PUC

ISSN 0103-9741

Monografias em Ciência da Computação
nº 25/06

A Metamodel for Configuring Collaborative Virtual Workspaces: Application in Disaster Management of Oil & Gas Offshore Structures

Enio Emanuel Ramos Russo
Alberto Barbosa Raposo
Marcelo Gattass
Terrence Fernando

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 25/06 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena July, 2006

A Metamodel for Configuring Collaborative Virtual Workspaces: Application in Disaster Management of Oil & Gas Offshore Structures *

Enio Emanuel Ramos Russo[1] Alberto Barbosa Raposo[1]
Marcelo Gattass[1] Terrence Fernando[2]

[1]Laboratório de Computação Gráfica (TecGraf) – Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
[2]School of Construction and Property Management, The University of Salford, UK

[enio, abraposo, mgattass]@tecgraf.puc-rio.br, t.fernando@salford.ac.uk

Abstract. Many companies have been creating virtual teams that bring together geographically dispersed workers with complementary skills, increasing the demand for CSCW (Computer Supported Cooperative Work) applications. In order to facilitate the development of a wide range of these collaborative applications, we should offer a general architecture that is adaptable to different situations, tasks, and settings in a flexible way. This work investigates how a distributed workspace environment can support disaster management, involving distributed collaborative technical teams. We first identify the requirements for the distributed workspace, from the stakeholders involved in a disaster, and analyse the commercial emergency systems available. We then elaborate a multi-perspective metamodel to support configuring this collaborative virtual workspace. Finally a prototype for oil & gas offshore structures disaster management based on our multi-perspective metamodel is derived and an HLA (High Level Architecture) compliant implementation for this prototype is developed as a proof-of-concept of the metamodel.

Keywords: computer-supported cooperative work, metamodel, virtual workspaces, oil & gas.

Resumo. Várias companhias têm criado equipes virtuais para agregar trabalhadores de diversas especialidades que estão dipersos geograficamente, aumentando a demanda por aplicações CSCW (Computer Supported Cooperative Work). De modo a facilitar o desenvolvimento de uma ampla gama destas aplicações colaborativas, devemos prover uma arquitetura genérica que seja adaptável a diferentes situações, tarefas e configurações de um modo flexível. Este trabalho investiga como um ambiente de trabalho distribuído pode apoiar o gerenciamento de desastres, envolvendo equipes técnicas colaborativas distribuídas. Primeiramente, identificamos os requisitos para o espaço de trabalho distribuído, a partir dos atores envolvidos em um desastre, e analisamos os sistemas de emergência comerciais disponíveis. Em seguida, elaboramos um metamodelo multi-perspectiva para auxiliar a configurar este espaço de trabalho virtual colaborativo. Finalmente, derivamos, a partir do metamodelo, um protótipo para o gerenciamento de desastres de estruturas offshore de óleo & gás e desenvolvemos uma implementação aderente ao padrão HLA (High Level Architecture) para este protótipo, como prova de conceito deste metamodelo.

Palavras-chave: trabalho colaborativo auxiliado por computador, metamodelo, espaços de trabalho virtuais, óleo & gás.

---

*This work has been sponsored by Petrobras.

In charge for publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

List of figures

List of tables

1
Introduction

There are serious risks involved in running offshore units, with many reported disasters. These disasters can not only cause fatalities and serious environmental impacts, but also have strong impact on business. Companies can lose billions of dollars when they lose an offshore unit and further billions of dollars as a consequence of the halt in oil production. For example, Petrobras would lose US$ 700 million by losing the P-40 offshore platform and more than US$ 6 billion by failing to accomplish its three-year production of 150,000 barrels of oil.

Among the worst disasters there was the Exxon Valdez oil tanker in Alaska, USA, in 1989, with direct cleanup costs of 4 to 8 billion dollars and 10 years of efforts to allow the ecosystem to revert the area back to its natural state. Similarly, the disaster of the Piper Alpha oil platform in 1988 in the North Sea caused 167 casualties.

Petrobras also faced two major accidents in the beginning of this decade. In 2001, P-36, the largest semi-submersible platform in the world (40-story high, weighing 31,000 tons), sunk, killing 11 employees and ceasing a daily production of 84,000 barrels of oil and 1.3 million cubic meters of natural gas. In 2002, the P-34 FPSO (Floating Production, Storage and Offloading) unit, with a daily production of 35,000 barrels and a storage capacity of 58,000 m^3 of oil, weighing 62,000 tons, had a stability problem and almost sunk, immediately ceasing its operation. This time, Petrobras managed to rescue the unit without loss of lives.

As a direct result of these huge accidents, oil & gas companies usually take measures in two main directions: *(i)* one that has the objective of correcting and improving operational procedures; and *(ii)* a second one that has the aim of planning a set of projects to improve the technological level of the company in order to minimise the risk of future accidents. Also the organisational learning process of these companies has improved the protection level of the environment

and guaranteed high standards of operational security for both employees and physical installations (Costa, 2004).

Considering the second aspect mentioned above and the necessity of minimising disaster impacts, we have verified the need to develop a system architecture capable of bringing people together to work as a virtual team in order to explore various rescue plans and work towards consensus.

Many companies have been creating virtual teams that bring together geographically dispersed workers with complementary skills, increasing the demand for CSCW (Computer Supported Cooperative Work) applications. In order to make the development of a wide range of these collaborative applications more effective, we should offer a general architecture that is adaptable to different situations, tasks, and settings in a flexible way. The motivation for this work has been the necessity of developing a collaborative virtual workspace for disaster management of oil & gas offshore structures for a global company (Russo et al., 2004).

1.1. Motivation

Petrobras, as one of the oil & gas companies seeking to employ efficient processes and technologies to respond to such events, has taken important actions in order to ensure safety. For example, in 2000, Petrobras launched a programme called *Environment Management and Operational Security Excellence Program* (PEGASO), with an investment of 1.7 billion dollars and the development of 4,000 projects in 4 years (Petrobras, 2004). As a direct result of the P-36 accident, the enquiry commission recommended the implementation of another operational excellence program (PEO) specific for offshore production units (Petrobras, 2001).

However, the implementation of such processes involves bringing together a large number of diverse and geographically distributed groups and resources to make appropriate decisions within a short period of time. Such groups are comprised of many technical experts and decision makers such as naval engineers, structural engineers, risers analysts and oceanographers, as well as managers. Typically, a high-level decision group will operate from the operational unit and

the technical staff from a land base near the disaster, from the headquarters and/or from various research centres. These groups need to be in constant communication with operators inside the unit, divers, and security team, and perhaps with experts who are travelling to execute the rescue plan. The main task of the technical staff in such situations is to rapidly come up with a solution to stabilise the unit, by running various simulation programmes which take into account waves, winds, currents, and other forces acting on the unit. The members of the technical staff can also be at distributed locations, in connection from various research centres and the headquarters.

In face of all of these requirements, there is the need to develop a system architecture capable of bringing people together to work as a virtual team, in order to explore various rescue plans and work towards consensus. The purpose of this thesis is therefore to explore how collaborative workspace technologies can support such decision-making and consensus-building process. We then elaborate a multi-perspective metamodel to support configuring this collaborative virtual workspace.

1.2. Aims and Objectives

The main aim of this work is to investigate how a distributed workspace environment can support disaster management, involving distributed collaborative technical teams. Specifically, this research will focus on a distributed workspace for technical groups to work as a collaborative virtual team, to explore various simulation options and to communicate their results to decision makers.

This aim will be achieved through the following objectives:

- conducting a survey:
  - identifying the requirements for the distributed workspace, from stakeholders involved in a disaster scenario;
  - analysing the commercial emergency systems available;
- elaborating a metamodel to configure collaborative virtual workspaces;
- conducting a survey to analyse the most important distributed systems;

- defining a distributed workspace environment based on this metamodel for the technical team to engage in rescue efforts.

1.3. Thesis Outline

The sequence of chapters of the present thesis is organised as follows.

In Chapter 2, we process requirements gathering focussing on two Petrobras' case studies: P-36 and P-34. We also summarise a survey on the main commercial emergency systems available, identifying the need to develop a system architecture capable of supporting the distributed resources present in an emergency scenario, mainly distributed simulators running on high performance visualisation systems. This architecture should provide synchronous communication among different equipments with virtual co-location as one of the features, constituting a collaborative virtual workspace for disaster management.

In Chapter 3, we elaborate a multi-perspective metamodel to help define and configure the collaborative virtual workspace architecture. We also show how models derived from the metamodel can be reconfigured to accomplish new situations, which is particularly important in emergency scenarios.

In Chapter 4, we summarise a survey on the main technologies used for developing distributed environments, from both middleware and specific distributed virtual environment approaches. We select two of these technologies to validate our metamodel, namely High Level Architecture (HLA) and InfoGrid.

In Chapter 5, we derive a first model for oil & gas offshore structure disaster management based on our multi-perspective metamodel. We also develop an HLA-compliant prototype as a proof-of-concept of the metamodel and discuss how the prototype could be implemented using InfoGrid. Still for the disaster management application, we present a second model and its prototype showing that we can derive different models for the same application. Finally, to validate the generality of the metamodel, we also delineate a model for another application, namely CAD (Computer-Aided Design) visualisation in virtual environments.

In Chapter 6, we present the conclusions and possible future work.

Finally, in the Appendix, we present some screenshots of the HLA-compliant prototype that was developed.

2
Requirements Gathering

Requirements gathering was obtained through Petrobras' case studies P-36 and P-34. These case studies were used to identify the roles and attributes of people involved in a typical disaster management operation. Semi-structured interviews were also carried out to identify procedures and user expectations about the collaborative workspace. In this type of environment, it is important to model the users' relationships and to identify the main collaborative features that the users would like to have.

Once the users' requirements capture phase was completed, the next step was to define the technical requirements in terms of collaboration models, simulation steering, personalised and global workspaces, synchronised viewing, video-streaming, etc. The intended collaborative workspace should have distributed nodes with personalised interfaces representing various user perspectives.

We then conducted a survey on the main commercial emergency management systems available to gather their principal characteristics and the main features still underdeveloped.

2.1.
Evolution of Disaster Management in Petrobras

There were two main disaster incidents in Petrobras (P-36 and P-34). This section provides a summary of these two incidents with the purpose of illustrating the complexity of the problem in terms of processes and groups of people involved in such disaster incidents. From this discussion, we show that Petrobras has been continuously active in improving its disaster management program.

During the P-36 disaster, there was a mechanical explosion and a chemical explosion with loss of lives, which caused difficulty in acting quickly to save the unit. During the P-34 disaster, there was no explosion, enabling the teams to react quickly, although communication among them could still be improved. This

research aims to perform the following step in terms of using ICT (Information and Communication Technology) to improve the collaboration among the stakeholders involved in disaster incidents.

In the case of the P-36 disaster, Petrobras identified the need for updated emergency procedures and for executing the actions within a short period of time in order to save the unit. This case raised the need to investigate collaborative and decision-making models to help complex teams avoid disasters.

In the case of P-34, there was already an updated model of the offshore unit and a form of distributed work that did help the rescue team to act quickly. There was also a static simulator that allowed specialists to run different simulations. Nevertheless, the team still lacked an adequate environment to work as a virtual team to share knowledge, jointly discuss possible rescue plans, and work quickly towards consensus.

As a result, it was necessary to bring people together into the same physical location, causing some delay in the process. Furthermore, some of the information was not directly available to the decision makers. This incident showed the necessity of strengthening collaboration among the distributed teams, providing better interaction, simulation, and discussion during the whole rescue operation.

2.2. Requirements

From the analysis of the described disasters, we observe the need for going a step further and developing a paradigm for reacting to emergency scenarios. This paradigm should provide, as its essential core, collaboration among all the participants in this emergency scenario. Ideally each participant node of this distributed environment should be able to share and discuss his results with other participants and to access whatever data he needs for his simulation tasks.

We are now going to discuss the distributed nature of the teams and the resources that need to be brought together in the proposed collaborative workspace paradigm.

2.2.1.
Distributed Nature of the Teams

In the particular case of Petrobras, when an accident occurs, the head office is immediately contacted and the General Manager of the operational unit is in charge of crisis management. All the work will be under his control in the decision workspace. The Security, Environmental, and Health Department then starts emergency procedures and at the same time technical specialists begin to act. Constituting this technical workspace, there are naval engineers, structural engineers, risers analysts, and oceanographers. When they are working together in a collaborative way, there are usually the following main distributed groups:

- the high-level decision team at the operational unit;
- a task force group leading the decision-making process:
  - at the Business Unit (in Rio, for platforms located in Campos Basin) if the platform is not heavily damaged, with two or three operators remaining inside the unit and performing the operations required; or
  - in a city which is the nearest place to the accident over land (Macaé, for platforms located in Campos Basin), from where orientation is given to divers, who do the only possible work when the unit is heavily damaged and has security problems;
- a technical support team at the company headquarters in Rio, at the Business Unit, and at the research centre;
- mobile experts, who sometimes are overseas or travelling and who must also be connected.

In addition to these groups, and working together with them, there are security teams in rescue units which are moved towards the region of the accident and provide help during the whole crisis period.

2.2.2.
Distributed Nature of the Resources

Not only the experts, but also the system resources are distributed in this scenario:

- computer-intensive simulators have to remotely run on a super computer or on a cluster of computers to get quick results;
- the computer system needs access to remote databases which maintain CAD models and simulation models of the unit;
- each site participating in the crisis solution can have different configurations, such as a Virtual Reality (VR) Centre, an intranet desktop, and a laptop connected to the network;
- experts who are travelling may have to be linked via mobile technologies;
- the connection between the unit and the people on land may vary. In the best case, the unit is one node of the network and, in the worst case, the communication with the operators can be done only by radio or telephone.

2.3. Commercial Emergency Management Systems

After having determined the collaborative disaster management workspace requirements, we conducted a survey on the main commercial emergency management systems available. We identified the principal characteristics of those systems, the main areas already covered, what is the state-of-the-art and what are the main features which are still underdeveloped.

While performing this survey, existent emergency management systems from some vendors were investigated, as well as crisis intervention methods being practiced in companies such as Statoil, Norsk Hydro, Elf, and British Petroleum (BP). The systems investigated are summarised in the next subsections.

2.3.1. U.S. Department of Justice's Feature Comparison Report

The U.S. Department of Justice has developed a Feature Comparison Report of the main Crisis Information Management Software (CIMS) commercially available (Hart, 2002). That survey resulted in many important findings, from which the ones directly related to this work are listed below. The software should:

- allow for remote access by authorised users located outside the LAN;
- comply with the provisions and standards for Incident Command System (ICS). ICS is the model tool for command, control, and coordination of a response and is built around five major management activities related to an incident:
  - Command;
  - Operations;
  - Planning;
  - Logistics;
  - Finance/administration;
- integrate with other systems, such as mapping, other CIMS, and telephonic alert notification systems;
- integrate public health into emergency management;
- operate within a variety of network configurations;
- have a wide range of features consistent with the four phases of emergency management operations: planning, mitigation, response, and recovery.

They also concluded that there is no best product and no perfect fit. The best product should be chosen based on budgets, system environment, scale of operation, sophistication of operation, discipline to implement, and political considerations.

2.3.2. L-3 CRISIS Command and Control System

One of the main vendors of emergency systems is Ship Analytics, with whom Petrobras has already begun establishing a commercial relationship. One of its main products is the L-3 CRISIS Command and Control System (MPRI Ship Analytics, 2003), which is a standard off-the-shelf computer system. It provides support for emergency managers, being the nerve centre during disaster response and an educational tool to train for disaster response. It also provides computer simulations which allow for evaluation of alternative responses, a planning system

for risk management and environmental damage mitigation, and a cost accounting system for management of alternative assets. It also provides incident management teams with a Geographic Information System (GIS) and Standard Operating Procedures (SOPs) for responding to different types of disasters such as flood, windstorm, toxic gas release, etc.

As a practical matter, emergency managers find it useful to develop checklists from the SOP for each functional position which is event specific. These checklists are designed to be easy to read and easy to implement. L-3 CRISIS provides a means for automating the checklist function.

Firewalls are employed where necessary to protect sensitive information while still allowing users to access the information they require while doing their job. The system has been designed to utilise inputs from a variety of real-time sensing devices such as infrared imaging, satellite imagery, and hydrocarbon sensing buoys. It is a web-enabled system incorporating an imbedded browser which enables system users to access the most current and up-to-date information available from the Internet, Intranet, and the system's own databases.

2.3.3. Oil Spill Crisis Management Simulator

One of the simulators based upon L-3 CRISIS is the Oil Spill Crisis Management Simulator. It supports all stages of incident management from initial prevention/mitigation through preparation, response, and remediation. It functions as a complete training centre which prepares Incident Management Teams, On-Scene Commanders, and Responders to handle a variety of incidents via exercise simulation, response plans, operating procedures, and checklists.

The Spill Management Simulator (SMS) incorporates existing, standard oil spill models, and oil weathering models. The exercise participants interact with the scenario through the use of L-3 CRISIS response modules, with an interface to complex relational databases, a Geographic Information System (GIS), and science-based fate and trajectory hazard models.

The SMS also provides a central repository that contains information critical to the effective management of crisis response, such as the locations and specifications of available response equipment, available personnel, and

environmental and economic sensitivity data. The simulation component of the system, which models the physical fate and trajectory of spilled oil, and tracks the simulated response vessels and countermeasure operations, stores its output data in the central database.

To conduct a simulation exercise, scenario-specific personnel and equipment databases are created by associating all or a subset of the Master Resource databases with a given scenario. Once the exercise is running, instructors/operators can modify and/or add scenario-specific resource records without affecting the Master databases. The resource data is used by the Incident Management Team or Exercise Participants to allocate and track equipment and personnel.

Spill scenarios are developed or modified prior to the conduction of an exercise using a scenario definition utility.

2.3.4. U.S. Automated Resource Management System (ARMS) Systems Requirements Document (SRD)

Another important document which has been studied was the one by Booz Allen Hamilton (2003), who has been tasked to support the U. S. Federal Emergency Management Agency (FEMA) - Preparedness Division, in the development of a requirements document for the Automated Resource Management System (ARMS). The ARMS Systems Requirements Document (SRD) provides a high-level specification of ARMS system requirements by identifying and defining the corresponding data, business rules, functional, operational, and technical requirements for a web site needed to help state and local governments improve their capability to carry out mutual aid during emergency situations.

ARMS is defined as an “automated system which assists emergency managers in locating resources to enhance their response to emergencies.” Resources include personnel, equipment, and supplies. ARMS is the computerised portion of the U. S. National Mutual Aid and Resource Management Initiative which will enhance the mutual aid process.

2.3.5.
Crisis Intervention and Operability (CRIOP) Analysis

Another important document describing a scenario method for crisis intervention and operability analysis was CRIOP (Johnsen, 2004), jointly developed by Scandpower, SINTEF, STATOIL, and NTNU, with support from Norsk Hydro, Saga, Elf, NORSOK, BP, Safetec, DNV, and Aker.

CRIOP is a methodology used to verify and validate the ability of a control centre to safely and effectively handle all modes of operations including start up, normal operations, maintenance and revision maintenance, process disturbances, safety critical situations, and shut down. Such methodology can be applied to central control rooms, drillers' cabins, cranes and other types of cabins, onshore, offshore and emergency control rooms.

The key elements of CRIOP are checklists covering relevant areas in the design of a Control Centre (CC), Scenario Analysis of key scenarios, and a learning arena where the workforce with operating experience, designers, and managers can meet and evaluate the optimal CC. A CRIOP analysis is initiated by a preparation and organisation phase to identify stakeholders, gather necessary documentation, establish analysis groups, and decide when the CRIOP analysis should be performed.

The CRIOP method focusses on the interaction among people, technology, and organisations. One of the most important principles of the CRIOP method is to verify that a focus is kept on important human factors, in relation to operation and handling of abnormal situations in offshore control centres, and to validate solutions and results.

2.3.6.
Conclusions about Emergency Systems

Finalising the emergency management survey, it can be observed that most emergency management systems have some common characteristics which we describe below:

- they usually serve as incident management as well as training and planning tools;

- they have strong integration capabilities, not only with internal databases and systems but also with public emergency management systems;
- they usually have a flexible architecture in terms of networks and software integration, mainly being integrated with one or more simulators related to the type of disaster being treated;
- normally they integrate to a graphic system, such as a Geographical Information System (GIS), which is responsible for displaying real-time data of the incident being considered;
- they are capable of logging and tracking activities and resources, which is important not only during a real emergency case but also as an efficient tool for training purposes;
- most of the emergency systems also use checklists as an efficient and fast method to address the multiple simultaneous requirements present during an emergency scenario. The capability of providing a means for automating the checklist function as much as possible can be determinant for solving an emergency case safely and timely.

In spite of all the features listed above, we identified two main drawbacks of current emergency management systems:

1. Lack of full and suitable integration of simulators with high performance visualisation systems;
2. Inadequate security and access control features.

The survey has demonstrated that, in spite of the integration of most of the emergency management systems with simulators, there is the need to develop a system architecture capable of supporting distributed resources, especially distributed simulators running on *high performance* visualisation systems. This architecture should also provide synchronous communication among different equipments with virtual co-location as one of the features.

