara, Japan, (April 1998) 336

[30] Zhao, W., Krishnaswamy, A., Chellappa, R., Swets, V., Weng, J. : Discriminant Analysis of Principal Components for Face Recognition. In: Face Recognition: From Theory to Applications, H. Wechsler, P.J. Phillips, V. Bruce, F.F. Soulie, and T.S. Huang, eds., Springer-Verlag, Berlin, (1998)73-85

Sessão 4

Programação Orientada a Aspectos

Testing Java based GUIs using an AOP approach

Pedro Mendes, José Moura, and Pedro Abreu

Faculdade de Engenharia da Universidade do Porto Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal

Abstract. This paper presents a technique for automating the interaction over Java GUIs using an Aspect Oriented Programming approach based on AspectJ for Java. Known methods for automatic and independent GUI interaction require GUI object recognition features by the tools involved in the process. Also the recognition procedures are dependant from the inherent operating system in which the GUI runs. These methods ignore several of the GUI's common characteristics over distinct operating systems and are closely attached to the data related to specific execution sessions. By using aspects, it is possible to modify the GUI building process in the virtual machine by adding extra default listeners that will work as hidden loggers of user actions. Replaying the recorded behavior is assured by Java's Robot classes, while storage is assured via XML files. This method proved to be feasible when used over GUIs with *Java Swing* based components although there are still some performance issues to be improved.

1 Introduction

Introducing automation features in the test phase of any software development process is the correct measure to take in order to assure high quality standards. Doing so with graphical user interface (GUI) projects brings additional difficulties, and the process requires the usage of specific tools to perform the code interaction with these artifacts. These tools, named as *capture-replay*, provide a recording function, with the purpose of making the interaction script creation simpler and automatic. Although the test scripts created by the tools using only the capture feature constitute a good base to initiate the development of a test project, they still have some limitations and drawbacks [1]:

- Hard-coded data values;
- Non-modular, hard-to-maintain;
- Lack of standards for reusability.

There are other approaches that make use of model based techniques, but they require a certain analysis expertise by the testers. Briefly, test automation is achieved through the exploration of a given model that contains enough iuformation on the GUIs' transitions according to user actions on the components. Graphically these models correspond to fiuite automata or Unified Modeling Language (UML) behavioral diagrams[2].

In either approach, the most common method to recognize the GUI components is to use the operating system application programming interface (API) at hand, to retrieve their specific identifiers so that the properties can be externally manipulated by an intrusive process. The fact is that there is no real gain in externalizing these mechanisms. If it was possible to add logging code to the original application or even to the framework in which it is based on in a modular way, in order to retrieve the information that is expected to be tested, efficiency would doubtlessly be raised.

- Getting the desired object properties could be achieved through a transparent procedure;
- Coding language could as complex as the application under test (AUT) source code language;
- Security would be raised as there would be no longer the need to use the operating system API.

Aspect Oriented Programming (AOP) is a recent programming concept concerning software. Its adoption may have an impact similar to the one observed with the introduction of object-oriented programming (OOP)[3]. The AOP concept raises applications' modularity and goes further than traditional methodologies like the ones used with OOP. Although these mechanisms also have their own reusability patterns, they do not respond to certain issues like the ones that are transversal to the entire application such as: state machine manipulation or logging. These *cross-cutting concerns* are the basis for the creation of an aspect. An aspect is the entity that codes the behavior to be added to some application in given points of its execution. The code itself is named as *advice-code* and the points where such intersection occurs, *pointcuts*[4].

The most mature implementation of the concept is AspectJ[5]. It corresponds to both an extension to the Java programming language and a proper virtual machine with certain differences, when compared to the one supplied by Sun. These differences are related to dynamic pointcut matching features although most of the work is done with static analysis.

Therefore the case study presented in this paper refers to *Java Swing*[6] based GUIs running on an AspectJ virtual machine with aspects developed on this platform. Aspects code the addition of an extra listener to every loaded GUI component through the match of a pointcut explained in the next section.

For each interaction an appropriated object is created. These objects' information is later used for the replaying features.

Since this case study uses only Java based GUIs, behavior replay is achieved through the Java Robot classes.

The paper details the process in section 2; section 3 concludes the paper and in the end some acknowledgements are made.

2 Automating Interaction

On the architectural level, the developed solution is structured in three components, which correspond to the three technologies used, and to the following

subsections. Briefly, it is divided into the record, storage and event replay modules. The second module mentioned is responsible for the serialization of Java objects to eXtensible Markup Language (XML)[7] files, which follow the proposed schema by Sun, to represent them in this format. Figure 1 illustrates the proposed architecture.

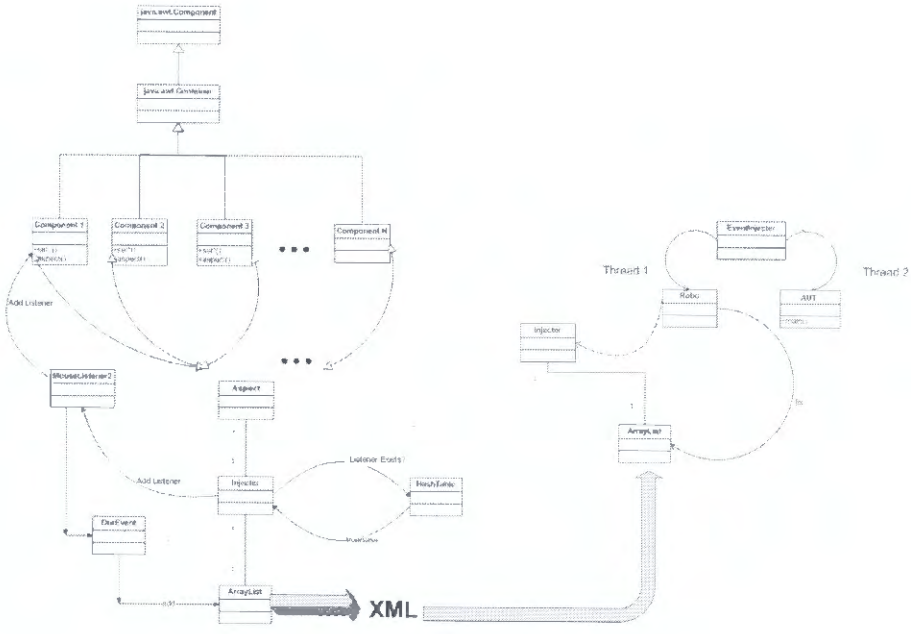

Fig. 1. Solution Architecture Diagram

2.1 Record

The defined strategy to capture user behavior, was to extend the *JComponent* constructor by adding an extra listener with the purpose of logging user actions. *JComponent* is the root class for all *Java Swing* components, and because of that it is the natural target of all the *super* methods that are called by the virtual machine when it is building some GUI[8]. However, AspectJ has not got sufficient privileges, to access methods and inject code, so near the *Class* class, on user mode.

Three options emerge to solve this problem : reconfigure the virtual machine to run aspects in administrator mode, change its security settings or inject code in other methods which can be accessed in user mode, even if they do not belong to the *JComponent* class. The third solution was the adopted one because it is

the least intrusive and also the only one that does not alter the default security settings.

After analyzing the Java Swing source code it is observable that every *JComponent* and *Component* makes at least one call to a *setter* method (method which alters an attribute of a class and matches the regular expression set*). This led to the use of this property for the creation of the aspect pointcut as shown in code listing 1.1.

Listing 1.1. Injection Aspect

```
1   pointcut adds(Component comp) : (call(* Component+.set
        *(..)) && target(comp) );
2       after(Component comp) : adds(comp) {
3       Injecter.injectMouseListener(comp);
4       Injecter.injectKeyListener(comp);
5    }
```

As can be seen in the previous listing, this aspect adds to the component at hand an extra mouse listener which logs every mouse interaction performed on it. Code listing 1.2 illustrates the listener declaration.

Listing 1.2. MyMouseListerner

```
1  public class MyMouseListener2 implements MouseListener {
2          public int eventcard=0;
3          ...
4
5          public void mouseClicked(MouseEvent e) {
6                  int x=MouseInfo.getPointerInfo().getLocation()
                       .x;
7          int y=MouseInfo.getPointerInfo().getLocation().y;
8          Injecter.injectMouseListener((Component)e.getSource())
               ;
9          Injecter.eventMemory.add(new OurEvent(x,y,++eventcard,
               "mouseClicked"));
10         }
```

For each action performed by the user, an *OurEvent* object is generated containing: the type of action performed, (which listener was triggered), the action's properties when referring to the mouse input interface (x,y coordinates relative to the object), and the hash code ID of the object. Since the injected code is viewed by the application as its own, these properties are directly accessible without any kind of security override, or intrusive mechanism.

For the purpose of this case study this addition is sufficient to prove the correctness of the method; although other aspects are present in order to make this approach complete. These aspects focus on adding listeners regarding other input interfaces. As an example of this possible extension another aspect is shown in code listing 1.3. This one however is slightly different from the common ones. It was developed in order to make all the target applications closable with the "CTRL+ALT+F9" key combination.

Listing 1.3. MyMouseListerner

```
1 public class KeyListener2 implements KeyListener {
2 ...
3         public void keyPressed(KeyEvent arg0)
4         {
5                 if(arg0.getKeyCode()==KeyEvent.VK_F9 && arg0.
                      isShiftDown() && arg0.isControlDown())
6         }
7         ...
```

2.2 Storage

After obtaining interaction data from the recording module, in the form of a collection of *OurEvent* objects, it is necessary to store it. There are several hypotheses to solve this issue. In this case study the solution found was to serialize in the file system the collection as an XML file. The supporting reason for this option is the fact that XML is human readable. Later on, these files are loaded as *OurEvent* instances for the replay stage, and because the files are both understandable and editable by the human tester, replay behavior can be changed without having to repeat the recording process. Figure 2 shows an example output produced by the record module.

```
- <java version="1.5.0" class="java.beans.XMLDecoder">
  - <object class="java.util.ArrayList">
    + <void method="add"></void>
    + <void method="add"></void>
    + <void method="add"></void>
    - <void method="add">
      - <object class="OurEvent">
        - <void property="pos">
            <int>34</int>
          </void>
        - <void property="type">
            <string>mouseExited</string>
          </void>
        - <void property="x">
            <string>788</string>
          </void>
        - <void property="y">
            <string>29</string>
          </void>
        </object>
      </void>
    </object>
  </java>
```

Fig. 2. Example Output File

2.3 Replay

Since the application domain of this case study is Java, the Robot classes introduced in the *Java Development Kit* version 1.3.2 have been used. The replay module is actually quite simple, it just loads the XML file containing the info to rebuild a series of *OurEvent* objects according to a certain order and then inputting its information in a proper format for the Robot classes to act. Mouse clicks' coordinates are relative to the designated component, so the GUI can be anywhere on the desktop, as the user behavior will be exactly reproduced. Components are matched according to the hash code ID presented in section 2.1. The replay module follows a multi-thread architecture, with a thread to perform robot actions and another listening to interruption requests. This step is exemplified in code listing 1.4.

Listing 1.4. Actions performed by Java Robot

```
1  ...
2  import java.awt.Robot;
3  ...
4  public class EventInjecter extends Thread{
5          Robot slave;
6          ...
7          public void process(OurEvent event){
8                  String tipo = event.getType();
9                  int x = event.getX();
10                 int y = event.getY();
11                 int xActual=MouseInfo.getPointerInfo().
                       getLocation().x;
12         int yActual=MouseInfo.getPointerInfo().getLocation().y
               ;
13                 if (tipo.equals("mouseClicked"))
14                         gotoAndClick(xActual,yActual,x,y);
15                 else if (tipo.equals("mousePressed"))
16                         gotoAndpress(xActual,yActual,x,y);
17                 else if (tipo.equals("mouseReleased"))
18                         gotoAndRelease(xActual,yActual,x,y);
19         }
20         public void gotoXY(int x1, int y1, int x2, int y2) {
21                 ...
22         slave.mouseMove(x1,y1);
23         slave.delay(25);
24                 }
25         ...
26     }
```

3 Conclusion and Fnture Development

This study proves the correctness of the method for automating the GUI interaction, and shows that the AOP approach can be used even when the cross cutting concerns are part of the framework supporting the developed code. Performance levels of this approach are still far from the ones achieved with conventional methods, but several studies are being conducted in order to achieve acceptable efficiency levels[9,10].

This is still work in progress, and so, there are still some requirements to be fulfilled before this approach can be considered to be complete. During the record stage it should be possible to add timed pause instructions. This should also be possible by editing the XML files. It would also be interesting to time execution pauses according to the GUI execution state.

Adding flow control to the captured interaction is also an emergent requirement. Achieving this feature by using this approach, at its current state, would involve introducing some extra code in the replay stage, however a future iteration of the project will involve partitioning the several flows possible, into distinct XML files that will have an extra tag in order to include information about the XML file to be loaded next.

There are some possible extensions to the work that has been exposed as well.

Storage could be improved across all modules. If the tester tries to code a lot of actions in a single usage session, there may be too many *OurEvent* objects lurking in the virtual machine memory, so storing the data in a transactional database using something like hiberuate [11] would probably solve this problem. Storing the XML files, in a native XML database such as eXist[12] would also ensure better performance levels, for the replay module in presence of large XML files.

Integratiug the replay module with a code coverage tool would expose this uetric, which is used for validating test procedures, across several companies depending on trustworthy and critical applications.

Adding the possibility to take a screen snapshot for each action taken or just for some actious defined by the tester would also constitute an interesting improvement. Obtained screens during the replay stage, could later be compared with snapshots from previous executions providing the means to produce a visual report.

Finally iutegration of the three modules through a GUI would also be a reasonable addition.

4 Acknowledgments

Regards go to Mik Kersten for replying to the emails sent and being very fast iu doiug so. Fiually a special mention goes to professor Ademar Aguiar for introducing the AOP world to the authors aud supervising the project.

References

1. Elfriede Dustin - Effective Software Testing - 50 Specific Ways To Improve Your Testing, Chapter 8,Addison Wesley, 2002
2. Paiva A., Faria J., Tillmann N., and Vidal, R,. - A Model-to-implementation Mapping Tool for Automated Model-based GUI Testing, CFEM 2005 - 7th International Conference on Formal Engineering Methods, Manchester, 2005
3. Sabbah D.,- Aspects - from Promise to Reality, AOSD 2004, Lancaster, UK, March 2004
4. Kiczales G., - Aspect-Oriented Programming, ECOOP'97, Finland, June 1997
5. AspectJ Official Website - `http://www.eclipse.org/aspectj/` - 2006/12/04
6. Java Swing - `http://java.sun.com/javase/technologies/desktop/desktop-documentation.jsp/` - 2006/12/08
7. Extensible Markup Language - `http://www.w3.org/XML/` - 2006/12/09
8. Java SDK Source Code - `http://download.java.net/jdk5/` - 2006/10/24
9. Li Jingwu, and Hendren Laurie - Improving the Compiling Speed of the AspectBench Compile - McGill University School of Computer Science Sable Research Group - August 2006
10. Avgustinov P., Christensen A., Hendren A., Kuzins S.,Lhoták J., Lhoták O., Moor O., Sereni D., Sittampalam G., and Tibble J. - Optimising AspectJ - Oxford University, University of Aarhus, McGill University - November 2004
11. Hibernate Relational Persistence for Java - `http://hibernate.bluemars.net/` - 2006/12/09
12. eXist open source native XML database - `http://exist.sourceforge.net/` - 2006/12/09

FanIN: A Framework for Automatic Aspect Mining

Jorge Almeida

Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal
pro06020@fe.up.pt

Abstract. Aspect mining attempts to identify crosscutting concerns in legacy systems and thus supports the refactoring into an aspect-oriented design. This paper introduces FanIN, an aspect mining tool implemented as an Eclipse plug-in for automatic identification of crosscutting concern seeds in source code. Furthermore, this tool allows for combination of aspect mining and visual techniques and facilitates code navigation. The results show that the aspect mining technique used is suitable for a high degree of automation.

Keywords: aspect mining, aspect-oriented programming

1 Introduction

The research area of aspect mining is directly related with the new paradigm of programming called aspect-oriented programming (AOP). Despite the majority of the developed systems following the object-oriented paradigm (OO), many software engineers try to incorporate the AOP in its software systems, that is, the adoption of aspects reduce the complexity, increase the legibility and improve the modularization, becoming more adjusted for future evolutions [1].

To include AOP in OO software systems, it is necessary to carry through an analysis to identify crosscutting concerns, whose implementation is spread across many modules as tangled and scattered code, to be converted into aspects. However, manually applying this analysis to a legacy system is a difficult and error-prone process. For the fact of being difficult to explore some classes of a system that, are possibly dispersed in many archives, having each class a variable size in terms of attributes, methods and code lines, becomes necessary to make this exploration process automatic. Code-level techniques, tools and methodologies have been designed to automatize the process. The aspect mining research community proposes a great number of such techniques, making it difficult for software engineers to choose which tool is most suitable [2].

This paper introduces FanIN a framework that automate the process of aspect discovery. FanIN combines aspect mining and visualization techniques. The tool support an aspect mining technique based in fan-in analysis. This analysis looks for candidate-seeds among methods invoked from a large number of scattered callsites, and hence has a high fan-in value. The visualization technique makes easier the code navigation and the decision to transforming crosscutting concerns into real aspects.

An evaluation of the tool is made using the JHotDraw[1], a Java framework for drawing two-dimensional graphics, which has been proposed as common benchmark for aspect mining.

The paper is organized as follows: Section 2 defines aspect mining and some aspect mining techniques. Section 3 gives an briefly overview of FanIN framework. Section 4 describe and discuss experimental results applied to JHotDraw. In section 5 the paper concludes with possible directions for future work.

2 Aspect Mining

Aspect-oriented programming [3] is a new programming paradigm with constructs dedicated to handling crosscutting concerns. In an object-oriented system, it often happens that functionalities, such as persistence, exception handling, error management, logging, are scattered across the classes and are highly tangled with the surrounding code portions. Thus tangled code is the interference across different concerns in order to implement one concern, while scattered code is the code of one concern spread across the system. Aspect-oriented programming addresses the issue of crosscutting concerns with a new modularization unit, the aspect, which encapsulates such crosscutting behaviours.

Aspect mining tools try (partly automatic) to identify such concerns [4]. The development of techniques and tools for the identification of concern code is inevitable given the complexity and size of nowadays software systems. These tools permit engineers to employ a strategy to help divide a software system in conceptually coherent pieces (concerns), which they can study and understand in isolation, without worrying about the other code. Eventually in the end of the process, aspect mining allows developers to refactor the system into an aspect-oriented system, where some or all discovered crosscutting concerns are transformed into real aspects. Aspect mining is consequently an important prerequisite for aspect refactoring. (See Figure 1)

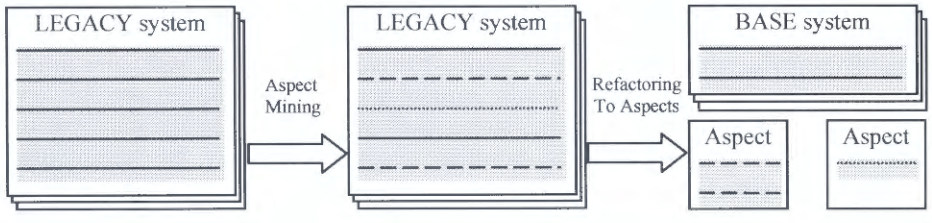

Fig. 1. Migrating a legacy system to an aspect-oriented system

As referred, the process of migrating a legacy system into a system using aspects consists of two steps: the discovery of aspect candidates and the refactoring of some these candidates into aspects. Three kinds of aspect discovery approaches can be distinguished: early aspect discovery, dedicated browsers and aspect mining. The first

[1] http://www.jhotdraw.org/

one tries to discover aspects in earlier phases of the software life-cycle [5, 6]. The other approach is advanced code browsers that provide support for navigating source code and explore cross-cutting concerns [7, 8, 9]. Finally, aspect mining approach automate the process of aspect discovery and proposes aspect candidates [10, 11, 12].

2.1 Aspect Mining Techniques

Several automated code-level aspect mining approaches have been proposed by the aspect mining research community. These approaches can be grouped in: static, dynamic and hybrid aspect mining.

Static Aspect Mining. Based in static program analysis and often require user interaction. The Aspect Browser [7] identifies cross-cutting concerns with textual-pattern matching (much like "grep") and highlights them. The Aspect Mining Tool (AMT) [8] combines text-based and type-based analysis of source code. Krinke and Breu [13] propose an automatic static aspect mining based on control-flow. The fan-in analysis by Marin, van Deursen, and Moonen [14] determines methods that are called from many different places. FEAT [15], the Feature Exploration and Analysis is implemented as a plugin for the Eclipse Platform. FEAT visualises concerns in a system using so-called "concern graphs". A concern graph abstracts the implementation details of a concern by storing the structure implementing that concern.

Dynamic Aspect Mining. DynAMiT (Dynamic Aspect Mining Tool) [10] analyses program traces reflecting the run-time behaviour of a system in search for recurring execution patterns of method relations. Tonella and Ceccato [11] suggest a technique that applies concept analysis to the relationship between execution traces and executed computational units. This dynamic aspect mining approche has as its main limitations the fact of being partial (i.e., not all methods involved in an aspect are retrieved), heing based on a dynamic analysis, and it can determine only aspects that can be discriminated by different execution scenarios (e.g., aspects that are exercised in every program execution cannot be detected). Additionally, it does not deal with code that cannot be executed (e.g., code that is part of a larger framework, but that is not used in a specific application).

Hybrid Aspect Mining. Combine dynamic analysis with static types. Breu reports on a hybrid approach [16] where the dynamic information of the previous DynAMiT approach is complemented with static type information such as static object types.

3 Tool Description

FanIN is a plug-in for Eclipse[2], an extensible, open-source platform that provides a reusable framework for source code analysis. To execute any of the analyses supported by FanIN, tbe user selects a program element in the default Package explorer view of Eclipse; the view shows the Java element hierarchy of the Java

[2] http://www.eclipse.org/

projects in the active workbench. The analysis of FanIN, fan-in analysis, looks into the selected element, builds and stores the call graph for its methods and their callees, together with class hierarchy information.
The results of the analysis are displayed in the Fan-in analysis view as a tree structure of callee-callers elements, sorted by name or fan-in value. From this view, shown in Figure 2 , the user can inspect the source code of each of the displayed elements and apply two filters to restrict the elements in the view to the most relevant candidates: (1) The fan-in value filter selects elements for which the number of callers is higher than a chosen threshold; (2) the utilities filter allows the user to select sub-elements of the analyzed element which are to be excluded from the set of results; (3) elements that could have a large number of callers but typically do not implement a crosscutting concern, like, for instance, collection classes.

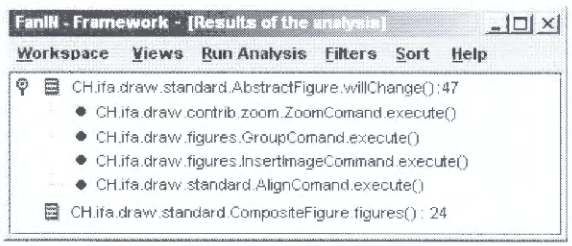

Fig. 2. The FanIn window

The filters also allow indicating sets of elements that should not contribute to the fan-in value of a candidate. For example, the user can indicate that calls from (JUnit) test packages have to be ignored when computing the fan-in metric.

3.1 The Fan-In Metric

Crosscutting functionality can occur at different levels of modularity. Classes, for instance, can assimilate new concerns by implementing multiple interfaces or by implementing new methods specific to superimposed roles. At the method level, crosscutting in many cases resides in calls to methods that address a different concern than the core logic of the caller. Typical examples include logging, tracing, pre-condition and post-condition checks, and exception handling. It is exactly this type of crosscutting that is capture with fan-in analysis. Finding concerns present themselves as a number of distributed calls to a method implementing a crosscutting functionality and the amount of calls (fan-in) is a good measure for the importance and scattering of the discovered concern [17].
The fan-in of a method *m* represents the number of distinct bodies that can invoke *m*. Because of polymorphism, one method call can affect the fan-in of several other methods. A call to method *m* contributes to the fan-in of all methods referred by m as well as to all methods that are referring *m*.

4 Experimental Results

This section presents the results of applying FanIN techniques to version 5.4b1 of JHotDraw, a Java program with approximately 18000 non-commented lines of code and around 2800 methods. JHotDraw is a framework for drawing structured 2D graphics and was originally developed as an exercise to illustrate good use of object-oriented design patterns [18] in a Java program. These particularities recommend it as a well-designed case study, a prerequisite for considering improvements through aspect-oriented techniques. Furthermore, it also shows that modularization limitations are present in even well-designed (legacy) systems.

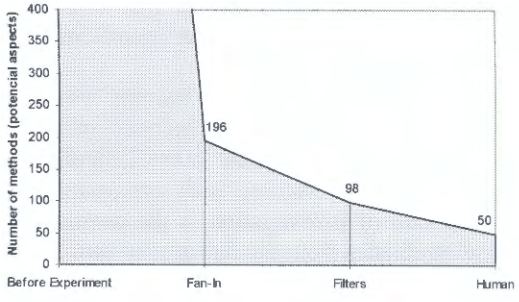

Fig. 2. Evolution of the methods number (potential aspects) during the phase experiment

As described in section 3, fan-in analysis first performs a number of successive steps to filter the methods in the analyzed system. The threshold-based filtering, which selects methods with high fan-in values, kept around 7% of the total number of methods. The filters and utility methods eliminated around half of the remaining methods. (See figure 3)

In the remaining subset, more than half of the methods (52%) were categorized as aspect seeds: methods that by the structure of the calls and the call sites strongly connote a crosscutting concern and thus offer the core elements to further explore and understand the whole extent of the concern's implementation. This seed/non-seed categorization was largely based on manual analysis.