The integration of simulators using high performance visualisation systems in a synchronous distributed environment is the aspect of the emergency scenario on which we are going to focus. In order to support the definition of the architecture for this environment, a metamodel has been elaborated. We then have

developed an HLA-compliant prototype as a proof-of-concept of the metamodel, considering that HLA is a standard for high performance real-time simulations.

3
A Metamodel for Configuring Collaborative Virtual Workspaces

In this chapter, we elaborate a multi-perspective metamodel to help define and configure the collaborative virtual workspace architecture.

Collaborative systems have been defined as “a combination of technology, people, and organisations that facilitates the communication and coordination necessary for a group to effectively work together in pursuit of a shared goal, and to achieve gain for all its members.” (Haynes et al., 2004) Employing this concept, many companies have been creating virtual teams of their own employees that bring together geographically dispersed workers with complementary skills (Handel & Herbsleb, 2002; Bos et al., 2004), increasing the demand for CSCW applications. In order to make the development of a wide range of these collaborative applications easier and more effective, we should offer a general architecture that is adaptable to different situations, tasks, and settings in a flexible way (Schuckmann et al., 1996).

CSCW research to date on how to address the architecture characteristic mentioned above has largely focussed on issues concerning differences between: *(i)* co-located work and working across distance (Bos et al., 2004; Dourish & Bly, 1992; Fussell et al., 2000; Herbsleb & Grinter, 1999; Herbsleb et al., 2000; Lauche, 2005); or *(ii)* work with people from the same culture, or common ground (affinities, mutual knowledge, beliefs, goals, attitudes, etc), and work with people from different cultures (cross-cultural work), or common ground (Fussell et al., 2000; Greenspan et al., 2000; Setlock et al., 2004). These perspectives have been named, respectively: *Place-Centred* and *People-Centred* (Jones et al., 2004).

Instead of employing one of those two perspectives to derive a metamodel for designing CSCW applications, we propose to adopt a different view on the problem based on the activities carried out by the teams participating in the collaborative work. We name this perspective *Activity-Centred*. The term *Activity*

is incorporated here from the very broad and multi-layered concept from activity theory.

As it will be detailed in the next sections, the Activity-Centred perspective may be seen as a multi-perspective concept (also in accordance to the way by which the term was incorporated from activity theory), since it not only encompasses the Place-Centred and the People-Centred perspectives, but also allows adopting each one or more typically both of them (in a hybrid way) to the desired extent, and admits changes from one perspective to another.

It is important to stress that this is not a technology-driven approach but a user or human-centred approach (Ishii et al., 1994; Laurillau & Nigay, 2002; Palen, 1999; Sachs, 1995), focussing on a qualitative research (Neale et al., 2004) of the involved teams instead of on a particular technology.

The motivation for this work was derived from the necessity of configuring a collaborative virtual workspace for disaster management of oil & gas offshore structures for an oil & gas global company. In order to identify the system's requirements, the authors have conducted semi-structured interviews with key individuals and not only have observed their work practice but in fact have been participating with them in joint projects and activities for more than one decade.

3.1. Activity-Centred Metamodel

We begin this section defining *metamodel*: it is a logical – as opposed to physical or implementation – model, emphasising the declarative semantics rather than the implementation details of the model. It is expected that the implementations that are based on the metamodel should conform to its semantics (Athanasopoulos et al., 2003).

Various researchers have been developing architectures based on this concept. Dewan's generic collaborative architecture (Dewan, 1999) structures a groupware application into a variable number of layers from the domain-dependent level to the hardware level, where a layer is a software component corresponding to a specific level of abstraction. Similarly, the Clover architectural metamodel (Laurillau & Nigay, 2002) also structures a groupware application into

a variable number of layers, decomposing each layer into three functional sub-components dedicated to production, communication, and coordination. Dourish, in the Prospero toolkit, proposes an approach in which a metalevel architecture is used by programmers to separate the high-level representation and functionality from the lower-level facilities, mapping the first ones onto the second ones (Dourish, 1998).

Our proposed metamodel adopts a similar multi-level approach, according to Leontjev's (1978) activity theory version, in which a three-level scheme describes the hierarchical structure of activity. This scheme has been summarised by Tuikka (2002) as follows:

- The central level is that of actions. Actions are oriented towards goals. Usually, goals are subordinated to other goals, which may be subordinated to further goals, and so on.
- Moving up the hierarchy of goals we finally reach a top-level goal, which is not subordinated to any other goal. This top-level goal, which in activity theory is designated as *motive*, is the object of a whole activity.
- Moving down the hierarchy of actions, we could reach automatic processes providing adjustments of actions to current situations. According to activity theory, they are *operations*.
- Activities, which are driven by motives, are performed through actions which are directed at goals and which, in turn, are implemented through operations.

Orthogonally to this approach, similarly to the Clover metamodel, the Activity-Centred metamodel also allows the breakdown of the components correspondent to a specific level. These two orthogonal approaches applied together contribute to the generality of the metamodel.

3.1.1. Metamodel Abstraction Levels

The top-most level of our metamodel, the motive, is represented by a single complex *node* which encompasses the whole activity. This motive can be as

diverse as the elaboration of a paper (Figure 1) or the disaster management of an oil & gas offshore structure (Figure 2). The level immediately below the top-most level contains the main actions that should be performed in order to accomplish the whole activity. These actions are the result of the interactions of groups, with each group represented by a complex node and the interactions among them represented by *edges*, which means that each abstraction level could be represented by a graph.

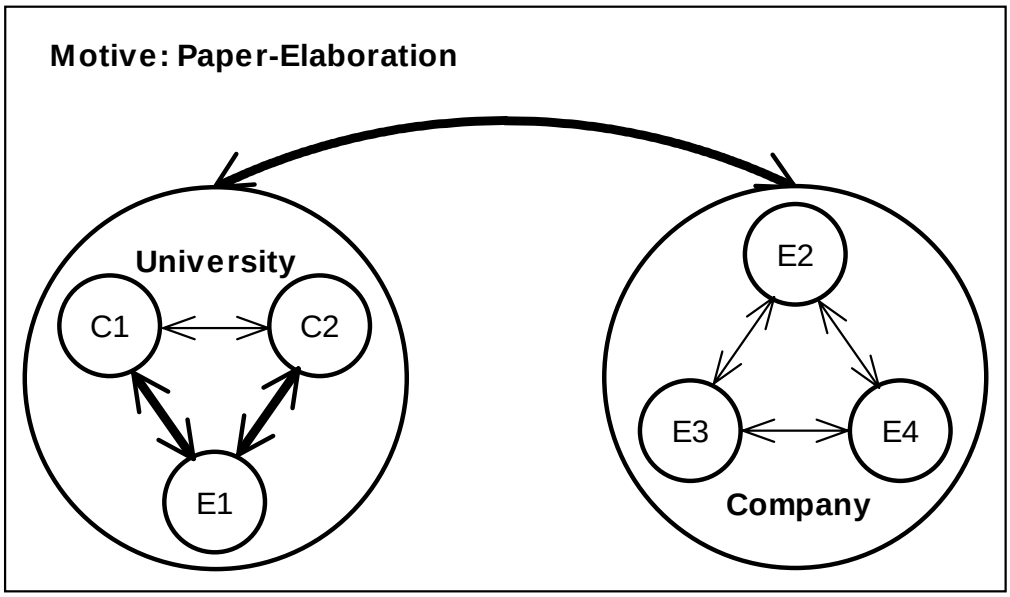

Figure 1 - A paper-elaboration collaborative model: C1 and C2 are Computer Science researchers, while E1, E2, E3, and E4 are Engineers

We continue this downward process splitting each complex node of the upper abstraction level into more elementary nodes until we reach a *leaf node*, which will typically be a *person*. To those leaf nodes we then associate implementation and hardware attributes such as the application to be executed and the host in which it should be run. Sometimes the leaf node is not a real person but a *software agent,* responsible for a specific set of tasks (Ellis et al., 1991).

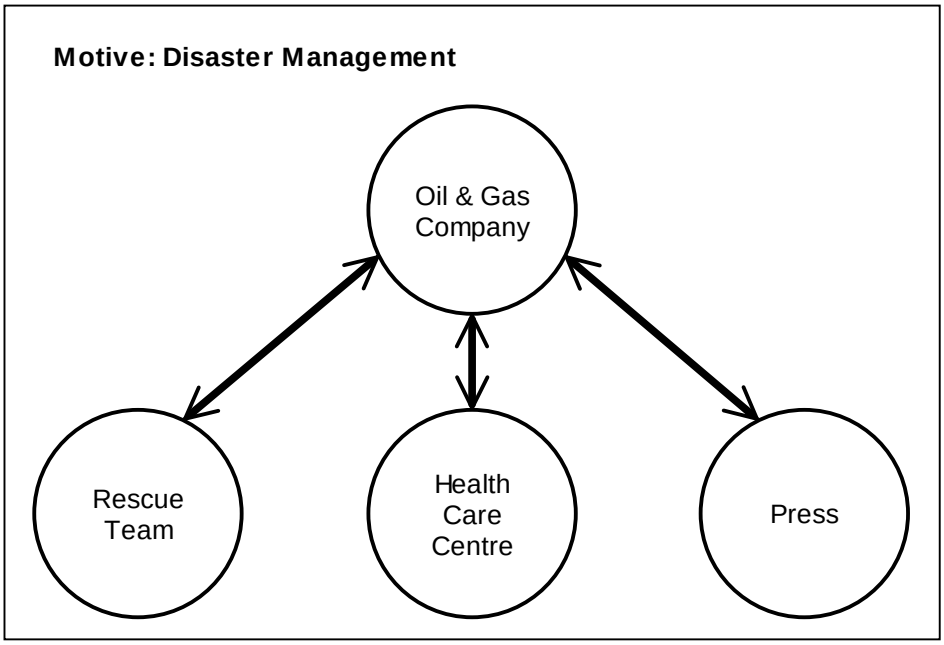

Figure 2 - An oil & gas offshore structure disaster management collaborative model

From Figure 2, we can see that this metamodel seems sufficiently general to accommodate even models that include inter-organisational groups or nodes. In these cases, traditional groupware access control mechanisms should be extended for dealing with new aspects such as privacy, confidentiality, mutual and contradictory interests, different cultures, etc (Cohen et al., 2000; Godefroid et al., 2000; Setlock et al., 2004; Stevens & Wulf, 2002).

3.1.2. Breaking Down the Components of a Specific Abstraction Level

Orthogonally to the top-down process, the Activity-Centred metamodel also allows the breakdown of the components correspondent to a specific level. The main idea of this mechanism will become clearer if we illustrate it with a practical example.

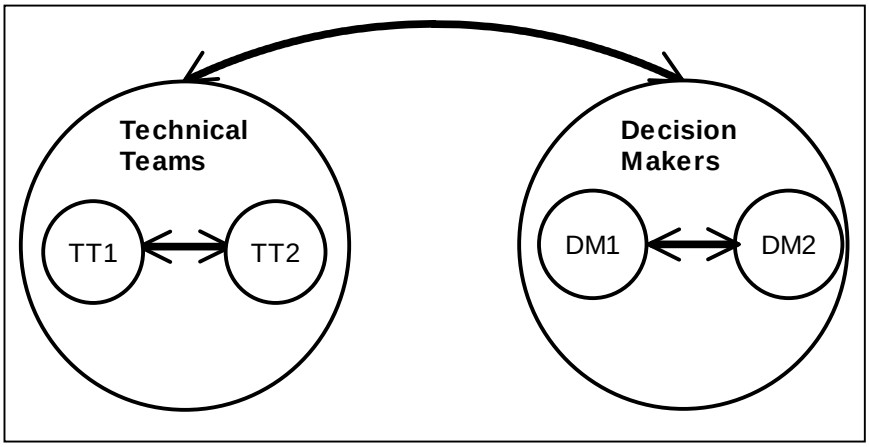

Figure 3 - First downward level of the oil & gas company from the disaster management collaborative model: TT1 and TT2 represent Technical Team 1 and Technical Team 2, respectively, while DM1 and DM2 represent, respectively, Decision Maker 1 and Decision Maker 2

Let us consider a specific abstraction level of the disaster management example, namely the first one downward of the oil & gas company (Figure 3). There we can observe two main groups: *Technical Teams* and *Decision Makers*. To facilitate the comprehension of the breakdown mechanism, we also represent in the same figure the level immediately below the one being considered: group *Technical Teams* is decomposed into sub-groups *Technical Team 1* and *Technical Team 2*, and group *Decision Makers* is decomposed into sub-groups (persons) *Decision Maker 1* and *Decision Maker 2*. In such a configuration, both *Decision Makers* are being considered with the same background and level of interaction with the *Technical Teams*. Now suppose that *Decision Maker 1* is a more technical manager and that *Decision Maker 2* is a manager of a higher organisational level inside the company, such as a director, and is not so technically involved with the problem. The simplest way to accommodate this difference is to split the *Decision Makers* group into two new groups of this same abstraction level, namely *Decision Maker 1* and *Decision Maker 2*, each one with one single manager. We should also replace the old edges by new ones representing the new interactions among the new groups formed. The resultant configuration is shown in Figure 4. Now it becomes clear that the middle level manager is the one responsible for directly communicating with the *Technical Teams* and also for communicating with the director.

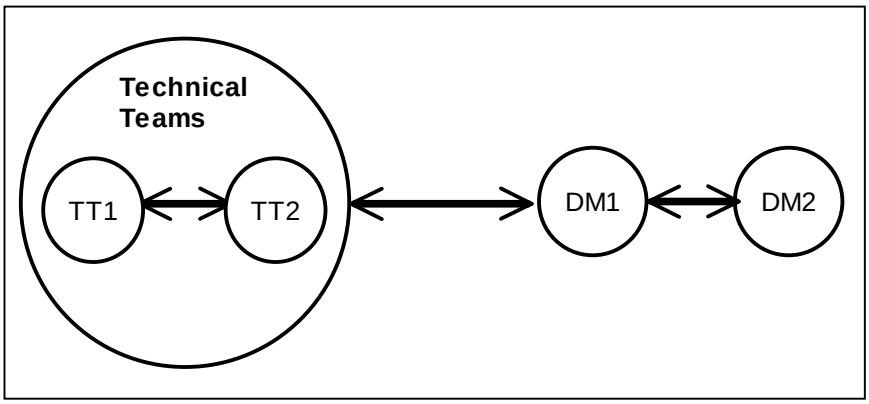

Figure 4 - First downward level of the oil & gas company from the disaster management collaborative model, now with Decision Maker 1 placed between the Technical Teams node and Decision Maker 2

This simple example illustrates the usefulness of this breakdown mechanism, allowing us to experience and derive the most appropriate model to a specific situation.

3.1.3. Metamodel Components

We are now going to describe the main components of our metamodel.

3.1.3.1. Nodes

Nodes are essential components of our metamodel, going from the top-most node representing the whole activity (motive) through many nodes of different levels representing groups and sub-groups until the *leaf nodes* representing a *person* or a *software agent*. Nodes have a set of *attributes* such as user interface preferences and language used, which are applied using a hierarchical class concept: the value of a node attribute is valid for all sub-nodes below the node considered unless explicitly redefined, with this redefinition also valid for all levels below the one considered.

Nodes also have an attribute called *artefacts* defined as “all objects on which users can operate” (Gross & Prinz, 2004). Examples of artefacts are drawings, sketches, physical models, prototypes and product descriptions, documents and files, pens and blackboards, reports, papers and review comments,

spreadsheets and artwork, all in their electronic version, or more complex ones such as electronic calendars (Palen, 1999), shared drawers and cabinets (Pankoke-Babatz & Syri, 1997), CAD and computerised maps (Pettersson et al., 2004), shared virtual prototypes (Tuikka, 2002), shared interactive surfaces (Brignull et al., 2004), shared tabletop displays (Morris et. al, 2004a), shared multi-user tabletops (Morris et al., 2004b), and interactive shared displays (Paek et al., 2004). Following the class concept, an artefact associated with a group node is shared by all members in the group, unless otherwise explicitly stated. In this case, a mechanism such as an access control list will determine who shares access to the artefact.

An artefact has two basic *properties* (Dourish & Bellotti, 1992): *synchrony*, which can assume values *synchronous* (being worked on at the moment) or *asynchronous*, and *persistency*, which can assume values *persistent* (lasts after being worked on) or *volatile*.

Leaf nodes have the low-level attributes that are going to be mapped into a specific implementation such as the application to be executed (Gross & Prinz, 2004).

3.1.3.2. Edges

Edges in our metamodel represent the *interaction paths* among the nodes. They can be uni- or bi-directed, representing the possible directions of interaction. When an edge is represented by a thin arrow, this means that the nodes on its extremities are co-located. When the arrow is thick, this represents that one node is placed remotely in relation to the other.

An edge has one or more *channels*, each one representing the electronically mediated channel that allows communication between two nodes (Greenspan et al., 2000).

According to Nardi et al. (2000), communication is primarily about information exchange. The information exchanged can be of many types, such as text, graphics, voice, and video. Fuks et al. (2005) consider some elements that communication channels should have in order to make this information exchange occur: media (textual, spoken, pictorial, or gestured), transmission mode (in

blocks or continually; synchronous or asynchronous), restriction policies, meta-information (about the message, such as subject, date, priority, and category), conversation structure (linear, hierarchical, or network form), and conversation paths.

A channel has four basic properties:

- Synchrony (Dourish & Bellotti, 1992; Greenspan et al., 2000): the synchronous-asynchronous distinction depends upon when the content is communicated or chosen. Examples of synchronous or real-time channels include: instant messaging, chat, and video conferencing. An important characteristic of these synchronous applications is that they convey presence awareness information (Dourish & Bellotti, 1992; Godefroid et al., 2000; Pankoke-Babatz & Syri, 1997) about the members participating in the collaborative session. Some examples of asynchronous channels are: electronic mail, forum-style computer conferencing, and web pages. While synchronous channels can provide real-time feedback, asynchronous channels, on the other hand, can afford greater control over content creation and make it easier to archive and forward messages. Beyond the two traditional modes, Dourish and Bellotti (1992) introduce the term *semi-synchronous*, which means supporting both synchronous and asynchronous modes, associated with shared workspaces: semi-synchronous systems present information on synchronously co-present collaborators, at the same time as representation of past activities by other collaborators who are not synchronously present.
- Persistency (Dourish & Bellotti, 1992; Pankoke-Babatz & Syri, 1997): the channel is called *volatile* if it can provide information only at the moment in which the event occurs, such as a video conferencing. On the other hand, it is called *persistent* if it can provide information after a certain time of absence to inform about intermediate progress, or on request to provide more details, or to inform about the history of an object. An example of persistent channel is a chat that has the ability to retain discussion history (Handel & Herbsleb, 2002; Ribak et al., 2002) or a forum-style computer conferencing.

- Symmetry (Greenspan et al., 2000): a channel is considered *symmetric* if the receiver of a message can respond with the same type of message. An example is an e-mail tool that allows reading together with composition and sending. Videotelephony is a fully symmetric channel providing audiovisual backchannel feedback, while television and radio broadcasts are fully *asymmetric*. Distance-learning applications introduce the possibility of hybrid forms of communication in which both parties can hear one another, but only the presenter is visible to the students.
- Media: this property is related to the media richness (Greenspan et al., 2000) of the communication channel, involving, for example, audio and visual aspects. Non-computer-mediated face-to-face communication is considered more "media rich" than audiovisual media, which is more "media rich" than audio-only or visual-only communication media.

3.2. Instantiating the Activity-Centred Metamodel: Activity-Centred Models

In this section, we are going to focus on the most important characteristic of our Activity-Centred metamodel: being capable of accommodating multiple perspectives, corresponding to different models, each one being an instance of the metamodel.

In order to facilitate the understanding of the multiple perspectives, we will derive models for the paper-elaboration collaborative model introduced in Subsection 3.1.1, the only difference being that we now have researchers in two different universities.

3.2.1. A Place-Centred Model

Suppose that the most important aspects of our collaborative application are related to the place where people are effectively working. This could be motivated, for example, by the necessity of employing "media rich" communication such as face-to-face (Greenspan et al., 2000) or because the

different places have different facilities such as desktop computers and visualisation rooms, hence having different work behaviours. A possible model using this Place-Centred perspective for the paper-elaboration collaborative application is the one shown in Figure 5.