Before the experiment we have 2800 methods, after the fan-in analysis exist 196 methods (potential methods). Appling the filters the number of methods went down for 98. With a manual analysis those methods went down to 50.

For evaluate the performance of the proposed technique it is considered the following expression:

$$\text{Performance}_{FanIN} = 1 - \left(\frac{\text{Methods}_{Filter} - \text{Methods}_{Human}}{\text{Methods}_{BeforeExperiment}} \right). \quad (1)$$

The expression depends on the number of methods at the final of the experiment, the number of the methods after fan-in analysis and the filters application. When the difference between the two values is close to 0 the performance is near 100%.

In the experiment with the JHotDraw the FanIN framework obtained a performance of 98.29%, only an error of 48 methods in around 2800 methods.

5 Conclusions and Future Work

The framework proposed shows that fan-in technique is indeed useful for aspect mining. Migration of existing applications to AOP can be supported by FanIN semi-automated aspect identification method. Interpretation of discovery aspect is instead a human intensive activity.

In future work, new techniques of aspect mining would be investigated and incorporated in the FanIn framework. Those techniques must include both static and dynamic mining. However, in order to draw more general conclusions, it will be necessary to conduct further case studies with programs of different sizes and complexity. Another direction of investigation is the development of a filter which extracts the refactorable candidates from the discovered candidates.

References

1. Hannemann, J., Kiczales, G.: Overcoming the prevalent decomposition in legacy code. In: Workshop on Advanced Separation of Concerns, International Conference on Software Engineering (ICSE) (2001)
2. Kellens, A., Mens, K., Tonella, P.: A Survey of Automated Code-Level Aspect Mining Techniques. In: "To appear in 'Transactions on Aspect-Oriented Software Development, special issue of Software Evolution'" (2006)
3. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M, Irwin, J.: Aspect oriented programming. In: Proc. of the 11th European Conference on Object Oriented Programming (ECOOP), vol. 1241 of LNCS, Springer-Verlag (1997) 220–242
4. Ceccato, M., Marin, M., Mens, K., Moonen, L., Tonello, P., Tourw´e, T.: A qualitative comparison of three aspect mining techniques. In: International Workshop on Program Comprehension (IWPC 2005), IEEE Computer Society Press (2005) 13–22
5. Baniassad, E., Clements, P.C., Araujo, J., Moreira, A., Rashid, A.,Tekinerdogan, B.: Discovering early aspects. IEEE Software 23(1) (2006) 61–70
6. Rashid, A., Sawyer, P., Moreira, A.M.D., Araujo, J.: Early aspects: A model for aspect-oriented requirements engineerin. In: Joint International Conference on Requirements Engineering (RE), IEEE Computer Society Press (2002) 199–202
7. Griswold, W., Kato, Y., Yuan, J.: Aspect browser: Tool support for managing dispersed aspects. In: Workshop on Multi-Dimensional Separation of Concerns in Object-oriented Systems. (1999)
8. Zhang, C., Jacobsen, H.: Extended aspect mining tool. http://www.eecg.utoronto.ca/~czhang/amtex (2002)
9. Zhang, C., Jacobsen, H.A.: Prism is research in aspect mining. In:OOPSLA, ACM (2004)
10. Breu, S., Krinke, J.: Aspect mining using event traces. In: Conference on Automated Software Engineering (ASE). (2004)

11. P. Tonella and M. Ceccato. Aspect Mining through the Formal Concept Analysis of Execution Traces. In: 11th Working Conference on Reverse Engineering (WCRE), IEEE Computer Society (2004) 112-121
12. Breu, S., Zimmermann, T.: Mining Aspects from Version History. In: Proc. 21st Intl. Conference on Automated Software Engineering (ASE 2006), Tokio, Japan, (2006)
13. Krinke, J., Breu, S.: Aspect Mining Based on Control-Flow. Softwaretechnik-Trends (2005)
14. Marin, M., Moonen, L., van Deursen, A.: A common framework for aspect mining based on crosscutting concern sorts. In: Proceedings of the 13th Working Conference on Reverse Engineering (WCRE), IEEE Computer Society (2006) 29-38
15. Robillard, M., Murphy, G.: Concern Graphs: Finding and Describing Concerns Using Structural Program Dependencies. In: 24th International Conference on Software Engineering (ICSE) (2002) 406–416
16. Breu, S: Extending Dynamic Aspect Mining with Static Information. In: 5th International Workshop on Source Code Analysis and Manipulation (SCAM 2005), Budapest, Hungary, (2005)
17. Sellers, B.: Object-oriented metrics: measures of complexity. Prentice-Hall, Inc., (1996)
18. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)

Aspects: Conflicts and Interferences
A Survey

André Restivo and Ademar Aguiar

Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL
arestivo,aaguiar@fe.up.pt

Abstract. Aspect Oriented Programming (AOP) is a programming paradigm that tries to solve the problem of crosscutting concerns being normally scattered throughout several units of an application.
Although an important step forward in the search for modularity, by breaking the notion of encapsulation introduced by Object Oriented Programming (OOP), AOP has proven to be prone to numerous problems caused by conflicts and interferences between aspects.
In this paper, current research paths regarding this particular problem will be analyzed in order to create a solid knowledge base for future works in the area.

1 Introduction

Separation of concerns is at the core of software engineering. In its most general form, it refers to the ability to identify, encapsulate, and manipulate only those parts of software that are relevant to a particular concept, goal, or purpose. Concerns are the primary motivation for organizing and decomposing software into manageable and comprehensible parts [1].

Aspect-Oriented Programming(AOP) [2] is a new programming paradigm that builds on the success of proved paradigms, like Object-Oriented Programming(OOP). The main idea behind AOP is that concerns crosscutting several modules of an application can be developed as single units of modularity and weaved into the application, through a process of composition, in specific points called joinpoints. The motivation behind AOP is that by being able to develop crosscutting concerns as separate units the overall modularity of an application will be improved.

AOP also aims for obliviousness [3], stating that a developer should not have to know about any aspects that are being weaved into his code. To make this a reality, aspects have to be allowed to attach virtually to any position of the original source code. In this way, each developer only has to be concerned about his own modules.

However, by allowing anyone to change the behavior of any piece of code, *encapsulation*, one of the pillars of OOP, and also one of its greatest advantages, is broken. In OOP, encapsulation was the sole warranty that modules where

not misused by external entities. Without that warranty, interferences between aspects will be prone to occur.

Several solutions to this problem are currently being researched. In this paper, several of these solutions will be presented and discussed. The objective behind this work is to create a solid base of knowledge for future works in this specific area.

This paper is organized in the following way: Section 2 will start by reviewing several possible classifications for aspects and the interferences caused by them. Sections 3 and 4 will explain how different authors are approaching the interference detection and interference resolution problem. Avoiding interference problems before they arise is another line of study that will be reviewed in Section 5. Section 6 will present some alternatives to AOP and finally Section 7 will draw some conclusions.

2 The Anatomy of Aspect Interferences

Before starting to solve the aspect interference problem it is important to understand which kinds of interferences are prone to occur when using AOP. This important point is a cause of disagreement between most of the authors approaching this problem, each defining his own terminology.

In the next sections three different approaches to aspect interference terminology and cataloging will be presented. These approaches look at the same problem from different angles and each one of them has an important perspective into the problem.

2.1 Types of Interferences

The most important work done in this area has been the classification of aspect interferences by Tessier [4]. In his work the author points out several different ways in which aspects can interfere with each other:

Crosscutting Specifications - The use of joinpoints, and specially with '*' wildcards, can lead to accidental joinpoints or infinite recursions.

Aspect-Aspect Conflicts - When multiple aspects exist in the same system problems like mutual exclusions between aspects, the importance of aspect ordering or conditional execution of an aspect by another aspect can occur.

Base-Aspect Conflicts - Circular dependencies between aspects and basic classes.

Concern-Concern Conflicts - Aspects changing a functionality needed by other aspect and composition anomalies normally happening due to subtype substitutability.

2.2 Application Affection

According to Katz [5], three types of aspects can be described in respect to how they affect an application. Although not exactly types of interferences this

classification is important as some interferences only happen with some types of aspects. The three different aspect types are the following:

Spectative aspects only gather information about the system to which they are woven, usually by adding fields and methods, but do not influence the possible underlying computations;

Regulatory aspects change the flow of control (e.g., which methods are activated in which conditions) but do not change the computation done to existing fields;

Invasive aspects change values of existing fields (but still should not invalidate desirable properties).

2.3 Aspect Dependencies

Kienzle [6] approached the problem from a different point of view by considering only the relationships of dependency between aspects and the original code. Three different kinds of aspect dependencies have been identified:

Orthogonal aspects provide functionality to an application that is completely independent from the other functionalities to the application. No data structures are shared between these aspects and the rest of the application. This kind of aspects are very uncommon.

Uni-directional aspects depend from some functionality of the application. These can be further divided as preserving, if the application functionality is maintained or enhanced without any current functionalities being altered or hidden, or modifying, if the application functionality is altered or hidden.

Circular aspects are aspects that are mutually dependent of each other. These kind of aspects are so tightly coupled than one can argue if they should really be considered as separate aspects or as one unique aspect.

3 Detecting Aspect Interferences

In order to solve the problem posed by the interference of aspects, a second problem must be solved first: how to detect that an aspect interferes with another aspect or module? Literature has many different ideas about how to solve this problem. In the following sections these ideas will be explained.

3.1 Program Slicing

Balzarotti [7] claims that this problem can be solved by using a technique proposed in the early 80's called program slicing. A slice of a program is the set of statements which affect a given point in an executable program. According to the author the following holds:

> Let A1 and A2 be two aspects and S1 and S2 the corresponding backward slices obtained by using all the statements defined in A1 and A2 as slicing criteria. A1 does not interfere with A2 if $A1 \cap S2 = \emptyset$;

This technique is accurate enough to identify all interferences introduced by an aspect but can also detect some false-positives. Furthermore, the existence of pointcuts that are defined based on dynamic contexts, forces the analysis of every execution trace increasing the number of these false-positives. However the approach has the advantage of removing the burden of having to declare formally the expected behavior of each aspect.

3.2 Aspect Integration Contracts

Contracts have been introduced by Meyer [8] as a defensive solution against dependency problems in OOP. Some authors claim that contracts can be imported into the AOP world in order to assist programmers in avoiding interference problems.

Lagaisse [9] proposed an extension to the Design by Contract (DbC) paradigm by allowing aspects to define what they expect of the system and how they will change it. This will allow the detection of interferences by other aspects that were weaved before, as well as the detection of interferences by aspects that are bounded to be weaved later in the process. According to the author, for an Aspect A bound to a component C the following should be defined:

1. The aspect should specify what it requires from component C and possibly from other software components.
2. The aspect also needs to specify in which way it affects the component C and the functionality it provides (if applicable).
3. The specification of component C must express which interference is permitted from certain (types of) aspects.

This approach has the disadvantage of forcing the programmer to verbosely specify all requirements and modifications for each aspect as well as permitted interferences. On the other hand, the formal specification of behaviors has proven to be a valuable tool in Software Engineering.

3.3 Regression Testing

Katz [5] proposed the use of *regression testing* and *regression verification* as tools that could help identifying harmful aspects. The idea behind this technique is to use regression testing as normally and then weave each aspect into the system and rerun all regression tests to see if they still pass. If an error is found, either the error is corrected or the failing tests have to be replaced by new ones specific for that particular aspect.

3.4 Service Based Approach

It has been noticed by Kienzle [6] that aspects can be defined as entities that require services from a system, provide new services to that same system and removes others. If some way of explicitly describing what services are required by each aspect it would be possible to detect interferences (for example, an aspect that removes a service needed by another aspect) and to choose better weaving orders.

3.5 Introduction and Hierarchical Changes Interferences

Störzer [10] developed a technique to detect interferences caused by two different, but related, properties of AOP languages.

Störzer claims that the possibility of aspects introducing members in other classes can lead to undesired behaviors as it can result in changes of dynamic lookup if the introduced method redefines a method of a superclass. He calls this type of interference *binding interference*.

The other problem Störzer refers to is the possibility of aspects changing the inheritance hierarchy of a set of classes. He claims that this type of changes can also give place to *binding interferences* as well as some unexpected behavior caused by the fact that *instanceof* predicates will no longer give the same results as before.

To detect this kind of conflicts the author proposes an analysis based on the lookup changes introduced by aspects.

Kessler [11] also studied how structural interferences could be detected. However, his approach is based in a logic engine where programmers can specify rules (ordering, visibility, dependencies, ...). In [11], Kessler also described the different type of interferences that are possible with introductions and hierarchical changes and proposed solutions for each one of them.

4 Aspect Interference Resolution

4.1 Stateful Aspects

Douence [12] [13] proposed a framework that allowed programmers to solve aspect interferences by using a dedicated composition language. The idea behind this language is to allow an explicit composition of aspects at the same execution point. The interferences solved by this approach are those that occur when the same crosscut is used by two different aspects.

5 Avoiding Aspect Interferences

5.1 Robust Pointcuts

Recently, Braem [14] proposed a method based on Inductive Logic Programming in order to automatically discover intensional pattern-based pointcuts. This method aims at solving the *fragile pointcut problem*, that states that pointcuts defined by enumeration do not cope well with program evolution and that the use of wildcards to solve this problem can cause interferences by means of accidental joinpoints.

5.2 Crosscutting Interfaces

Crosscutting Programing Interfaces, or XPIs, have been introduced by Griswold [15] as a form of making AOP programming easier. By using abstract interfaces to expose pointcut designators, this approach decouples aspect code from the unstable details of advised code without compromising the expressiveness of existing AO languages or requiring new ones. The author expects that integrated-development-environment support could aid programmers by showing the scope of an XPI applicability.

5.3 Joinpoint Encapsulation

The reason why AOP is so powerful and at the same time so easily misused is that joinpoints are available for weaving without any knowledge of the programmer that originally developed the code. On one hand, this allows the developers to be oblivious about what code is going to be weaved in their code, but on the other hand is the source of interferences and conflicts. Larochelle [16] proposed the idea of adding joinpoint encapsulation by introducing a new kind of advice: join point encapsulation advice, or restriction advice. Restriction advice serves to encapsulate the join points selected by a pointcut against modification by other aspects thus enabling the modular representation of the encapsulation of crosscutting sets of join points.

6 Alternatives to Aspects

Several other modularization mechanisms related to multi-dimensional separation of concerns have appeared as alternatives to AOP, promising to solve some of the problems pointed out about aspects, like the interference problem. In the following sections some of these alternatives will be described.

6.1 Layered Designs

The *mixin layers* approach [17] looks at different concerns as different collaborations of the same classes but with different roles. As mixin layers preserve the structure of the design it enhances the maintainability of an application. If changes are introduced in the design it is easy to isolate them.

6.2 Composition Filters

Composition Filters [18] is another alternative to AOP that claims that separation of concerns can be achieved by applying different filters at the message passing interface of objects.

6.3 Hyperspaces

The Hyperspaces approach [1] considers all artifacts that constitute an application as being concerns (called *units* in this approach). These artifacts can be a class, an interface or even a requirement specification, as long as it can be written in any given language.

Separation of concerns is achieved in three different stages: *identification*, where concerns are first identified and written in their own specific language, *encapsulation*, where units are prepared so to be used in a software system and *integration*, where concerns that have been encapsulated are used to create a complete application.

7 Summary and Conclusions

In OOP, encapsulation allowed programmers to establish a barrier between what could be modified in a unit without interfering with other units (the inner workings) and what had to be carefully preserved in any modification (its external interface). With the appearance of AOP this barrier has been broken as aspects can act on both sides of this barrier.

The current research state shows two different approaches to tackle this problem:

- Disallow the changing of any encapsulated code, or force programmers to specify which joinpoints are expected to be targets for aspects [14] [15] [16]. This solution will severely diminish the power of AOP, specially the concept of *obliviousness*.
- Detect aspect interferences and allow programmers to correct those interferences or specify them as being desirable.

The problem of aspect interference detection has also been studied under two different lights:

- Automatic detection [7] [10] [11], which has the problem of only detecting some specific types of interferences.
- Detection based on some kind of automatic verification methodology [5] [6] [9], like formal methods or unit testing. This approach is much more broader as all types of interferences can be tackled in this way.

Some interesting work as also been done in the areas of automatic interference solving [12] [13].

It is our belief that automatic detection, using unit testing, and manual interference resolution is the most promising research line at this point. We favor unit testing over formal methods as the latest are generally more complicated to use and are less widespread. However, in order for this approach to become useable, tools and methodologies have to be developed.

8 Acknowledgments

We will like to thank Miguel Pessoa Monteiro for the help in developing this paper and for the constant broadening of our perspectives.

References

1. Ossher, H., Tarr, P.: Multi-dimensional separation of concerns and the Hyperspace approach. In: Software Architectures and Component Technology: The State of the Art in Research and Practice. (2000)
2. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J.: Aspect-oriented programming. In Akşit, M., Matsuoka, S., eds.: 11th Europeen Conf. Object-Oriented Programming. Volume 1241 of LNCS., Springer Verlag (1997) 220–242
3. Filman, R., Friedman, D.: Aspect-oriented programming is quantification and obliviousness (2000)
4. Tessier, F., Badri, M., Badri, L.: A model-based detection of conflicts between crosscutting concerns: Towards a formal approach. In: International Workshop on Aspect-Oriented Software Development. (2004)
5. Katz, S.: Diagnosis of harmful aspects using regression verification. In: Foundations of Aspect-Oriented Languages. (2004)
6. Kienzle, J., Yu, Y., Xiong, J.: On composition and reuse of aspects. In: Software engineering Properties of Languages for Aspect Technologies. (2003)
7. Balzarotti, D., Monga, M.: Using program slicing to analyze aspect-oriented composition (2004)
8. Meyer, B.: Applying "design by contract". IEEE - Computer **25**(10) (1992) 40–51
9. Lagaisse, B., Joosen, W., De Win, B.: Managing semantic interference with aspect integration contracts. In: Software Engineering Properties of Languages and Aspect Technologies. (2004)
10. Störzer, M., Krinke, J.: Interference analysis for AspectJ. In: Foundations of Aspect-Oriented Languages. (2003)
11. Kessler, B., Tanter, É.: Analyzing interactions of structural aspects. ECOOP Workshop on Aspects, Dependencies and Interactions (ADI) (2006)
12. Douence, R., Fradet, P., Südholt, M.: Detection and resolution of aspect interactions. Technical Report RR-4435, INRIA (April 2002)
13. Douence, R., Fradet, P., Südholt, M.: Composition, reuse and interaction analysis of stateful aspects. In: Proceedings of the 3rd International Conference on Aspect-Oriented Software Development (AOSD). (2004) 141–150
14. Braem, M., Gybels, K., Kellens, A., Vanderperren, W.: Inducing evolution-robust pointcuts. ERCIM Evolution Workshop (2006)
15. Griswold, W.G., Shonle, M., Sullivan, K., Song, Y., Tewari, N., Cai, Y., Rajan, H.: Modular Software Design with Crosscutting Interfaces. IEEE - Software **23**(1) (January/February 2006) 51–60
16. Larochelle, D., Scheidt, K., Sullivan, K.: Join point encapsulation. In: Software Engineering Properties of Languages and Aspect Technologies. (2003)
17. Smaragdakis, Y., Batory, D.S.: Implementing layered designs with mixin layers. In: ECCOP '98: Proceedings of the 12th European Conference on Object-Oriented Programming, London, UK, Springer-Verlag (1998) 550–570

18. Akşit, M., Tekinerdoğan, B.: Aspect-oriented programming using composition filters. In Demeyer, S., Bosch, J., eds.: Object-Oriented Technology, ECOOP'98 Workshop Reader, Springer Verlag (July 1998) 435

Aspect-Oriented Programming in PHP

Nuno Beirão

CIULP, Universidade Lusíada do Porto
Rua Dr. Lopo de Carvalho, 4369-006 Porto, Portugal
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
nbeirao@por.ulusiada.pt

Abstract. Aspect oriented programming (AOP) is a new programming methodology, complementary to traditional languages, in which the fundamental principle is the separation of concerns and the encapsulation of cross-cutting functionalities. The PHP programming language is currently one of the major languages for web development, supported by a vast user base, with innumerous quality open source applications and with a strong presence in the enterprise market. The integration of a good implementation of AOP concepts on this language is a leveraging factor for a growing acceptance in a near future. This paper presents an introduction to AOP in the PHP language perspective, discussing various existing AOP implementations and identifying strong and weak points in each approach. Some practical examples of application are presented in order to demonstrate the various implementations. Finally, some conclusions are drawn along with future work possibilities.

Keywords: aspect oriented, AOP, PHP

1 Introduction

Aspect oriented programming (AOP) is a new programming paradigm, created by Xerox PARC [1], that aims to extend and to complement the traditional programming paradigms (object oriented, procedural, functional, etc.) in order to fill some of their limitations, introducing concepts and mechanisms to facilitate the modularization of cross-cutting concerns. However, despite being a generic technique for the separation of concerns [2], and not associated to one language in particular, AOP has been used predominantly with the object oriented languages Java, C++ and C#.

PHP is a programming language especially tailored for the development of web applications, used in the creation of thousand of open-source applications, with a vast community of programmers and with a growing presence in enterprise environments in the most varied activities. The PHP language supports the procedural and object oriented programming paradigms, and features an object model, since version 5, similar to the ones of Java and C#.

Integrating a good implementation of the AOP paradigm in PHP, natively or through extensions to the language, is a determinant factor to allow a continuous expansion in a near future and to make competition possible with languages such as Java and C#.

This paper presents a comparative analysis of the various AOP extensions for the PHP language. The rest of the document is structured as follows. Section 2 gives an overview and brief introduction to AOP, section 3 covers the multiple AOP integration projects for PHP together with some practical application examples, and finally, section 4 presents some final considerations and proposals for future work.

2 Aspect-Oriented Programming

This section gives a brief introduction to the main concepts and terminology of AOP paradigm and languages that implement it.

AOP aims to help in the separation of core concerns from cross-cutting concerns. Core concerns are concerns that capture the central functionalities (business logic) of a software module and shonld be implemented by the underlying language or framework. Cross-cntting concerns are functionalities that cross multiple domains (system logic), are common to various software modules, and should be implemented and encapsulated through aspects.

Recurrent examples of cross-cutting concerns are functionality like logging, security and authentication, debugging, persistence or caching, something that many applications implement and that normally are common to various modules, components or classes.

The separation of concerns allows preventing problems like the presence of code scattering and code tangling. Code scattering occurs when one concern is implemented in multiple modules and is not conveniently encapsulated. Code tangling occurs when a module has several simultaneously concerns implemented. These problems, normally originate difficulties in bug identification, less productivity and code reuse, worse code quality and difficulties of maintenance and evolution.

Languages that implement AOP should adhere to two fundamental principles, quantification and obliviousness [3]. Quantification is the ability to affect many places in the underlying source code with minimum effort. Obliviousness means that the underlying source code does not need to be aware of the aspect actions being implemented.

AOP languages allow the implementation of cross-cutting concerns under the form of aspects. They make it possible to insert code that is triggered when the program flow passes by some well defined points in the code, such as object instantiation, method/function invocation, variable/property access and exception handling.

Considering the terminology created by AspectJ [4], the first and more popular AOP language, these points are called *joint points*. The programmer can give *advice* for what to make, or which code to execute, in the joint points. Advice can normally be applied *before*, *after* or *around* the joint points. A group of related joint points is called a *pointcut*. The aspect, the modular unit in AOP, includes the set of pointcuts, joint points and advices related with the cross-cutting concerns that it implements. The way of specifying the instructions and the type of joint points available varies much depending on the AOP implementation in use.

The recomposition process (Fig. 1) that involves integrating the aspects with the traditional source code originating in one final application is called *weaving*. This

process can be made at various levels and at different periods in time depending on the implementation in use.

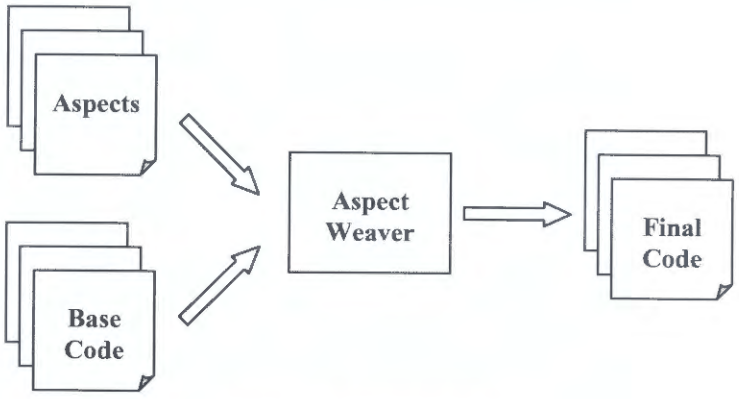

Fig. 1. The weaving process

3 AOP Implementations for PHP

This section presents some available implementations of AOP for the PHP language and some practical examples of its applicability.

We can consider that there are two main methods for the implementation of extensions to support AOP in PHP, the static one and the dynamic one. In the static method, the weaving process is carried through previously to the execution of the application, producing final PHP code that will be executed later. In the dynamic method, weaving is made dynamically, that is, in runtime without previous steps of source code generation to produce the final application.