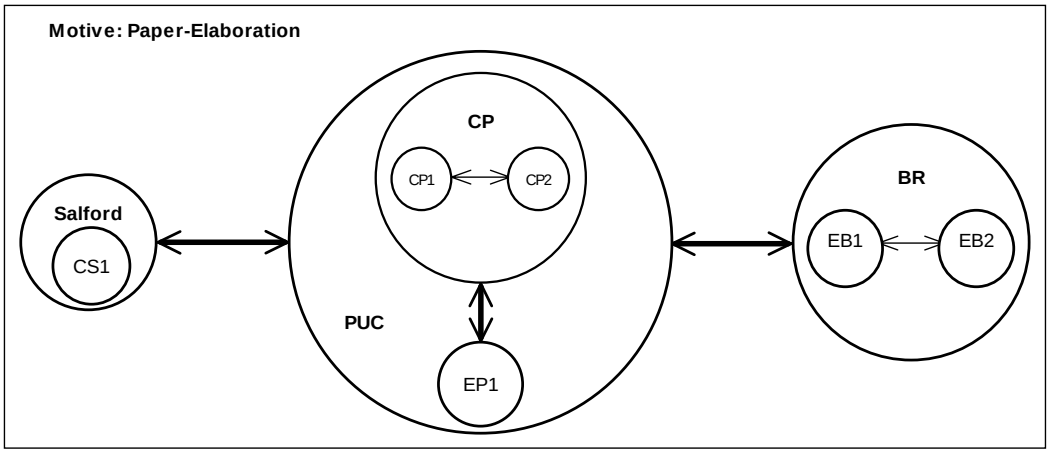

Figure 5 - The paper-elaboration collaborative model: a Place-Centred perspective

In this picture we can observe that the groups are formed having a Place-Centred perspective:

- We have three main nodes, which are: PUC University, Salford University, and Petrobras (BR, the Brazilian oil & gas company). Since in our example the paper is focussing on Computer Science, the central node, in this case, playing the main role in writing the paper, is PUC, which communicates remotely with both Salford University and Petrobras (in this model, there is no direct access from Salford University to Petrobras).
- Inside PUC University, we have two sub-groups: one is the Computer Science (CP, C for Computer and P for PUC) department, which has two co-located Computer Science researchers working together, CP1 and CP2, and the other is the Engineering department, which has one single Engineer (EP1). The two departments, being in different buildings, also communicate remotely. Only to avoid visual overload, when there is only one member in a specific node, we are designating this node with the name as the leaf node. Of course, we have to be careful with this notation, because it can be interesting to have properties associated to different levels, even in the case when

we have only one single member in each parent node for the example being considered. This could be the case of the hierarchy University → Department → Researcher, with specific attributes related to each level, even with only one leaf node representing all the groups.

- Inside Salford University, there is only one Computer Science researcher, namely CS1 (C for Computer and S for Salford). In this case, only to show the Salford University inheritance, we place the leaf node CS1 inside the parent node Salford (according to the above notation, we could only write CS1 inside this node).
- Finally, inside Petrobras (BR) node, we have two other co-located Engineers working together, namely EB1 and EB2 (E for Engineer and B for BR).

Therefore, in Place-Centred models we group the various nodes privileging the aspects related to the environments in which the work is being performed, such as especial rooms (e.g., visualisation rooms) or especial interactive devices (e.g., interactive shared displays), the necessity of having face-to-face communication, etc.

3.2.2. A People-Centred Model

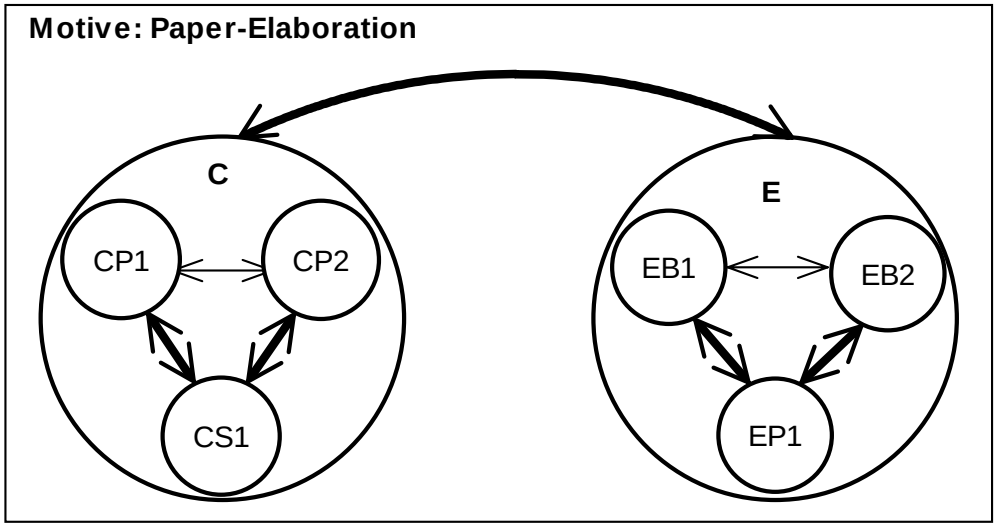

Figure 6 - The paper-elaboration collaborative model: a People-Centred perspective

Now consider that the main concerning issues of our collaborative application are related to culture and common ground barriers. In this case we should derive a model with a People-Centred perspective, such as the one shown in Figure 6:

- We now have only two main nodes: the Computer Science researchers' (C) group and the Engineers' (E) group, communicating remotely.
- Inside the C group, we have three Computer Science researchers working together, namely CP1, CP2, and CS1. CP1 and CP2 work co-located and CS1 works remotely to each one of them, possibly virtually co-located. It is important to note that, although CS1 is from a different university when compared to CP1 and CP2, their common ground is so intense that he belongs to the same sub-group as CP1 and CP2, although placed remotely to them. If this relationship is not so intense, or has a cultural barrier, we should split C into two sub-groups, one with two co-located CP1 and CP2 researchers communicating remotely with CS1, single member of the other Computer Science sub-group.
- The same reasoning can be applied to the E group. There we find three Engineers, namely EP1, EB1, and EB2. EB1 and EB2 are co-located Engineers, and EP1 works remotely to each one of them, possibly virtually co-located. Analogously to the CS1 analysis, if there is some sort of cultural barrier between EP1 and both EB1 and EB2, we should split the E group into two sub-groups, one with co-located EB1 and EB2 Engineers communicating remotely with EP1, single member of the other Engineering sub-group.

In People-Centred models, thus, the main concerning aspects while defining the groups are related to the participants' characteristics: their culture or common ground, their specialties, their skills, their behaviours, etc.

3.2.3.
Mixing the Perspectives: an Activity-Centred Model

We now mix the two previous perspectives in what we call an Activity-Centred perspective. In the case of the present collaborative application, it seems more adequate to focus on the whole activity being performed – the paper-elaboration – and then derive the groups to be formed. To elaborate the paper, the authors, CP1, CP2, CS1, and EP1, try to derive a new theoretical model based on the requirements identified through field study, working together with Engineers EB1 and EB2. So we aggregate those people in two main groups formed based on their main activity: the Theory group and the Field group. The groups are then arranged according to Figure 7:

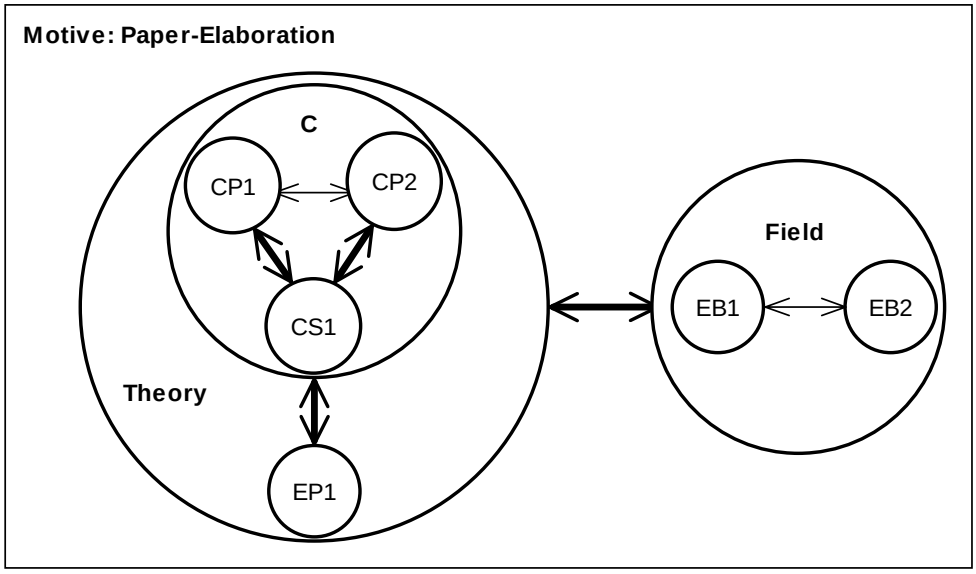

Figure 7 - The paper-elaboration collaborative model: an Activity-Centred perspective

- The two main groups, Theory and Field, communicate remotely with each other.
- Inside the Theory group, we have two sub-groups: one with three Computer Science researchers (C), namely CP1 and CP2 working co-located together with CS1 working remotely to them, possibly virtually co-located; and the other with a single Engineer, EP1, working remotely to the C group.
- The Field group is equivalent to the BR group in Figure 5, with two co-located Engineers working together, namely EB1 and EB2.

It is important to note the role being performed here by EP1. Clearly EP1 is being considered somehow different from the other members of the Theory group, probably acting more like a consultant for engineering issues. Nevertheless, if EP1 could be considered to have the same common ground and culture as the other members in the Theory group, capable of having the same level of discussion, it should probably be better to have him integrated in the same group as the others, with the new Theory group being the one shown in Figure 8.

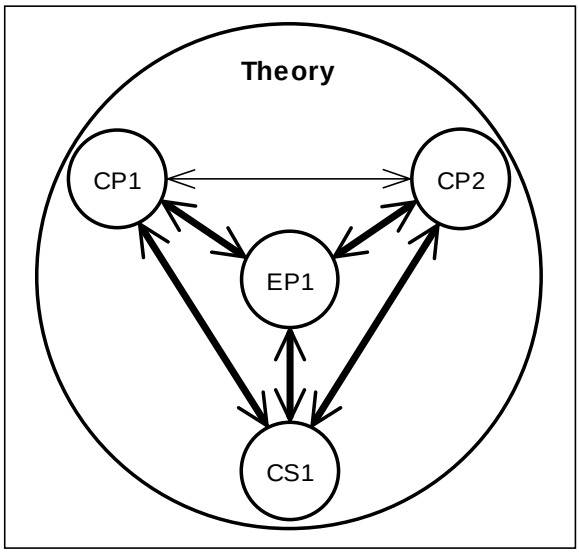

Figure 8 - New configuration of the Theory group

3.2.4. Activity-Centred Metamodel: Some Conclusions

From the examples above, we observe the usefulness of this Activity-Centred metamodel, capable of accommodating different perspectives and allowing the designer to experience models and choose the one that best fits his requirements. We can also observe that people necessities and physical space resources, such as a shared workspace, guide the model configuration.

Another important aspect guiding the model's topology is the information that should be available at each defined node of the model. While elaborating a model, it is important to consider not only the topological aspects, but also the aspects involving the collaboration itself, i.e., how group members can better cooperate by producing, manipulating, and organising information, and by building and refining cooperation objects.

It is also important to know how to reconfigure the model, since it seems that there are some configurations that work better than others. Without an automatic reconfigurator, it should be interesting at least to derive some pre-defined adequate configurations that could be selected when starting a collaborative application.

It should also be stated that this multi-perspective metamodel intrinsically does not constitute a different metamodel when compared to many others already developed, in such a way that we can use many of the specification languages available to describe it, as we will see further in this chapter. What really seems to be this metamodel's contribution is its expressiveness capacity, which helps to clarify, via a simple graphical representation, the relationships among the groups involved in the main activity being performed. We have observed with real users that this simple representation provided them with a better understanding of the problem and also stimulated them to discuss and propose modifications in the configuration of the models. Also the thin and thick edge arrows rapidly provide the notion of who is locally or remotely located to the other, as well as indicate the necessity of lower or higher requirements in terms of communication network.

Through a detailed analysis, we can identify the three components of the 3C model in our Activity-Centred metamodel, which are: communication, coordination, and cooperation (Ellis et al., 1991; Fuks et al., 2005). The cooperation component sometimes is given different names, but representing the same aspects: collaboration (Ellis et al., 1991), production (Laurillau & Nigay, 2002), and work coupling (Neale et al., 2004).

The communication component is the one related to the exchange of messages and information among people (Fuks et al., 2005). In our metamodel, it is represented by the edges that link the nodes.

The cooperation component is the joint operation taking place during a session on a shared space. Group members cooperate by producing, manipulating, and organising information, and by building and refining artefacts. We can then conclude that in our metamodel the cooperation component is formed by the association of the many different abstraction levels, which represent the whole activity being performed during the collaborative session, including all the applications being executed in the leaf nodes and the artefacts being produced or manipulated – i.e., the workspace that is being modelled. All these elements are

being stored during the collaborative session and may be used as a knowledge base for post-auditing or reconfiguration of the model in future sessions.

Finally, the coordination component is related to the management of people, their activities, and resources (Raposo & Fuks, 2002). In loosely coupled group work, in which the processing sequence is to result from the course of the participants' actions, the coordination is done implicitly. On the other hand, if the work involves tightly coupled activities, there is the need to prescribe the order of actions, as occurs in workflow systems (Pankoke-Babatz & Syri, 1997). Our metamodel, with the elements already considered, is capable of representing loosely coupled group work. In order for the metamodel to also accommodate tightly coupled activities, we have identified the necessity to introduce new elements in the metamodel, which will correspond to the coordination component of the 3C model and will be described in the following section: *edge specialisation elements*, *role rules*, and *message attributes table*.

3.3. Activity-Centred Metamodel: Coordination Components

As already mentioned, we have identified the necessity to introduce new elements in the metamodel, corresponding to the coordination component of the 3C model, which are going to be described in the next subsections: *edge specialisation elements*, *role rules*, and *message attributes table*.

3.3.1. Edge Specialisation Elements

Analysing a message across the interaction path from a sender to a receiver node, we have identified some additional requirements that our metamodel should accomplish. Suppose for example a situation where you have a group of engineers using desktop computers that send a video to another group of engineers located in a visualisation room, with larger display size. It would be interesting for this second group to automatically adjust some video properties to their condition before displaying it on the visualisation room screen. This could be done by what we call a *post-communication processing module*, being executed by the receiver group node immediately after receiving the message via the

communication channel and immediately before delivering it to the group members.

Another example could be the use of a video capture tool during a simulation collaborative session. Inside the technical group, all members should receive all the frames from a simulator being executed in one leaf node. On the other hand, for the managers being part of the decision makers' group it would be sufficient to receive only one every ten frames. The solution here would be to add a *pre-communication processing module* being executed by the sender group node, acting like a filter, immediately before sending the message (one frame) through the communication channel.

A third and even broader example would be sending a message from one group of Brazilians to one group of Chinese with their common language being English, considering, for the purpose of this example, that there is no Portuguese-Chinese-Portuguese translator available. In this case, we should use both pre- and post-communication processing. The first one would be executed by the sender node, translating the message from Portuguese to English immediately before sending the message through the communication channel. The message would then traverse the communication channel reaching the receiver node. There, a post-communication processing module would finally translate it from English to Chinese, then sending it to the receivers.

From the above examples, we conclude that it would be interesting to separate these pre- and post- communication processing modules into two different classes:

- The first class is constituted by the pre- and post-processing modules directly associated with the leaf nodes. They represent the processing modules to be executed particularly onto a specific message being passed between two nodes and are stored in an especial table with keys (*message_id*, *receiver*), as it will be described in Subsection 3.3.3.
- The second class is the one constituted by the pre- and post-processing modules associated with groups on different levels of the metamodel hierarchy, representing the policies of these groups when respectively sending (out-policies) and receiving (in-policies) messages. Since these are not particular policies related to one

specific message, but more general ones associated with groups on different levels of the metamodel, they can be stored in the metamodel itself.

In order to better illustrate this idea, we show in Figure 9 the different pre- and post-communication processing modules that could be executed while sending a message from a Computer Science Researcher CR1 of the Computer Science Department CD1 of University U1 to a Computer Science Researcher CR2 of the Computer Science Department CD2 of University U2.

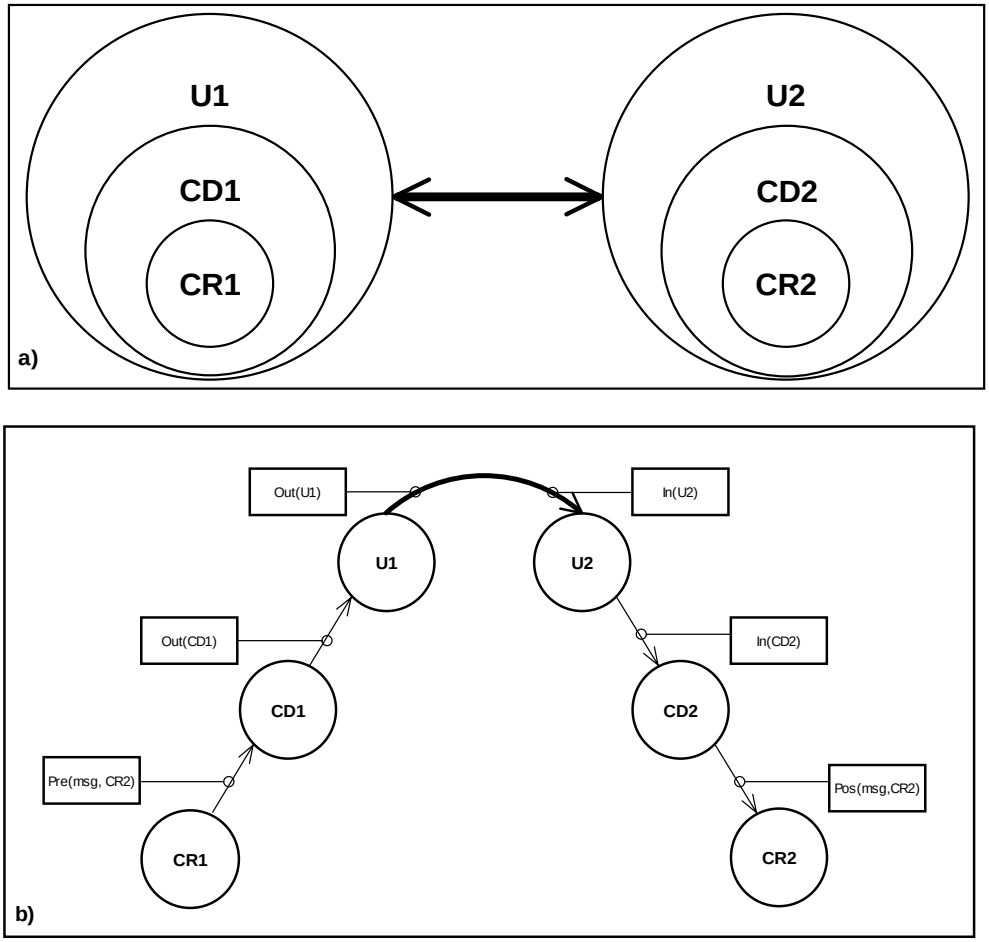

Figure 9 - Activity-Centred metamodel: pre- and post-communication processing: a) metamodel point-of-view; b) pre- and post-processing point-of-view

In Figure 9, we can follow the sequence of processing modules executed:

- The first pre-processing module to be executed is the one associated with the particular message being sent from CR1 to CR2.
- Then the out-policy associated with department CD1 is executed.

- Finally the out-policy associated with university U1 is executed.
- The message is then sent through the specific communication edge (the one linking the top-most nodes at either side).
- Now at the receiver side of the edge, the first post-processing module to be executed is the in-policy associated with university U2.
- Then the in-policy associated with department CD2 is executed.
- Finally a third post-processing module, associated with the particular message being sent from CR1 to CR2, is executed.

From a further analysis of this last example, two important questions arise. The first is related to who will execute these pre- and the post-communication processing modules. At the sender side of the edge, the natural candidate to execute the pre-processing modules is the leaf node who is sending the message. At the receiver side of the edge, this could be accomplished by adding an attribute to the first group node pointed by the edge (in our example, university U2), correspondent to the leaf node of this group which will execute the post-processing modules.

The second question involves the algorithms to be executed when sending a message at both the sender and the receiver side. We will present these algorithms later, after defining the two other coordination components of our metamodel in the next subsections: the role rules and the message attributes table.

The edge specialisation elements increase the metamodel's flexibility in two different aspects. One is the concentration of attribute values in modules that can be dynamically changed during run-time. The second aspect emphasises the model's adherence to social protocols (Morris, 2004b) involving collaborative applications. As an example of this social protocol perspective, in our metamodel the post-communication processing gives the responsibility to a leaf in a receiver group node to decide to which members of its group to send a received message. This seems reasonable, since it knows its group well and was designated by them to assume this role. As another example, retaking the practical example discussed about the video capture tool, the sender group could decide that it would be more polite and convenient to send all the frames to the decision makers' group and let them decide to cut or not some frames (the filter would be included in the post-

communication processing). Of course, this second approach would increase the network load, but this would be the price paid for a more “political” social protocol.

We can give some other practical examples of use of pre- and post-processing:

- File format conversion (ex.: CAD) from one system at one side of the edge to another system at the other side. This could be done at either side or at both sides if, for example, a neutral file is transmitted through the communication channel.
- UTM to XYZ coordinates transformation (at either side).
- Virus detection and firewall policies (at either side).

Ending this subsection, it is interesting to note that these concepts of pre- and post-processing seem to be usually considered in studies about collaborative applications, although not always having the same approach. For example, employing the same approach, Cortés and Mishra (1996) use *pre* and *postexecution* functions in their DCWPL coordination programs. Also David and Borges (2001) propose the use of input and output filters by the application as well as by the user, to assure privacy of information, select the events, the awareness information, and/or modify the processing of these events. Dourish (1998) uses these same concepts in his Prospero toolkit, now naming them *before-methods* and *after-methods*, which are executed by the toolkit. Finally Stevens and Wulf (2002), now employing a different approach, use the terms *ex-ante*, *uno-tempore*, and *ex-post* associated with the time in which access control permissions are defined.