The methods used to implement each one can vary much, as we will see. To exemplify the integration of aspects, the *Foo* class is used (Listing 1) and a logging aspect will be associated to the method *bar*.

Listing 1. Class Foo

```
class Foo {
   public function bar() {
      echo "Inside bar...";
   }
}
$foo = new Foo;
$foo->bar();
```

3.1 Static Weaving

The AOP extensions using the static weaving process make use of a pre-processor that applies code transformations, integrating the aspects and generating final PHP code for the application. Three implementations exist currently: PHPAspect [5], aoPHP [6] and Transparent PHP AOP [7].

PHPAspect. PHPAspect, inspired by AspectJ and one of the most promising solutions, extends the PHP language adding a set of instructions for the definition of aspects, pointcuts and advices. It uses a compiler (written in PHP) that carries all the source code transformations after being called on the command line. The compiler gets XML representations for the syntax trees (using Lex & Yacc [8]) and applies a set of XSL transformations originating in the final code.

PHPAspect supports a varied set of joint points (method execution/call, attribute writing/reading, object construction/destruction and exception throwing), the before, after and around advices, and pointcut composition with logical operators.

In Listing 2 we can observe an example implementing a logging aspect applied to method bar from class Foo defined previously.

Listing 2. PHPAspect

```
aspect Logging {
   pointcut logFooBar:exec(public Foo::bar(0));
   after logFooBar {
      log("End of method bar.");
   }
}
```

aoPHP - Aspect-Oriented PHP. The aoPHP project uses a pre-processor written in Java that modifies PHP code applying the aspects defined in files with *.aophp* extension. For the pre-processor activation, it is necessary to substitute the traditional *<?php* (beginning of PHP code) for *<?aophp filename="aspect.aophp"* in files where aspect integration is intended.

In terms of joint points, aoPHP only supports functions, allowing before, after and around advices, and point cut composition with logical operators. A C version of the pre-processor is being developed in order to increase the performance and to eliminate the Java dependencies.

Listing 3 presents an implementation of a logging aspect applied to the function *foobar*.

Listing 3. aoPHP

```
<?aophp filename="logging.aophp"
function foobar {
   echo "Inside bar...";
}
?>

// logging.aophp
after(): exec(bar()) {
   log("End of function bar.");
}
```

Transparent PHP AOP. Transparent PHP AOP is a recent implementation where the weaving process is simplified and significantly different from the previous ones. The aspect definition is made with XML files, XAD (XML Aspect Definition) as the author calls them, which are integrated in the base code on first use. For this integration to occur, it is necessary to substitute the instructions used for file inclusion (require, require_once, include and include_once) by calls to the function *require_aop* (see Listing 4) indicating which aspect or set of aspects to weave. The referenced files are then compiled and stored in a cache folder for later use. The files are compiled whenever some change to the base code or aspect is made.

Listing 4. Transparent PHP AOP – Aspect weaving

```
require_once "../func.require_aop.php";

require_aop("class.Foo.php", "logging.xml");
$foo = new Foo;
$foo->bar();
```

This extension only supports method or function invocations as joint points and allows the before and after advices to be applied.

In Listing 5 we can see an example aspect to implement the functionality of logging in method *bar* from class *Foo*.

Listing 5. Transparent PHP AOP – Aspect definition

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE aspect SYSTEM "../aop.dtd">

<aspect>
   <pointcut auto="after" class="Foo" function="bar">
   <![CDATA[
      log("End of method bar.");
   ]]>
   </pointcut>
</aspect>
```

3.2 Dynamic Weaving

The extensions that implement this method use aspect weavers that work in application runtime, taking advantage of the interpreted nature of the PHP language, and weaving the aspects on demand. Four projects with totally distinct approaches are known: aspectPHP [9], AOP Library for PHP [10], MFAOP [11] and GAP (Generic Aspects for PHP) [12].

aspectPHP. This extension is a reimplementation of aoPHP in C language and a patch to PHP 4.3.10, allowing it to directly support AOP constructs natively with its interpreter.

This project supports the same functionalities of project aoPHP and suffers from the same limitations. The biggest disadvantage is to work only with an obsolete version of PHP (4.3.10).

AOP Library for PHP. This library, developed by Dmitry Sheiko [13], implements a very basic form of the AOP paradigm. Its use requires major changes to the base source code of the application, ignoring two fundamental principles of the AOP paradigm, quantification and obliviousness.

The aspects are defined by instantiating objects of class *Aspect* and the definition of pointcuts through calls to the method *pointcut*. The only supported joint point type is the method invocation, to which can be applied the before and after advices.

For the execution of the specified advices it is necessary to modify all class definitions and to place *Advice::_before()* in the beginning of each method and *Advice::_after()* before the return as illustrated in Listing 6.

Listing 6. AOP Library for PHP

```
class Foo {
   public function bar() {
      Advice::_before();
      echo "Inside bar...";
      Advice::_after();
   }
}
...
$logAspect = new Aspect();
$pcFoobar = $logAspect->pointcut("call Foo::bar");
$pcFoobar->_after('log("End of method bar.")');
Aspect::apply($logAspect);
```

MFAOP. The MFAOP extension is similar to the previously described project, with the advantage of not having to modify the methods of the original class definitions to allow the association of advice code. To make this possible, the authors used the PHP *Classkit* [14] extension allowing changes to class definitions to be made at runtime, for the creation of new methods or renaming existing ones. This implementation supports the before, after and around advices, applied to method invocations.

Listing 7 illustrates the creation of a new aspect with MFAOP.

Listing 7. MFAOP

```
$pcFoobar = new PointCut();
$pcFoobar->addJoinPoint('Foo', 'bar');
$logAspect = new Aspect($pcFoobar, after, 'log("End of
method bar.");');
```

GAP - Generic Aspects will be PHP. This is a project under development by Sebastian Bergmann, the author of PHPUnit [15] (software testing framework for PHP based on JUnit of Java), for the creation of an AOP extension that takes full advantage of the new functionalities made available by PHP 5, such as the *Reflection API* or the overloading methods *__call()*, *__get()* and *__set()*. This project, which is not available for public consumption yet, uses the PHP *Runkit* [16] extension, a new and powerful reimplementation of the Classkit extension mentioned previously.

In this implementation, the definition of aspects is obtained through the creation of classes whose methods correspond to the kind of advice to apply (before, after or

around). The creation of pointcuts and association of advice to the joint points is carried through with the aid of annotations that precede the class definition.

Listing 8 shows the definition of a logging aspect that will be applied to method *bar* from class *Foo*.

Listing 8. GAP – Generic Aspects for PHP

```
/* @pointcut callPointCut : method(Foo->bar(..));
 * @after callPointCut : LoggingAspect->after();
 */
class LoggingAspect {
   public function after($joinPoint) {
      log("End of method bar.");
   }
}
```

4 Conclusion and Future Work

This article presented various solutions and approaches to support the AOP paradigm in the PHP programming language. The implementations were classified as static or dynamic accordingly to the method used for the weaving process. In order to demonstrate each implementation, a practical example of aspect weaving was used, applying a logging aspect to a method of an existing class.

The dynamic weaving approaches present the advantage of not needing a previous process of pre-processing or compilation in order to weave or integrate the aspects with the base code, augmented by the fact of being able to take advantage of the interpretive nature of the PHP language. These advantages alone are enough for one to realize that the dynamic weaving process is much more useful and powerful than the static one. Besides reducing development time by not having to deal with the pre-processing stage, allows, for example, applying aspects selectively in runtime depending on the result of some condition or event, like user input.

Compared to the static weaving process, the dynamic also has its share of limitations mainly due to the lack of native AOP support by the PHP interpreter or PHP extensions. The dynamic AOP solutions are generally slower than their counterparts because of the overhead in managing the aspect weaving in runtime, and also, do not implement new language constructs (like PHPAspect does) to support the paradigm.

From all the covered proposals, the most promising seems to be GAP, but unfortunately, at the time of writing, there is no public release available to test.

The ideal solution would be for the PHP developers at Zend to add native AOP support on the next major version of PHP (version 6). Meanwhile, while we wait, the

emergence of a native AOP extension (or PECL module) could leverage the use of this programming methodology in the PHP community.

References

1. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Videira Lopes, C., Loingtier, J.-M., and Irwin, J., Aspect-Oriented Programming, In Proceedings of ECOOP 1997, (Finland, June 1997). 25pp.
2. Hursch W.L., and Lopes C.V., Separation of concerns, Technical Report NU-CCS-95-03, College of Computer Science, Northeastern University, Boston, MA, February 1995.
3. Filman R.E., Friedman D.P., Aspect-Oriented Programming is Quantification and Obliviousness. In Workshop on Advanced Separation of Concerns, OOPSLA 2000, (Minneapolis, October 2000).
4. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and Griswold, W.G., An overview of AspectJ. In Proceedings of ECOOP 2001, pages 327–353, 2001.
5. Candillon, W., Vanwormhoudt, G., PHPAspect website. Visited on December 21st 2006 at http://www.phpaspect.org.
6. Stamey J.W., Saunders B., Aspect-Oriented PHP website. Visited on December 6th 2006 at: http://www.aophp.net.
7. Blanco, G., Transparent PHP AOP website. Visited on December 18th 2006 at http://www.phpclasses.org/aop.
8. Levine, J.R., Mason, T. and Brown, D., Lex & Yacc (2nd Ed.). O'Reilly & Associates, Inc., Sebastopol, CA, 1992.
9. Yu, Y., aspectPHP website. Visited on December 29th 2006 at http://www.cs.toronto.edu/~yijun/aspectPHP/.
10. Sheiko, D., AOP Library for PHP website. Visited on December 28th 2006 at http://www.phpclasses.org/aopinphp.
11. MFAOP website. Visited on December 29th 2006 at http://www.mfaop.com/.
12. Bergmann, S., Kniesel, G., GAP: Generic Aspects for PHP, In Proceedings of EWAS 2006, (Netherlands, August 2006).
13. Sheiko, D., Aspect Oriented Software Development and PHP, In php | architect, Volume 5 Issue 4, (April 2005), pages 17-25.
14. Golemon, S., Classkit extension for PHP website, Visited on December 29th 2006 at http://pecl.php.net/package/classkit.
15. Bergmann, S., PHPUnit website, Visited on December 29th 2006 at http://www.phpunit.de.
16. Golemon, S., Runkit extension for PHP website, Visited on December 29th 2006 at http://pecl.php.net/package/runkit.

Sessão 5

Engenharia de Software e Sistemas de Informação

Deriving User Interfaces from Contracts

António Miguel Rosado da Cruz

Department of Electrical and Computer Engineering
Faculty of Engineering, University of Porto
Porto, Portugal
antonio.cruz@fe.up.pt

Abstract. User Interface model-based approaches usually make use of several component models of their own, like task model, abstract and concrete presentation models or application model, which are difficult to integrate in order to have a view of the whole that enable automatic generation of the UI. This paper presents a model-driven approach for deriving user interface abstract presentation models from the application's business logic model, which is represented in the form of UML class diagrams complemented with contracts in VDM++. A set of rules for mapping the business logic model onto a form-based user interface is sketched. An approach for leveraging the usability and user-friendliness of the UI generated is also presented, based on the generation of validation procedures from the methods' pre-conditions and the classes' invariants. The work reported is on-going work being developed in the author's doctorate investigation project.

1 Introduction

Software development processes have been using, for decades, software models that help software engineers cope with complexity, when analyzing the problem domain or designing a software solution. Software models capture relevant parts of the problem and solution domains and are typically used as a means of communicating with the stakeholders (e.g.: clients, users, team members). Several software modeling methodologies have been developed. A review of the most popnlar methods and modeling techniques from the last 20 years can be found in [1].

Most of the methodologies use disparate views, or components models, to capture different aspects of the domain (task model, abstract and concrete presentation models or application model) [2]. It is, often, difficult to understand the whole because the views of the parts do not integrate well, i.e. it is typically hard to have an integrated view of every component models. This situation compromises the possibility of having code generated from the software models.

Model-driven development approaches, like OMG's Model Driven Architecture[1] (MDA), refocus the aim of modeliug from human understandable to system recognizable [3]. Model driven development processes, typically based ou formal

[1] "Model Driven Architecture" is a trademark of OMG.

or semi-formal (with a core suitable to unambiguous interpretation) specification methods, have already achieved some capabilities for automatic generation of code for some of the code layers (e.g., data layer, business logic layer, user interface layer). Tipically, the code for the data layer (e.g., database SQL script) can be fully generated from the model [4], and all or part of the code for the business logic layer can also be generated for a given platform. But the user interface layer has been forgotten in the efforts to achieve automatic code generation from integrated software models. User interface model-based technologies often produce UI models that do not integrate well with the core of the application. The user interface layer has to cope with complexities both from the application core and the user.

The main contribution of this paper is to provide a model-driven approach to the generation of default user interfaces, starting from the system's business logic model represented as UML class diagram along with class contracts in VDM++. The class of system models considered is restricted to data oriented (business) applications.

The next section presents some concepts of modeling systems through contracts and model-driven development. Section 3 addresses the specification of contracts in VDM++ notation. Section 4 addresses the problem of derivating form-based user interfaces from design componeuts' contracts. Section 5 concludes the paper and sketches some directions for future research.

2 System modeling through contracts

Design by Contract[2] (DbC), aims at developing software that is trustable, in which people can rely on. In DbC, a contract is a means by which two elements agree in an explicit roster of mutual obligation and benefits[5,6,7,8]. In an object-oriented setting, components and classes act as providers and clients of services. Through a contract, two components or classes *agree* in a protocol for communicating, that is, which methods are exported (publicly visible) by the intervaining classes, and also in the conditions that have to be met in order for the communication to happen.

A contract not only specifies the protocol for instances of classes or components' communication, but also specifies, as will be seen in section 3, what conditions must the caller (client) assure in order to be allowable to call a given method and what conditions must the callee (provider) guarantee that bold after the method is returned.

A contract may also specify conditions that must allways hold, as invariant conditions that are imposed to the state of a class' instances.

DbC can be viewed as a means to develop a system's design model and, as it is abstractly but formaly defined, the room for ambiguity is strongly reduced, meaning that it may be machine understandable and that it may be used for model transformation activities. This way, DbC can be an approach to model-driven development.

[2] "Design by Contract" is a trademark of Eiffel Software.

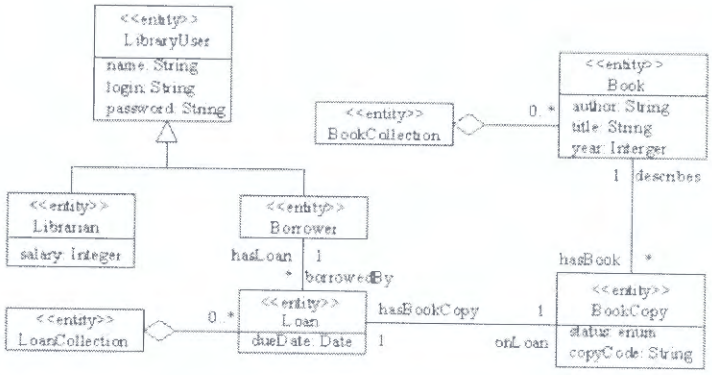

Fig. 1. Domain Model of the Library System (taken from [2])

3 Contract Specification in Vdm++

In order to present the VDM notion of contract, the UML class diagram of a library system case study is borrowed from [2]. A UML domain model of the Library System is presented in figure 1. The domain classes' diagram is composed of ≪ *entity* ≫ classes that model objects identified in the design process. In the UML diagram presented, the methods are not specified. They will be specified later in the VDM notation.

LibraryUser, *Librarian*, *Borrower*, *Book*, *BookCollection*, *Loan*, *LoanCollection* and *BookCopy* are the ≪ *entity* ≫ classes of the Library System. The contract associated to each class is not clear in the UML class diagram. OMG provides a language for the specification of such contracts, though, namely Object Constraint Language (OCL). Nevertheless, due to the familiarity of the author with VDM/VDM++, in this paper, will be used the VDM++ notation.

VDM++ is an object oriented extention to the Vienna Development Method (VDM) notation. VDM is a formal method's language for the specification of programs, developed in 1973 in the Vienna laboratory of IBM, which has been stated as an ISO standard (ISO/IEC 13817-1) in 1996 as VDM-SL[9]. VDM-SL enables the development of algebraic models, either purely functional or composed of modules with inner state. VDM++ adds the notion of class and all its assotiated concepts (inheritance, encapsulation, information hiding, etc.) to the abstraction facilities provided by VDM-SL.

By using the VDMTOOLS©[10,11], it is possible to create a complete document that incorporates both textnal non-formal english descriptions of the system and the VDM-SL or VDM++ model, through literate programming.

In VDM++ , a class may have an inner state, purelly functional methods, called functions, or non-purelly functional methods, called operations. In purelly functional methods, the return value depends only on the parameter values and

has no side effects, while in non-purelly functional methods, if they return a value, it may depend both from the parameters and the inner state of the class' instance, and it may, as a side effect, update the inner state.

Both in VDM-SL and in VDM++ , specifications can be made at an abstraction level where the semantic of the operators is fully specified or, at a higher abstraction level where the semantic of the operators is specified through pre- and post-conditions. Invariants can be specified over abstract data types or over modules' inner variables or classes' instance variables, depending on weather VDM-SL or VDM++ is being used. In the second case, specifications can be called contracts.

Next is the VDM++ contract for class *Book*:

```
class Book
instance variables
    private author : char*;
    private title : char*;
    private year : ℤ;
    private describes : BookCopy-set;
    inv (self.year > 1900) ∧ (self.year < 2050)

operations
public
    get-author () result : char*
    pre true
post
result = self.author;
public
    get-title () result : char*
    pre ...
...
public
    get-year () result : ℤ
...
public
    set-author (x : char*)
    pre true
post
self.author = x;
...
end Book
```

And, the VDM++ contract for class *BookCollection*:

```
class BookCollection
instance variables
    private bookCollection : Book-set;
    inv ∀ b1, b2 ∈ self.bookCollection ·
            b1.get-author () = b2.get-author () ∧
            b1.get-title () = b2.get-title () ∧
            b1.get-year () = b2.get-year ()

operations
public
    addBook (b : Book)
    pre ¬∃ x ∈ self.bookCollection ·
            x.get-author () = b.get-author () ∧
            x.get-title () = b.get-title () ∧
            x.get-year () = b.get-year ()
post
∃ x ∈ self.bookCollection ·
            x.get-author () = b.get-author () ∧
            x.get-title () = b.get-title () ∧
            x.get-year () = b.get-year ()
end BookCollection
```

The contracts specify what invariant conditions must allways hold in each of the classes, what (pre-) conditions must hold for each one of the methods and what (post-) conditions must be verified after each method returns. For example, in class *Book* an invariant is defined that states that the value for *year* must be an integer greater than 1900 and less that 2050. Pre-condition of method *addBook* in class *BookCollection* states that the book being added to the book collection must not previously exist in the book collection.

4 UI derivation approach

For being able to systematize, and hopefuly automate, the generation of a user interface for a given system model, a set of mapping rules has been established for deriving a default user interface form from each of the classes, and a user interface navigation map from the assotiations between the classes. Figures 2 and 3 show the mapping rules for a simple class, like *BookCopy*, and for a class with an aggregated class, like *BookCollection*.

Figure 4 illustrates, for the three classes specified earlier for the library system case study, the rules for mapping the UML class model to forms-based UI.

Each class gives origin to a form with a data entry field (e.g.: text box, or any other field for entering text), for each class' public attribute and/or each get-/set- methods pair. A get- method with no corresponding set- method originates a read-only output field (e.g.: non-editable text box). Each class also originates

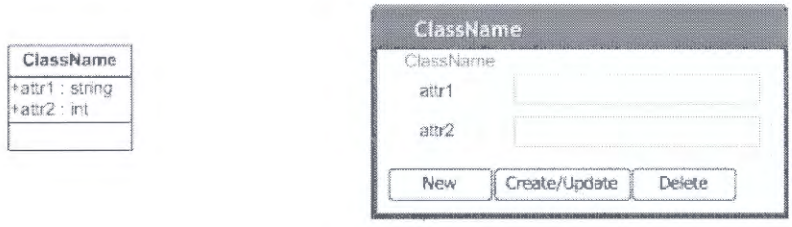

Fig. 2. UI mapping rule for a simple class.

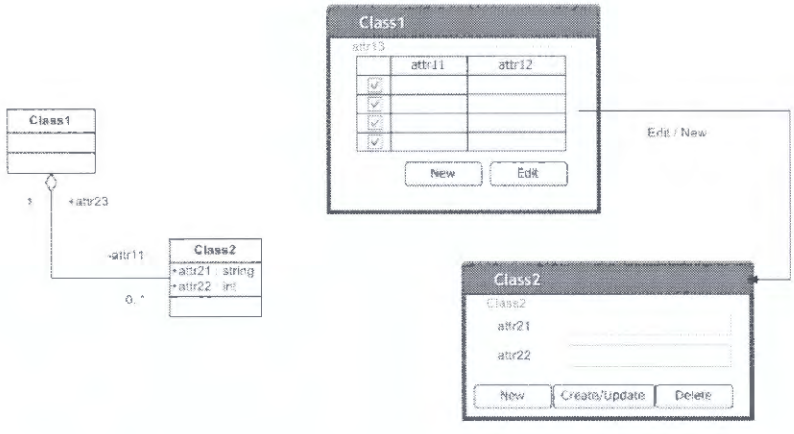

Fig. 3. UI mapping rule for a class with an aggregation.

a set of action fields (e.g.: buttons), namely *New*, *Create/Update* and *Delete*, except those classes that only have one field that aggregates instances from another class. An aggregation of classes like the one between classes *BookCollection* and *Book* (*BookCollection* has an attribute which type is *setofBook*) originates a collection (multiple items) output field (e.g.: data grid) with an *Edit* action field (e.g.: button) for each object in that list.

In order to better understand the UI generated by the mapping process, figure 4 shows a very concrete UI model. The widgets presented in this concretization of the UI are only an example. The process shall generate not a concrete UI but rather an abstract one using, for example, a notation for abstract prototypes like the one presented in [12].

Each class invariant may also originate a validation rule that shall run after each operation has finished. From each method pre-condition a validation rule may also be derived, that shall run before effectively running the operation's

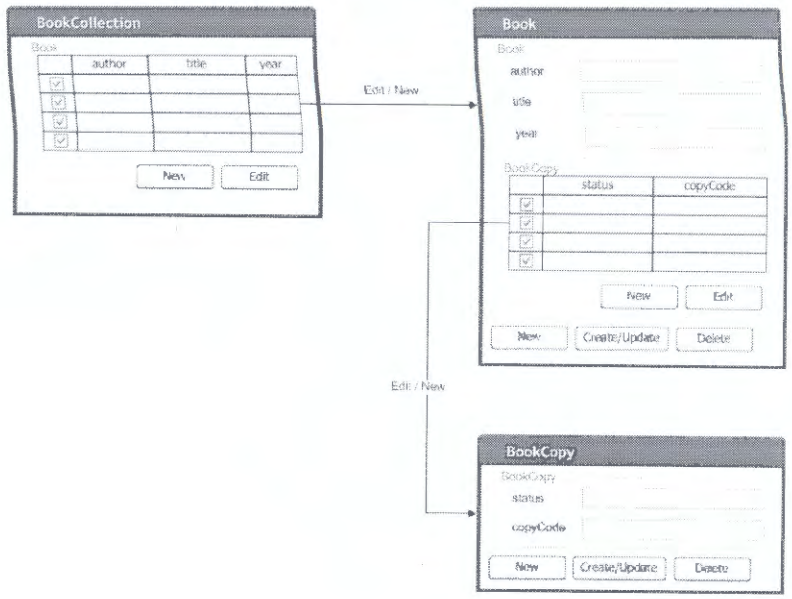

Fig. 4. User Interface abstract model for the classes *BookCollection*, *Book* and *BookCopy*.

code. Indeed, each method's code shall only run if the validation rule succeeds, i.e. if the operation's pre-condition holds. Validation rules that only relate fields from the UI form, like the one derived from the *Book*'s invariant, may run at the UI layer. Rules that relate fields from the form with data from other instances, of the same or any other class, shall be coded as services from the business logic layer, and invoked from the UI layer. Examples of this last scenary are the validation rules that are derived from the *BookCollection*'s invariant and from the *BookCollection*'s *addBook*() method.

5 Conclusions and Directions for Future Research

Developing software systems through the specification of contracts may be understood as a model-driven development technique, in which the system is modelled using object oriented methods complemented with a *Contract* for each class. The contract involves the specification of logic conditions that must always hold during the lifetime of the class' instances and also of conditions that must be verified in order to be "legal" to invoke a given method and conditions that the method must guarantee that hold after returning.

Invariants that guarantee the integrity of the system or compouent are partially implicit in the UML class diagram. For example, an instance of a *BookCopy*

referenced by a *Loan*, through attribute *hasBookCopy* must be a member of set attribute *describes* of an instance of class *Book*. This is no more than a foreign key constraint, implicit in the class diagram, but, in VDM++ , if one wants to specify a clear and explicit contract that crosscuts several classes, one has to create a structure (class) that aggregates everything else, and define the crosscuting invariant in that top class. In [13,14] a study has been made of aspect contracts for contractualize crosscutting concerns, such as invariants that span several classes.

From the classes' contracts, a set of mapping rules has been specified, in order to be able to systematize the process of generating form-based user interfaces. From the contracts' invariants and pre-conditions a set of validation rules may be derived for increasing the usability of the generated nser interface. Further study is needed to formally specify a mapping table from invariants and pre-conditions to validation rules at code level.

Also, the specification of the generated UI through Abstract Prototypes may not be sufficient for enabling an automation of the generation process. The abstract prototype, is only another model in the model transformation chain created by the model-driven development approach. A default and customizable concrete UI should also be generated, either driven by a usability knowledge based repository, by a given user model or historic nsage model or by any other means.

Several works [15,16] present UI generation approaches for executable XML-based user interfaces, but their start point is a declarative model of the user interface, not a model of the system's business logic. Declarative User Interface Models (UIMs) provide an abstract description of the UI, that can be reused and can be refined to more concrete models. UIMs can be fonnd at different levels of abstraction during the UI design process. UIM provide an infrastrncture for allowing automated tasks in the UI design and implementation processes. In [17] a comparative stndy between 14 model-based user interface development environments is presented.

As referenced earlier, the approach presented herein starts from the (business logic) application model and obtains a default user interface abstract presentation model. Also a default task model is implicit in the derived navigation map. Further stndy is needed in order to test and refine the mapping rules between the application model (class contracts) and the UI model generated. Also, a default concrete presentation model along with the data validation rules, generated from the classes' invariants and the methods' pre-conditions, is something that the on-going work will try to achieve.

The generated UI has several applications. One is the obvious generation of much of the work that is necessary to build a UI layer, that may be *a posteriori* customized. Other application is the easy UI prototyping of the contracts that may be used to validate and refine the contracts themselves, by the client or end user, before advancing in the model transformation process towards a complete and executable application.

Being based in the same complete application model that is used for deriving the data layer and the business logic layer, the presented approach for deriving a user interface layer, promotes a complete and integrated model of the system being developed.

References

1. Wieringa, R.: A survey of structured and object-oriented software specification methods and techniques. ACM Comput. Surv. **30**(4) (1998) 459–527
2. Pinheiro da Silva, P.: Object Modelling of Interactive Systems: The UML$_i$ Approach. PhD thesis, Faculty of Science and Engineering, University of Manchester (2002)
3. MODELWARE: Learning MDD (set of videos and other training material). http://www.modelware-ist.org [visited on the 27th December 2006].
4. Chaves, T.M.L.: VooDooM: Suport for understanding and re-engeneering of VDM-SL specifications. Master's thesis, University of Minho (2006)
5. Schoeller, B., Widmer, T., Meyer, B.: Making specifications complete through models, Springer-Verlag Lecture Notes in Computer Science (2006) to appear in "Architecting Systems with Trustworthy Components".
6. Meyer, B.: Dependable Software. Lecture Notes in Computer Science (2006) to appear in Dependable Systems: Software, Computing, Networks. eds. Jürg Kohlas, Bertrand Meyer, André Schiper. Springer-Verlag.
7. Artima: Contract-driven Development - A conversation with Bertrand Meyer, Part III - by Bill Venners (2004) www.artima.com/intv/contestP.html.
8. Artima: Design by Contract - A conversation with Bertrand Meyer, Part II - by Bill Venners (December 2004) www.artima.com/intv/contractsP.html.
9. Fitzgerald, J., Larsen, P.: Modelling Systems. Practical Tools and Techniques in Software Development. Cambridge University Press (1998)
10. IFAD: The IFAD VDM-SL Language. (2000) Revised for V3.6.
11. IFAD: The IFAD VDM++ Language. (1999) Revised for V6.3.0a.
12. Constantine, L., Windl, H., Noble, J., Lockwood, L.: From abstraction to realization in user interface designs: Abstract prototypes based on canonical abstract components. (2000) Working Paper. Last version can be found at http://www.forUse.com/articles/.
13. Balzer, S., Eugster, P., Meyer, B.: Can aspects implement contracts? In: Proceedings of RISE 2005 (Rapid Implementation of Software Engineering techniques), Beraklion, Greece (September 2005) to appear as Springer Lecture Notes in Computer Science.
14. Zhang, J., Gray, J., Lin, Y.: A Model-driven approach to enforce crosscuting assertion checking. Workshop on Modeling and Analysis of Concerns in Software (MACS 2005) (2005) Copyright 2005 ACM.
15. Kuo, Y., Shih, N., Tseng, L., Hu, H.C.: Forms-XML: generating form-based user interfaces for XML vocabularies. Advances in Web-Age Information Management. 6th International Conference, WAIM 2005. Proceedings (Lecture Notes in Computer Science Vol.3739) (2005) 925 –
16. Forbrig, P., Dittmar, A., Reichart, D., Sinnig, D.: From models to interactive systems tool support and XIML. In: MBUI. (2004)
17. Pinheiro da Silva, P.: User interface declarative models and development environments: A survey. LNCS **1946** (2000) 207–226

Grapheme-to-phoneme conversion using recurrent neural networks

Nuno Fonseca

Escola Superior de Tecnologia e Gestão de Leiria (IPL), Morro do Lena, Alto do Vieiro, 2411-901 Leiria, Portugal
nfonseca@estg.ipleiria.pt

Abstract. Although there are several methods to implement grapheme-to-phoneme conversion, most of them rely on previous alignment with some sort of human interaction. This paper presents a method based on a recurrent neural network that by the use of feedback (using previous outputs as inputs) can dismiss the previous alignment task, handling all the learning by itself, and achieving a phoneme success rate of more than 78% and a word success rate near 50%.

Keywords: Grapheme-to-phoneme recurrent neural networks text-to-speech

1 Introduction

Speech synthesis is a major tool to improve the interface between machines and men. This type of systems, usually called text-to-speech (TTS), are constructed internally with 2 sub-systems: grapheme-to-phoneme conversion and voice synthesis. The first one is responsible to convert the written words to their correspondent phonetics, and the second one is responsible to convert the phonetics to an audio stream.

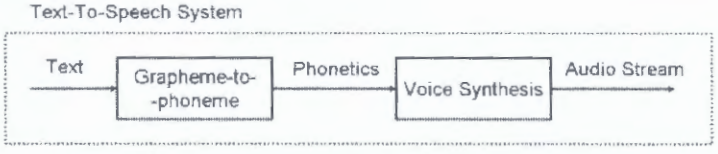

Fig. 1. A Text-To-Speech System

There are several ways to accomplish the grapheme-to-phoneme conversion. One way is to construct a big dictionary to store the maximum possible amount of words and the phonetics of each one, which may correspond to tens or hundreds of thousands of words. Another way to do the same is to create some method that do the conversion based on predefined rules or based on previous knowledge about how

words are read. Pronunciation dictionaries have the advantage of giving almost the perfect phonetics for each word (errors may appear when the same word could be read in different ways, depending on the context), but fail completely when they need to handle an unknown word. Besides that, they consume a big amount of memory that could be prohibitive on some devices (handhelds, etc). The rules/knowledge-based methods can handle any kind of word (even unknown ones), because they will apply the same knowledge they got from their previous learning. In this paper we will work on these second types of systems.

One of the major difficulties with grapheme-to-phoneme conversion is that the grapheme (written word) and its phonetics (sequence of phonemes) may have different lengths. Although most letters correspond to one phoneme, there are others that may correspond to several phonemes, or even correspond to zero phonemes.

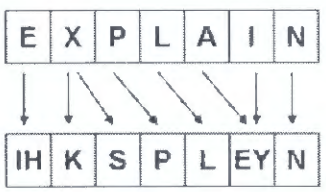

Fig. 2. Words like "Explain" can include letters that correspond to 1 phoneme, letters that correspond to 2 phonemes and letters that correspond to 0 phonemes. Although there are the same number of letters and phonemes, they are not aligned.

If the alignment were of 1 to 1 (one letter to one phoneme), the grapheme-to-phoneme conversion would be more or less trivial, by simply applying some sort of machine learning algorithms (decision trees, neural networks, etc). But this is not the case. As such, there are two possible approaches: using a previous alignment algorithm or use a method that could handle both (alignment and conversion). In the first case, most methods usually consider some kind of human intervention: full human work (all alignment made previously by someone [1]) or partial human intervention (someone enters some kind of alignment examples and the computer figure-it-out how to align the rest [2]).

At the moment, several research work have been done using different types of machine learning method: pre-alignment (human seed) with decision trees[2], Feedforward neural networks with [1] or without [3] previous (human) alignment, Hidden Markov Models [4] (without pre-alignment).

This paper presents a recurrent neural network that handle grapheme-to-phoneme conversion without any type of previous alignment processing. Section 2 presents the model and the major decisions, section 3 present the tests and results, section 4 discuss the obtained results and section 5 presents the conclusions and future work.

2. Approach

Since most Text-to-Speech systems run in real time, the grapheme-to-phoneme conversion must be done in a very small amount of time, which discards some machine learning approaches (genetic algorithms, etc). Neural Networks (NN)[5] are one of the most used machine learning methods, and although they need a large amount of time for learning proposes, they are able to give "answers" in a very quick time. NN results are mainly dependent of its architecture, the right choice of inputs and learning process.

In most languages (including English) the relation between letters and phonemes are not direct (for instance, we can not simply create a table with letters and the phonemes to be used with each one). The right phoneme depends on the letter, but also on the letter's neighborhood. Most grapheme-to-phoneme systems consider that a neighborhood of 3 letters before and 3 letters after the current letter is sufficient to guess the right phoneme [1][2].

For representation proposes lets consider L(t) as the current letter, L(t+n) as the n^{th} letter after the current letter, L(t-n) as the n^{th} letter before the current letter.

As such, we could use these 7 letters (current letter and neighborhood of ±3 letters) as inputs of our neural network, but only, if we already had the alignment problem solved (as in [1]). To create a neural network capable of handle misalignment, a decision as made to use recurrent neural networks (RNN)[6], e.g., neural networks that use feedback, linking previous outputs as inputs. This additional input will help the NN to handle misaligned letters/phonemes.

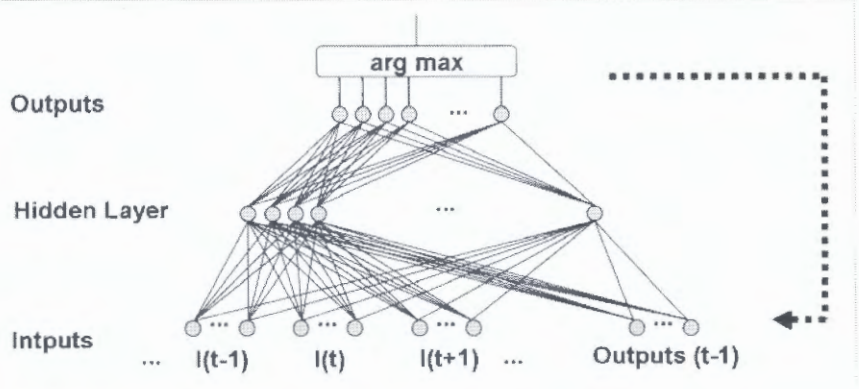

Fig. 3. Recurrent Neural Network architecture used to do the grapheme-to-phoneme conversion.

Since there are 26 different letters in the English language, each input letter is represented by 26 different inputs (only one of these 26 inputs is activated at each time). In the output layer, there are 40 outputs, each one representing a different phoneme (39 phonemes) and one representing the end of phoneme sequence. The output with the higher value is chosen as the right phoneme. The number of hidden

units is equal to the number of inputs and sigmoid is used as the activation function. Both inputs and target outputs use values of 0.1 (0) or 0.9 (1).

Although there is some sort of feedback, this feedback is made of outputs, which have perfectly defined targets to compare with, which mean that a standard back-propagation algorithm is sufficient. All learning is done using online weight changes (weights changing in each interaction).

Figure 3 shows the architecture of the NN, where it is clear that end up to be a mixture of a Jordan Neural Network (outputs connected as inputs in delay) and a Time Delay Neural Network (inputs delayed in time) with look ahead feature (t+n).

This neural network is capable of presenting only a phoneme at a time. To get the complete phoneme sequence, the neural network must be run in sequence. Table 1 shows an example of the sequence used to learn/test the word "explain".

Table 1. The representation over the time of the sequence of learning/testing for the word "explain".

Inputs								Target
L(t-3)	L(t-2)	L(t-1)	**L(t)**	L(t+1)	L(t+2)	L(t+3)	Out(t-1)	**Output**
-	-	-	**E**	X	P	L	-	**IH**
-	-	E	**X**	P	L	A	IH	**K**
-	E	X	**P**	L	A	I	K	**S**
E	X	P	**L**	A	I	N	S	**P**
X	P	L	**A**	I	N	-	P	**L**
P	L	A	**I**	N	-	-	L	**EY**
L	A	I	**N**	-	-	-	EY	**N**
A	I	N	-	-	-	-	N	-

3. Tests

The presented RNN was implemented in C++ and both learning and tests were made using the pronouncing dictionary from CMU [7] – 70% for learning and the rest 30% for testing proposes. This dictionary, that doesn't have any kind of alignment information, has 125.000 words in English and their phonemes, and had been used as a reference in many papers in grapheme-to-phoneme area.

Nevertheless, this dictionary presents two major issues. First of all, it includes several acronyms, "words" that don't contribute for a good learning process (for instance "AAA" that is read as "triple A"). Secondly, the dictionary also includes for some words, several available phonetics, since the same word can be read in different ways, which raise some questions about how to handle them in the test set.

To resolve the first issue (acronyms), and since it was not possible to verify all words on the dictionary, the decision as made to verify only the words that presented a big discrepancy (>=2, in this case) between the number of letters and the number of phonemes (words like "AAA"/"triple A"). All acronyms found in this verification process were simply removed from the dictionary.

Regarding words with several different phonemes, it was chosen to remove them from test-sets, although they continue to be used in the learning process. Since most words have several letters, the learning set (70% of CMU dictionary) have almost 900.000 different phonemes situations, which in a NN of this size (60.000 different weights) may turn out to be a very slow process. Table 2 present tests made with several learning set sizes.

Table 3 shows the results of the tests made with the recurrent neural network with and without feedback, using the full learning set (70% of CMU dictionary) for learning and tested with full test set. Results are presented both with percentage of correct phonemes, and percentage of correct words (considering a correct word as a word made entirely of corrected phonemes).

Table 2. Different learn set sizes. Regardless of the learning set size, tests were made with 30% of [7].

Learn Set Size	Correct phonemes	Correct words
0.7% of [7]	58.724%	17.744%
7% of [7]	71.105%	35.350%
70% of [7]	78.877%	48.213%

Table 3. Tests made with and without feedback

Full Learn Set (70%)	Correct phonemes	Correct words
With feedback	78.877%	48.213%
Without feedback	69.622%	26.133%

4. Discussion

It is obvious that the percentage of correct words is always smaller than the percentage of corrected phonemes.

Also, as expected, the use of feedback increases the number of correct phonemes.

Regarding the global results, and doing a comparison with other grapheme-to-phoneme methods [1][2][3][4], it can be shown that this method present worst results (most methods guess between 85% to 90% of phonemes). Nevertheless, most of these grapheme-to-phoneme methods use some kind of human interaction to align letters and phonemes.

5. Conclusions and future work

This approach presents several positive characteristics: all learning process can be done without any type of human knowledge or intervention; it requires small amount of memory; and it provides response in a quick time.

Although many techniques present better results than the ones in this paper, that doesn't mean that this type of architecture should be abandoned.

There is a large amount of work to be done to improve the results, such as explore different recurrent neural networks architectures (Elman, Jordan, fully/partial recurrence, etc) and different back-propagation methods (BPTT, etc.); apply different learning methods (batch/stochastic). Also, since the learning process is very time demanding, longer learning time tests need to be made.

6. References

1. Sejnowski, T., Rosenberg, C.: Parallel Networks that Learn to Pronounce English Text, Complex Systems 1, (1987) 145-168
2. Pagel, V., Lenzo, K., Black, A.: Letter to sound rules for accented lexicon compression, In 1998 Internation Conference on Spoken Language Processing, Sydney (1998).
3. Bullinaria, J.A.: Connectionist Modelling of Spelling, In: Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society (1994) 78-83
4. Taylor, P.: Hidden Markov Models for Grapheme to Phoneme Conversion, In INTER-. SPEECH 2005, Lisbon, Portugal (2005) 1973-1976
5. Michell, T.M.: Machine Learning, McGraw-Hill International Editions (1997) 81-126
6. Haykin, S.:Neural Networks – a comprehensive foundation, Prentice Hall, second edition (1999) 732-789
7. CMU Pronouncing Dictionary, version 0.6, "http://www.speech.cs.cmu.edu/cgi-bin/cmudict" (November 2006).

Understanding a Framework through Design Patterns Recovery

Nuno Flores[1]

[1] FEUP, Universidade do Porto - Rua Dr.Roberto Frias s/n - 4200-465 Porto, Portugal
nuno.flores@fe.up.pt

Abstract. Object-oriented frameworks are a powerful technique for large-scale reuse but they often require a lot of time to learn how to effectively reuse them, especially when the available documentation is poorly written, lacks detail and is clearly outdated. Frameworks usually encompass several design pattern instances as these provide good generic solutions to recurring design problems, thus becoming the building blocks for adding flexibility and extensibility. This paper presents an approach for recovering these building blocks, through a semi-automatic reverse engineering multi-phased process that recovers design elements of increasing level of abstraction. A supporting tool was developed and validated over the JUnit framework. This approach helps on understanding a framework's design by providing useful intermediate results (hot spots, metapatterns and design patterns), improving the learning process and supporting an effective design and code reuse: a framework's main goal.

Keywords: Framework understanding, design patterns, design recovery, framework documentation, reverse engineering.

1 Introduction

Frameworks are a powerful technique for large-scale reuse that helps developers to improve quality and to reduce costs and time-to-market [9]. However, before being able to reuse a framework effectively, developers have to invest considerable effort on understanding them, especially first time users, because its design is often very complex and hard to communicate, due to its abstractness, incompleteness, superfluous flexibility, and obscurity [22]. Good design and implementation are therefore necessary pre-requisites for successful framework reuse, but not sufficient. Although not always recognized, good documentation is another important pre-requisite, as it can significantly help on the process of learning new frameworks.

The problem with documentation is that producing good quality documentation for a framework is hard, costly, and tiresome, especially for large frameworks or when not supported by appropriate tools and methods [17].

Approaches to help on framework understanding focus on existing artifacts, such as framework code or existing instantiations, relieving developers from the creation of additional artifacts, and therefore saving considerable time and effort. Among these approaches we can include software visualization and reverse-engineering techniques.

Reverse engineering techniques concern on how to extract relevant knowledge from source-code and other existent artifacts and present it in a way that facilitates comprehension, by showing components and its relations at different levels of granularity and abstraction using many forms (i.e. patterns, models, beacons, idioms, call graphs and control flow diagrams).

Considering the issue of framework understanding, focusing on its effective reuse, this paper presents an approach for recovering framework design, namely design patterns. Design patterns proved to be extremely useful artifacts to capture design experience and enclose meta-knowledge about a framework's flexibility.

The presented approach relies on a semi-automatic reverse engineering multi-phased process that recovers design elements of increasing level of abstraction, already presented in [11]. This paper summarizes the approach and its supporting tool and presents results and possible perspectives for future work.

2 Frameworks and Design Patterns

Frameworks are considered a powerful reuse technique because they lead to one of the most important kinds of reuse, the reuse of design.

The main concern of framework design is to separate the aspects that are invariant along several applications in a domain — the frozen spots — from the other domain aspects that vary among applications and thus must be kept flexible and customizable — the hot spots, also called variation points [14], [22].

The key elements used to design a framework can be divided in three levels of abstraction: template-hook mechanisms, metapatterns, and design patterns. There can be other artifacts present in a framework [29], but for the purpose of the approach here presented, these three levels are sufficient.

The possible ways of composing templates and hooks in the variation points of a framework were enumerated and presented in [22], [23] under the form of a set of design patterns, called metapatterns. These are a useful abstraction that can be applied to categorize and describe framework hot spots on a metalevel and are repeatedly found in design patterns.

A design pattern is commonly defined as a generic solution to a recurring design problem that might arise in a given context [3], [15]. Design patterns are extremely useful to provide flexibility and extensibility, and are particularly good to document frameworks because they capture design experience and enclose meta-knowledge about how the flexibility was incorporated.

Generally, it is preferred to attach metapatterns to design patterns, and design patterns to concrete participants that instantiate them, instead of attaching metapatterns directly to concrete participants, mainly due to the redundancy introduced and the level of detail being too fine to be useful.

3 Related Work

On the subject of recovering design patterns (DPs), several reverse engineering approaches have been developed in recent years. These range from automated detection tools to pattern extraction heuristics, all with variable outcomes regarding availability, flexibility, effectiveness and efficiency. These approaches have been briefly reviewed and are summarized below.

The Pat [19] system is a design recovery tool for C++, which uses a PROLOG fact base for formalizing design patterns and matching them upon C++ header files, using the PROLOG inference engine. The lack of semantic information affects dealing with behavioral patterns.

KT [5] reverse-engineers design diagrams from SmallTalk code and matches it against a both static and dynamic model of design patterns. The detection methods are, however, hard-coded into the tool, hindering its flexibility.

Through a Pipes&Filters style, Seeman and Gudenberg [24] use a successive step-like approach to recover DPs from Java. Each step uncovers information of different layers of abstraction, based on inheritance, method calling and naming conventions, and is successively transformed into a matching-ready form for a final matching task.

The SPOOL [18] reverse engineering environment, aimed at industrial development, has three-tier architecture: object-oriented database, repository schema and end-user tools. These layers provide storage, knowledge representation (both static and dynamic) and visualization, and specific domain tools for design pattern recovery, all in an integrated environment.

By matching a source-code instantiation of a predefined meta-model against design pattern representation in that same meta-model, Albin-Amiot and Guéhéneuc [2] rely in both structural and behavioral information to uncover existing design patterns.

JBOOTRET [20] approach intends to separate data extraction from information representation, thus preventing repeating the analysis process for each higher-level model extraction. It parses C++ code through a data extractor, mediated by a knowledge manager and shown upon an information presenter. These three major components are designed with a fine degree of flexibility to adapt to other languages.

Heuzeroth et al. [16] developed an automated step-by-step process that filters tuples of program elements such as classes, methods or attributes, which conform to certain design patterns rules. Despite combining both static and dynamic analysis (code construction and runtime behavior), the user has to intervene to verify the results.

SPQR [25] is a Pipes & Filters based approach where the source code is progressively filtered and converted into intermediate notations until it finally reaches a state suitable for formal proof assertions of design patterns. This process requires an extensive formalization of the design patterns made a priori, based on its structures and relationships between its components. Its representation relies on production rules. It is relevant to point out the reasonable high-abstraction level of this approach, similar to the metapattern level.

DPVK [28] is a reverse engineering tool to detect design patterns in Eiffel source code. This approach relies on a phased process starting on a static behavior analysis of candidate design patterns and ending on a dynamic behavior analysis. It relies on

information stored in a repository where the design patterns have previously been catalogued according to those two aspects of behavior. It presents itself as a plug-in for IBM's Eclipse development environment.

Reviewing these approaches emphasized a number of aspects to be considered while developing a design patterns recovery approach. These were:

Formalization. An adequate formalization of design patterns improves the detection of its components, whether structural or behavioral.
High-level of abstraction. Formalization must address high-level abstractions first. If only low-level abstractions are formalized, the detection process will be short-sighted and pattern variations would have to be included.
Harnessing behavioral information. As a complement to structural information (classes, attributes, methods), behavioral information (method invocation, generalization, delegation, etc.) is a vital key for the effectiveness of the recovery process.
Flexibility. Allowing for the extension of the pattern knowledge base to accommodate new discovered patterns widens the applicability of the process and its adaptation to specific domains.
Clarity and traceability. Besides providing the user with a clean and clear view over the results, it is equally important to provide traceability back to the source-code and between intermediate results, if they exist.
Iterative, intermediate results. Without full-proof detection heuristics (mainly due to the nature of design patterns, implementation variations and overlapping of common concepts), adopting a phased, iterative process with added-value intermediate results is preferred to a one-step fully automated detection process.

4 Design Recovery Approach

As described earlier, associations of templates and hooks can be mapped to metapatterns, which capture the essence of design patterns by emphasizing the flexible aspects of these elements. Taking these facts into consideration, we have devised a high-level reverse engineering process to detect design patterns [13].

At its conception the process had the following goals:

1. To adopt an approach based on metapatterns, where each step of the process delivers clear and relevant information about the possible reusability of an object oriented software system.
2. To use a representation for design patterns based on its composition of metapatterns. It should be generic enough to be able to represent any design pattern, whether already discovered or yet to unfold.
3. To provide automation and tool support, by developing a tool to automate the reverse engineering process, and be flexible, interactive and capable of visually deliver its results to the user.

Formalization

According to Tourwé [27], tool support involving design patterns requires, first of all, a formal definition of those design patterns. A good abstraction is therefore mandatory and metapatterns provide exactly the kind of abstraction required to define design patterns. This approach uses a formal representation of metapatterns developed by Tourwé. In his work, a formal model for representing metapatterns is presented. An example is shown below:

```
unificationFundamentalMP(H,H::Mt,H::Mh) with:
participants:
      hookHierarchy(H)
      templatemethods(H::Mt)
      hookmethods(H::Mh)
   constraints:
      understandsMessage(root(H), H::Mt)
      definesMethod(root(H), H::Mh)
      understandsMessage(leafs(H), H::Mh)
      invokes(H::Mt, H::Mh, self)
```

Phases

The process starts from the source code, be it a snippet, a module, package, component or an entire framework, and progressively evolves, step by step, until reaching the desired detail of information about the framework design. Each consecutive step delivers concrete results, such as: templates, hooks, hot spots, and instances of metapatterns and design patterns. The approach relies on probing the source code for variation points and successively grouping them into structures of increasing level of abstraction.

Template and hook methods are identified and grouped into hot spots and metapatterns, which are then used to identify instances of design patterns. This is done with some level of certainty, based on a knowledge base of existing pattern definitions. Each and every step will produce intermediate value-added results, providing elements to complement or generate new documentation over the identified framework variation points, possibly in several formats, such as UML [21], Javadoc [26], XSDoc [1], or SVG [10].

The phased process unfolds as depicted in Figure 1, each step leading into the next level of abstraction and presenting itself with deliverable results.

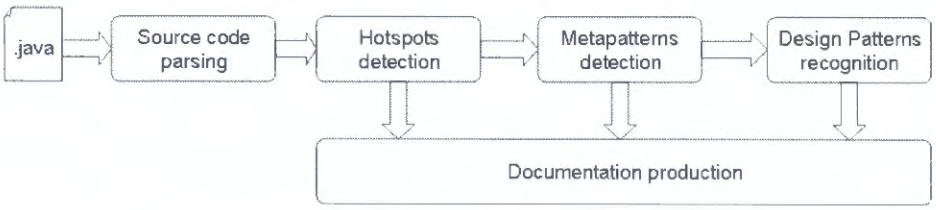

Fig. 1. Phases of the design recovery process

5 JFREEDOM Tool

To support our reverse engineering approach, a semi-automated tool was developed [11]. Existing available tools don't address reusability information in such a straightforward manner and are rather available for use.

The main concerns regarding the development of this tool were to make it:

- *Flexible*, i.e., it should be pluggable into an integrated development environment.
- *Interactive*, i.e., the user should be able to control its process and customize its behavior and preferences.
- *Able to generate visual information*, whether graphics or text, to store the produced data.

As a result, JFREEDOM was conceived as an Eclipse plug-in, whose architecture is depicted in Figure 2

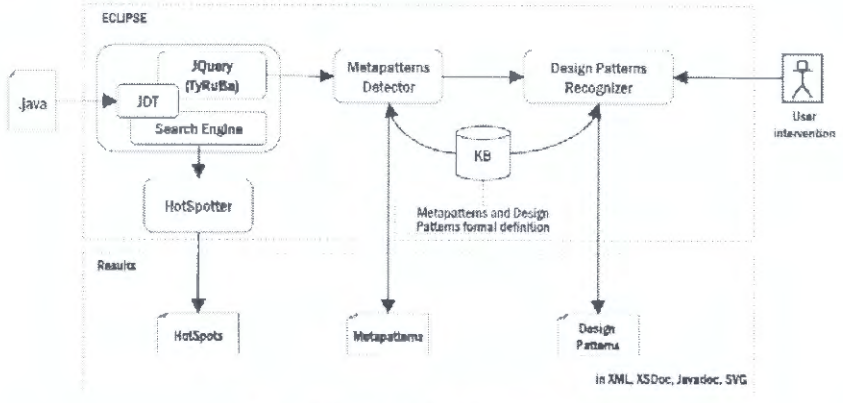

Fig. 2. JFREEDOM's Architecture

The tool was developed in Java, using the TyRuBa inference engine [7] as a source code fact base. The TyRuBa engine was developed in conjunction with JQuery [8], a query-based code browser plug-in for Eclipse. It is defined as a set of TyRuBa predicates which operate on facts generated from the Eclipse JDT's abstract syntax tree.

To speed up development, the second step of grouping templates and hooks into hot spots was done directly using the JDT Search Engine, by the HotSpotter [12] module.

The subsequent modules, Metapatterns detector and Design Patterns recognizer use the inferred results from TyRuBa to produce their own. To this aim, the formal definitions of metapatterns and design patterns that uses Tourwé's work, were translated to facts and rules into TyRuBa's knowledge-base. The tool's source code is available upon author's contact.

6 Case-Study and Experimentation: JUnit

The JFREEDOM tool was tested, verified, and validated using the JUnit framework. JUnit [4] is an open-source framework for unit testing of Java source-code. It is build based on the xUnit [6] architecture and was originally develop by Erich Gamma and Kent Beck. Its most important features include: assertion of expected results; "Test fixtures" for test grouping; "Test Suites" for test serialization and a text and graphical user interface for test execution.

While applying the tool to the JUnit framework, it uncovered the following metapattern instances:

- 1 recursion fundamental metapattern instance;
- 28 connection fundamental metapattern instances;
- 1 hierarchy fundamental metapattern instances;
- 10 creation fundamental metapattern instances and
- 21 unification fundamental metapatterns instances.

Through a preliminary inspection of the results, the authors could identify some expected GoF [15] design pattern instances from their recovered candidate metapattern instances, namely Template Method, Factory Method, Decorator and Composite. An example of the Decorator recovered pattern is shown in Figure 3.

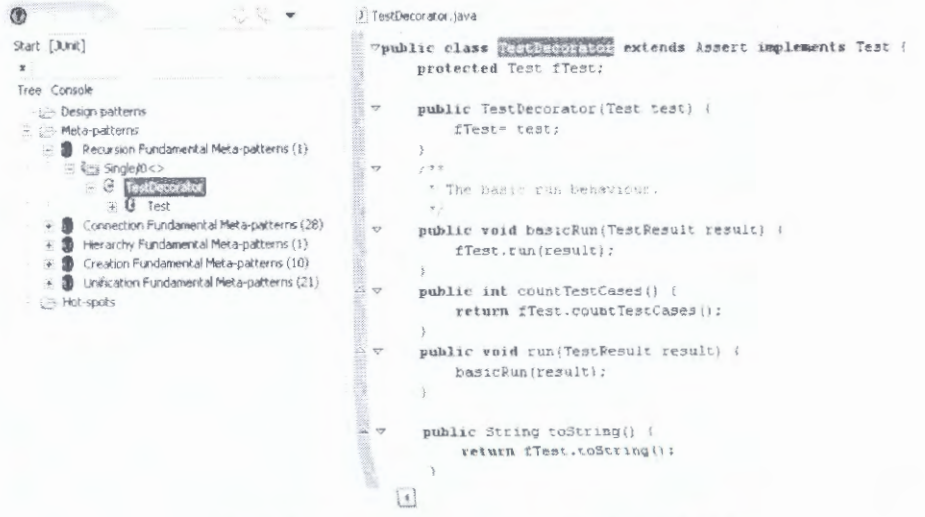

Fig. 3. Decorator pattern recognized by JFREEDOM

A further analysis of the results led to pinpointing some positive aspects...

- *Highlighting of the framework flexibility areas.* By showing where metapatterns instances are located, the user has a clearer scope of the framework's flexibility points.

- *User intervention and fast traceability.* It proved to be quite easy to find out what design pattern could be present around a certain discovered metapattern instance.
- *Design pattern documentation support.* The fast access to design pattern oriented in-tool documentation greatly improved the process of inspecting the results and match them to design patterns.

and less positive aspects...

- *Generic containers.* The tool can't cope with generic structures which can contain any kind of instances. This hinders the detection process and prevents detection of some patterns and their variations (e.g. Composite).
- *False candidates.* There are quite a few false candidates, which could be marked as such and removed from a next iteration.
- *Result ordering.* The order by which the components of each metapattern instance appear could be better suited for traceability purposes.

Overall, JFREEDOM assisted on identifying and restricting JUnit flexibility "hot spots" by showing where metapatterns instances can be found. The quick and easy code inspection around these elements provided the user with a fast assertion of the design patterns present in the source-code, thus improving the framework understanding process.

6 Conclusions

In order to efficiently understand a framework, one must be aware of its design, at different levels of abstraction. Most commonly, the accompanying design documentation of frameworks is poor or even inexistent, thus leading to a steep learning curve.

Uncovering the design of an existing framework is a hard and tiresome task, therefore an automated aiding tool comes in order. Several solutions exist to this problem, yet none deals with the aspect of reusability in a clear, straightforward manner, and are unable to provide useful intermediate results.

The approach proposed focuses purely on identifying the elements responsible for providing framework reusability and flexibility, at various levels of abstraction. Aimed at obtaining the same results as other existing approaches it provides however usable intermediate results (hot-spots, template-hooks, and metapatterns), even if the final results (design patterns) aren't sufficiently accurate.

The proposed multi-phase process supports itself on the concept of metapatterns, a meta-level representation of an abstract design pattern that describes the relationship between the elementary template and hook methods.

The automated tool aimed at supporting the proposed approach, being flexible, interactive and visually delivering its results to the user. An Eclipse plug-in was decided to be the preferred way to satisfy such requirements.

The experimentation with JUnit proved its usefulness and applicability and also revealed some limitations and room for future improvement.

6 Future Work

Both the design recovery approach and the developed tool have space for improvement and several directions for future work. A few guidelines to follow are, for example:

- Refine the last phase of the recovery process to rely on a formal definition of design patterns. That is, improve the design pattern formalization language.
- Extend the knowledge-base to hold more definitions of design patterns for recovery.
- Try to solve the "generic container" tool limitation.
- Enhance user's intervention to improve results, and diminish false candidates at early iterations.
- Introduce dynamic analysis techniques to improve the recovery process.
- Adapt the recovery process to support program understanding cognitive tasks.
- Use alternative and more adequate software visualization techniques.
- Directly support a round-trip process of documentation production.
- Recover other kinds of framework artifacts, like architectural primitives [29], beacons, idioms, etc...

As a jumpstart point for a future PhD work, the main idea is to try and evolve both the approach and tool to support a more extensive framework understanding process. How do users actually learn how to use a framework? What cognitive processes do they undertake? Is there a way to automatically support some or all of these processes? To knowledge, there are several cognitive ways of learning and reading a framework or software system. Assessing which ways are there and how they can be supported could improve the overall learning process. Associating software visualization techniques with the recovery of software artifacts may prove to be a successful combination.

References

1. Aguiar, A., David, G. and Padilha, M.: XSDoc: an Extensible Wiki-based Infrastructure for Framework Documentation. In E. Pimentel, N. R. Brisaboa, and J. G´omez, editors, JISBD (2003), 11–24.
2. Albin-Amiot, H. and Guéhéneuc, Y.-G.: Meta-modeling design patterns: Application to pattern detection and code synthesis. In B. Tekinerdogan, P. V. D. Broek, M. Saeki, P. Hruby, and G. Suny´e, editors, proceedings of the 1st ECOOP workshop on Automating Object-Oriented Software Development Methods. Centre for Telematics and Information Technology, University of Twente (2001)
3. Alexander, C., Ishikawa, S. and Silverstein, M.: A Pattern Language. Oxford University Press (1977).

4.Beck, K. and Gamma, E.. Junit homepage (1997). Available from http://www.junit.org.
5.Brown, K.: Design reverse-engineering and automated design pattern detection in Smalltalk. PhD thesis, North Carolina State University (1996).
6.Beck, K.: Simple Smalltalk Testing : with patterns, (2003). URL: http://www.xprogramming.com/testfram.htm.
7. De Volder, K.. TyRuBa Home Page (2004). Available from http://tyruba.sourceforge.net/.
8. De Volder, K.: JQuery Tool Home Page, 2004. Available from http://www.cs.ubc.ca/labs/spl/projects /jquery/.
9. Fayad, M. E. and Schmidt, D. C. (1997). "Object-oriented application frameworks". Communications of the ACM 40(10), 32–38.
10. Ferraiolo, J. et al.: Scalable vector graphics (SVG) 1.1 specification, w3c recommendation, January 2003. Available from http://www.w3.org/TR/SVG11.
11. Flores, N. and Aguiar, A. : "JFREEDOM: a Reverse Engineering Tool to Recover Framework Design" in 1st International Workshop on Object-Oriented Reengineering, ECOOP'05, Glasgow (2005)
12. Flores, N. et al.: HotSpotter: : using JavaML to discover hot-spots. In Proceedings of XATA 2005, XML: Aplicações e Tecnologias Associadas (2005).
13. Flores, Nuno: Engenharia Reversa de Padrões em Arquitecturas Reutilizáveis, MsC thesis, MEI, FEUP (2006). Available from www.fe.up.pt/~nflores/pmwiki/pmwiki.php/FEUP/MsC
14. Fontoura, M. et al.: The uml profile of framework architectures (2000).
15. Gamma, E. et al.: Design Patterns — Elements of reusable object-oriented software. Addison-Wesley, 1995.
16. Heuzeroth, D.: Automatic design pattern detection. In Proceedings of 11th IEEE International Workshop on Program Comprehension (2003), 94–103,.
17. Johnson, R.: Documenting frameworks using patterns. In A. Paepcke, editor, OOPSLA'92 Conference Proceedings, ACM Press (1992), 63–76.
18. Keller, S. R. et al.: Pattern-based reverse engineering of design components. In Proceedings of 21th Conference on Software Engineering (1999), 226–235.
19. Kramer, C.: Design recovery by automated search for structural design patterns in object-oriented software. In Proceedings of the Working Conference on Reverse Engineering, (1996), 208–215.
20. Mei, F. Y. H. and Xie, T.: Jbooret: an automated tool to recover oo design and source models. In Proceedings of 25th Annual International Computer Software and Applications Conference (2001), 61–76.
21. OMG. Unified modeling language (2003). Available from http://www.omg.org/uml.
22. Pree, W. : Design Patterns for Object-Oriented Software Development. Addison-Wesley / ACM Press (1995).
23. Pree, W.: Hot-spot-driven development. In Building Application Frameworks — Object-Oriented Foundations of Framework Design, John Wiley & Sons, (1999), 379–394..
24. Seemann, J. and Gudenberg, J. W.: Pattern-based design recovery of java software. In SIGSOFT '98/FSE-6: Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering, New York, USA, ACM Press (1998), 10–16
25. Smith, J. and Stotts, D.: Spqr: Flexible automated design pattern extraction from source code. In In 18th IEEE International Conference on Automated Software Engineering (2003).
26. Sun Microsystems. Javadoc Tool Home Page, 2003. Avaliable from http://java.sun.com/j2se/javadoc/.
27. Tourwé, T.:Automated support for framework-based software evolution, PhD thesis (2002).
28. Wang, V. T. W.: Dpvk - an eclipse plug-in to detect design patterns in eiffeil systems. In Department of Computer Science, York University, Toronto, Canada (2004).
29. Zdun, U. and Avgeriou, P.: Modeling architectural patterns using architectural primitives. In Proceedings of the 20th Annual ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages, and Applications.OOPSLA (2005).

An Approach to Modelling Informal Communication in Business Networks

Firmino Oliveira Silva[1], João José Pinto Ferreira[1]

[1] Faculty of Engineering, University of Porto
Rua Roberto Frias, s/n
4200-465 Porto, Portugal
{pro06015, jjpf}@fe.up.pt

Abstract. Nowadays, business market forces enterprises to make fast decisions for taking advantages upon opportunities. Many times, the strategic partnerships among enterprises are the only way to provide a fast response to those opportunities. This paper develops in a context in which enterprises have to establish negotiated agreements for successful networking among business partners. To this end, enterprise organizational structures, supported by adequate information systems, should enable the fast deployment of negotiated decisions. A considerable number of these decisions will be the result of negotiation processes, typically informal processes, for which there is no information system support capable of not only supporting the actual negotiation process but also enable its effective deployment in the organization. In this paper we propose an approach to modelling enterprise informal communication interaction in a negotiation setting. The proposed modelling framework is briefly introduced as well as its rationale in the enterprise collaborative environment fostering the support of informal communication across enterprise borders both at tactical and strategic management levels.

Keywords: Enterprise integration, Networking, Collaboration, Negotiation.

1 Introduction

The heavy competitive pressure of the market forces all competitors to design strategies of continuous adaptation to business environment, creating agile and flexible structures for responding, with the highest total quality level, to market demands. Under a competitive and dynamic business environment, the move to agility requires the ability to respond to unanticipated change [1] in a way to guarantee the interconnection with the remaining players, as each enterprise operates in the market as a node in the network of suppliers, customers, service providers and partners [2].

This environment further enhances the need for the interaction between the enterprise and its business partners in order to collaborate together against an increasingly aggressive competition. In fact, interaction is realized at different levels, strategic, tactical, operational and real-time [3]. In this context we could further detail that upper level interaction among enterprise partners tend to be informal, whereas the proximity of enterprise operations (e.g. Production) tend to allow information systems

and business process integration to reign and provide IT support to this daily operation. In this paper we address the upper layers of the enterprise management where informal interaction both within and across enterprise borders are the rule and where negotiation among partners is a normal activity along any business cycle.

This informal process communication, usually recorded as in meeting or negotiation minutes, is not supported by any integration technology which is a mean of difficulty, at least, when we want to rapidly recover the information within the context it happens. Significant problems [4] with information from multi informal communication channels, makes management and history analysis almost impossible.

In this paper we present a rationale and an approach to a model that structure the informal process of decision communication at the highest level of the organization, fostering the support of informal communication across enterprise borders both at tactical and strategic management levels.

This paper is organized as follows: chapter 2 frames the problem to be solved by framing its scope and context by integrating several enterprise integration concepts and reference architectures. This approach ensures the application generality of the concepts to be developed in the following chapters. Chapter 3 defines the approach to modelling the informal communication along the business negotiation process. At the end of this paper we present the conclusions and the future work to be developed.

2 Framing the Problem

The concept unfolded in this paper started in the analysis of how to use the Zachman Framework in the context of enterprise modelling and integration [5] where time is to be explicitly modelled.

Fig. 1 illustrates a first approach to time integrate in the Zachman Framework from a life-cycle perspective.

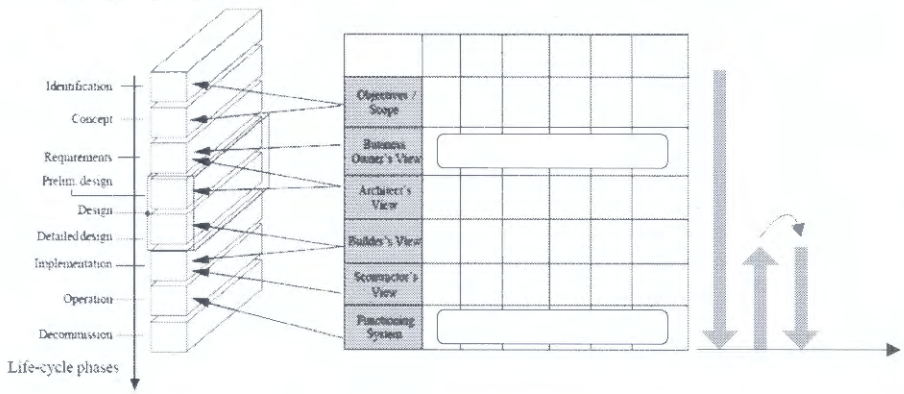

Fig. 1. Mapping the Zachman perspectives to the life cycle phases of GERA

In fact, this clearly pictures that depending on how the enterprise evolves, one could go through either a full engineering of the enterprise entity (i.e. enterprise production line), or through a simple continuous improvement process. Moreover, this further enhances the Zachman view by implicitly introducing the GERAM Enterprise Modelling and Integration Services, where model reuse from modelling time to execution time is the key to a successful integration deployment.

This flexibility of representation fails, however, to tackle the actual operation time where decisions have different sliding time horizons. To this end, [6] introduces the time horizons from the GRAI Integration Methodology [3] and looks in particular at the business model layer of the Zachman Reference Architecture, which, in fact, maps to the requirements phase of the GERA Entity life-cycle [7]. Fig. 2 pictures this view by illustrating that the Zachman Business Model layer can be structured from different perspectives, Strategic, Tactical, Operational and Real-time. This further illustrates that model reuse can, and should, be promoted through the so-called Enterprise Modelling and Integration Services.

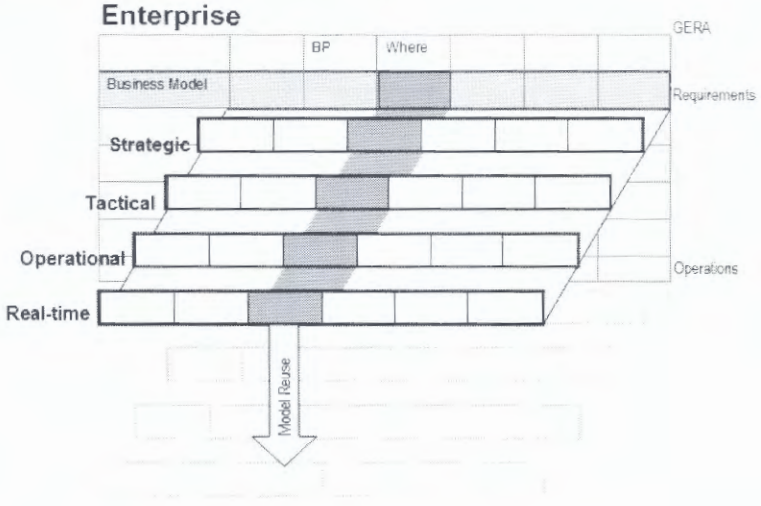

Fig. 2. Time Horizons and Enterprise Model Deployment

Having in mind the above framework, Fig. 3 finally frames the scope of this paper, where we aim at giving IT support to the informal communication that unfolds both inside and across enterprise borders. For the sake of scope reduction, we will concentrate on the tactical and strategic levels.

The picture illustrates the interaction between the two upper management levels having in mind that these interactions may occur in different locations and across different enterprises. Our objective is to add value to this information communication used to support all kinds of activities developed by each enterprise players (e.g. executive managers, marketing people, sales managers, etc.). To this end we propose

the usage of a simple modelling approach that would allow both the support of this interaction and the knowledge gathering as these interactions unfold.

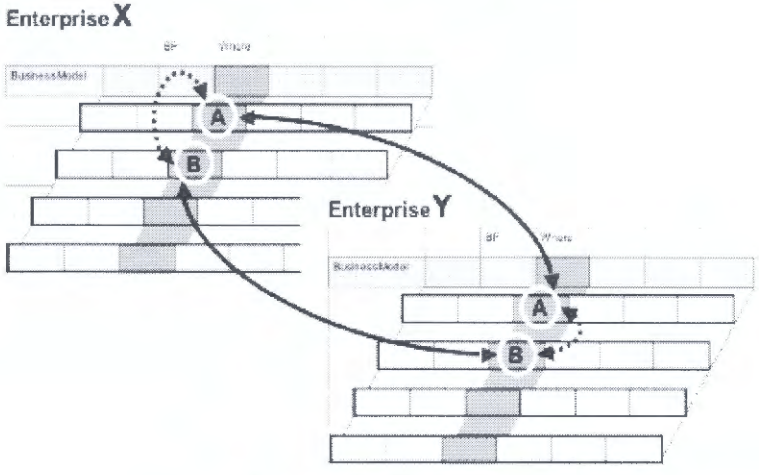

Fig. 3. Cross Border Integration

As a conclusion to this chapter we would refer that the whole introduction to this illustration further enhances its scope. In fact, while talking about this IT support we clearly aim at providing some kind of tool to allow the interaction of managers across enterprise borders. This demands some kind of process to model informal communication as well as its operational support (GERA Life-cycle) that will enact the actual interaction model.

3 Modelling the Informal Communicatiou

The departure point to the topic of this paper was the paper entitled "Introducing Time Horizons to Enterprise Networking Architecture" [6] that proposes an integration of decision time horizons in the Zachman Framework. The proposed concept enabled the explicit modelling of business process interaction at different management levels (strategic, tactical, operational and real-time) [3] both within and across enterprise borders. To this end, we use the principle of Negotiation Utility Space [8] to understand and support the possible outcomes of a negotiation process, which represents, in fact, a strategy to get a decision.

3.1 Negotiation Utility Space

According to [8], there are four main procedures to deal with opposing preferences: Mediation, Struggle, Arbitration and Negotiation. In this paper we assume

Negotiation is the appropriate way to get enterprises, with different objectives and individual strategies, involved in discussion with the goal of reaching agreement.

Negotiation is a process of making decision in a context of a strategic interaction or of interdependence and has several advantages as a mean of resolving opposing preferences. The two main advantages are: it is usually not costly (e.g. struggle often requires heavy expenditure of resources); and makes it easier to find and adopt a mutually acceptable solution (e.g. arbitrators often fails to find mutually acceptable solutions, thereby endangering the relationship between parties). However there are many risks associated to Negotiation [8]: the communication process between parties may be difficult; trusting levels of each other may be so low that they might not dare to get an explicit agreement; one of the parties may be too proud or too angry to do anything that favours the other's welfare.

The Actor-Network Theory explains, in a certain way, those difficulties for the establishment of an agreement in a negotiation. The fact is that each enterprise representative embodies a wide range of convictions [9] of the most different sources (political, religious, ethical, business practice, etc.) that guides the actuation in its deployment and therefore influences its results. Such convictions can be either helpful or harmful with respect to reaching agreement, depending on whether they are shared by the negotiation parties. Thus, the principle of Negotiation Utility Space [8] (based on those constraints) will be used to describe the possible outcomes of a negotiation (as shown in Fig. 4):

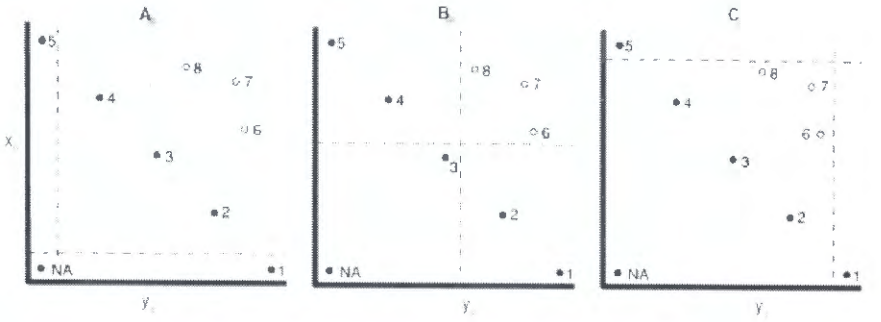

Fig. 4. Three possible contrasting joint utility spaces [8]

Within the Negotiation Utility Space, negotiation ends if: neither x nor enterprises agreed (NA in the Figure); or one of the companies accepted whatever the other proposed, without any compensation in return (dot 1 or 5 in the Figure); or a simple compromise is established (e.g. various options: 2, 3 or dot 4); or, finally, a win-win agreement is reached (e.g. where 6, 7 and 8th options are possible mutually beneficial solutions in which x adds several improvements that benefit y more than they cost x).

3.2 An Approach to Modelling Negotiation

The illustration below (Fig. 5) presents the proposed model to integrate Information Technology Negotiation Support mechanisms at the enterprise desktop. Our vision is that as the pressure increases for new ways to save energy, protect environment while remaining competitive, enterprises will have to integrate the use of videoconferencing, voice-over-IP and other technologies with enhanced desktop functionalities capable of handling informal tasks such as negotiation with business partners.

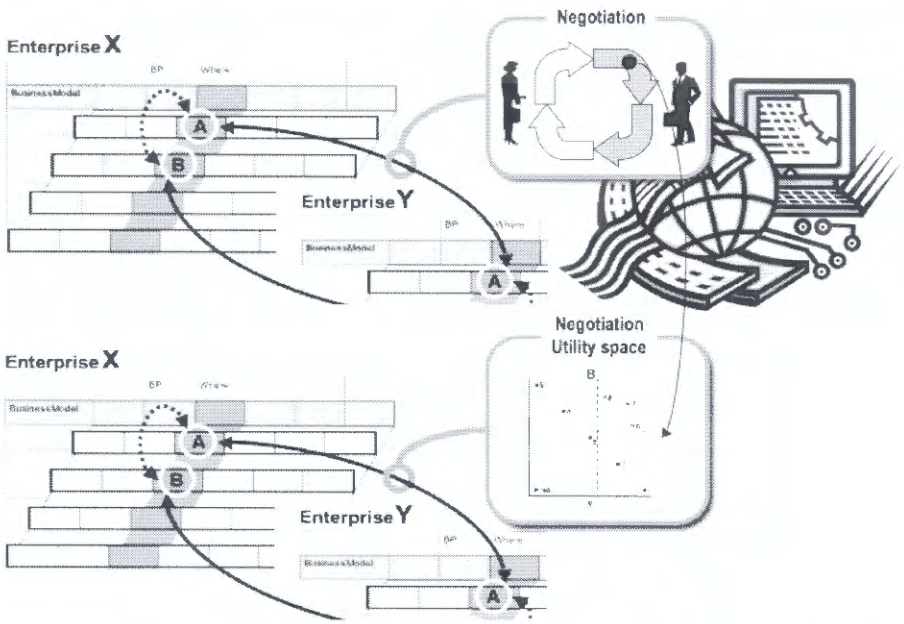

Fig. 5. Modelling a negotiation Interaction

Business interactions can be typically modelled as a "client-server" interaction between two parties that can be modelled as a "speech-act" [10, 11, 12]. This approach has been used in the past to model formal business processes as speech-acts. In our proposal we would like to go a further in this concept arguing that, from the moment the actual negotiation starts, specially if it is done live through a professional videoconferencing system or through SKYPE, informal interaction would be the rule, whereas total information sharing among negotiation parties will be limited to what each partner wants its "opponent" to know.

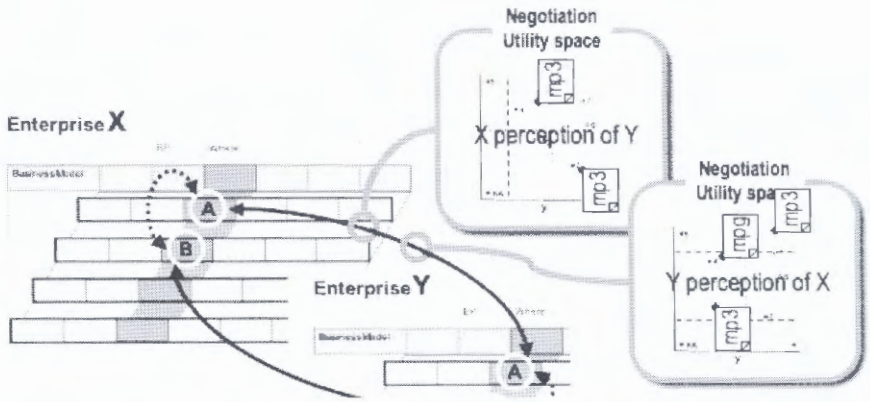

Fig. 6. Not shared perspectives of the same negotiation

The result is that each negotiating partner will have his own view of whatever the other party has to offer and of his "reservation prices". This means that each negotiation party has and builds its own vision and perception of what is going on as the actual negotiation unfolds. Fig. 6 illustrates just that by showing the flexibility granted to each party of not only picturing its own view of the negotiation but also to document it by attaching any type of document to his utility space diagram. These documents have two objectives. On one hand they support the meeting minutes elaboration and the actual decision-making. On the other hand they provide the means for conveying the outcome of the negotiation to the other management levels of the enterprise: the actual negotiation results as well as the whole contextual framework (information, reports, recorded video/voice meetings) under which the actual negotiation occurred.