3.3.2. Role Rules

Role rules for the coordination structure have been employed in CSCW studies for more than one decade. For example, Furuta and Stotts (1994) present an evolution of the Trellis model in which group interaction protocols are represented separately from the interface processes that use them for coordination. Protocols are interpreted, so group interactions can be changed as a collaborative

task progresses. Changes can be made either by a person editing the protocol specification “on the fly” or by a silent “observation” process. Trellis mixes hypermedia browsing with collaboration support – a *hyperprogram* – with its model based primarily on a synchronously executed, transition-timed Petri net as the structure of the hyperprogram. The net notation used in Trellis is referred to as *colored timed nets*, or *CTN*. A Trellis hyperprogram is completed by entering annotations for the components of the CTN: the CTN is the task's description and the different annotations are the information required by the task. One important category of annotation is *content*. Fragments of information (text, graphics, video, audio, executable code, other hyperprograms) are associated with places in a CTN. Another category of annotation is *events*, which are mapped to the transitions of the CTN. A third category of annotation is the *attribute/value (A/V) pair*; a list of A/V pairs is kept with each place, each transition, each arc, and for the CTN as a whole.

Edwards (1996) presents Intermezzo, a system that allows users to control collaboration by enacting *policies* that serve as general guidelines to restrict and define the behaviour of the system in reaction to the state of the world. Policies are described in terms of access control rights on data objects, and are assigned to groups of users called *roles*. Roles represent not only statically-defined collections of users, but also dynamic descriptions of users that are evaluated as applications are run. This run-time aspect of roles allows them to react flexibly to the dynamism inherent to collaboration. Intermezzo provides mechanisms for creating both static and dynamic roles, an implementation of policies on top of the “raw” access control system, and a specification declarative language for roles and policies that can be used to control and configure the policy subsystem. While static roles are implemented using simple access control lists, dynamic roles are implemented by associating a predicate function to a set of access control rights.

Cortés and Mishra (1996) propose that a collaborative program should be divided into two main components: a computational program that models the shareable artefacts and a coordination program that specifies the way these artefacts need to be shared. The coordination programs then can be easily modified, often without any change to the computational program. They have developed a coordination language – DCWPL (Describing Collaborative Work Programming Language – pronounced “decouple”) – and its run-time interpreter

to allow programmers to create coordination programs, specifying control mechanisms decoupled from computational programs written in some traditional programming language. A coordination program in DCWPL is composed of one or more coordination primitives: artefacts, roles, storage, coordination functions, policies, and session management.

Li and Muntz (1998) propose COCA (Collaborative Object Coordination Architecture) as a generic framework for developing evolvable collaborative systems and modelling the coordination policies. As in the three previously mentioned works, they also favour the idea of separating coordination and computation. COCA provides the developers a logic-based specification language for modelling coordination policies. In COCA, the participants in a collaboration are divided by roles, and active rules are defined for each role to guard its interactions with other collaborators. These policies are interpreted at run-time by the COCA virtual machine, a copy of which runs on each participant machine. This approach makes it convenient to make changes both during development and run-time. Policies in COCA include not only access control, but also session control, concurrency control, floor control, etc.

Roussev et al. (2000) take a component-based approach for separating data and control. The shared state of a data component is modelled as component properties and state changes as property change events. The composable design is based on programming patterns that eliminate the implicit binding between the logical structure of a shared object and particular system-defined abstractions, thereby increasing the range of supported objects and supporting extensibility.

Laurillau and Nigay (2002) present the Clover architectural model, resultant from the combination of the layer approach of Dewan's (1999) generic architecture with the functional decomposition of the Clover design model. The Clover design model defines three classes of services that a groupware application may support, namely production, communication, and coordination services. These three classes of services can be found in each functional layer of the model. The Clover functional partitioning establishes a direct mapping between the design concepts and the software architecture modelling. This partitioning serves as a guide in the organisation of the functionalities identified during the design phase. In addition, it is complementary to the traditional partitioning of functionalities into components, where each of them corresponds to one level of

abstraction, as for instance in Dewan's architecture. Indeed, for a component corresponding to one level of abstraction, the Clover metamodel advocates three sub-components dedicated to production, communication, and coordination. This modular implementation is in accordance with the modifiability, reusability, and extensibility properties. In the Clover metamodel, the coordination, production, and communication functionalities are all treated in the same way. This is in contrast to other studies, such as in Dewan's architecture, where the coordination functionalities are treated as especial whereas the production and communication functionalities are not. The conceptual architecture is mapped into an implementation architecture by assigning processes to components; the processes in turn must then be assigned to hosts. Different approaches to distribution can be applied to a conceptual architecture. Distribution and layer decomposition (or software component decomposition) are two orthogonal mechanisms. In particular, a shared layer at the conceptual level does not imply a central site at the implementation level.

Yang and Li (2004) then retake the component-based approach considered previously by Roussev et al. (2000), also separating data and control, but now supporting adaptable consistency protocols in collaborative systems. On clearly separating data and control, they allow consistency protocols to be dynamically attached to shared data at the object level. Protocols can be switched at run-time without modifying the source code.

In accordance to all these studies, except somehow for the Clover model, we decided to adopt the strategy of separating the coordination structure and the computational program. In our metamodel, we use a logic-based specification language for specifying the coordination policies. We will provide more details about this specification language in Chapter 5. For the time being, it suffices to emphasise the following points:

- We declare a collaboration bus which is used to connect all the participants in this collaboration, with the collaboration bus having at least one channel declaration. We allow many different collaborations being executed at the same time, each one with its correspondent collaboration bus.
- Roles are defined specifying individual behaviours and constraints in different sets of logical rules.

- Communication among the participants occurs through one or more message channels associated with one collaboration bus.
- The basic tasks of receiving and sending out messages are performed by:
  - For receiving messages, we use an active rule named *on-arrive* with arguments *channel*, *receiver*, *message_id* (and *sender*).
  - For sending out messages, we use a *send* formula with arguments *channel*, *sender*, *message_id* (and *receiver*).

We now explain why the last arguments of both tasks appear in parenthesis:

- For the task of receiving messages, the *sender* is optional. For the task of sending out messages, if for example IP multicast is adopted as the group communication model, the message is delivered to all the receivers, making the *receiver* argument unnecessary (in case of a unicast service, we would need the *receiver* argument).
- However, the main reason for placing those arguments in parenthesis is that, to enhance flexibility, we decided to build a *message attributes table* which includes the *sender*, the *receiver*, and other attributes of each message, as we will see in the next subsection. In this case, changing the receiver of a message would simply be a line change in a table instead of modifying the coordination program. We then make some modifications on the rule and on the formula used in COCA (Li and Muntz, 1998), which turn respectively into:
  - on-arrive(channel, message_table(*dummy*,message_id,receiver));
  - send(channel, message_table(sender, message_id, *dummy*)).

A typical coordination program in our metamodel would have a form similar to the one shown in Table 1. There, *self* is a functor denoting the current participant, *source* associates the participant id with the message, and *display* is used to exhibit the message content on the screen.

Following this coordination program line by line, we have:

- The first line defining the collaboration with name *integrated_simulation*.

- The second line defining a collaboration bus with one single *remote* channel.
- The third line opening a set of rules correspondent to role *technician_0* (which can be instantiated by participants during the collaboration execution).
- Then, we define the three rules constituting the role *technician_0*:
  - The first rule, *on-init*, is fired when the collaboration *integrated_simulation* starts. When this happens, a message with *message_id* 1 is sent to the receiver determined by the line in the message attributes table with key *sender*=*source*(*self*) (i.e., the present participant instantiating role *technician_0*) and *message_id*=1.
  - The second rule, *on-arrive*, is activated when a message with *message_id* 2 is received by *source*(*self*). When this happens, the message is displayed on the participant's console.
  - Finally the third rule, another *on-arrive*, is activated when a message with *message_id* 3 is received by *source*(*self*). When this happens, the message is displayed on the participant's console.

```
collaboration integrated_simulation
{ collaboration_bus { channel(remote). }
  role technician_0 // a technician responsible for coordinating the simulations
  {
        on-init(integrated_simulation) :-
              send(remote, message_table(source(self), 1, dummy)).
        on-arrive(remote, message_table(dummy, 2, source(self))) :-
              display(message_table(dummy, 2, source(self))).
        on-arrive(remote, message_table(dummy, 3, source(self))) :-
              display(message_table(dummy, 3, source(self))).
  }
}
```

Table 1 - Activity-Centred metamodel: a typical coordination program

In general, role rules define coordination policies for each role present in a collaboration. When these rules also determine the order in which the events occur, they also define workflow rules.

3.3.3. Message Attributes Table

We have introduced pre- and post-communication processing modules associated with each message sent, so it would be a natural choice to build a table where we could define which processing modules to be executed at each time. Also it would be easy to change the name of a particular pre- or post-processing module even in run-time (besides having the possibility to also change the code in run-time before it is invoked).

We built a *message attributes table* in our metamodel to enhance the flexibility of the coordination program, separating the coordination rules from data related specifically to each message. This is adherent to the overall concept of our metamodel, in separating data and control, with this indirection allowing dynamic reconfiguration. The only exception to this concept is that we decided to include the *receiver* (related to control instead of data) as a table column. This was motivated by the fact that, in tightly coupled workflows, it seems to be interesting to at least change receivers of a particular message in run-time, without having to change a coordination program. For example, a person from the sender group possibly would not know exactly to whom at the receiver side he should send the message. Then he could send the message to the abstract receiver group node, allowing it to determine via its post-processing module to which receivers the message would be sent. Another example could be that of a latecomer observer (e.g., a manager not responsible for making the decisions) joining a collaborative session and wanting to receive every message from that moment onwards. It would be a matter of editing the table to include new lines associating the future messages with the latecomer. Note that the edges connecting this latecomer to the other participants should have been anticipated and included in the model before the beginning of the collaboration, and that the latecomer should assume an already defined role that does not interfere in the main process.

We now show in Table 2 the first lines of a typical message attributes table associated with a tightly coupled collaborative application.

sender	message_id	receiver	edge	pre-processing	post-processing
T0	1	T1	e1	Pre_1_T1	Pos_1_T1
T1	2	T0	e1	Pre_2_T0	Pos_2_T0
T1	2	G1	e2	Pre_2_G1	Pos_2_G1
T1	3	T0	e1	Pre_3_T0	Pos_3_T0
T1	3	G1	e2	Pre_3_G1	Pos_3_G1

Table 2 - Columns and first lines of a typical message attributes table

The keys of this table are *sender*, *message_id*, and *receiver*. As we have already seen in Subsection 3.3.2, we enter the table using the pair (*sender*, *message_id*) when sending a message and we use the pair (*message_id*, *receiver*) to receive a message.

3.3.4. Sender and Receiver Algorithms

We now present the algorithms executed when sending a message at both the sender and the receiver side.

First we note that each message has one initial sender, which is necessarily a leaf node, and one or more final receivers, which can be either leaf or group nodes. After having presented the role rules and the message attributes table, we can now derive the algorithms to be executed at either side, as shown in Table 3.

We consider the message attributes table of the previous subsection (Table 2) to analyse the algorithms and derive in which moments the pre- and the post-processing modules are executed for each message. It is important to emphasise two points:

- G1 (for Group 1) is not a leaf node, but a group node. According to the algorithms, the post-processing module *Pos_2_G1* is responsible for determining to which leaf nodes in group G1 the message (2, G1) should be delivered.
- Although sending the same message to each receiver gives our metamodel more flexibility, with the possibility of executing pre- and post-communication processing modules particular to each receiver, with many advantages such as the ones mentioned in the examples given in Subsection 3.3.1, we should also consider the

impact that this strategy has on performance. As Li and Muntz (1998) reported, early models were largely based on grouping of multiple point-to-point communications. In this case, a packet with *n* intended receivers must be delivered *n* times by the sender, regardless of the physical location of the involved parties. This approach incurs overhead on both the message senders and the communication paths, and latency increases with the size of the receiver population. We should then balance our necessities when choosing a strategy to be used in a collaborative application. For example, it seems adequate to choose our algorithm if we have few nodes and great diversity among the nodes. On the other hand, it could be more efficient to choose an IP multicast communication model if we have many nodes with not so much diversity among them.

sender (sender, receiver, flag)
- until the receiver is found repeat
  - at the current level, search for the sub-tree that contains the receiver
  - if the receiver is found (and all the path from the sender to the receiver is determined)
    - if flag = in_table
      - execute the pre-processing module associated with the pair (message, receiver)
    - else
      - create a new line in the message attributes table with pair (message, receiver) indicating the post-processing module to be executed
    - execute all the out-policies associated with groups on levels in the path beginning at the sender until the communication edge is reached
    - send the message with the receiver to the leaf node which is assigned to the post-processing attribute of the receiver group node, or to the receiver itself
  - else
    - go to the upper level

Sender side of the communication edge (executed by the initial_sender leaf node):
- for each final_receiver associated with the message
  - sender (initial_sender, final_receiver, in_table)

Receiver side of the communication edge:
- receive the message
- execute all the in-policies associated with groups on levels in the path beginning at the present node until the final_receiver node is reached
- if the final_receiver is a leaf node
  - if it is equal to the post-processing execution node
    - execute the post-processing module associated with the pair (message, final_receiver)
  - else
    - send the message to the final_receiver
- else
  - execute the post-processing module associated with the pair (message, final_receiver), which in this case should determine the leaf node(s) or group node(s) to receive the message
  - for each of the node(s) determined above (current_node)
    - sender (final_receiver, current_node, not_in_table)

Table 3 - Activity-Centred metamodel: sender and receiver algorithms

We now describe both algorithms, at the sender and at the receiver side, shown in Table 3. First we detail a common method, named *sender*, which is used

at both sides. The basic purpose of this method is to find, in the network component of our metamodel, the receiver node to receive a message, determining the complete path from the sender to the receiver. The method has three arguments: the *sender*, the *receiver*, and a *flag* determining if the present message is or is not already in the message attributes table. It has a main loop *until…repeat* to find the *receiver*, which is executed beginning by searching for the *receiver* in the sub-tree defined by the level immediately above the *sender* (a leaf node) level, going upwards executing the same step until the *receiver* is found. When this happens, we have two possible situations:

- The message is already in the message attributes table:
  in this case, the pre-processing module associated with the pair (*message_id*, *receiver*) is executed.
- The message is not in the message attributes table:
  this means that this is a new message being created in run-time by a group node at the receiver side, defining a new line in the table including its post-processing module.

Ending the *sender* method, we have two more steps:

- The first one executes all the out-policies associated with groups on levels in the path beginning at the *sender* until the communication edge is reached.
- The second one finally sends the message with the *receiver* to the leaf node which is assigned to the post-processing attribute of the *receiver* group node, or to the *receiver* itself (in case of a leaf node).

Now we describe the sender side algorithm. With the *sender* method defined above, this algorithm, executed by the *initial_sender* leaf node, is simply a *for* loop repeating the following step for each *final_receiver* (determined in the lines of the message attributes table) associated with the current message being sent:

- *sender* (*initial_sender*, *final_receiver*, *in_table*).

We can observe that the method is invoked with *flag=in_table*, meaning that those are messages already pre-defined in the message attributes table.

Finally we now describe the receiver side algorithm:

- The first step, executed by the post-processing attribute of the *receiver* group node or by the *receiver* itself (in case of a leaf node), is exactly to receive the message.
- The second step is to execute all the in-policies associated with groups on levels in the path beginning at the present node until the *final_receiver* node is reached.
- The third and last main step is executed differently, depending on whether the *final_receiver* is or is not a leaf node:
  - If the *final_receiver* is a leaf node:
    - The *final_receiver* can be the post-processing execution node itself, when it simply executes the post-processing module associated with the pair (*message_id*, *final_receiver*), or they are different, in which case the post-processing execution node still has to send the message to the *final_receiver* leaf node.
  - If the *final_receiver* is a group node, two steps are executed:
    - The post-processing module associated with the pair (*message_id*, *final_receiver*) is executed, which in this case should determine the leaf node(s) or group node(s) to receive the message.
    - A *for* loop repeating the following step for each node determined in the step above (the *current_node*) is executed:
      - *sender* (*final_receiver*, *current_node*, *not_in_table*).

      We can observe that, in this case, the *sender* method is invoked with *flag=not_in_table*, indicating that those are new messages being created in run-time by a group node.

3.4. Activity-Centred Metamodel: Specification Language for the Network Component

As we have seen in Section 3.2, the main component of our Activity-Centred metamodel is the network constituted by nodes and edges defining,

respectively, the groups performing the activity and the interaction paths among them.

We should use a specification language to describe this network, choosing among many of the specification languages available, such as Process Algebra (Milner, 1989), Petri Nets (Murata, 1989), Unified Modeling Language (UML) Diagrams (UML, 2006), and Use Case Maps (UCM) (Buhr & Casselman, 1996). We have then selected the specification language proposed (Russo, 1988; Santos, 1986) for the PUC-Rio EITIS – Environment of Integrated Tools for Interactive Systems – project (Melo, 1987) mainly because of two reasons:

- Its simplicity, inspired on a modified BNF (Backus-Naur-Form) notation, allowing us not to lose focus on the present work being developed.
- The fact that we have already used it in a previous project of the Computer Science Department of PUC-Rio.

We now give a brief description of the notation used in our specification blocks:

- In capital letters, we represent the entities and the relationships, as well as the attributes being described and the types and functions of the entities.
- In small letters, we represent the attributes that are not being described and the syntax associated to the attributes being described.
- The + signal indicates a possible occurrence of more than one instance of a specific object.
- The | signal represents a logical “OR”.
- The relationships or attributes in brackets may or may not occur in a specific instance of the entity being considered.

The specification blocks defining the main entities and relationships of the network component of our metamodel are shown in Table 4.

NODE: node-id
DESCRIPTION: text
NAME: text
TYPE: LEAF | GROUP
[PARENT: node-id]
[POST_PROCESSING EXECUTION NODE: node-id]
[HOST: host-id]
[ATTRIBUTES: {attrib-name attrib-type attrib-value}+] //user interface preferences, language
[SHARES: artefact-id+]
[IN-POLICY: policy-id]
[OUT-POLICY: policy-id]

EDGE: edge-id
DESCRIPTION: text
DISTANCE: LOCAL | REMOTE
DIRECTION: UNIDIRECTED | BIDIRECTED
SENDER: node-id
RECEIVER: node-id
HAS: channel-id+

ARTEFACT: artefact-id
DESCRIPTION: text
NAME: text
TYPE: many
SYNCHRONY: SYNCHRONOUS | ASYNCHRONOUS
PERSISTENCY: PERSISTENT | VOLATILE
[ACL: acl-id] //Access Control List

CHANNEL: channel-id
DESCRIPTION: text
NAME: text
TYPE: many
SYNCHRONY: SYNCHRONOUS | ASYNCHRONOUS
PERSISTENCY: PERSISTENT | VOLATILE

Table 4 - Network component: specification blocks defining entities and relationships

These specification blocks should be then stored in a data structure with a Data Description Language (DDL) similar to the one shown in Table 5 (there we describe only the main entities of the network component, i.e., nodes and edges).

```
RECORD node
ITEM node-id NUM 2 KEY
ITEM description CHAR 35
ITEM name CHAR 30
ITEM type CHAR 5
ITEM parent NUM 2
ITEM post NUM 2
ITEM host NUM 12
.
.

RECORD edge
ITEM edge-id NUM 4 KEY
ITEM description CHAR 35
ITEM distance CHAR 6
ITEM direction CHAR 11
ITEM sender NUM 2
ITEM receiver NUM 2
.
.
```

Table 5 - Network component: Data Description Language (DDL)

Finally, we present in Table 6 the load records for nodes and edges correspondent to the Data Description Language of Table 5.

Load Records

Node

rec-type	node-id	description	name	type	parent	post	host	…

Edge

rec-type	edge-id	description	distance	direction	sender	receiver	…

Table 6 - Network component: load records for nodes and edges

3.5.
Activity-Centred Model: A Simple Complete Example

We now consider a simple, but complete, example (Figure 10) to illustrate how we describe the three main components of our metamodel: the network, the role rules and the message attributes table.

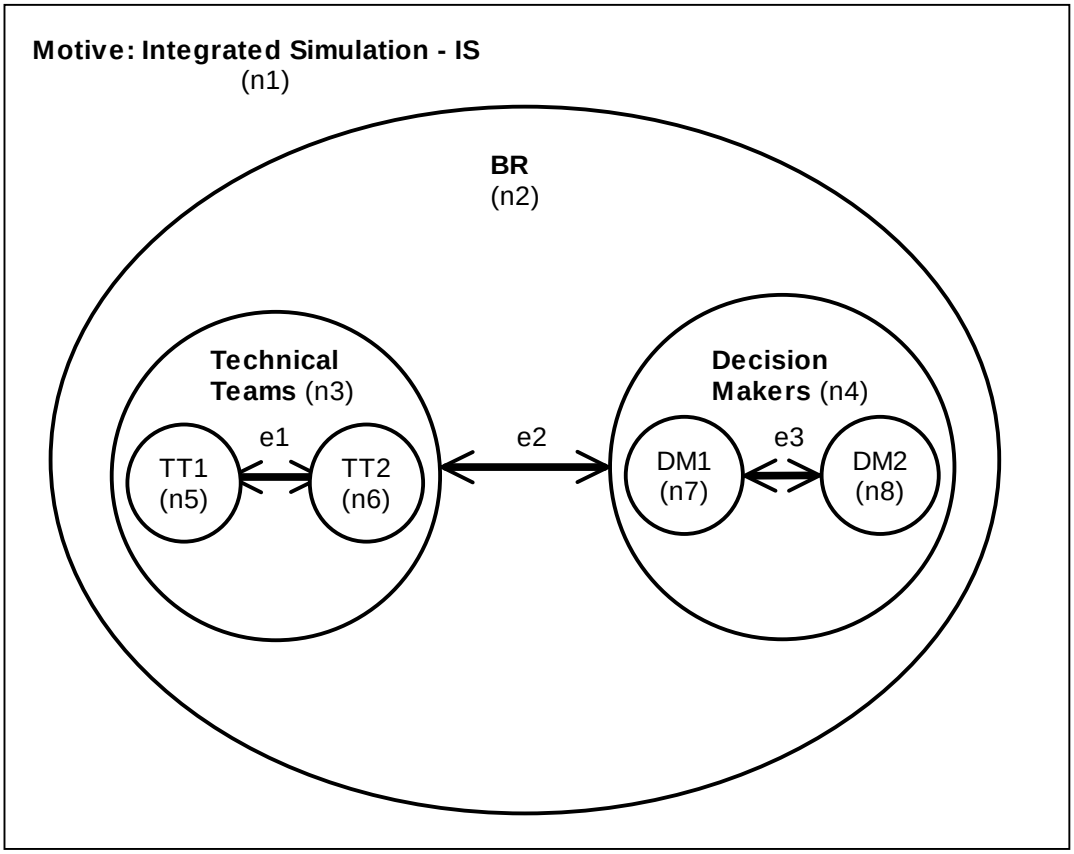

Figure 10 - A simple complete example of an activity-centred model: Integrated Simulation

The top level node of this example, the motive, is Integrated Simulation (n1). It has one main node, the oil & gas company BR (n2), constituted by two groups: Technical Teams (n3) and Decision Makers (n4).

Technical Teams is constituted by two leaf nodes, technician TT1 (n5) and technician TT2 (n6), while Decision Makers is constituted by two other leaf nodes, decision maker DM1 (n7) and decision maker DM2 (n8).

Also represented in the network component there are the edges e1, e2, and e3, showing the interaction paths among the nodes, all with thick arrows (remote).

The load records for these nodes and edges are shown in Table 7.

1 | 1 | Integrated Simulation | IS | GROUP | 0 | 5 | …
1 | 2 | Oil & Gas Company | BR | GROUP | 1 | 5 | …
1 | 3 | Technical Teams | TT | GROUP | 2 | 5 | …
1 | 4 | Decision Makers | DM | GROUP | 2 | 7 | …
1 | 5 | Technician 1 | TT1 | LEAF | 3 | 0 | …
1 | 6 | Technician 2 | TT2 | LEAF | 3 | 0 | …
1 | 7 | Decision Maker 1 | DM1 | LEAF | 4 | 0 | …
1 | 8 | Decision Maker 2 | DM2 | LEAF | 4 | 0 | …
2 | 1 | technicians channel | REMOTE | BIDIRECTED | 5 | 6 | …
2 | 2 | technical-manager channel | REMOTE | BIDIRECTED | 3 | 4 | …
2 | 3 | managers channel | REMOTE | BIDIRECTED | 7 | 8 | …

Table 7 - Activity-centred model: load records for the network component

In this table, we can identify the seven nodes and the three edges constituting the network component of the integrated simulation model. In terms of nodes, we can see that the records define which nodes are groups and which ones are leaves, what are the parents of the nodes (0, when the parent is the motive), and what are the post-processing execution nodes for the group nodes (0, when the nodes are leaf nodes). For example, the third line (1 | 3 | Technical Teams | TT | GROUP | 2 | 5 | …) starts by defining that it refers to a node (record type 1), followed by its *id* (3), description (Technical Teams), and name (TT). Then, the node type is defined as GROUP, and its parent is defined as the node with *id*=2 (defined in the previous line – BR). Finally, the last attribute shown defines the node that executes the post-processing module associated to this group (in this case, it is the node with *id*=5, defined two lines below – TT1).

In terms of edges, we identify which are the remote (all of them in this example) and the local ones, which are the bidirected (again all of them in the present example) and the unidirected ones, and the nodes defining the edges. For example, the last line (2 | 3 | managers channel | REMOTE | BIDIRECTED | 7 | 8 | …) starts by defining that it refers to an edge (record type 2), followed by its *id* (3) and description (managers channel). Then the edge is defined as REMOTE and BIDIRECTED. Finally, the last two attributes identify the nodes connected by the edge (in this case, nodes with *ids* 7 and 8 – DM1 and DM2, respectively).

We then show in Table 8 the role rules defined for *technician_1* instantiated by TT1 from the Technical Teams group.

```
collaboration integrated_simulation
{  collaboration_bus { channel(remote). }
   role technician_1 //Technician 1 from Technical Teams, responsible for coordinating the simulations
   {
   on-init(integrated_simulation) :-
     send(remote, message_table(source(self), 1, dummy)). //send “Technician 2, please begin simulation.”
   on-arrive(remote, message_table(dummy, 2, source(self))) :-
     display(message_table(dummy, 2, source(self))). //display “Simulation  has begun.”
   on-arrive(remote, message_table(dummy, 3, source(self))) :-
     display(message_table(dummy, 3, source(self))), //display “End of simulation.”
     send(remote, message_table(source(self), 4, dummy)). //send “Decision Maker, please validate the simulation.”
   on-arrive(remote, message_table(dummy, 5, source(self))) :-
     display(message_table(dummy, 5, source(self))). //display “Simulation validated.”
   }
}
```

Table 8 - Activity-centred model: role rules for technician_1

Following this coordination program line by line, we have:

- The first line defining the collaboration with name *integrated_simulation*.
- The second line defining a collaboration bus with one single *remote* channel.
- The third line opening a set of rules correspondent to role *technician_1* (which in this example is instantiated by participant TT1).
- Then, we define the four rules constituting the role *technician_1*:
  - The first rule, *on-init*, is fired when the collaboration *integrated_simulation* starts. When this happens, a message with *message_id* 1 and content “Technician 2, please begin simulation.” is sent to the receiver determined by the line in the message attributes table with key *sender=source*(*self*) (i.e., TT1) and *message_id*=1. In Table 9, this line is the first one, and indicates that the *receiver* is TT2.
  - The second rule, *on-arrive*, is activated when a message with *message_id* 2 is received by *source*(*self*) (i.e., TT1). When this happens, the message is displayed on the participant's console: “Simulation has begun.”

- The third rule, another *on-arrive*, is activated when a message with *message_id* 3 is received by *source*(*self*) (i.e., TT1). When this happens, message 3 is displayed on the participant's console – “End of simulation.” – and a message with *message_id* 4 with the content “Decision Maker, please validate the simulation.” is sent to the receiver determined by the line in the message attributes table with key *sender=source*(*self*) (i.e., TT1) and *message_id*=4. In Table 9, this line is the sixth one, and indicates that the *receiver* is DM.
- Finally the fourth rule, yet another *on-arrive*, is activated when a message with *message_id* 5 is received by *source*(*self*) (i.e., TT1). When this happens, the message is displayed on the participant's console: “Simulation validated.”

We could similarly define role rules for the other participants of the collaborative application. We should note that, since in this example the role rules are defining the order of the events, they can also be considered workflow rules.

We finish our simple example showing the message attributes table (Table 9) containing all the messages to be sent by the participants, which, together with the role rules, define the coordination structure of our metamodel.

sender	Message_id	receiver	edge	pre-processing	post-processing
TT1	1	TT2	e1	Pre_1_TT2	Pos_1_TT2
TT2	2	TT1	e1	Pre_2_TT1	Pos_2_TT1
TT2	2	DM	e2	Pre_2_DM	Pos_2_DM
TT2	3	TT1	e1	Pre_3_TT1	Pos_3_TT1
TT2	3	DM	e2	Pre_3_DM	Pos_3_DM
TT1	4	DM	e2	Pre_4_DM	Pos_4_DM
DM1	5	TT1	e2	Pre_5_TT1	Pos_5_TT1

Table 9 - Activity-Centred Metamodel: message attributes table for the BR model

In this table, we can also see the different pre- and post-processing modules to be executed while sending the messages, particularly messages 2 and 3, which have different processing modules associated with different receivers. Also it is important to note that message 2 sent from TT2, message 3 sent from TT2, and message 4 sent from TT1, all of them are destined to DM group. This means that the post-processing modules *Pos_2_DM*, *Pos_3_DM*, and *Pos_4_DM*, respectively, all of them executed by the post-processing execution node of the

DM group, DM1 (see Table 7, in which it has *node-id* 7), should determine which leaf node(s) of group DM should receive the messages. In our present example, we define that messages 2 and 3 are re-routed to both DM1 and DM2 (as they are simply follow-up messages), and message 4 is re-routed to DM1, since he is the one responsible for making the final decision about the integrated simulation.

We finalise this section noting that we should guarantee consistency among all the three parts of our metamodel. This means for example that the types of channel present in the role rules, local or remote, should be the same represented in the network component (thin or thick arrows). Also messages present in the message attributes table should only be defined if the correspondent edges linking the senders and the receivers in the network component exist. The sender and the receiver algorithms should verify this consistency during run-time. Finally, we should note that, in the *send* formulas used in the role rules, since the receivers are only indirectly determined via the message attributes table, we only indicate the channel used to communicate with the main receiver of the message. The channels used to communicate with the other receivers are determined while identifying the receivers and the edges used (and their channels) in the message attributes table.

4
Technological Approaches for Developing Distributed Virtual Environments

After elaborating a metamodel to help configure the collaborative virtual workspace architecture, in this chapter we focus on reviewing and analysing existent Distributed Virtual Environments (DVEs) in order to select which ones have the more appropriate features to serve as a basis to support the implementation that will be developed to validate our metamodel.

We cover two of the current technological approaches for creating and building distributed systems, namely Middleware and Pure Distributed Virtual Environments, summarising the systems that have been evaluated.

4.1. Considerations in Developing Collaborative Environments

An efficient Collaborative Virtual Environment (CVE), which is another name for DVE, should guarantee distribution support, an efficient network with large bandwidth, adequate communication, and data storage models. Furthermore this distributed environment should offer appropriate collaborative interfaces, operations and metaphors to enable the users to work effectively.

For a virtual environment to be collaborative, it must be distributed among the participants who wish to share it. The choice of communication architecture is parameterised by the degree to which the data structures representing the virtual environment are replicated or cached among the computing nodes and the underlying transportation technology (West & Hubbold, 2001).

However, whatever the technology, communication latencies are ultimately insurmountable. Real-time shared interaction is particularly sensitive to lag, especially in the case of immersive interfaces. The application's sensitivity to such lag depends on the degree of genuinely shared interaction desired. For example, in a passive walk-through of a model, small temporal discrepancies among the participants' experience of the world may well go unnoticed. At the opposite

extreme, the two people operating a stretcher through the confines of an oil rig will find coordination difficult or impossible with any appreciable lag.

If it is not possible to achieve the adequate synchrony, one solution is to at least focus resources upon those activities which are most sensitive to lag, i.e. those which produce the most pronounced discontinuities of perceptual experience when lag is present. This involves determining metrics for significance and prioritising urgency within the communication infrastructure accordingly (West & Hubbold, 2001). Ultimately, as the components of a CVE system interact in complex ways, the designer must regard the application as a unified system. Invariably, trying to optimise one element of a CVE can adversely impact the behaviour of other components. In effect, CVE development is a difficult balancing act of engineering tradeoffs (Singhal & Zyda, 1999).

It is impossible to predict the network requirements of CVEs in isolation; rather, we need a model of CVE operation which encompasses the application, user, software, and hardware concerns. Greenhalgh (2001) proposes a model that has six layers:

1. Task/application/collaboration requirements: what people want or try to do.
2. User behaviour: what particular actions people do and when. It is important to consider user behaviour when trying to understand the network requirements of CVEs because almost all of those requirements derive from what the users choose to do and when they choose to do it. For example, if users speak only rarely, and never at the same time, then the network requirement for audio could be very limited. On the other hand, for some scenarios of use there must be enough bandwidth for every user to speak at the same time. This could be the case, for example, of emergency systems.
3. Process behaviour: how the application responds. Every CVE system has its own set of capabilities and its own ways of representing users and virtual worlds. For example, each system supports a different subset of possible user actions. Once again, the emergency scenario could be a good example: while people ahead of the whole operation could execute any command, other specialists could only execute the tasks they were asked to.

4. Distribution architecture: what communicates with what. The choice of distribution architecture determines which information must be communicated to which parts of the system. The distribution architecture will therefore determine how virtual worlds are organised and divided, which users can communicate with each other, and whether or not there are any central coordinating processes or servers. The distribution architectures can basically be divided into three models: client/server, peer-to-peer, and hybrid.
5. Communication protocols: how information is exchanged. Protocols can be either unicast or multicast. The developer has to choose the best distribution architecture/communication protocol combination that fits the particular application being elaborated.
6. Network communication: what actually happens in the network. There are some network requirements that depend on exactly where in the network each application is running and how the network is connected (e.g. LAN or WAN), since the actual bandwidth in different parts of the network will vary. This also includes mobile systems.

The choice of distribution architecture is the main factor determining much of the multi-user "feel" of a CVE. For example, the distribution architecture will typically determine:

- The possible structure of a virtual world, e.g. whether it copes well with large outdoor spaces or with complex building interiors. In the oil & gas industry, the users have to deal with both situations, for example when building a complex offshore structure (complex interior) or monitoring oil pipelines disposed over a mountain (large outdoor space). The challenge becomes even more complex in some emergency scenarios, when specialists possibly have to go through many details of the structure using an egocentric point-of-view (indoor space), while having to observe the simulation effect of possible emergency operations using an exocentric point-of-view (outdoor space). In these cases, the system has to address simultaneously the indoor and the outdoor requirements.

- How many users may share a single virtual world. Typical oil & gas applications nowadays are held in no more than a half dozen visualisation rooms simultaneously, with no more than 20 specialists in each one. In some cases, specialists spread over the outdoor space may have the need to enter the virtual world using a mobile system.
- The dynamics of the virtual world, including the ways in which behaviours may be realised and what things may and may not be changed. For example, it is extremely hard to dynamically change basic geometries in some systems, whereas it is relatively easy in others. This is a very important issue of oil & gas applications considering the complexity of the data involved. What is usually done is the replication of the data model in the sites before the collaboration session begins. To avoid network overload, normally during a session the users are only allowed to do some incremental updates, transmitting only small data and/or action messages.

For the moment, it is fair to say that there is no universal choice of distribution or communication architecture, but rather a range of trade-offs in performance and deployment issues.

Another important requirement in distributed environments is the need for a data storage model for supporting collaborative activities. According to Macedonia and Zyda (1997), the data storage models can be classified as: *(i)* centralised and shared database; *(ii)* homogeneous worlds replicated database; *(iii)* distributed and shared database with peer-to-peer modifications; and *(iv)* distributed and shared client/server database.

Centralised databases provide an easy programming model. The developer can access information at any time, ignoring the existence of the network. The only indication of whether the data is cached or remote will be in the access time. Additionally synchronisation is assured as (ignoring caching) there is only one representation of each object in the database. The main drawback of centralised databases is a potential performance hit associated with accessing the data across a network.

Replicated databases are popular in many VR systems. Local representations of the data mean that local updates are fast.

Due to the high value of their data, oil & gas applications have strict database consistency and security requirements, suggesting a centralised and shared database model. The grid concept seems to match those requirements, since it is conceptually centralised with real data spread at various places transparently to the application. Nevertheless, due to performance requirements and as the grid is still a new concept, it can be observed that almost all the practical applications at the moment use the replicated model, some of them allowing the users to make what is called incremental updates.

4.2. Architectural Approaches

After discussing general considerations about developing collaborative environments, we are now going to focus on two current technological approaches for creating and building distributed systems, namely Middleware and Pure Distributed Environments, summarising the systems that have been evaluated.

4.2.1. Middleware

Systems based on the Middleware approach have the goal to create an interface that establishes a practical and portable standard for process communication. In the next subsections, we summarise five of those systems.

4.2.1.1. Message Passing Interface (MPI)

The goal of the Message Passing Interface simply stated is to develop a widely used standard for writing message-passing programs (Dongarra et al., 1993). The main advantages of establishing a message-passing standard are portability and ease-of-use. In a distributed-memory communication environment in which the higher-level routines and/or abstractions are built upon lower-level message passing routines, the benefits of standardisation are particularly apparent.

Message passing is a paradigm used widely on certain classes of parallel machines, especially those with distributed memory. Although there are many variations, the basic concept of processes communicating through messages is

well understood. Over the last ten years, substantial progress has been made in casting significant applications in this paradigm. Each vendor has implemented its own variant.

MPI-1 (Dongarra et al., 1993) was completed in early 1993. It focussed mainly on point-to-point communication. It did not include any collective communication routines and was not thread-safe.

MPI-2 (2006) added some extensions to the basic standard. MPI/RT is the latest instantiation. It incorporates numerous features, such as an API, support for heterogeneous environments, C and Fortran bindings, point to point and collective channels, a reliable communication interface, and thread safety.

4.2.1.2. CORBA and TAO

The Common Object Request Broker Architecture (CORBA) (OMG, 1995) is an open standard for communicating between local and remote processes. On top of packet transmission, CORBA offers a number of "standard" services which automate many common network programming tasks, such as object registration, location, and activation; request demultiplexing, and operation dispatching. At the base level, communication is defined by Remote Procedure Calls (RPCs) between processes. The developer defines the interface to the processes using the Interface Description Language (IDL). The IDL is compiled into an interface that exists between client and server. Thereafter, the communication details are handled by CORBA. The services mentioned above are then layered over this communication framework.

Initial implementations of CORBA did not deal with real-time processing issues overtly (although this did not preclude a small number of CORBA-based VR applications, e.g., COVRA-CAD (Junghyun et al., 1998)).

Doug Schmidt and colleagues at the University of Washington State developed The ACE ORB (TAO) (Schmidt, 2006), a CORBA-compliant middleware framework that addresses some of the real-time challenges of distributed processing. Schmidt and colleagues identify a set of patterns and framework components that can be applied systematically to eliminate many

tedious, error-prone, and non-portable aspects of developing and maintaining distributed applications.

4.2.1.3. Grid Computing and Globus

The grid can be defined by the following notional checklist:

"(A Grid…)

- Coordinates resources that are not subject to centralised control – (A Grid integrates and coordinates resources and users that live within different control domains – for example, the user's desktop vs. central computing; different administrative units of the same company; or different companies – and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings.)
- Using standard, open, general-purpose protocols and interfaces – (A Grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorisation, resource discovery, and resource access...)
- To deliver nontrivial qualities of service – (A Grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater than that of the sum of its parts.)" (Foster & Kesselman, 1998).

There are a number of software infrastructures for Grid computing (e.g. Globus (Foster & Kesselman, 1997), Legion (Grimshaw & Wulf, 1997), and SNIPE (Fagg et al., 1997)). Services such as authentication, program startup, and data transfer mechanisms are all included in the infrastructures.

Globus is arguably the most popular of these infrastructures, and has been implemented as the distribution mechanism in a number of VR frameworks (including CAVERNSoft (Park et al., 2000)).

Globus is designed to offer features such as uniform access to distributed resources with diverse scheduling mechanisms, information service for resource publication, discovery and selection, and enhanced performance through multiple communication protocols (Foster & Kesselman, 1997).

4.2.1.4. Common Component Architecture (CCA)

The Common Component Architecture (Common Component Architecture Forum, 2006) defines a component-based communication framework that conceptually sits above middleware such as CORBA or Globus, or lower-level IP.

There are numerous advantages to component programming when implementing high-performance visualisation tools. This is realised when one considers that the development of virtual reality applications is increasingly a multidisciplinary undertaking. By providing flexible core frameworks, the broader development effort can incorporate research teams with a significant range of skills. Component programming supports the different approaches and requirements inevitable in a multidisciplinary development effort. However, possibly the most interesting facet of component programming to the present work is that components can be configured to execute locally or at remote locations.

CCA provides a transparent means of defining and managing software components. In doing this it addresses component-level interoperability, that is, the flexible connection to an architecture by a compliant component through a well defined interface. There are a number of CCA-compliant frameworks currently in use, including CCAFFEINE (Armstrong et al., 1999) and XCAT (2006). The dataflow model in SCIRUN (Parker, 1999) also incorporates a CCA-compliant framework.

4.2.1.5. InfoGrid

InfoGrid, previously CSGrid (Lima et. al, 2005), is a client/server system for grid environments which, in addition to support for usage and management of distributed computational resources, offers facilities to integrate applications and

manage data and users (Figure 11). InfoGrid presents to its users, through a web browser, a workspace with all available applications and with each user's data files organised by project. A user can extend the system, adding new applications. InfoGrid also provides its users with some collaborative work facilities.