4 Conclusions

With this paper we aimed at describing a model that would be able to handle the enterprise changing environment. The basic principles for this work were launched by the research conducted by [6], which enriched the Zachman framework with the time horizons concept developed within the GRAI Integration Methodology. From there we derived the context for the business negotiation activities performed by enterprise managers across enterprise borders, typically through informal communication as conversation between two or more people.

The approach to modelling enterprise informal communication in a negotiation setting was then presented. The Negotiation Utility Space seems to be an adequate tool to integrate changing information along with documents used to log each negotiating party perception of its opponent's moves.

We believe that an IT Negotiation support mechanisms (as described in chapter 3) will be decisive in a near future as a mean enhance enterprise networking abilities and therefore, its competitiveness.

The proposed structured approach to negotiation support enables an improved effectiveness of management negotiation relationships among enterprises. The enterprise perception of the other party position can be registered, live, during the actual negotiation process, by any combination of multimedia documents attached to any point in the Utility Space Diagram. This approach further enables negotiation track records saving for later reference in the enterprise database.

As future work we are planning further detail the application scope for laboratory validation with the cooperation of negotiation experts and business people so that the best feedback can be obtained for the fine-tuning of the proposed concept.

Acknowledgments

The authors want to acknowledge the Informatics Engineering Doctoral Programme of the Faculty of Engineering of the University of Porto for this opportunity to start the development of this research work.

References

1. Barnett, W., Presley, A., Johnson, M., Liles, D.H.: An Architecture for the Virtual Enterprise. In IEEE Systems, Man, and Cybernetics. Humans (1994), Vol. 1, pp. 506-511
2. C.-H. Kim, Y.-J. Son, T.-Y. Kim, K. Kim, K. Baik: A modelling approach for designing a value chain of virtual enterprise. In International Journal of Advanced Manufacturing Technology. Springer-Verlag (2005), Vol. 28, No 9, pp. 1025-1030
3. Bernus, P., Uppington, G.: An Co-ordination of management activities – mapping organizational structure to the decision structure. In Coordination Technology for Collaborative Application – Organizations, Processes and Agents, Springer-Verlag (1998), Vol. 1364, pp. 25-38
4. Bussler, C.: A Systematic Approach for Informal Communication During Workflow Execution. Advances in Database Technology – Proceedings of EDBT 2000: 7th International Conference on Extending Databases Technology (2000), Konstanz, Germany, Springer Berlin / Heidelberg, Vol. 1777/2000
5. Zachman, J.: A framework for information systems architecture. In IBM Systems Journal (1987), Vol. 26, No 3, pp. 276-292
6. Álvares-Ribeiro, N., Martins, Â., Pinto Ferreira, J.J.: Introducing Time Horizons To Enterprise Networking Architecture. Proceedings of PRO-VE2004 IFIP Working Conference on Virtual Enterprises and Collaborative Networks (2004). Toulouse, France, pp. 63-70
7. Noran, O.: An analysis of the Zachman framework for enterprise architecture from the GERAM perspective. In Annual Reviews in Control (2003), Vol. 27, pp. 163-183

8. Carnevale, Peter J. and Pruitt, Dean G.: Negotiation and Mediation. In Annual Review of Psychology (1992), Vol. 43, pp. 531-582
9. Monteiro, E.: Actor-Network Theory and Information Infrastructure. In: C. Ciborra (ed.), From control to drift. The dynamics of corporate information infrastructure, Oxford University Press (2000), pp. 71-83
10. Williams, M.: The Speech Act Method: Studying Power and Influence in Conversation Interaction and a Critique of Conversation Analysis. The University of Sheffield, In Sheffield Online Papers in Social Research (2002), ShOP Issue 6, ISSN: 1470-0689
11. Goldkuhl, G.: Communicative vs material actions: Instrumentality, sociality and comprehensibility. Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling (LAP 2001), Montreal, Canada
12. Medina-Mora, R.,Winograd, T., Flores, R., Flores, F.: The action workflow approach to workflow management technology. Proceedings of the 1992 ACM conference on Computer-supported cooperative work (1992), Toronto, Ontario, Canada, pp. 281-288

Sessão 6

Redes e Computação Distribuída

Video Surveillance for Front Office Systems in a Retail Company

Pedro Manuel Abreu[1] and Pedro Miguel Gomes[2]

[1] Faculdade de Engenharia da Universidade do Porto
Rua D. Roberto Frias, 4200-465 Porto,Portugal
Telephone: (+351) 22 508 14 40
email: pha@fe.up.pt
[2] Modelo Continente Hipermercados, SA
Centro Empresarial da Lionesa
Rua da Lionesa, 446 edifcio C35
4465-671 Lea do Balio, Portugal
Telephone: (+351) 22 016 10 52
email: pgomes@modelocontinente.pt

Abstract. The retail companies are characterized by the fact of constantly dealing with merchandise. The business processes involved in this reality must focus on productivity but also in security. Shrinkage is the term and variable that measures the loss of merchandise. This loss can have several causes like depreciation, theft and process errors. This paper is focused on dealing with the security issues that concern shrinkage due to theft in stores, specially in the Front Office area - where typically the customer pays before leaving with the purchased merchandise. For this it is presented a new approach which involves a video surveillance system fully integrated with the Front Office. The system is being tested in Modelo Continente Hipermercados, SA which is the major Retailer in Portugal. The innovation factor lies on the type of integration stated before along with the reduced cost of the solution. The first results show the value of solution in two main perspectives: the dissuasive factor and the close integration with the Front Office.

1 Introduction

Sbrinkage in the business of retail is the term that designates the lost of merchandise in stores, warehouses or distribution centres. This economic lost influences the results of a retail company and is a factor which must be constantly analysed. Shrinkage can be divided into three major categories: depreciation, theft and process errors [1]. In the opinion of the *ECR* Europe (Efficient Consumer Response) [2] the external theft, which is not performed by employees, is a consequence of poor processes and deficient security systems [1]. Internal theft must be analysed considering processes and technology but is also very dependent on the human resources policies. This kind of theft is responsible for almost 37% of shrinkage in the world [1]. Modelo Continente Hipermercados (MCH) is the largest retailer in Portugal and in the year of 2004 the results demonstrates

the importance of the external theft as a factor of shrinkage (47% of the total cases) [3]. In this universe of supermarkets, the combination of the internal theft, which is performed by the employees of the shop, and the external theft totalizes 80% of the verified cases. In this year 2004, *MCH* lost around 44,6 millions of Euros with this kind of business threats.

The best way today to prevent a merchandise from being stolen is to attach a security *RFID* (Radio Frequency IDentification) tag to it. This tag allows to trigger an alert whenever someone tries to leave a store without having paid the merchandise. This type of technology is still too expensive to use in every product, like in an pack of rice. This technology is also effective for the external theft but not for the internal one. For that we need different approaches, more human independent and not so expensive - this approaches regard video snrveillance systems.

Nowadays efforts have been developed by Mitsubishi to analyse the scenes in terms of objects detected [4]. This problematic is a very complex. The fundamentals of this approach are the segmentation of the scenes and after that the tracking of objects from a stream of images. This process is steal in an investigation domain, because there is not yet a satisfactory solntion to use in practice. There are also other approaches types, more commercial, based in *CCTV* (closed circuit television) systems used especially for crime prevention [5]. They record all movements in a restrict area, using cameras all over the area which are connected to a central (may be a *DVR* (digital video record). In case of incidents this video capture can help as a prove.

The main contribution of this paper is to present a unique and innovator video surveillance system integrated with the Front Office system. Based on a traditional *CCTV* system and using a simple architecture, it connects one camera per *POS* (point of sale) in a checkout's area in the shop with a server. It brings to the operations supervisor in a store a powerful real time information to support the security process. We believe that at the eud of this project, we can have a tool that helps us to: improve security in the checkout's area, increase the client's comfort and the agility and productivity of the employee's. In the same way, we hope reduce the factors of shrinkage. For a better comprehension, this paper is organized in five sectious. In the introduction section we described the problenatic that we try to fixed it with this new solution. in the second section we explained the methods and procedures used in the study. We have also a section reserved to the results and another to the conclusion and feature implementations that can be development in this area.

2 Overview of Video Surveillance Implementation

2.1 The Front Office System

In a retail company the Front Office System is a system that supports all the process of sales products iu a store. It works as a normal computer which iutegrates one server per shop, all *POSs* existent in the shop, scale nachiues, price

checkers and more recently *PDAs* (Personal Digital Assistant). The Front Office' server is a crucial component for the business because it aggregates the information of all sales in a shop, does the management of stocks and modes of payment and so on. It also diffuses the products information (specially prices) through the *PDAs*, controls the operations (transactions) in the shop, analyses the operator's productivity and produces the necessary information for Back Office System.

The application of the Front Office Server is in a web platform, which is only accessible for the company intranet, and it's used by the shop's operators and by the company administrators. One of the functionalities supported by this web platform is the *SCC* (checkouts control systems) panel (Figure 1), made with *PHP* (Hypertext Pre-Processor) technology. Through this, we could see all the checkout's line of the shop for example if the shop had fifty *POSs* we will have their checkout's state (open, pause or close) represented in different colours. In this panel *SCC* we also do, in real time, the treatment of some different occurrences in the *POSs* and so, it's not rare see messages like "allowing saw things like annnlations of the product 26 (*EAN* internal number) in *POS* 40"," *POS* 50 change it state to pause" and so on. As we explained, the panel is a good tool for the bnsiness bnt it conld be better if we add more information to it, and if the total information contained on it, appears at the same time, for example the information of checkout's line with their real time image and their real time client's coupon information of the *POS*.

Fig. 1. Example of a SCC Panel

2.2 Architecture Used

The most important and critical fact for a retail company is to sell its products. As we analysed in the last section this process is supported by a Front Office

system implemented in the stores. One failure in this system could result in an error on the sales transaction and more critical it could result in the shop's close. In the beginning of this project, we needed to know if the infrastructure of the shop, where we will do the pilot test, will supports a new debit in it's local network. This study proved that the infrastructure of that shop was saturated and a new extra debit could provide a collapse in the system. Consequently we defined a new independent architecture for this project (Figure 2). This architecture consists in a new network constituted by one camera per *POS*, which connects them, via coaxial cable, to a *DVR* system located in the Back Office of the shop. After that we connect the *DVR* to our new network to attribute to this system one *IP* (internet protocol). Internally the *DVR* attributes one hexadecimal mask per camera and so, after we access to the *DVR*, we could easily access to a specific camera.

In this approach we have two types of debit. One of them, via *IP*, when we access to the *DVR* and the other, via camera, resultant of the transmission of images' frames. In this sequence, the resolution of the cameras was 320X240 because we assumed that it provides a good debit for the network and so, we obtained a good images' quality for the posterior analyses.

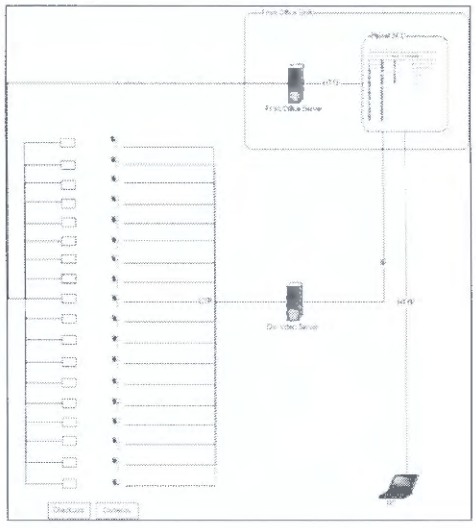

Fig. 2. Architecture used in this project

2.3 Integration with Front Office

Our objective, with the integration at our new video surveillance system with the Front Office system is to add new functionalities to the *SCC* panel in particular

introducing real time video for a specific checkout and a near real time purchase coupon stub for a client in that *POS*. For us each information presented alone in this panel isn't so useful for the business but all functionalities together at the same time could turn this panel a powerful tool for the company. So, first off all, we needed to guaranteed that, when the *SCC* panel made its refresh, which occurred in 60 to 60 sec, the information presented their, like checkout's line and the treatment of errors, didn't disappear in the screen. For that, we used the *AJAX* (Asynchronous JavaScript and XML) technologies to make all the refresh in background, becoming this process total transparent for the user and guarantying that the information didn't disappear. To obtain real time checkout's image we need to, in which number of checkout represented in *SCC* panel create a link to a specific camera (the number of the camera is the same of the checkout to turn the implementation's process more simply). Like we referred in the last section we have our *DVR* system identified in the network by an *IP* address and it accesses via hexadecimal mask, to a specified camera. So to do the link to a checkout's image we only need to have the *DVR IP* address with the specific camera mask to the *SCC* panel. In the process of product's transaction the *POS* generates a log file and after that, when the length of this is superior than 3kbytes it transmits it to the Front Office server. To obtain the pnrchase coupon stub we have the possibility to make a select in Front Office server and to achieve a near real time client's coupon for a particular *POS* obtaining the result demonstrated in Figure 3. This approach have many advantages such as the transaction of this log never pnts the system in danger because it doesn't modify the normal process implemented in the shop particulary the connection between *POS* and the server. Furthermore, if we don't use this approach when we do a direct select in *POS* system we need to reconfignre all the *POS's* kernel and, with this we conld provide extra debit in the shop's network. Conseqnently we will put in risk all the process of product's transaction. That's why we follow our approach. However, our project has disadvantage too. The main of them is that, we obtain only the information after 3KBytes captnred; that corresponds to 3 lines in a normal conpon. Because of this, we talk about near real time client coupon information.

3 Results and Discussion

The results were based on inquiry with the key users, but further analysis will be made concerning the evolution of shrinkage factor along with the clients perception of the system intrusion.

One of the most important results of this implementation was that the system is considered to be very discrete in terms of visibility of machinery. The cameras are very small and are hidden. This is important for the cnstomers, which dont feel too have a big brother on top of them. On the other side, for the operators, and because there were already a lot of cameras on the store the general feeling was good.

Informações do Talão			
POS	014	Talão	011519
Data	30/12/2006	Hora	18:46:18
Estornado	Não	Finalizado	Não

Artigos		
EAN	Valor	Anulado
5410063037163	4.99	Não
2815987000009	2.35	Não
5601212001874	0.56	Não
5601038240203	0.87	Não
5601027000729	1.15	Não
2614276000069	1.03	Não
2612914000007	3.68	Não
2612878000006	3.69	Não
2815980000005	1.84	Não
2612871000003	5.61	Não
5601131021433	1.71	Não
5601192201104	0.43	Não
5601165003148	1.06	Não
5601131107003	1.33	Não
5601493048926	4.88	Não

Formas de Pagamento	
Código	Valor
45	110

Fig. 3. Result of clicked in a specific *POS* in *SCC* panel

Another important result is the dissuasive factor: those who execute theft pay more attention to the security technology, and having a camera in each *POS* accentuates this factor. Typically a store has several cameras spread out, but this approach requires one camera per *POS*, and for that there is immediately a sense of more security than with other systems. This system allows a all new type of interaction with the operator that in on the *POS*. Since the supervisors are always watching to the abnormal operations in the central console (*SCC*), they can now make a phone call with video asking for the rational of the operation. They can ask during the phone call to show the merchandise, the bags, and so on, which allows more control in the exceptions. This interaction puts on the operator a sense that someone is always watching them. With this type of architecture, with a separated network for the cameras, the adicional information around the network didnt increased in a significant way (Figure 4). Nevertheless it is very important not to have to many users watching the console at the same time.

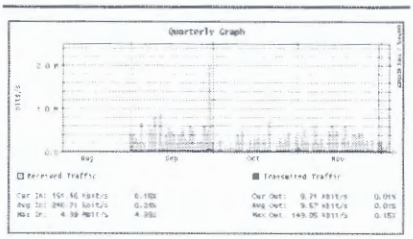

Fig. 4. Result of a network debit study

The cost of the solution is in fact very low which was on of the objectives but for that the quality of the presented images are not suitable for analysing the small details.

4 Conclusion and Future Work

We have presented a video surveillance system, which reduces the factors of shrinkage in a retail company. This system is characterized by the flexibility of its architecture, independent one, in terms of networks and resources existents in the shop, never putting in danger the Front Office system. Thus, the security of the product's transactions is well guaranteed. It is also a cheap system because it uses low cost cameras connected to a *DVR*, enabling to visualize many information in the *SCC* panel, like errors in *POS*, real time image on the checkout, near time coupon, and so on. We believe that this system is original and its principal innovation is the integration of many technologies such as: the front office system, the cameras and the *DVR* system. Together, they form a very important beat to a retail company and certainly in the future it will constitute an important case of study for many companies in this area.

Our future work will be, in a first phase, try to visualize a real time coupon and after that, substitute the *EAN* code to a description of the product. It will make the analysis more simple and correct. In another phase, we will try to record all the day images captured, during a day, in the shop, in each *POS*, using a *DVR* system. This will enable to see a transaction, in a specific *POS*, in a given interval of time defined (this interval is defined by the shop's supervisor or by other systems user). This new functionality is practically implemented because we already used a *DVR* system in the shop. However, it's only used only to allow access to all cameras. Another functionality that can be added is to integrate this system with other system available at the company called *IntelliQ* system [6]. This is an information system, that receives information all over the company such as data sent by the *ERP* (enterprise resource planning), by the Front Office of each shop and so on. The goal of the use of data minding technologies is to generate reports with useful information for the company. So, if we send via our video surveillance system, some critical events (that were

detected in a day in the shop) for example annulations of product 21 in *POS* 42 in the next day the *IntelliQ* sends to the shop the report indicated the period of time that its occurs and the respective *POS*. The supervisor only had to access the *DVR* and analysed the images to decide if had occurred any irregularity in the checkout. Finally we will try to extend this project to solve other kind of retail company's problems more directed to the satisfaction of the client snch as the clients' queues in the checkouts of the shop.

Acknowledgment

In this project we like to thank to Eng Pedro Lago for all the support and trust demonstrated in this entire project and for the possibility to testing this new idea in a real world (in shop's environment). We like also to thank Sr Henrique Barbosa (shop's director) for the enthusiasm and good reaction in all the project steps (implementation pilot) and future functionalities. Finally we like to make a specially acknowledgment to all members of nnifo's team for the support and friendship.

References

1. *Skrinkage - a collaborative approach to reducing stock loss in the supply chain - now tried & tested!*, by Leicester University and Cranfield Business School 2004., http://www.ecrnet.org/, acess in 4/4/2006
2. organization to help industries in breaking down non produtive barriers., http://www.ecrnet.org/, acess in 4/4/2006
3. *Internal Presentation*, by *MCH* directory 2004., 2004 Business Report. ppt, acess in 15/3/2006
4. Scene Analysis using Camera Arrays, http://www.merl.com/projects/camarray/, acess in 15/4/2006
5. Video Surveillance to Fight Crime, http://www.epic.org/privacy/surveillance/, acess in 17/4/2006
6. The Intelliq System, http://www.intelliq.com/, acess in 23/5/2006

Distribution Management Systems:
A new approach for Managing Edits

Manuel Cláudio de Magalhães Freire

Department of Computer Science, Faculty of Engineering, University of Porto.
Doctor Roberto Frias Road, 4200-465 Porto, PORTUGAL

Abstract. This work proposes an alternative to the data (Electric Net Models) storage management problem generated on a DMS (Distribution Management System). That is an alternative to expensive commercial solutions based on GIS (Geographic Information Systems) or GDS (Geospatial Data and Systems). The new MMVE (Model of Management of Versions of Edits[1]) is an efficiently implemented solution that uses a temporal database (TimeDB) and a version management system to play the role. This new approach makes a significant difference to overcoming costs by running on a RDBMS (Relational Database Management System), making easier for the observation of a time Edit evolution and his corresponding version. The MMVE is a basic prototype for Edit data storage validation. All the tests and results achieved, proved almost all the goals proposed.
Key words: Temporal database, Versions model, time-evolving systems specification, Distribution Management Systems, Relational DataBase Management System, Supervisory Control And Data Acquisition.

1 Introduction

In DMS applications, different alternatives or versions of a Edit are kept in the database. Historically, first researches about versions were related to the areas of CAD (Computer Aided Design), CASE (Computer Aided Software Engineering), and SCM (Software Configuration Management) [8].

An Edit describes a model of an electric power distribution network in a period of time or from a certain point of view. Although some older design alternatives are stored as versions, not all the history of data changes is recorded. Important modifications may have occurred where related data are lost. The full history is only accessible if a temporal database is used.

A temporal data model specifies both static and time-varying aspects of the application. By definition, a temporal database must follow the principle that all data excluded by the user is kept in order to preserve the complete data history. In this context, presenting versions with their respective temporal aspects has the advantage of getting model information related to specific periods.

The purpose of this paper is to bring together both versioning and temporal features using a RDBMS. The database stores the versions of an Edit and, for

[1] The version of a electric distribution network model or simply a electric net model.

each version, the history of data changes. The model that we build to the project is called: **MMVE**. This model uses a regular RDBMS to store all the data, which keeps the system low-priced. Alternatively GDS and GIS are better commercial solutions but high-priced.