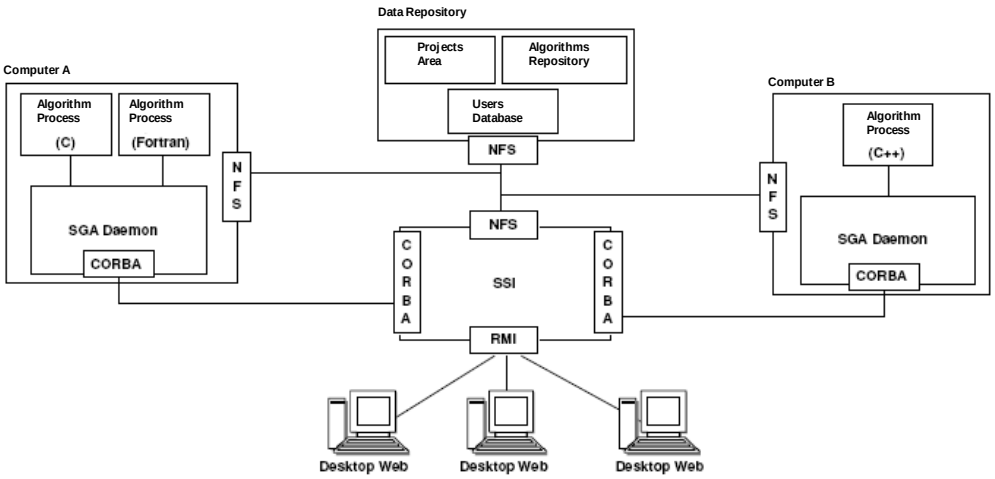

Figure 11 - InfoGrid architecture

Applications which are executed in the client utilise InfoGrid's available services to have access to and to manage distributed computational resources. One of these services is the remote execution of algorithms which are in computers linked to InfoGrid. For InfoGrid, algorithms are defined as executable programs implemented in any language which accept input parameters, generate an output, and do not have any type of interaction with the user during their execution. Many computers can be incorporated to the grid environment to serve as a platform for algorithm execution. New algorithms can be easily made available in the environment and the process to execute them is turned into a transparent task for the user.

An important characteristic of InfoGrid is the interoperability between the main server of the system and the servers being executed in the different computers for algorithm execution. Independently from the implementation language of the algorithms and the computers in which they are being executed, InfoGrid offers a standard interface for remote execution and monitoring of the environment computers.

InfoGrid has a *desktop web* where the users utilise the distributed computational resources as if they were in their local computers. Through this

client, they can create their own workspace, monitor the grid's computers, install algorithms, execute applications installed in the desktop, command the remote execution of algorithms, and monitor the processes which are executing these algorithms in the environment's computers.

InfoGrid was implemented as an extension of CSBase (2006), a framework for resource management and algorithm execution in a distributed and heterogeneous computational environment. CSBase is a result of different projects being developed by the joint partnership between PUC-Rio/Tecgraf and Petrobras. The first of these projects, WebSintesi, has implemented and made available an integrated environment for analysis and synthesis of geophysical data. In WebSintesi, multi-processing computers and clusters are linked to a main server that performs, transparently to the users, the remote execution of programs which require high performance processing and handle seismic data of hundreds of gigabytes.

After the first year of the WebSintesi development, the Dynamic InfoPAE project was started to attend Petrobras users who utilise emergency management applications, such as offshore stability analysis programs. To allow the reuse of the services already implemented for WebSintesi and to facilitate the development of new services and particular applications for the WebSintesi and for the Dynamic InfoPAE projects, the CSBase framework was created.

Because of all these developments, we decided to take InfoGrid as an example of this first approach (Middleware) to delineate a prototype (described in Chapter 5) that will be one of the proofs of our metamodel.

4.2.2. Pure Distributed Virtual Environments

In this subsection, we investigate the second approach for developing distributed virtual environments, namely Pure Distributed Virtual Environments, in which specific distributed environments are developed to meet the requirements. This approach offers a higher abstraction level when compared to Middleware, in the sense that their protocols, architectures and systems provide the developers with APIs (Application Programmer's Interfaces) which are closer to the application level than to the implementation level.

4.2.2.1. SIMNET, DIS, and HLA

Distributed Interactive Simulation (DIS) and its predecessor SIMNET are standards for distributed interactive simulations (Locke, 1993; Youngman, 2006). DIS retains a replicated database and uses dead reckoning to keep track of others. Dead reckoning is a strategy for dealing with network latency problems. It involves packaging an entity's location, a time stamp, and a velocity vector. The host application receiving the data can then predict where the object should be. Each entity runs a local simulation together with the dead reckoning model, and when the two diverge by more than a certain tolerance a message is sent out to all participants to reconcile the discrepancy.

DIS brings together several interesting ideas for distributed virtual reality:

- Standard message formats.
- No central server.
- Dead reckoning.
- Area of Interest Management (AOIM).
- Potential use of multicasting.

Multicast messages can be sent to a specified group of machines. Multicasting is popular in distributed virtual reality systems, particularly with systems that support large numbers of participants. Macedonia et al. (1995) state that some simulations have been shown to reduce network traffic by 90% by employing multicasting.

A number of applications implement the DIS protocols, such as VR-Link (Morrison, 1995), Close Combat Tactical Trainer (CCTT) (Mastaglio & Callahan, 1995), PARADISE (Singhal, 1996), and NPS Networked Vehicle Simulator (NPSNET) (Macedonia et al., 1995). NPSNET-V (Capps et al., 2000) is developed by the Naval Postgraduate School. It is implemented in Java, and incorporates the concept of an entity to represent a virtual world object. NPSNET-V defines object entities which contain information about artefacts in the virtual environment (e.g., tanks, people). The entities define state information, such as

location and velocity. From a list of defined protocols, each entity can be imbued with a set of behaviours.

The High Level Architecture (HLA) effort (Defense, 2006; Dahmann et al., 1999) aims to facilitate the interoperability and composability of the broadest range of component-based simulations. The HLA did not originate as an open standard, but was later recognised and adopted by both the Object Management Group (OMG, 2006) and the Institute of Electrical and Electronics Engineers (IEEE, 2006). HLA development occurred in much the same way as did the DIS standard – namely, the initial design was produced within the U.S. Department of Defense and then delivered to an external body (IEEE and/or OMG) for standardisation (Singhal & Zyda, 1999).

The HLA architecture can be thought as an object-oriented net-VE design. Each simulator, known as a *federate*, is a component that represents a collection of objects, each having a set of attributes and capable of initiating and receiving events. The federate registers each of its objects with a piece of middleware called Run-Time Infrastructure (RTI). The RTI collaborates with RTI instances in other hosts to learn about remote participants (objects) and delivers information about those participants to the local federate. The local federate, in turn, typically instantiates local objects representing those remote participants. Attribute updates and events are also exchanged through the RTI, which is responsible for handling area of interest management, time synchronisation, and other low-level net-VE services on behalf of the application. The collection of federates, along with their associated RTI instances, is termed a *federation*. Simulators in the federation send and receive state information via calls to and from the RTI (Singhal & Zyda, 1999). HLA as an IEEE standard will be better discussed later in this chapter.

4.2.2.2. DIVE

DIVE (Distributed Interactive Virtual Environment) (Swedish, 2006) is developed by the Swedish Institute of Computer Science. The development effort focussed primarily on the distribution component of the virtual environment. Certainly, in its early days DIVE emerged as an important example of how to develop distributed virtual reality.

The distribution element is conceptually quite simple. Shared memory is provided over a network and the system controls the sending of signals to processes. DIVE implements a distributed fully replicated, dynamic database. It has the capability to add new objects and modify existing databases in a reliable and consistent fashion. Reliable multicast protocols and concurrency control are employed through a distributed locking mechanism to facilitate database updates.

4.2.2.3. MASSIVE

MASSIVE (Greenhalgh et al., 2000) is an object-oriented system developed at the University of Nottingham to support multi-user collaborative virtual environments. MASSIVE is based on a 'spatial model' of interaction (Benford et al., 1993). It is mainly developed as an academic research tool, for researchers investigating online collaboration in virtual environments. The work is focussed on promoting awareness of others in the virtual environment.

The system supports interaction with the virtual environments via text, 2D and 3D graphical interfaces, and has support for real-time audio, allowing users to communicate with one another using speech. The emphasis is on user-to-user interaction, rather than on a user's interaction with objects in the environment. MASSIVE uses a combination of peer-to-peer and client/server processes. It employs multicasting as a collaboration baseline.

4.2.2.4. Avango

Avango is developed by the German National Research Centre for Information Technology (GMD) (Bierbaum & Just, 1998; Tramberend, 1999). It is ostensibly an object-oriented framework built atop Performer for constructing networked virtual environments.

The major subsystems (interaction, graphics, networking) in a VR application are integrated into one. It provides a replicated scene graph across the network. The objects can be instantiated as either local or distributed (public or private). It defines two categories of objects:

1. Nodes – define scene graph elements for rendering.

2. Sensors – import external device data into the application.

Avango is built on a dataflow model using fields within the objects to support a generic streaming interface. The dataflow graph defines the behaviour of the objects in the world.

Avango has a C++ API and a binding to an interpreted language, Scheme (Dybvig, 1996). Typically complex and performance-critical functionalities are implemented in C++. From then on, the application is implemented using Scheme scripts.

Avango uses a process group model. Each group member is guaranteed to receive delivered network messages in exactly the same order. In addition, when a new member joins the group, all communication is suspended until the new member's state is updated. This guarantees state consistency.

4.2.2.5. DEVA/MAVERIK

DEVA (Pettifer et al., 2000) is a distributed virtual reality kernel developed by the Advanced Interfaces Group at Manchester University.

DEVA supports databases comprising virtual objects. DEVA decouples a virtual object's representation into client-side and server-side objects. The server-side part represents the virtual object and defines its behaviour, some of which may be generic and some may be specific to the application. The client-side part contains interpretations of this behaviour, and only responds to instructions from the server. The messages from the server to the client are similar to a high-level vocabulary used to describe the effect of the behaviour. DEVA establishes a locus of control by managing states in one place. This means that the state of an entity can be managed locally rather than in the server if this is appropriate.

Rather than using dead reckoning to combat lag and jitter, DEVA uses “twines” (Marsh et al., 1999), which smooth over irregular updates.

DEVA is tightly coupled to a VR kernel (MAVERIK (Hubbold et al., 2001)), which provides rendering, input, output and spatial management capabilities. The main advantage is that application data does not need to be duplicated and stored in a specific data structure such as a scene graph; instead,

MAVERIK promotes the use of an application's own data structures. This avoids the need to conform to a rigid and potentially inappropriate data structure.

4.2.2.6. Other DVEs

Virginia Tech's DIVERSE toolkit (DTK) (Arsenault et al., 2001) is designed to aid implementation of distributed VR applications and is available both under the Gnu Public License (GPL) and the LGPL. The toolkit is implemented as a C++ API to a server and clients. It provides an extensible but relatively low-level library for distribution rather than a VR framework.

COVISE (Collaborative Visualisation and Simulation Environment) (Rantzau et al., 1998) exploits high-speed network infrastructures in distributed computing and collaborative engineering. It focusses predominantly on visualising high end supercomputing problems. COVER (2006), based on IRIS Performer, provides visualisation support. COVISE mediates a distributed session through a system of turn-taking. The database is fully replicated across nodes at connection. All user input, system output, and system administration is handled at a single point, reducing the complexity of keeping all users synchronised, but at the expense of round trip delays and the potential liability of the central point of control becoming a bottleneck as more users are added.

CAVERNsoft (Park et al., 2000) combines a networking library and a database. It facilitates the development of collaborative interactive virtual environments. CAVERNsoft is based on a client-server model. Applications use the Information Request Broker (IRB) to mediate communication between network instantiations. Users may request the type of network connection they wish to use. CAVERNsoft supplies the option of TCP/IP, UDP or IP/Multicast. The user defines the desired bandwidth and acceptable network attributes (latency, jitter), and the remote IRB attempts to comply. In this way the user specifies the desired quality of service on startup.

There are also more general VR frameworks that provide support for implementing DVEs. For example, VR Juggler (Bierbaum et al., 2001) has a networking support provided through an abstract Network manager. Another example is Bamboo (Watsen & Zyda, 1998), built on top of the ADAPTIVE

Communication Environment (ACE) (Schmidt, 1994), which provides the system with networking, concurrency (threading) and synchronisation functions.

4.2.2.7. HLA

Distributed simulation is an application of distributed system technology that enables models to be linked together over networks such as the Internet so that they work together (or interoperate) during a simulation run. The HLA (High Level Architecture) is a standard that defines the distributed system technology to make this interoperability possible. Rather than a networking protocol (wire standard) like DIS, HLA defines an architecture with a set of API Standards. Simulation applications (known as federates in HLA) communicate by making calls to the HLA APIs. A piece of software known as the RTI (Run-time Infrastructure) implements the HLA API, and is responsible for transporting data from one federate to another. Like DIS, HLA Standards are owned by IEEE. There are three documents that comprise the HLA Standard, all available from IEEE (2006):

- IEEE 1516-2000 – IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules: provides the rules and definitions for implementing and using HLA. Its IEEE product code is SH94882;
- IEEE 1516.1-2000 – IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification: defines the various services provided by an HLA RTI (see Figure 12), and contains the APIs. Its IEEE product code is SH94883;
- IEEE 1516.2-2000 – IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specification: defines the format used for describing object models in HLA. An object model dictates what kinds of data a particular set of HLA federates will be exchanging. The IEEE product code for this document is SH94884.

There is a fourth document that is not technically part of the definition of HLA, but that defines some of the recommended practices for using HLA. It is called IEEE 1516.3-2003 – IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP), and its IEEE product code is SH95088.

While the IEEE 1516 series of standards represents the "current" version of HLA, many HLA simulations are still using an earlier version known as HLA 1.3. This version was maintained by DMSO (Defense Modeling and Simulation Office) and the U.S. Department of Defense prior to IEEE Standardization.

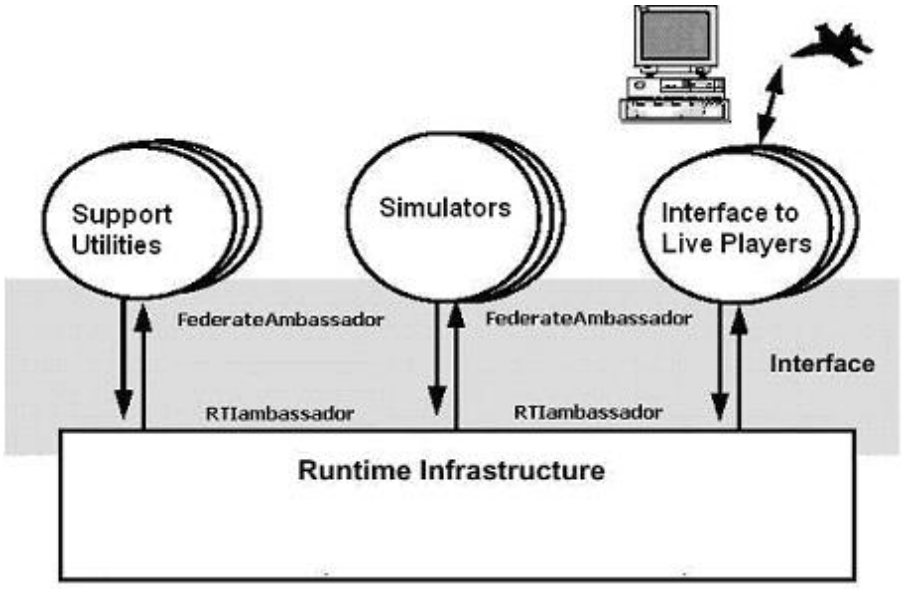

Figure 12 - HLA and RTI (Dahmann et al., 1999)

The HLA was initially developed for the US Department of Defense for simulating battlefield scenarios. The HLA emergent standard from IEEE has the following characteristics:

- Supports simulations composed of different simulation components.
- Supports real-time environment and large complex simulation problems.
- Supports reusability: component simulation models can be reused in different simulation scenarios and applications.
- Supports interoperability: reusable component simulations can be combined with other components without the need of re-coding.

We can thus conclude that HLA fulfils our requirement of real-time collaboration support as well as seems to be a flexible component-based architecture, in accordance to all the principles we have been pursuing so far. Also HLA seems to suit many aspects of a collaborative workspace for disaster management in the oil & gas area, except for the fact that it is mainly developed to have many nodes with precise real-time update, while in the oil & gas area there are fewer nodes but with higher visualisation demands.

The use of the HLA standard is not restricted to distributed virtual environments, which means that, as the demand for integration of processes, systems, and databases within companies continues to grow, this is one of the approaches that should be considered while defining integration solutions.

The fundamental concepts in HLA are:

- Federate: when a simulation is implemented as part of an HLA-compliant simulation, it is referred to as a *federate*.
- Federation: is a collection of federates working together to solve a specific problem.

Note that federations can include more than simulations. They can also include interfaces to human operators/players, to real hardware, and to general software performing functions such as data collection, data analysis, and data display.

The three main components in HLA are:

- HLA Rules:
  - Ensure proper interaction of federates in a federation.
  - Describe the responsibilities of federates and federations.
- Interface Specification:
  - Defines Run-time Infrastructure (RTI) services and interfaces.
  - Identifies “callback” functions each federate must provide.
- Object Model Template (OMT):
  - Prescribes the format and syntax for recording information.
  - Establishes the format of key models:
    - Federation Object Model (FOM).
    - Simulation Object Model (SOM).

- Management Object Model (MOM).

Figure 13 shows a high level, logical view of an executing HLA Federation. All of the components shown in the figure are part of a single federation with the exception of RtiExec. A brief description of each of these components is provided below:

- Federate: an HLA-compliant simulation component program, plus a SOM.
- Federation: a simulation composed of a set of federates interacting via the RTI services, plus a FOM.
- FedExec: manages the federation. It allows federates to join and to resign from the federation, and facilitates data exchange between participating federates.
- FDD file: FOM Document Data file, contains information derived from the FOM and used by the RTI at runtime.
- RtiExec: a global process that manages the creation and destruction of FedExec's.
- RID file: RTI Initialization Data. RTI vendor-specific information needed to run an RTI.

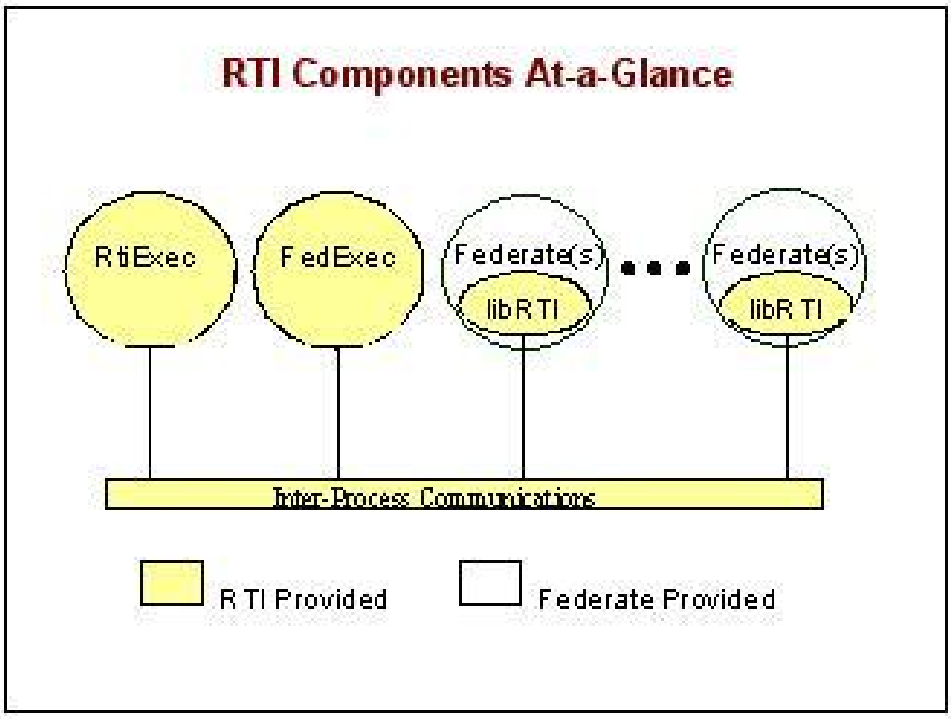

Figure 13 - Logical View of an HLA Federation (McLeod, 2006)

We then consider a system using HLA with multiple Federations in execution. This system is running two federations: Federation 1 and Federation 2. Though one system may run two federations, the HLA Rules specify that they are independent from each other and may not exchange any information. This system is illustrated in Figure 14, and has the following components:

- There is a single RtiExec and RTI.RID file, shared by both federations.
- Each Federation contains its own FedExec and FDD file. FedExec1 controls the execution of Federation 1, FedExec2 controls the execution of Federation 2.
- Federation 1 contains two federates: the White Federate and the Green Federate.
- Federation 2 contains three federates: the Purple, Orange, and Blue Federates.