The remainder of the paper is organized as follows: Section 2 briefly motivates this approach vis-a-vis conventional system based on GDS or GIS and discusses the background of the work. Section 3 describes the MMVE system that was implemented. Section 4 presents results of the prototype. Section 5 presents conclusions and discusses future research.

2 Motivation and Background

A typical Power System consists of Generation, Transmission and distribution. The basic structure of Power Transmission and Distribution system covers a huge network consisting of wide range of Equipments, Feeders and Facilities. To efficiently manage utility operations different disparate systems exist and the role of each system is different and unique. Typical systems used in the Electrical Utility are SCADA[2]/DMS-GIS [2]. GIS is used to superimpose the complete electrical network assets from Generation to service point on top of the land base data.

2.1 Problem and Motivation

Energy Systems and Automation - A division of EFACEC[3] is developing a new DMS system over JFRK[4] which needs a new database support for electric net configuration models. The system must support a set of informations: electrical equipment; electrical characteristics; topology, among others.

EFACEC does not want to use an SCADA/DMS-GIS integration - it is too expensive and requires a permanent skillful team, so they specified a set of requirements [1] to validate an alternative solution:

1. **Performance and quality engineering requirements** - 5 messages per minute, is the minimum that the system is able to respond to operator calls. In an overload, the system is able to respond to a maximum of 500 calls per minute.
2. **Safety requirements** - Multiple operators can work simultaneously and make changes on Edits, with total control and safety.

[2] SCADA (Supervisory Control And Data Acquisition) is a category of software application program for process control, the gathering of data in real time from remote locations in order to control equipment and conditions.

[3] The EFACEC Group is the largest Portuguese organization in the field of electromechanics and electronics, with a strong presence in different international markets.

[4] The JFRK is a JAVA/XML Framework based on RMI/IIOP/SOAP protocols with capacity to link several disperses components using the Publish/Subscribe paradigm architecture. A Java BUS [9] implementation.

3. **Functional requirements** - creation of an empty Edit; an Edit based on another Edit (Historical and RealWorld Edit); only one operator can make changes to an Edit at the same time; all the changes must be confirmed by the operator and those changes must remain in the database; every changes can be rollbacked to the previous state; when an operator close the Edit session, all the changes not confirmed are removed from the database; only a completed, tested and closed Edit can replace the current RealWorld, in this case the RealWorld replaced passes to historical Edit state; an Edit can be removed from the database.

Problems with the conventional system. The main advantages of SCADA/DMS - GIS integration are improved network operation efficiency, reduced cost of data management, more consistent asset information distribution among the various users and finally, and perhaps most important, better end user services. The problem is to put together the two systems, because they do not speak the same language. A SCADA/DMS-GIS can be easily implemented once a CIM[5]-based integration system is in place. Nevertheless, to accomplisb those requirements, specialized teams and a development team to create tbe integration must be put in place, which astronomically increases the price of the solution.

Further Advantages of the MMVE Approach. MMVE offers an alternative strategy for recording DMS information. It resolves the problems of integration and produces a cheaper solution. The MMVE concept is realized with a relational DBMS that interfaces with a temporal database. MMVE offers several additional advantages:

1. **A temporal DBMS frontend to a relatioual DBMS** - by translating temporal SQL into standard SQL statements, it supports temporal functionality for a non-temporal relational DBMS;
2. **One development team to implement the system** - by using the company know-how and company development team resources. The advantage of this approacb is that the development team can continue to work with the RDBMS they know better, reducing the development time and project budget;
3. **Royalties or GIS utilization licenses are simply removed** - the advantage of removing those licenses makes the product cheaper and generates an economic solution for new potentials customers;
4. **Build on Java** - implemented in a multi-plataform language.

Some Disadvantages of the MMVE approach over SCADA/DMS-GIS systems. GIS is both a database system with specific capabilities for spatially referenced data as well as a set of operations for working with the data. MMVE

[5] International Standards - IEC 61970: Energy management system application programme interface (EMS-API) encompassing the Common Information Model (CIM).

is not a spatial database; GIS solutions have more than 30 years old in business and are improved every day. Moreover they are implemented in thousands of places and practically bug-free. MMVE is not a mature solution and it is not bug-free.

2.2 Related Work

It should by now be clear that the main of MMVE project is to create a cheaper and functional data storage solution for a DMS system. By using in this approach a temporal database frontend and a version management system, the goal can be achieved.

Temporal Database concepts [14] and Version management concepts [6] like some others works in the field are referencied on the dissertation book that support all this work [5].

What is being done in the field. Some software houses, especialized on Electric Power software, have developed simmilar products, replacing GIS by others in-house solutions. For the DMSGroup, the Windows platform and an RDBMS was adopted to develop their SCADA/DMS system [11]. All the electric facilities are saved in the RDBMS and controlled by specialized Windows applications; For the SNC-Lavalin Energy Control Systems company, the SCADA/DMS software [12] is based on the GEN-3 (product developed by them). The distribution network model is maintained in an Oracle RDBMS, and is complemented by a set of memory-resident real-time databases that are replicated to all of the servers and workstations on the local area network for optimal performance.

3 The MMVE System

This section presents the MMVE system . The MMVE project is a small part of the new SCADA/DMS system developed by EFACEC. The MMVE project is only a small component of the EFACEC DMS architecture, nevertheless a very important piece.

The MMVE is composed by TimeDB 2.0, a temporal database frontend linked to a Oracle 10g RDBMS and an Edit version management based on the schema version management system TVM2.

3.1 The temporal database

The evolution of an Edit is recorded in a temporal database. By using a temporal database we are able to visualize the Edit evolution over time. One advantage of this approach is the possibility for the operator to rollback the Edit to a previous state.

The parameters and the reasons to choose TimeDB v2.0 as the temporal database solution, can be found in our dissertation work [4]. The main characteristics of TimeDB 2.0 are:

- TimeDB 2.0 supports a temporal version of SQL called ATSQL2 [13];
- TimeDB 2.0 , however, is not a temporal DBMS itself but is a frontend to a relational DBMS;
- TimeDB 2.0 was implemented in Java and thus is platform independent;
- TimeDB 2.0 uses JDBC and thus can be used with many different DBMS;
- TimeDB 2.0 has a GUI and thus is easier to install and use;
- TimeDB 2.0 is optimised with respect to the creation of auxiliary tables;
- TimeDB 2.0 has a native call interface (TDBCI) which Java applications can use to execute ATSQL2 statements.

3.2 The Edit Version Management system

Each Edit is an independent version. The system must be able to manage three type of Edits: **RealWorld Edit** - the model version that contain the real state of the electric net; **Historical Edit** - an old model version, completed and closed that remains in the database for historical purposes; **Working Edit** - a working model version of an electric net (not yet finished). When an Edit is completed and closed, it can be the next RealWorld or be sent to others applications (for example: electric study).

The Edit version management is based on the schema version management system TVM2 [3]. TVM2 (Temporal Version Model 2) is a new approach of the original TVM [10] model (a temporal data model object oriented). TVM2 use states to alter the evolution of a schema. On our case the MMVE is like TVM2 but with some differences: MMVE is not Object Oriented and does not record changes to the schema model level (each time an object of the schema changes his attributes a new version is created with all the data associated). Instead MMVE use database schemes as repositories to record Edit versions (each schema is an independent Edit and the attributes of the objects never changes, only the data). Like TVM2 the MMVE model use states to change the evolution of the Edit (time and version).

Edit version management characteristics.

1. **List** for relationship between versions - one version is allowed to derive from another Edit version and can only be the source to another Edit;
2. **Explicit** for derivation version - the changes are controlled by the operator and not automatically by the system; asynchronous for data management - the requests and answers are made asynchronously;
3. **Edit version** for version management type - the changes happen only at the Edit level, for example, an operator may need to analyze a new electric net model and create a new Edit based on the RealWorld to study some power low scenarios;
4. **Partial** for version management mode - allow changes only to the current Edit;
5. **Multiple repositories** for storage alternative - each new Edit is stored in a new repository.

6. **Bi-temporal** for Edit version management type - transaction and validation timestamps shall be recorded on each Edit data change;
7. **Second derivation** for derivation form - the last version would not be the version that can suffer a new derivation. Any version (Edit) can serve as base for derivation, but only one version is considered the current version.

3.3 Requests and states classification

To better understand the MMVE mechanisms, this subsection import some definitions of states and event requests used in the MMVE system.

States. The Edit's evolution in MMVE is controlled by states, just like TVM2. We defined four Edit states: **Open** - the Edit is in edition mode. The operator can make changes to the Edit version. He can add, update or remove objects from it; **RealWorld** - this is the real and operacional electric distribution network model; **Close** - this is the state when a Edit is finished. An Edit "closed" can be the next RealWorld or can be tested on Power load scenarios; **Historic** - an RealWorld state returns to the historic state when it is replaced by a new "closed" Edit.

Only **Historic** and **Close** state Edits can be removed from the system. Edit at this stage are always in visualization mode: the operator can review step by step the evolution of the Edit however he can not make any changes. An new Edit can be created from Edits at the Historic, Close or RealWorld state.

Requests. The MMVE is able to understand several "requests" (XML events that translates actions made by the SCADA/DMS system's operators). In the JFRK middleware events travels all around the SCADA/DMS system, however only recognizable events (events with topics that MMVE can subscribe) are imported to the MMVE framework. Their are three requests types:

1. **Metadata requests** - requests that simply query the MMVE metadata[6]. These requests return always a result from the MMVE system. For example: the RealWorld Edit ID; the list of all Edits with historic state; the last timestamp that an Edit was saved (Open state); etc.
2. **Simple requests** - requests that requires actions and operations on the temporal database: update an Edit object; drop an Edit object; rollback a Edit version to a previous state (undo requests); etc.
3. **Special requests** - Requests that create, merge or delete temporal database repositories (each repository is an Edit version). For example, when an operator is creating a new Edit, the MMVE searchs the Edit ID in the metadata (internal metadata request); if the new Edit ID is valid, it searchs for the derivation Edit (an new Edit is always derived from an Edit at Historic, Close or RealWord state). If the derivation Edit is found then a new repository is created in the temporal database with all the derivation Edit data.

[6] MMVE metadata is a special repository of information that record special information like: Edits state, transactions time and validation time of a Edit session.

3.4 The MMVE architecture

The MMVE architecture (builded around three main components) is based on a multi-thread and multi-layer system implemented in Java with a Backend Process Manager thread [7] (to deliver operators calls). Those technics set performance, scalability and efficiency to the system. By using the TimeDB API, it was easy to implement the temporal database with minimum work. The big work remained in the Edit version management system, build from the scratch.

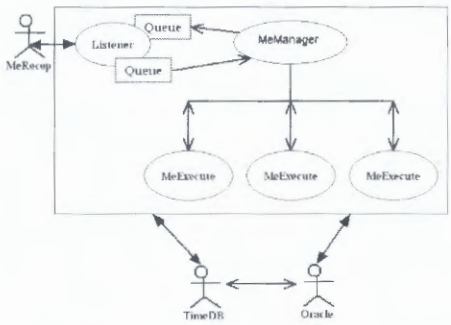

Fig. 1. MMVE Components Model interaction

MeRecep. It is responsible to translate and validate the XML events received from the JFRK as operator requests. It is responsible to send back to the JFRK the answers of the MMVE system as results of the operator requests. This component use two queues to deal with the operator requests and the MMVE responses.

MeManager. It is an agent that distribute work among multiple "MeExecute" processes (Figure 1 on page 7). When a new request arrives (a new request is placed in the MeRecep requests queue) it take care of the request by sending it to a MeExecute process. If all MeExecute processes are taken down, it may create more "workers" or wait a little while before doing that. By default only five workers are created at the MMVE startup. A similar process is made to send back MMVE results information to the operator, in this case the answers are put in a special queue (MMVE responses queue) .

MeExecute. It is the main process of the MMVE system. When the process receives a request, a set of actions are taken on it: first, the request is analyzed and break into smaller pieces. The pieces are sorted in a logical order and executed in sequencial mode. Any error message returned, breaks immediately the sequence of commands.

Some others capabilities builded around it: the capability to transform requests on ATSQL2 instructions (if necessary); the capability to open and close TimeDB sessions; the capability to update the MMVE metadata (if necessary).

Example of a new Edit creation

Example 1. [CREATE]{NAME::"Teste01"; ID::"15"}

The example above is an MMVE message request that represents the creation of a new Edit. The syntax of the message was firstly translated from component MeRecep.

The next commands are the result of the "breaking down" of the request message (returning a logical sequence of transactions):

- (01)[NOT_FIND_EDIT]{NAME::"Teste01"};
- (02)[FIND_EDIT]{ID::"15"};
- (03)[NOT_OPEN_EDIT]{ID::"15"};
- (04)[CREATE_EDIT]{NAME::"Teste01"};
- (05)[CREATE_TIMEDB]{NAME::"Teste01"};
- (06)[MERGE_TIMEDB]{NAME::"Teste01"; ID::"15"};

Every transactions in the sequence should return an valid value ("TRUE"). If a transaction return an invalid value ("FALSE") an error is immediatly generated and the request operation is aborted. Only a sucessfull transaction sequence can "commit" the changes in the database, otherwise when the process is aborted an internal database rollback is automatically taken.

4 Tests and Results

This section presents a quick overview of results and tests executed among the MMVE building process.

Performance tests with request/answers tests were the tools used to validate the system. We tested the system in a closed environment, well defined, with no exterior interference.

4.1 Test process

To record the system performance, we used the "black box" technique where a set of test cases under the form of XML are executed directly in the MMVE system. Automatically, a log is generated, with the execution time. The system answers are saved as XML files. After answer validation (manual validation), the next test is executed and so on.

When a completed scenario (a logic sequence of requests) of test cases are validated, a benchmark with all those tests is made. To validate the results obtained we realized a large number of tests and the final result is an average of those results.

For tests purpose we create a default Edit version (temporal database repository) with 500.000 rows among 23 objects.

4.2 Final Results

The functionality of the system was proved during the tests. All proposed functionalities were validated. Some bngs and problerms was found along the tests.

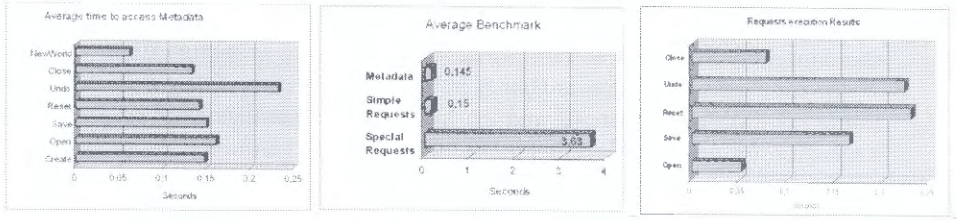

Fig. 2. Benchmarks

For performance purpose two main goals was defined: in a normal day the system may support 5 calls per minute (12 seconds per request), however in a bad day the system may snpport a maximum overload of 500 calls per minute (0.12 seconds per request).

Analyzing the benchmark resnlts (2) of the three request types, the results show:

1. Benchmark metadata requests results - the average time was: 0.145 seconds per request. This value represents 414 requests per minute. The system reached almost 83% of the initial goal. It is a good result.
2. Benchmark simple requests results - the average time was: 0.15 seconds per request. This value represents 400 requests per minute. The systeın reached almost 80% of the initial goal. It is a good result.
3. Benchmark special requests results - the average time was: 3.63. This value represents ~16 requests per minute. The results reflects the overload of an request of this kind. It demands several operations and among others operations: creation of the temporal database repository; copy of an repository to the new repository; etc.

5 Discussion and future work

This paper has presented work that spans the fields of temporal database, Database version management, and Distribution Management System. Our research was motivated in part by the realization of a simpler and controlled system that can reduce costs. Electric companies spent high resources to implement an SCADA/DMS-GIS integrated system.

With this product we are able to supply an SCADA/DMS data storage solution at a lower price. MMVE offers a software alternative to a SCADA/DMS-GIS

integration solution. It is not a GIS product and does not want to be one, however temporal and version management characteristics was implemented.

In a relatively short period of time we have successfully implemented the MMVE framework with a set of working functionalities.

The results are satisfactory and valid the approach used on this work. Only the performance tests did not fulfill their initials goals, however we were closed.

In future work we will endeavour to develop a more efficient system to support the complete collection of functionalities with faster response time.

In conclusion, it appears that a SCADA/DMS integrating a RDBMS with a temporal database frontend and an Edit version management system could be a solution for smaller electric company with fewer resources.

Acknowledgments. This research was made possible by the creativity, dedication, and cooperation of Marsh Dave, who specified the system requirements. We thank Silva Pedro and Rodrigues Alberto for their many important contributions. The work was partially developed on the Energy Systems and Automation premises, a division of EFACEC.

References

1. Dave A.Marsh. Edits in a relational environment. *EFACEC,SE: Brief Technical Description*, pages 1–35, Março 2004.
2. International Conference and Exhibition on Electricity Distribution. *Fourteenth International Conference on Electricity Distribution, Cired 97.* Inspec/Iee, softcover edition, 1997.
3. Renata de Matos Galante, Adriana Bueno da Silva Roma, Anelise Jantsch, Nina Edelweiss, and Clesio Saraiva dos Santos. Dynamic schema evolution management using version in temporal object-oriented databases. In *DEXA*, pages 524–533, 2002.
4. Manuel Claudio Magalhães Freire and Ademar Manuel Teixeira Aguiar. Temporal database analysis: Version management on an relational model with aplication on an electric network configurator. *Electric nets configuration, Electric nets Management, SCADA/DMS, Dissertation, MEI, Porto : [s.n.] 2005*, pages 82–86, 2005.
5. Manuel Claudio Magalhães Freire and Ademar Manuel Teixeira Aguiar. *Version Management based on a relation model with aplication on a Electric Network configurator.* Master technical thesis, Engineering Faculty, Porto University, Department of Electrical and Computer Engineering, 2005.
6. Fabio Grandi and Federica Mandreoli. Odmg language extensions for generalised schema versioning support. In *ER (Workshops)*, pages 36–47, 1999.
7. Edward Harned. The backend process manager. In *Multi-Threading – The Next Level*, November 2002.
8. Golendziner L.G. Um modelo de versões para bases de dados orientados a objectos. *CPGCC da UFRGS, Tese de Doutoramento*, 1995.
9. Dave Marsh and Paulo Viegas. The bus architecture. In *Proceedings of the 2001 Porto Tech Conference*, September 2001.
10. Mirella Moura Moro, Nina Edelweiss, and Clesio Saraiva dos Santos. Temporal versions model. In *UFRGS - Instituto de Informática*, pages 1–15, 2002.

11. Novi Said. Dms software: Windows for distribution networks. *DMS Group*, pages 1–21, 2006.
12. SNC-LAVALIN. Elektro gorenjska - scada/dms. *DMS Group*, 2003.
13. Richard T. Snodgrass, Michael H. Böhlen, Christian S. Jensen, and Andreas Steiner. Adding valid time to sql/temporal(change proposal). In *ANSI X3H2-96-501r2, ISO/IEC JTC1/SC 21/WG 3 DBL-MAD-146r2*, November 1996.
14. Abdullah Uz Tansel, James Clifford, Shashi K. Gadia, Arie Segev, and Richard T. Snodgrass. *Temporal Databases: Theory, Design, and Implementation.* Benjamin/Cummings, 1993.

Traffic Control on a Multi Service Edge Device

Joana Urbano

Centre for Informatics and Systems of the University of Coimbra
{stmaria@dei.uc.pt}

Abstract. Small and medium organizations are increasingly using the Internet as the communication vehicle for data and interactive streaming traffic (eg., streaming VoIP flows). As telematic applications evolve, these organizations also have a crescent need in differentiating and protecting traffic of high priority users, services or machines. Though there exist traffic control devices dedicated to this task, they are proprietary, rather expensive and complex to manage. Organizations need new solutions that are able to offer several IP services in one device, including traffic control services. This paper presents the traffic control system of Edge Device, a Linux multi-service router. More particularly, it proposes a DiffServ-based traffic control solution that seamless integrates with other IP services and that is able to differentiated traffic in both upstream and downstream directions, by modeling each interface to a value slightly lower than the link capacity and by creating different traffic classes at both directions. Testing performed on the prototype proved that our approach is successful in differentiating traffic and in protecting critical, high priority traffic. Results also showed that using a rudimentary TCP rate control mechanism at the downstream direction helps protecting critical flows at the expense of a slightly degradation of less priority service performance.

1. Introduction

Small and medium size networks with access to the Internet are each day more common in SOHO (Small Office/Home Office) environments, comprising domestic and small to medium size organizations. Inherent to the spread of these networks is the search for small access devices that allow the connection to the Internet in a very user friendly way but with a growing number of offered services, comprising basic IP functionalities (e.g., DNS, DHCP, SMTP and NAT), security and traffic control services. Traditionally, these services are available in dedicated, proprietary, and often-patented devices, with complex integration and high cost. More recently, new products appeared in the market that integrate several services in a single device, as is the case of 6WINDGATE from 6WIND [1], Access Point from Lucent Technologies [2] and ERX Edge Router from Juniper Networks [3]. However, these products still present some complexity at management and still have medium to high cost. To overcome an open space in the market, Critical Software and the LCT Lab from University of Coimbra developed a Linux, multi-service edge device's prototype, aiming at offering basic and advanced IP services at low cost and with trivial management requirements. This paper presents one of the key components of this device, the traffic

control system. Particularly, it presents a solution based on the DiffServ specification that is able to model upstream and downstream traffic in a single Linux router that is also an authentication, NAT (network address translation), firewall, service provider and application gateway machine.