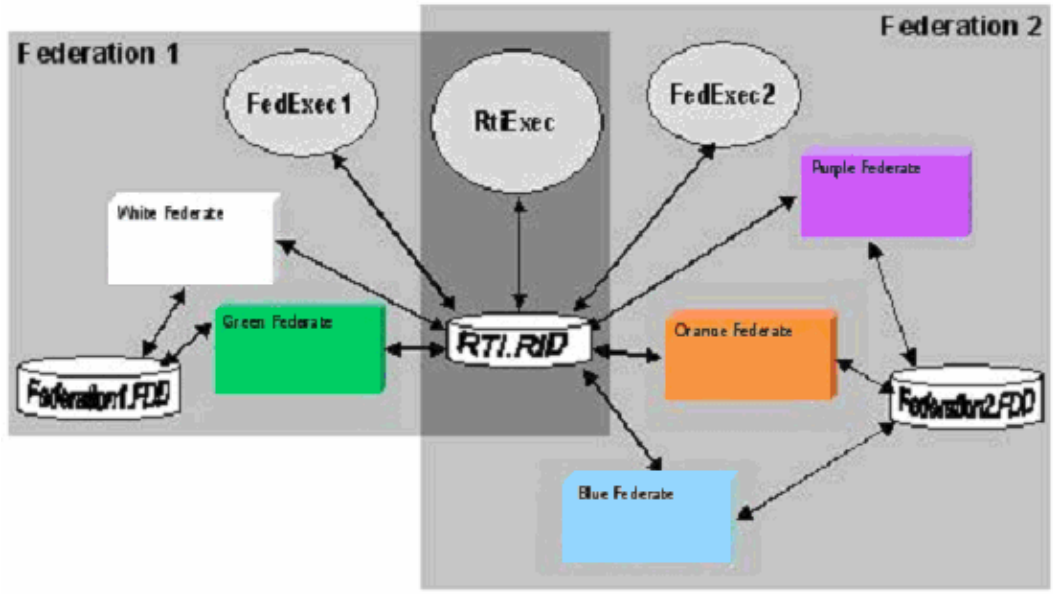

Figure 14 - The Big Picture – Federations in Execution (McLeod, 2006)

In Figure 15, we show the steps in the process of starting a federation execution:

1. When a federation is run, the RtiExec is started first.

2. Then a federate, acting as a manager, creates a federation execution by invoking the RTI method *createFederationExecution*.
3. Then a name is reserved with RtiExec, a FedExec process is spawned, and that FedExec registers its communication address with RtiExec. The federation execution is underway.
4. Once a federation execution exists, other federates can join it. RtiExec is consulted to get the address of FedExec, and *joinFederationExecution()* is invoked on FedExec. Additional federates can join via the same process.

In face of all these characteristics, we have chosen HLA as an example of this second approach (Pure DVEs) to delineate a prototype (Chapter 5) to be one of the proofs of our metamodel, as the InfoGrid was chosen as a platform of the first approach (Middleware).

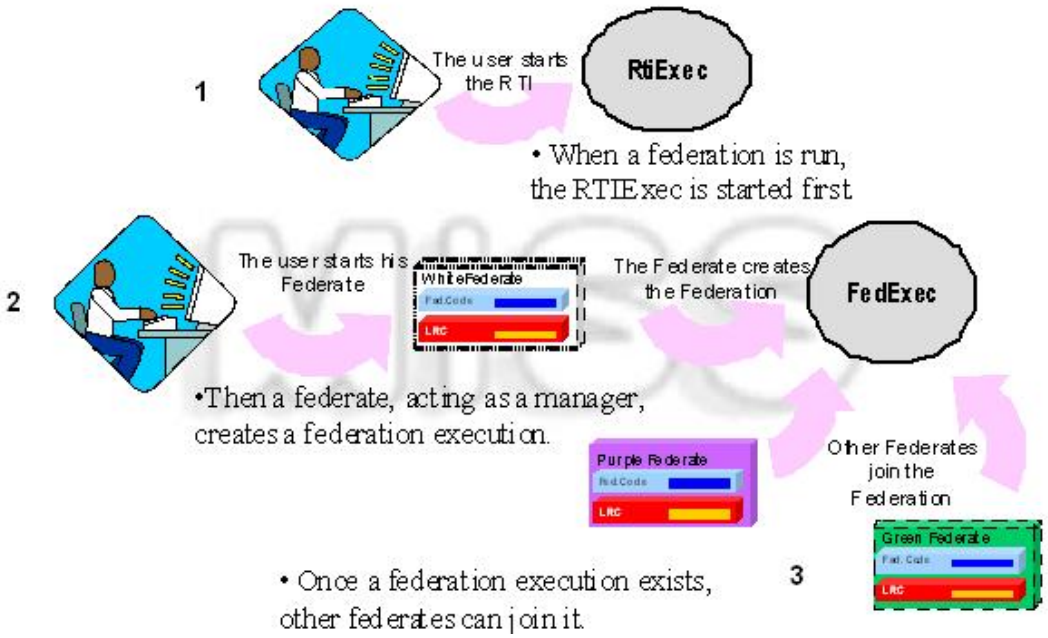

Figure 15 - Steps in the Process of Federation Execution (McLeod, 2006)

We still had to choose the HLA Run-time Infrastructure (RTI) to be used with our prototype. Our requirements were:

- To be open-source.
- To be freely redistributable.

Based on these requirements and on the fact that it was well documented in a master's thesis, we chose XRTI – The Extensible Run-Time Infrastructure – from Kapolka (2003). Its basic characteristics are:

- It has full compatibility with the HLA standard.
- It specifies a standardisable message protocol.
- It supports dynamic-object model extension and composition.
- It is written in Java and it uses XML object models.
- It uses a pure client-server topology in which federates only communicate with one another through the XRTI Executive, a server application.
- Federates maintain two channels to the Executive: a TCP (Transmission Control Protocol) channel for reliable communication and a UDP (User Datagram Protocol) channel for unreliable messaging.

5
Activity-Centred Metamodel: Deriving Models and Prototypes

In this chapter, we derive a first model for oil & gas offshore structures disaster management based on our multi-perspective metamodel. We also develop an HLA-compliant prototype as a proof-of-concept of the metamodel and discuss how the prototype could be implemented using InfoGrid. Still for the disaster management application, we present a second model and its prototype showing that we can derive different models for the same application. Finally, to validate the generality of the metamodel, we also delineate a model for another application, namely CAD visualisation in virtual environments.

5.1.
The Oil & Gas Offshore Structures Disaster Management Application

The oil & gas offshore structures disaster management application was the one that motivated the creation of our metamodel. Investigating the application's requirements, we have followed a general-consensus academic recommendation, mentioned by Lauche (2005), which states that only a thorough understanding of the activities one is designing for will allow ICT systems to become meaningful tools suited to the task and the context of use. It is therefore important to not only interview but also observe users in practice and to make sense of these findings before consolidating them in the form of requirements. We have then conducted semi-structured interviews with key individuals and not only have observed their work practice but in fact have been participating with them in joint projects and activities for more than one decade.

The disaster management of an oil & gas offshore structure – which can be a ship or a semi-submersible platform – is a complex operation involving several groups, such as the oil & gas company (in our particular case, the Brazilian company Petrobras), the rescue team, the health care centre, the press, among

others. We can then see that this is in fact an inter-organisational complex activity, with all the issues and characteristics already mentioned in Chapter 3.

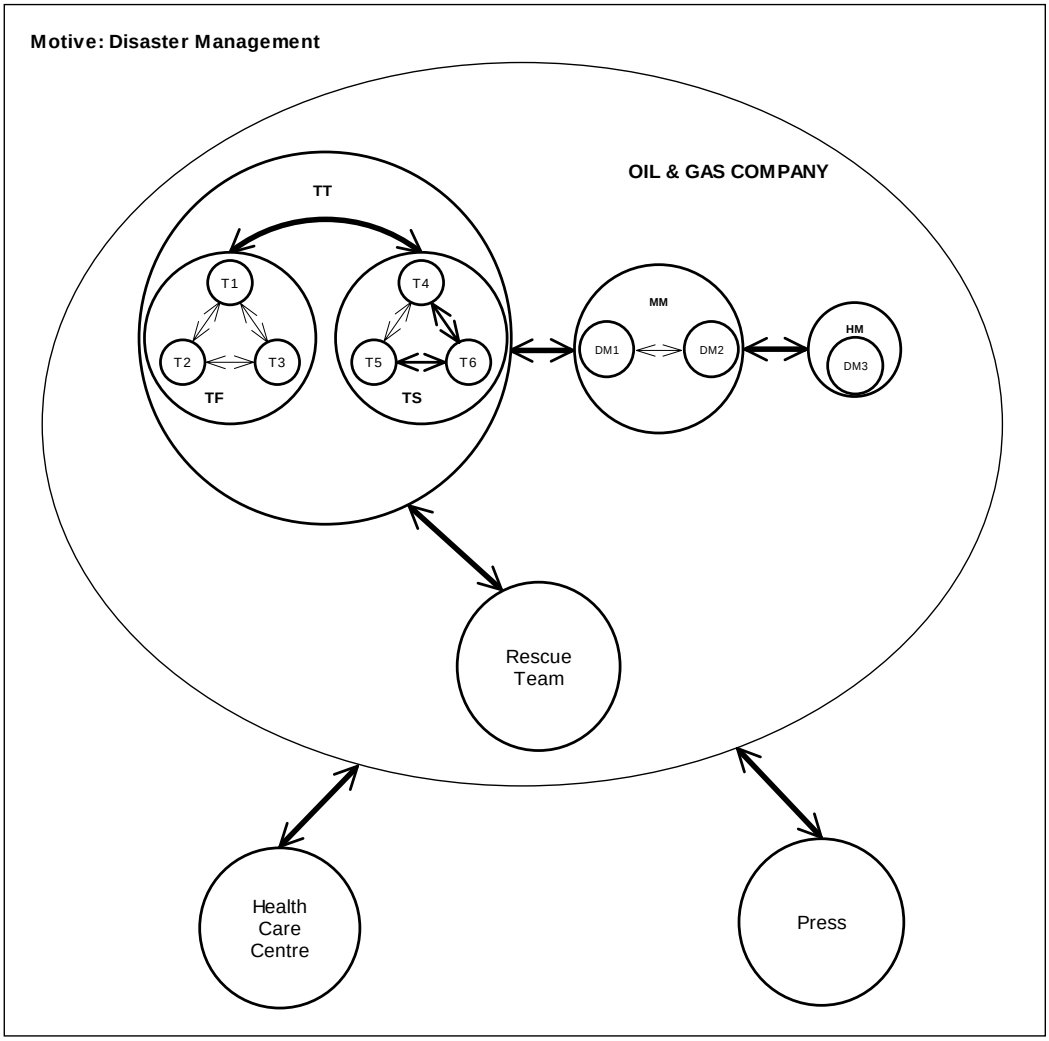

Figure 16 - Disaster management collaborative model: overall picture

We now describe the main nodes present in the overall picture of the disaster management model shown in Figure 16:

- Representing four organisations, there are four main nodes, Petrobras, Rescue Team, Health Care Centre, and Press, each one remotely located to the other. We can observe that Petrobras is in the centre of the whole activity, and we will concentrate our focus on and detail the Petrobras node. Before doing this, it is interesting to situate where most of the Petrobras offshore structures are located: in a region called Campos Basin, near Rio de Janeiro.
- Inside the Petrobras node, we identify three main groups: the Technical Teams (TT) group, the Middle-level Managers (MM)

group, and the High-level Managers (HM) group, each one remotely located to the other.

- The TT group is formed by two other technical sub-groups: the Task Force (TF) team and the Technical Support (TS) team. These two sub-groups are also remotely located to each other.
- The TF team plays the main role in the whole disaster management activity, leading the decision-making process. It is constituted by specialists such as naval engineers, structural engineers, risers analysts, and oceanographers, in this example represented by three co-located technicians, namely T1, T2, and T3, without losing generality. They are brought together to a work environment allowing face-to-face communication, which has many facilities, such as access to remote databases that maintain CAD models and simulation models of the unit. This work environment could be arranged either near or distant from the disaster – what is important is having communication with the disaster area. In this integrated environment, the TF team runs different simulators in order to derive the best solution to save the offshore unit, permanently communicating with the TS team. They also maintain contact with the MM group informing about their work evolution and asking for approval for their derived solution. Once their solution is approved, they pass to the unit operator or to the rescue team the sequence of commands to be executed. It can be observed that we have not represented the operator in our overall picture since, in the terms of the collaboration application, he is not directly involved with the integrated simulation, merely receiving its final result. Also in some cases he is not directly connected with the other participants in terms of ICT. In fact, two possible situations could occur: *(i)* if the unit has not been heavily damaged, two or three operators can remain inside it and can receive orientation from the TF team (if some sort of communication remained available, we could include the operator in the collaborative application); or *(ii)* if the unit has been heavily damaged and has security problems, only divers would be capable of working there.

- The TS team, represented in our example by technicians T4, T5, and T6 without losing generality, can be invoked by the TF team to perform specialised simulations focussing on some particular issues that would not be possible to be done in the TF work environment, or to obtain another opinion or view about the problem. T4, T5, and T6 are technicians working in the same fields as the TF team. In our particular example, T4 and T5 are working co-located, possibly in a Numerical Simulation Centre, with T6 working remotely to them (from any other company site, or even interacting via mobile).
- The MM is constituted by middle-level managers, in our example DM1 and DM2 (DM for Decision Maker) working co-located in a company office, with one of them usually being responsible for making the final decision. They have a good overall knowledge about the technical issues and work constantly interacting with the TT group. They also communicate with the HM group, informing about the work evolution and occasionally seeking confirmation or advice from this group when they need to make a more critical decision.
- The HM group, in our example a single manager, DM3, which could be a director working at the company headquarters, periodically receives from the MM group information about the work evolution and sometimes is requested by them to provide confirmation or advice concerning a particular critical issue.

After investigating the activities involved in this disaster scenario, identifying their requirements in terms of ICT, we decided to concentrate on the Technical Teams group to develop a first prototype of collaborative application implementing a particular model of our Activity-Centred metamodel.

This first prototype is more particularly related to the work performed by the Task Force group, including the simulators they run, their mutual communication and their interaction with the Middle-level Manager group. In terms of this last group, for the sake of simplifying the prototype, we are going to consider only one middle-level manager integrated in the collaborative application.

We first investigate how the Task Force group runs the different simulators and what the relationships among them are. During a crisis situation, Petrobras typically uses three simulators, which will be briefly described now.

The first simulator to be run is SSTAB (Coelho et al., 2003), the Floating Units Stability system, used to analyse the static conditions of the floating unit (Figure 17). SSTAB uses as inputs the unit model obtained from a centralised system called GIEN and updated data from the unit obtained through a monitoring system called ECOS. Its outputs consist of five files, including the inertia matrix. Also available in every unit, besides being used during emergency situations, SSTAB can also be used for planning maintenance operations as well as for projecting new units.

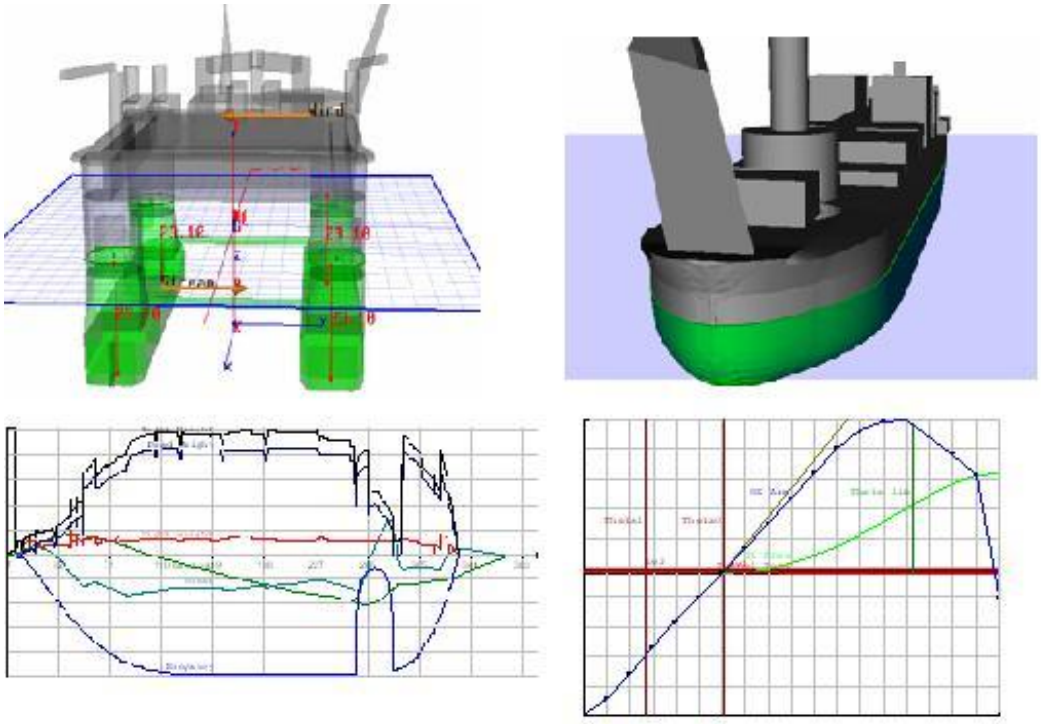

Figure 17 - SSTAB: Floating Units Stability system

The second simulator is called WAMIT (2006) and uses as inputs the output files generated by SSTAB. It works in the frequency domain, deriving the excitation forces of the unit and water forces' reactions to lateral displacement. WAMIT is activated by a user interface program called WMG.

Finally the third simulator to be executed is DYNASIM (Coelho et al., 2001), for Dynamic Stability (Figure 18). It uses as inputs the results obtained from WAMIT as well as the parameters H and P, representing respectively the

height and the *period* of the wave at the moment of the disaster. DYNASIM works in the time domain and in fact has two other modules apart from the central module: a pre-processor called Pre-Dyna and a post-processor called Pos-Dyna. DYNASIM calculates the forces acting on the mooring lines and risers, and the over-turning moment. When these forces are considered extreme, a retrofeedback process is started, performing all the simulations again, beginning with SSTAB, to find another stable condition for the unit.

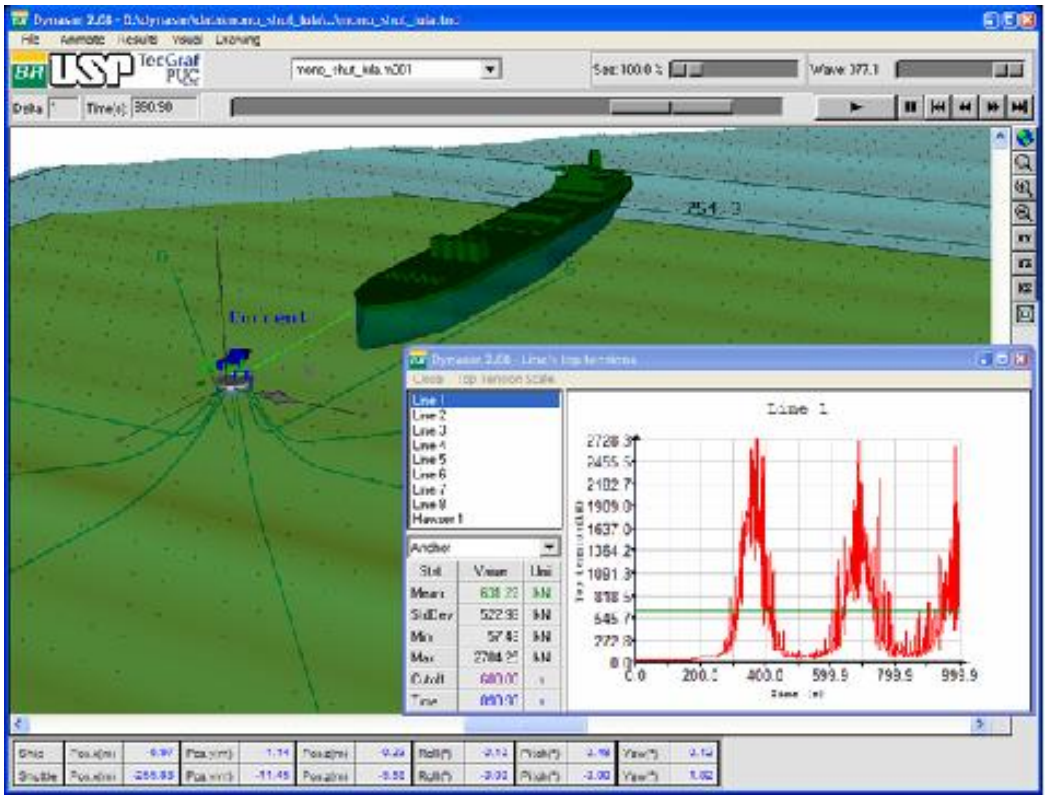

Figure 18 - DYNASIM: Dynamic Stability system

In order to better illustrate how the simulators communicate, we take the example of the P-34 FPSO (Floating Production, Storage and Offloading) unit and describe below the files and parameters that they use:

- At the beginning of the SSTAB simulation, the SSTAB operator opens the file *p34.sst*, which contains the geometric model of the unit.
- At the end of the SSTAB simulation, the SSTAB operator exports a WAMIT geometric data file with name *p34.gdf*. In fact, four other files are exported to WAMIT at this time, namely: *fnames.wam*, which contains the filename list; *p34.pot*, which is the Potential

Control File; *p34.frc*, which is the Force Control File; and *p34.cfg*, which is the WAMIT configuration file (WAMIT, 2006).