1.1 Traffic Control in an Edge Router

In an edge router accessing the Internet, it is important to control the bandwidth of traffic that leaves the LAN and enters the ISP link, i.e., the upstream traffic. In the example of a 128 Kbps upstream ADSL link connecting a medium size enterprise, aggregate uploads that exceed this rate will fill the ADSL modem (generally FIFO) queues, either leading to random packet drops or, in the presence of big FIFO queues, to values of latency far too high for interactive traffic. A better approach would be to conditioning the upstream traffic in the edge device up to somewhat less than the uplink bandwidth, and to prioritize this traffic according to its importance to the organization. In our approach, we will differentiate upstream traffic at the edge device by classifying it into one of the following categories: *Premium*, suitable for critical traffic, *Olympic* (*Gold*, *Silver* & *Bronze*), for somewhat less critical traffic, and *Best-Effort*, for the remaining traffic. These categories would be implemented using DiffServ classes, and the global interface would be shaped to a value smaller than the interface capacity, dynamically calculated based on a proprietary algorithm. At the *Premium* level, we further increase the separation of bandwidth, using the concept of *pipes* of individual, per IP, flows of traffic.

Looking at the downstream side, the problem is somewhat different, in that edge routers do not have effective control in what is send do the LAN. Downloads at a medium size enterprise, or even at home, easily exceed the downstream link capacity, leading to the queuing, delaying and possible dropping of packets at the ISP edge router, regardless the relative priority of the traffic flows. One possible approach for overcoming this situation would be to take profit from TCP rate control mechanisms by selectively dropping packets or even mangling the advertised receive window, at the edge device side, in the hope that sender side slows down the transmission. Both solutions present evident drawbacks, as mentioned in literature. [1] In our prototype, we are going to shape downstream traffic allowing for only two classes of priority, in a compromise between protecting critical traffic and, at the same time, minimizing the dropping of packets that already passed through the input interface.

With this strategy, we expect to achieve a solution that is simple to implement and that adequately integrates with the other modules of Edge Device. At the same time, we expect that our solution will be able to protect critical, more priority traffic without a significant damage on the overall interface capacity performance. Moreover, we expect that modeling traffic at the downstream in two classes, as a rudimentary implementation of TCP rate control, would be efficient in protecting priority traffic.

---

[1] By selectively dropping TCP packets we drop packets that have already consumed downstream link bandwidth. One the other hand, some authors claim that mangling the advertised *receive window* of TCP ACK packets works well on an individual flow basis, but has a negative impact on the system performance in a congested, mixed traffic link (see, e.g.,[4]).

This paper is organized as follows. This section introduces the paper. The next section presents the traffic control system of the Edge Device, pin pointing some problems that arise on the integration of multiple IP services into a single Linux machine, and detailing our solution for upstream and downstream traffic control. The paper proceeds with a description of the testing done over Edge Device's traffic control. Last section presents and discusses the results obtained, pin points questions that shall be worked as future work and concludes the paper.

2. Traffic Control System

This section starts with a brief description of the key features of Edge Device that directly interact with the traffic control system. We then conclude the section presenting the traffic control module of the router prototype. Edge Device is a firewall, NAT router operating at the edge of a LAN, developed under Linux 2.4.18, for wired and wireless environments. All users of the LAN, including wireless users, must authenticate before they connect. Edge Device is also a service server and an application gateway, integrating a Squid server for HTTP traffic, operating in transparent mode. In order to optimize the generally scarce resources at the access border, upstream and downstream traffic is classified by user into different DiffServ [5] traffic classes and prioritized according to its importance to the organization. All system resources are managed according to a policy based management.

2.1 Traffic Control Solution

Upstream traffic can be shaped at three different locations on the OSI layer 3 packet traveling. [2] In our solution, we chose to manage traffic at QoS Egress[3]. On this approach, we used some artifacts to overcome the effect of NAT and broken TCP connections issues. In fact, the transparent redirection of HTTP traffic to Squid at PREROUTING and the realization of NAT at POSTROUTING pose some difficulties to differentiating traffic per source IP at the QoS Egress, since the addresses seen by the traffic control would be the Edge Device address and the NAT address, respectively. To overcome Squid drawback, we decided to DSCP-mark HTTP traffic at Squid using the *tcp_outgoing_dscp* functionality. On the other hand, we forced NATed traffic that does not pass through Squid to be previously marked in the PREROUTING chain (using iptables) in a per user basis. This way, it is easily seen

[2] Linux traffic control can be done at QoS Ingress, QoS Egress or at the virtual device IMQ ([6]), called on PREROUTING and POSTROUTING Netfilter hooks. QoS Ingress uses a rather limited classless queuing discipline that only filters traffic, using a token bucket algorithm. On the other hand, IMQ modeling at PREROUTING or POSTROUTING is done in the exact same way as at QoS Egress, exploring the full capabilities of Linux queuing disciplines, classes and filters. As the IMQ module at POSTROUTING immediately precedes QoS Egress, it cannot perform any better than QoS Egress, having the drawback of introducing one more queue to the traffic control system ([7] [8]).

[3] The Kernel packet travel diagram can be found at [9].

that HTTP traffic that goes through Squid is differentiated and shaped at QoS Egress based on the DSCP field, and the remaining NATed traffic is differentiated and shaped by iptables mark.

As we mentioned before, we used a DiffServ approach to implement the upstream classes *Premium, Gold*, *Silver*, *Bronze* and *Best-Effort*. Our solution used a combination of the *dsmark* algorithm ([10]), as the root of the hierarchy of queuing disciplinators (*qdisc*), CBQ ([11]), TBF ([12]) and SFQ ([13]) for shaping at intermediary (CBQ) and leaf (TBF and SFQ) qdiscs, respectively, and GRED ([14]) for DiffServ drop precedence issues. *Premium pipes* were implemented by policy rules provided by these qdiscs. This overall implementation of upstream traffic control was ruled by PHB EF ([14]) (*Premium* service), PHB AF ([15]) (*Olympic* services) and *Default* PHB ([5]) specifications and by project generic requirements of sharing excess bandwidth between *Olympic* and *Best-Effort* services and total separation of bandwidth between these and the *Premium* service.

Downstream path is somewhat simpler than the other way out, as we chose to classify and shape traffic at the virtual device IMQ.[4] Moreover, we considered only two classes, one for critical traffic and the other one being a best-effort, catch-all class. Once again, we used a stratagem, which consisted in the registration of IMQ after NAT, at PREROUTING, using a modified version of Patrick McHardy's *imqnat patch* ([16]). By changing IMQ and NAT positions at PREROUTING, we can guarantee that traffic arriving at IMQ is already de-NATed and so can he differentiated by source IP. By this time, though it is still not possible to know the real source IP of non priority HTTP packets, as IMQ precedes INPUT chaiu (where packets are redirected to Squid), this traffic is naturally classified as best-effort.

Downstream classes are implemented by CBQ qdisc for root and intermediary nodes of the hierarchy of queueing disciplinators, and SFQ qdisc for leaf notes. *Premium* and *Best-Effort* services are bounded, meaning that they cannot share excess bandwidth between them.

3. Testing the Prototype

The evaluation of the traffic control system was planned in three different phases: the evaluation of queueing disciplinators algorithms and inherent configuration parameters; conformance testing, for evaluating the adequacy of our solution to the system requirements; and performance testing, for obtaining metrics that allow the evaluation of the efficiency of the proposed solution and, at a finer granularity level, of each one of the offered services. The first two phases were evaluated at earlier stages of the project and are out of the scope of this paper. Until the end of this section, we briefly describe the testbed used in our performance tests.

[4] We did not choose QoS Egress option this time because it implies the shaping of traffic – that can be severe, when concerning best-effort packets – that is already at the Squid cache, thus, reducing the potential benefit of Squid use.

3.1 Testbed Configuration

Figure 1 shows the equipment used in the testbed for performance testing. EP1, EP2, EP3 and EP4 served as internal LAN *endpoints*, and were connected to Edge Device through a switch. EP5 emulates a WAN *endpoint*. *WAN Shaper* is a Linux device that emulates a WAN router that shapes downstream traffic to 512 kbps and upstream traffic to 128 kbps, allowing traffic bursts.

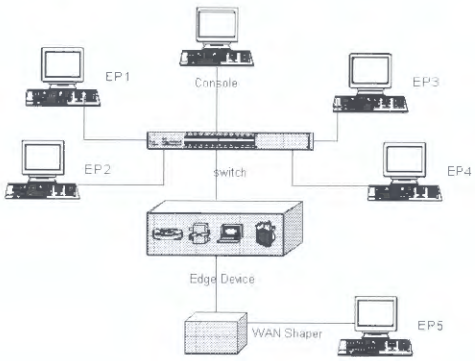

Fig. 1. Testbed configuration for Edge Device performance testing

We used NetIQ Chariot v4.1.934 ([17]) console and endpoints to emulate real traffic and measure performance parameters (eg., throughput, response time and packet loss) of the prototype. We chose three particular scripts, *IPTVa* for emulating Cisco Systems IP/TV application, with both default and variable packet size; *voip.g711* for emulating VoIP with G.711 codec and RTP protocol, with standard packet size; and *fileSndl* for emulating TCP long distance file transfers, using packets of 64, 512 and 1518 bytes (for space reasons, TCP tests are out of the scope of this paper). Each test ran for several iterations in fixed time intervals of 2 minutes. The Edge Device under test was compiled with full netfilter and QoS support, and there were installed iptables-1.2.7a package and IMQ, h323-conntrack-nat and imqnat patches. It was also installed version v.2 of Squid.

3.2 Performance Tests

Performance tests were done after the tuning of the traffic control system. We intend to compare QoS parameters results (throughput, packet loss, response time and jitter) of *Premium* and *Olympic* services with the results obtained without traffic control, in order to evaluate the benefits of our solution in terms of traffic differentiation and to measure the possible performance degradation associated with this differentiation.

3.2.1 *Premium* Service (Upstream)

Table 1 shows the UDP test suit aimed at evaluating *Premium* service performance, either with or without traffic control.

Table 1. Configuration of *Premium* service test suite

Test	*Script*	Proto	Pac. Size (*bytes*)	P1/P2/P3/P4 (%)	P1 (%)	P2 (%)	P3 (%)	P4 (%)
T1	IPTVa	UDP/RTP	[64-1460]	25/25/25/25	100	100	100	100
T2				25/25/25/25	150	150	150	150
T3				5/15/30/50	150	150	150	150

In all the tests, we have generated four *Premium* traffic flows. These flows should get the same treatment in T1 and T2, and they should be differentiated in T3, where P1, P2, P3 and P4 should receive 5%, 15%, 30% and 50% of all the bandwidth reserved on the output interface, respectively. Table 4 presents the results obtained in this test suite.

3.2.2 *Olympic* Service

Second test series evaluates the efficiency of the *Olympic* service in traffic differentiation, and compares the service performance with the one obtained without traffic control. Table 2 presents the test suite configuration.

Table 2. Configuration of *Olympic* service test suite

Test	*Script*	Proto	Pac. Size (*bytes*)	G1 (%)	S1 (%)	B1 (%)	BE1 (%)
T1	IPTVa	UDP/RTP	[64-1460] (Poisson)	100	100	100	100
T2				150	150	150	100
T3				50	100	150	250

In the test suit, we generate four different flows: *Gold* (G1), *Silver* (S1), *Bronze* (B1) and load flow *Best-Effort* (BE1). T1 and T2 evaluate the performance of each one of the *Olympic* classes, when the load generated by each flow is, first, one third of *Olympic* bandwidth and, then, 150% of this value. T3 simulates a more realistic use of *Olympic* and *Best-Effort* services, with less important flows generating more traffic and vice-versa. Table 5 shows the results obtained with UDP traffic.

3.2.3 Integration of All Services

Last test suite includes all services from input and output interfaces. Table 3 shows the test configuration.

Table 3. Configuration of "All Services" test suite

Flow	*Script*	Proto	Pac. Size (*bytes*)	Load (kbps)	Source	Target
Premium – Upstream	IPTVa	UDP/RTP	1278	93	EP1	EP6
Premium – Downstream	IPTVa	UDP/RTP	1278	93	EP6	EP1
Gold – Upstream	FileSndl	TCP	512	25	EP2	EP6
B. Effort – Upstream	FileSndl	TCP	512	25	EP3	EP6
B. Effort – Downstream	FileSndl	TCP	512	512	EP6	EP4

In this suite tests, output interface has 128 kbps, 80% of which were dedicated to *Premium* service, and the remaining 25.6 kbps were shared between *Olympic* and *Best-Effort* services. The input interface was configured with 512 kbps, where 30% of it was assigned to the *Premium* service. Upstream and downstream *Premium* flows simulated a streaming multimedia connection, where the other flows, in both ways, saturated the correspondent link capacities. The results of these tests are shown in Table 6.

4. Results and Discussion

This section presents and discusses the results of the test suits described below.

4.1 *Premium* Service (Upstream)

Table 4 shows the packet loss and jitter results for the first series of tests. As can be seen, Edge Device was efficient in equally treating the four flow tests (T1 and T2), as it was expected. The analyses of Wan Shaper counters also revealed that without traffic control, packets are drop in a rather random way.

Table 4. *Premium* service – Packet loss and jitter results (T1', T2' and T3' relate to tests done with traffic control on)

Loss				
Test	**P1** (%)	**P2** (%)	**P3** (%)	**P4** (%)
T1	0,29	0,00	0,63	0,00
T2	29,19	38,38	39,13	28,42
T1'	0,00	0,00	0,00	0,00
T2'	34,08	33,77	33,73	34,02
T3'	84,41	58,77	19,79	0,00

Jitter				
Test	**P1** (ms)	**P2** (ms)	**P3** (ms)	**P4** (ms)
T1	37,733	31,067	32,733	34,133
T2	34,574	34,442	29,963	28,293
T1'	32,000	29,467	35,733	31,000
T2'	45,218	46,866	38,977	39,093
T3'	41,750	33,333	24,500	21,409

Throughput results are presented in a graphical way in Figure 2.

We can see from the graphics that P2 performance was very poor when traffic control was off, and that the four flows received identical treatment when traffic control was on. Though throughput benefits using traffic control are evident, we can see from Table 4 that there have been a slightly performance degradation in terms of jitter associated with those benefits (T2).

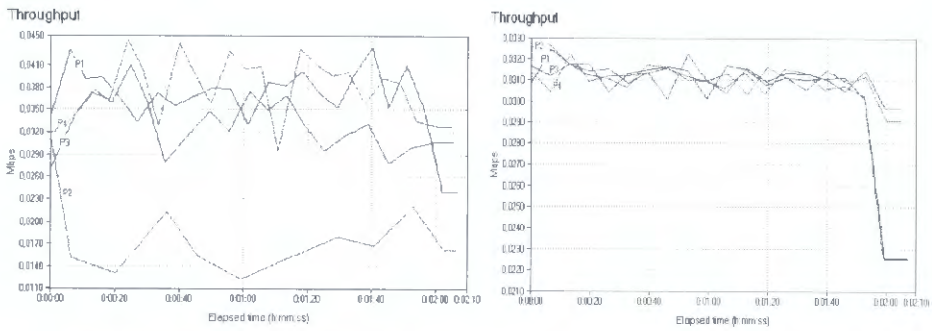

Fig. 2. *Premium* service – *Throughput* results (left: T2, without traffic control; right: T2', with traffic control)

Figure 3 shows the throughput results obtained in T3.

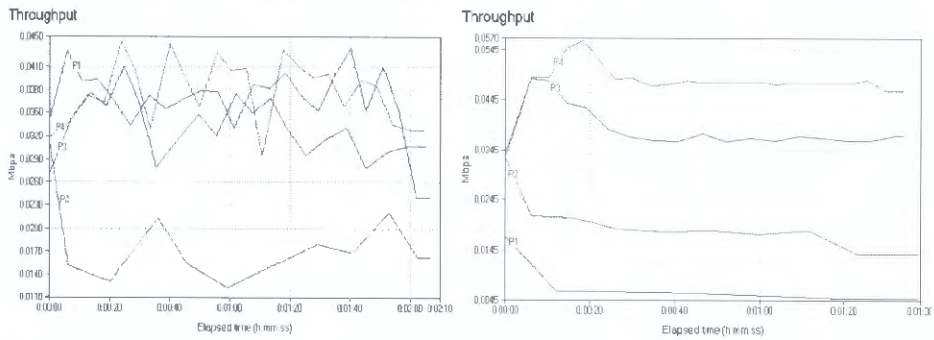

Fig. 3. *Premium* service – Throughput results (left: T2, without traffic control; right: T3', with *pipes* of different capacities)

Figure 3 shows that the bandwidth differentiation between different *Premium* flows was clearly effective when traffic control was on. Results obtained with TCP traffic (not shown in the paper because of space issues) confirm the effectiveness of the *Premium* service when differentiating flows with different priorities.

4.2 *Olympic* Service

Table 5 shows the packet loss and jitter results for the second test suite. T1 results shows that packet loss values were similar in both the presence and the absence of traffic control. There can be also seen that the activation of traffic control slightly increased jitter in G1 and S1, being more evident in B1 and BE1. T2 results are quite more interesting. When traffic control was off and output link capacity was overloaded, packets were dropped at WAN Shaper (that simulates an adsl/cable modem in the LAN → WAN direction) in a rather random way. *Olympic* classes behaved in random way, with G1 obtaining worse performance than S1 and B1, in terms of throughput and jitter.

Table 5. *Olympic* service – Packet loss and jitter results (UDP)

Loss				
Test	**G1** (%)	**S1** (%)	**B1** (%)	**BE1** (%)
T1	0,00	0,00	0,00	0,00
T2	38,93	30,09	18,19	25,19
T3	27,85	27,92	22,76	19,18
T1'	0,00	0,00	0,28	0,00
T2'	0,67	21,48	77,02	0,00
T3'	0,00	0,00	0,33	49,50

Jitter				
Test	**G1** (ms)	**S1** (ms)	**B1** (ms)	**BE1** (ms)
T1	31,200	35,867	32,867	39,000
T2	41,071	39,375	31,579	35,889
T3	40,167	41,636	32,500	29,667
T1'	37,333	37,800	43,533	60,583
T2'	45,957	57,889	75,167	57,750
T3'	46,500	41,267	39,739	119,000

Figure 4 shows throughput results obtained at T2, with and without traffic control. We can see that *Olympic* drop precedences worked as expected when an excess of 50% of the capacity of *Olympic* class was generated. In the scenario, *Gold* service obtained better results than *Silver* traffic and this one was better protected than *Bronze* service, in terms of packet loss and jitter. We should also note that there was a clear separation between *Olympic* and *Best-Effort* bandwidth, as expected by the project requirements.

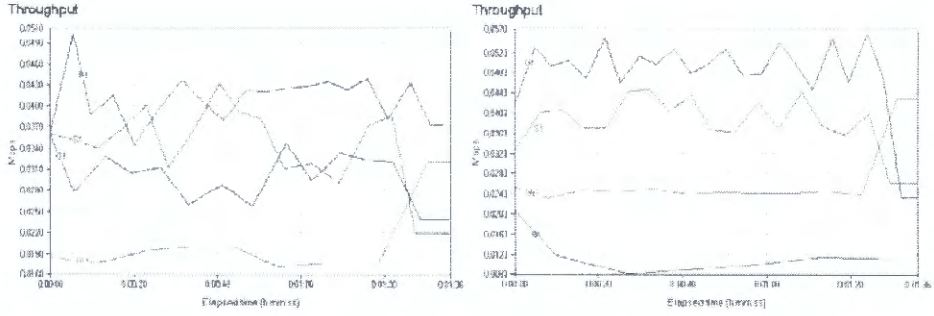

Fig. 4. *Olympic* and *Best-Effort* services –*Throughput* results (left: without traffic control (T2), G1 traffic is not protected; right: with traffic control (T2'), Olympic classes work as expected)

T3 results showed an almost optimal behaviour of *Olympic* service, with only *Bronze* service experiencing a residual lost of information. In Figure 5 (T3'), we can clearly see the point when *Olympic* service lended bandwidth to the *Best-Effort* service, when it had it in excess.

Results obtained with TCP traffic were quite similar in terms of throughput and are not shown here. In the same way, we do not show here the results concerning response time, due to shortage of paper space. The results obtained on this QoS parameter showed that there was no evident performance degradation when traffic control was turned on, and flows with higher priority received a slightly better service than less priority ones.

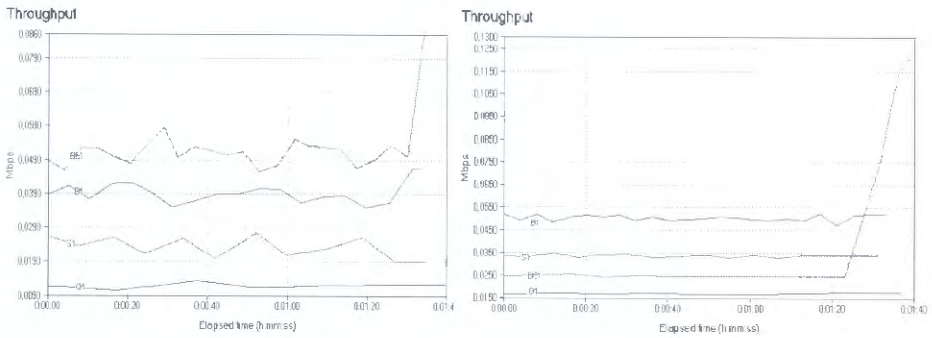

Fig. 5. *Olympic* and *Best-Effort* services – *Throughput* results (left: T2, without traffic control; right: T3', with traffic control)

4.3 Integration of All Services

Table 6 shows the results obtained in the third series of tests.

Table 6. Integration of all services – Results

Test	Premium (out)			Premium (in)			Gold		BE (out)		BE (in)	
	Tput (kbps)	*Loss* (%)	*Jitter* (ms)	*Thput* (kbps)	*Loss* (%)	*Jitter* (ms)	*Tput* (kbps)	*RT* (ms)	*Thput* (kbps)	*RT* (ms)	*Thput* (kbps)	*RT* (ms)
With TC	86	5,98	30,346	93	0,00	5,244	21	37,714	13	59,716	114	7,040
With/ TC	93	0,00	11,829	93	0,00	0,134	20	39,817	5	174,208	5	176,720

A first glance at the results shows that our solution was very effective in protecting streaming flows and in differentiating output traffic. However, downstream *Best-Effort* flow showed an incomprehensive bad result, hardly reaching 2% of the bandwidth available to this service. After some problem analyses, we concluded that ACK packets belonging to the problematic flow should have been dropped at the output interface. As a result, we promoted this flow to *Gold* class, first, and then to a *Premium pipe* with 8% of the overall *Premium* bandwidth. Results showed that the utilization rate of the flow increased to 12.8% and 72%, respectively. Figure 6 shows the throughput results of the testing.

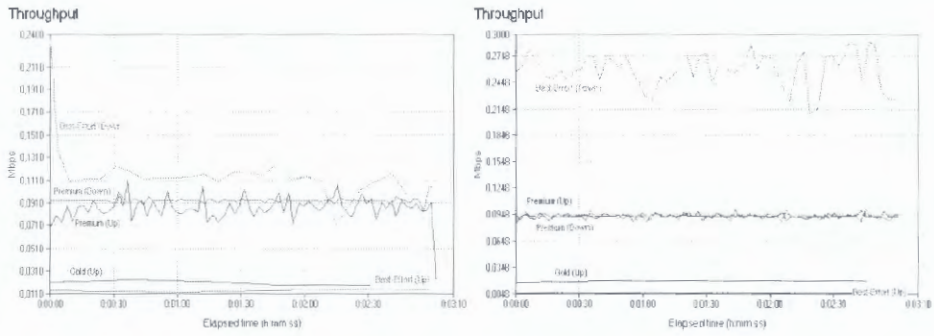

Fig. 6. Integration of all services – Results (left: without traffic control; right: with traffic control, Premium flows are protected and upstream Best-Effort flows reach their best results)

4.4 Discussion of Results and Future Work

The results obtained in the performance testing of the Edge Device revealed that our solution is effective in differentiating traffic in different service classes and in protecting high priority flows. Moreover, the results showed that these benefits generally came with non significant degradation in bandwidth utilization or jitter values. On the other hand, and though not explicitly shown in this paper due to space issues, the shaping of downstream link capacity and the distribution of incoming traffic in two services classes conferred major guarantees to high priority traffic, at the expense of some *Best-Effort*'s downstream bandwidth degradation. In face of these results, we can conclude that Edge Device traffic control system is well suited to small to medium enterprises that increasingly use the Internet as the communication vehicle for interactive streaming traffic (eg., streaming VoIP flows), and where there is a need to protect flows from high priority users, services or machines.

Tests also showed that a second major result of this project was the simplicity of the solution and the easy integration of the traffic control system with the other services of the router (as the firewall, authentication, content caching and application gateway services). After a short phase of integration and system testing, the Edge Device is now successfully working in production environment, being responsible for the control of all the traffic that enters and leaves Critical Software premises. We think that this was an important accomplishment and a major contribution to the academic and scientific community, since such integration is not trivial and public information on the subject was until now rather scarce or inexistent.

Finally, taking into account the results obtained and those expected, we would like to derive some considerations. First, we think that future versions of the Edge Device should be able to differentiate not only by user and IP, but also per service. One the one hand, this feature is present in top level traffic control devices (eg, Packeteer's Packet Shaper [18]), and on the other hand our prototype is already prepared for a simple implementation of the proposed extension. Secondly, we think that it would be critical to implement a mechanism that is able to protect ACK TCP packets. This can be done by using LARTC *u32* filter for the capture of packets and by allocating a pri-

ority traffic channel in both directions for this kind of traffic. Finally, the results obtained for downstream traffic suggest that the *Premium pipe* concept should be extended to downstream traffic, in order to adapt TCP flow control characteristics to different priorities flows that arrive at LAN.

5. Acknowledgements

This work was partially funded by Agência de Inovação, in the aim of the EDGEDEVICE project. We would like to thank Edmundo Monteiro from LCT and António Alves and António Raposo from Critical Software SA for the support given throughout the Edge Device project.

6. References

1. 6WIND, 2006, *6WIND Homapage* [online]. Available from: http://www.6wind.com [Accessed 01 February 2004]
2. Lucent Technologies, *Lucent Technologies Homepage* [online]. Available from: http://www.lucent.com [Accessed 01 February 2004]
3. Juniper Networks, *Juniper Networks Homepage* [online]. Available from: http://www.lucent.com [Accessed 01 February 2004]
4. Check Point, *Check Point Homepage* [online]. Available from: http://www.checkpoint.com [Accessed 01 February 2004]
5. X S. Blake et al., "An Architecture for Differentiated Services Framework", RFC 2475, December 1998
6. P. McHardy, *The Intermediate Queueing Device Homepage* [online]. Available from: http://luxik.cdi.cz/~patrick/imq/ [Accessed 01 February 2004]
7. LARTC, *Linux Advanced Routing & Traffic Control Homepage* [online]. Available from: http://www.lartc.org/ [Accessed 01 February 2004]
8. Netfilter, *Netfilter Homepage* [online]. Available from: http://www.netfilter.org/ [Accessed 01 February 2004]
9. Stef Coene, *Stef Coene Homepage* [online]. Available from: http://www.docum.org/ [Accessed 01 February 2004]
10. W. Almesberger, A. Kuznetsov, and J.H. Salim, "*Differentiated Services on Linux*," Proc. GlobeCom, pp. 831-836, 1999
11. S. Floyd and V. Jacobson, "Link Sharing and Resource Management Models for Packet Networks", ACM/IEEE Transactions on Networking, V. 3 (4), pp. 365-386, 1995
12. LARTC, *LARTC manpages*, [online]. Available from: http://www.lartc.org/manpages/ [Accessed 01 February 2004]
13. P. E. McKenney, "Stochastic Fairness Queueing", in *Proceedings of INFOCOM*, San Francisco, CA., June 1990
14. V. Jacobson et al., "An Expedited Forwarding PHB", RFC 2598, IETF, 1999
15. J. Heinanen, "Assured Forwarding PHB", RFC 2597, IETF, 1999
16. P. McHardy, *The Imqnat Patch Homepage* [online]. Available from: http://mailman.ds9a.nl/pipermail/lartc/2002q3/004725.html [Accessed 01 February 2004]

17. Chariot, *NetIQ: Chariot Homepage*. [online]. Available from: http://www.netiq.com/products/chr/default.asp [Accessed 01 February 2004]
18. Packeteer, *Packeteer Homepage* [online]. Available from: http://www.packeteer.com [Accessed 01 February 2004]