- WAMIT then reads the five files exported by SSTAB and performs its simulation.
- At the end of its simulation, WAMIT generates an output file called *p34.out*, containing the excitation forces acting on the unit.
- The DYNASIM operator then opens the DYNASIM project file *p34.prd*.
- He then opens the WAMIT output file *p34.out*, converting it to the WAMIT neutral file *p34.wnf*.
- He finally enters with the environmental parameters H and P received, beginning the DYNASIM simulation.

The method used to save an offshore unit has the goal of defining a sequence of commands to be passed to the unit operators or to the rescue team so that they can move the unit in a step by step mode from its initial unstable condition until it reaches back its normal equilibrium state. It is based on the following workflow:

- We first use these three simulators (possibly using retrofeedback) to derive the initial conditions of the offshore unit.
- We then define a next step configuration of tanks (e.g., moving water from a ballast tank of one side to a ballast tank of the opposite side, operating the tanks' valves) and simulate the unit in this new condition using again the three simulators. If we are not satisfied with the results obtained, we define another configuration of tanks and continue this process, experimenting iterative configurations, until we are satisfied with one of them. In this case, we say that we have reached the present step configuration of tanks.
- From the configuration of the previous step, we now try to derive a new step configuration of tanks, using a process analogous to the one just described.
- We repeat this process of deriving step configurations of tanks using our three simulators until we reach back a normal equilibrium state.

At the end of this whole process, we have a sequence of commands in terms of tank valve operations, corresponding to the achievement of each of the step configurations described above, in a step-by-step mode, which was exactly our goal. In the P-34 disaster, with no unit operators inside it, the actions were taken from outside of platform, using housings to fill the tanks.

What we obtain as a result of employing the method above is a sequence of commands that constitutes a single strategy. We could of course apply this method many times deriving many strategies. We could also use field feedback after applying each step on the real unit, refining future steps with this new information. Finally we could gather changing conditions from the unit or the disaster area and introduce them in our simulations. In P-34, a surveyor ship called Salgueiro and the oceanographic group supplied environmental data 24 hours a day.

It is important to note that the executions of simulators SSTAB and DYNASIM are highly interactive visualisation processes. In a crisis situation, when we need to rapidly experiment many alternatives to respond to the disaster, this characteristic of the simulators is intensively explored. Also we have to consider that, in emergency situations, it is very important to be as fast as possible, because every single minute lost could be crucial in saving the unit. Thus, searching for points where we could save time, we found that, if WAMIT receives the results from SSTAB, it can be activated automatically on ending the SSTAB simulation.

5.1.1. A First Model for the Disaster Management Collaborative Application

In this subsection, we derive a first Activity-Centred model for the disaster management collaborative application. We concentrate on the Technical Teams group activity, which is mainly performed by the Task Force group members running their simulators (Figure 19).

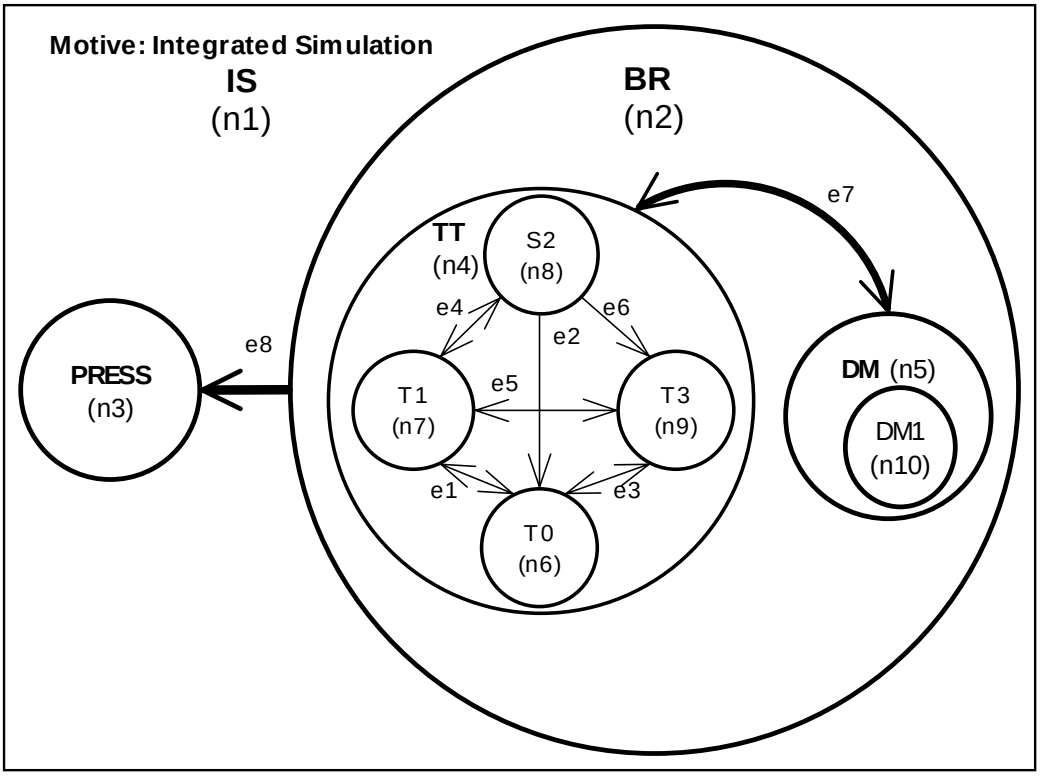

Figure 19 - First model for the disaster management collaborative application, focussing on the integrated simulation

The top-level node of the technical part of this crisis scenario, the motive, is the Integrated Simulation, named IS, which has two remotely located main nodes, one in relation to the other: BR, the oil & gas company, and Press.

Within BR, we created two remote groups, one in relation to the other, Technical Teams (TT) and Decision Makers (DM):

- TT is constituted by the Task Force (TF) team with members T0, T1, and T3, and the software agent S2, all of them co-located in the Task Force room and running their simulators. Since in our model TF is the only team inside TT, we are not explicitly representing it (if it has particular in- and out-policies, it should also be a node of our model).
- DM in this model is constituted by a single manager DM1 (Decision Maker 1). Without losing generality, DM1 can be considered a single representative of all the participants not directly involved with the technical part of the simulation activity such as operators and other managers, who only receive from the TT group follow-up messages, commands to be executed (in case of operators) or

approval requests (in case of managers), giving them simple answers.

The Press is represented by one single node that only receives periodical reports from the oil & gas company informing the disaster evolution.

There are some cases in which we have a technical decision maker, directly participating in the simulation activity. For example, in the P-34 disaster, the decision maker was in the war room also running the simulators. In this case, it would be more appropriate to allocate the DM node inside the TT group.

The load records for the nodes and edges correspondent to the network component of this first model are shown in Table 10.

1 | 1 | Integrated Simulation | IS | GROUP | 0 | 6 | …
1 | 2 | Oil & Gas Company | BR | GROUP | 1 | 6 | …
1 | 3 | Press | PRESS | LEAF | 1 | 0 | …
1 | 4 | Technical Teams | TT | GROUP | 2 | 6 | …
1 | 5 | Decision Makers | DM | GROUP | 2 | 10 | …
1 | 6 | Emergency Pilot | T0 | LEAF | 4 | 0 | …
1 | 7 | SSTAB Operator | T1 | LEAF | 4 | 0 | …
1 | 8 | WAMIT | S2 | LEAF | 4 | 0 | …
1 | 9 | DYNASIM Operator | T3 | LEAF | 4 | 0 | …
1 | 10 | Decision Maker 1 | DM1 | LEAF | 5 | 0 | …
2 | 1 | Pilot-SSTAB channel | LOCAL | BIDIRECTED | 6 | 7 | …
2 | 2 | WAMIT-Pilot channel | LOCAL | UNIDIRECTED | 8 | 6 | …
2 | 3 | Pilot-DYNASIM channel | LOCAL | BIDIRECTED | 6 | 9 | …
2 | 4 | SSTAB-WAMIT channel | LOCAL | BIDIRECTED | 7 | 8 | …
2 | 5 | SSTAB-DYNASIM channel | LOCAL | BIDIRECTED | 7 | 9 | …
2 | 6 | WAMIT-DYNASIM channel | LOCAL | UNIDIRECTED | 8 | 9 | …
2 | 7 | technical-manager channel | REMOTE | BIDIRECTED| 4 | 5 | …
2 | 8 | technical-press channel | REMOTE | UNIDIRECTED | 2 | 3 | …

Table 10 - First model for the disaster management application: load records

Apart from the interaction network component of the model just described, we also define role rules and the message attributes table in order to represent the following different roles played by the participant members in this disaster scenario:

- The Emergency Pilot T0 plays the main role in this disaster application, coordinating the collaborative session and leading the decision-making process. He asks the SSTAB operator (T1) to begin his simulation. After receiving a message from agent S2 indicating the end of its simulation, he asks the DYNASIM operator (T3) to begin his simulation, informing the current H and P values of the waves. On receiving a simulation conclusion message from T3, he makes a decision based on the force values acting on mooring lines and risers. If he understands that these forces are extreme, he asks T1 to begin the whole process again, in order to find a new stable condition of the unit, and this loop continues until he is satisfied with the force values obtained. In this case, the simulation for the initial condition of the unit is considered finished. T0 then repeats this process for each configuration step of the method already described until he determines the end of the simulation process by achieving the sequence of commands desired. In this case, he makes contact with the Decision Maker (DM1), asking for his approval to their solution. If DM1 approves it, T0 then passes the sequence of commands to be executed by the unit operator (not represented here) to save the unit.
- T1, the SSTAB operator, initiates a SSTAB simulation every time he receives an order from T0. He interactively studies many tank configurations trying to derive a step configuration of tanks according to the method described above and informs when he ends the simulation of this step.
- S2, Simulator 2, is in fact a reactive agent instead of a real person (used here to save time). Upon receiving a message from T1 indicating the end of his simulation, S2 automatically runs the WAMIT simulator using the information passed by T1 (SSTAB results). S2 informs the participant members when it initiates and finishes its simulation.
- T3, the DYNASIM operator, initiates a DYNASIM simulation every time he receives an order from T0. He uses the other two simulators'

results, H, and P as inputs and interactively studies the dynamic stability of the unit based on these conditions. When he finishes his simulation, he informs the participants his result, which can be *red* (extreme forces acting), *yellow* (moderate forces acting) or *green* (light forces acting).

- Finally DM1, the Decision Maker, receives from T0 the sequence of commands defining a strategy, and replies approving or not the defined strategy. In case he does not approve the strategy, he can ask T0 to restart the whole process again or he can further discuss the problem with the High-level Managers group, not represented here.

We should note that all participants should be informed about one another's work during the whole collaborative session.

Although the complete workflow is the one just described, to facilitate the understanding we are going to consider for the moment in our model only one configuration step with possible retrofeedback. With this simplification, the role rules correspondent to our workflow are the ones shown in Tables 11, 12, 13, and 14.

Apart from the *on-arrive* and the *on-init* rules, and the *send* and the *display* formulas already introduced in Chapter 3, we identified the need to define additional rules and formulas, following the Prolog terminology (SWI-Prolog, 2006), in order to completely describe the role rules of our model:

- *when* rule with arguments *term* and *term_value*: the rule is fired when *term* in the knowledge base has value *term_value*;
- *read* formula: is used to read inputs from the console;
- *write* formula: is used to write message contents in the message attributes table;
- *assert* predicate with argument *term*: is used to add a fact or clause in the database.
- *abolish* predicate with argument *term*: is used to remove a clause from the database (not being used in the present model).

We can now briefly describe the contents of Tables 11, 12, 13, and 14. In Table 11, we present the collaboration bus definition and the role rules for the Emergency Pilot. The bus in our example has one local channel and one remote channel. The rules defining the Emergency Pilot role are similar to those of the example already shown in Chapter 3, with a few new statements. When message 5 arrives, the values H and P have to be read from the console (using the *read* formula) and written in the line correspondent to message 6 in the message attributes table (using the *write* formula) before sending it to the receivers. Also, when message 8 arrives, the term *simulations_result* has its value read from the console (using the *read* formula) and added to the knowledge database (using the *assert* predicate). Then the sequence of commands to be executed is determined by two subsequent *when* rules based on the value of *simulations_result*. Finally, the last sequence of commands to be executed is determined by the two last *when* rules based on the value of *simulations_approval*. It is interesting to note that, in both sequences of commands fired by the two *when* rules, we wait for the user to press a button (using the *read* formula), the first one to stop the simulation and the second one to send the press release to the Press.

In Table 12, we present the role rules for the SSTAB operator and for the WAMIT simulator. The basic rules and formulas used are similar to the ones presented in Table 11, with a few differences. In the sequence of commands of the first *on_arrive* rule of the SSTAB operator role, we read *SSTAB_begin_button* and *SSTAB_end_button* (using the *read* formula) to respectively indicate the beginning and the end of the SSTAB simulation. In the WAMIT simulator role rules, as this simulator is automatically activated on arrival of message 3, we defined a method called *wamit()* to activate the WAMIT simulator. We also used a term called *wamit_end* to indicate the end of this simulator (used as an argument of the last *when* rule).

In Table 13, we present the role rules for the DYNASIM operator, also with the same basic rules and formulas as the ones presented in Tables 11 and 12. An interesting aspect we can highlight here is the sequence of three *when* rules used to determine the sequence of commands to be executed after the end of the DYNASIM simulation. The selection of commands is made based on the value of the term *DYNASIM_result*, which has been previously read from the console and can assume one of three possible values: *red*, indicating that there are extreme

```
collaboration integrated_simulation
{ collaboration_bus {
            channel(local).
            channel(remote).}
 role emergency_pilot //technician responsible for coordinating the simulations
 {
  on-init(integrated_simulation) :-
   send(local, message_table(source(self), 1, dummy)). //send “Technician 1, please begin SSTAB simulation.”

  on-arrive(local, message_table(dummy, 2, source(self))) :-
   display(message_table(dummy, 2, source(self))). //display “SSTAB simulation has begun.”

  on-arrive(local, message_table(dummy, 3, source(self))) :-
   display(message_table(dummy, 3, source(self))). //display “End of SSTAB simulation.”

  on-arrive(local, message_table(dummy, 4, source(self))) :-
   display(message_table(dummy, 4, source(self))). //display “WAMIT simulation has begun.”

  on-arrive(local, message_table(dummy, 5, source(self))) :-
   display(message_table(dummy, 5, source(self))), //display “End of WAMIT simulation.”
   read H and P, //read H and P from console
   write message 6, //write message 6 with H and P values in the message attributes table
   send(local, message_table(source(self), 6, dummy)). //send “Technician 3, please begin DYNASIM with h and p.”

  on-arrive(local, message_table(dummy, 7, source(self))) :-
   display(message_table(dummy, 7, source(self))). //display “DYNASIM simulation has begun.”

  on-arrive(local, message_table(dummy, 8, source(self))) :-
   display(message_table(dummy, 8, source(self))), //display the DYNASIM result
   read simulations_result, //read simulations_result (OK or not_OK) from console
   assert simulations_result. //assert the simulations_result in the knowledge base

  when(simulations_result, not_OK) :-
   send(local, message_table(source(self), 9, dummy)), //send “A new cycle of simulations will begin.”
   send(local, message_table(source(self), 1, dummy)). //send “Technician 1, please begin SSTAB simulation.”

  when(simulations_result, OK) :-
   send(remote, message_table(source(self), 10, dummy)). //send “Decision Maker, please approve the simulations.”

  on-arrive(remote, message_table(dummy, 11, source(self))) :-
   display(message_table(dummy, 11, source(self))). //display the decision made

  when(simulations_approval, not_OK) :-
     read stop_simulations_button. //read stop_simulations_button from console

  when(simulations_approval, OK) :-
   read send_to_press_button, //read send_to_press_button from console
   send(remote, message_table(source(self), 12, dummy)). //send “New press release posted.”
 }
```

Table 11 - Collaboration bus definition and role rules for the Emergency Pilot

forces acting; *yellow*, indicating the action of moderate forces; or *green*, indicating the action of mild forces.

```
role technician_1 //technician responsible for executing the SSTAB simulation
{
 on-arrive(local, message_table(dummy, 1, source(self))) :-
  display(message_table(dummy, 1, source(self))), //display "Technician 1, please begin SSTAB simulation."
  read SSTAB_begin_button, //read SSTAB_begin_button from console
  send(local, message_table(source(self), 2, dummy)), //send "SSTAB simulation has begun."
  read SSTAB_end_button, //read SSTAB_end_button from console
  send(local, message_table(source(self), 3, dummy)). //send "End of SSTAB simulation."

 on-arrive(local, message_table(dummy, 4, source(self))) :-
  display(message_table(dummy, 4, source(self))). //display "WAMIT simulation has begun."

 on-arrive(local, message_table(dummy, 5, source(self))) :-
  display(message_table(dummy, 5, source(self))). //display "End of WAMIT simulation."

 on-arrive(local, message_table(dummy, 7, source(self))) :-
  display(message_table(dummy, 7, source(self))). //display "DYNASIM simulation has begun."

 on-arrive(local, message_table(dummy, 8, source(self))) :-
  display(message_table(dummy, 8, source(self))). //display the DYNASIM result

 on-arrive(local, message_table(dummy, 9, source(self))) :-
  display(message_table(dummy, 9, source(self))). //display "A new cycle of simulations will begin."

 on-arrive(local, message_table(dummy, 10, source(self))) :-
  display(message_table(dummy, 10, source(self))). //display "Decision Maker, please approve the simulations."

 on-arrive(remote, message_table(dummy, 11, source(self))) :-
  display(message_table(dummy, 11, source(self))). //display the decision made
}

role simulator_2 //WAMIT simulator
{
 on-arrive(local, message_table(dummy, 3, source(self))) :-
  wamit(), //begin WAMIT simulation
  send(local, message_table(source(self), 4, dummy)). //send "WAMIT simulation has begun."

 when(wamit_end, OK) :-
  send(local, message_table(source(self), 5, dummy)). //send "End of WAMIT simulation."
}
```

Table 12 - Role rules for the SSTAB operator and the WAMIT simulator

```
role technician_3 //technician responsible for executing the DYNASIM simulation
{
 on-arrive(local, message_table(dummy, 2, source(self))) :-
   display(message_table(dummy, 2, source(self))). //display “SSTAB simulation has begun.”

 on-arrive(local, message_table(dummy, 3, source(self))) :-
   display(message_table(dummy, 3, source(self))). //display “End of SSTAB simulation.”

 on-arrive(local, message_table(dummy, 4, source(self))) :-
   display(message_table(dummy, 4, source(self))). //display “WAMIT simulation has begun.”

 on-arrive(local, message_table(dummy, 5, source(self))) :-
   display(message_table(dummy, 5, source(self))). //display “End of WAMIT simulation.”

 on-arrive(local, message_table(dummy, 6, source(self))) :-
   display(message_table(dummy, 6, source(self))), //display “Technician 3, please begin DYNASIM with h and p.”
   read DYNASIM_begin_button, //read DYNASIM_begin_button from console
   send(local, message_table(source(self), 7, dummy)), //send “DYNASIM simulation  has begun.”
   read DYNASIM_result, //read DYNASIM_result from console (red, yellow or green)
   assert DYNASIM_result. //assert the DYNASIM_result in the knowledge base

 when(DYNASIM_result, red) :-
   write message 8 with “Extreme forces: you should begin a new simulation cycle.”,
   send(local, message_table(source(self), 8, dummy)). //send message 8

 when(DYNASIM_result, yellow) :-
   write message 8 with “Moderate forces: you should decide on performing or not a new simulation cycle.”,
   send(local, message_table(source(self), 8, dummy)). //send message 8

 when(DYNASIM_result, green) :-
   write message 8 with “Mild forces: you can approve the present simulation.”,
   send(local, message_table(source(self), 8, dummy)). //send message 8

 on-arrive(local, message_table(dummy, 9, source(self))) :-
   display(message_table(dummy, 9, source(self))). //display “A new cycle of simulations will begin.”

 on-arrive(local, message_table(dummy, 10, source(self))) :-
   display(message_table(dummy, 10, source(self))). //display “Decision Maker, please approve the simulations.”

 on-arrive(remote, message_table(dummy, 11, source(self))) :-
   display(message_table(dummy, 11, source(self))). //display the decision made
}
```

Table 13 - Role rules for the DYNASIM operator

```
role decision_maker //manager responsible for making the decision
{
  on-arrive(remote, message_table(dummy, 2, source(self))) :-
    display(message_table(dummy, 2, source(self))). //display “SSTAB simulation has begun.”

  on-arrive(remote, message_table(dummy, 3, source(self))) :-
    display(message_table(dummy, 3, source(self))). //display “End of SSTAB simulation.”

  on-arrive(remote, message_table(dummy, 4, source(self))) :-
    display(message_table(dummy, 4, source(self))). //display “WAMIT simulation has begun.”

  on-arrive(remote, message_table(dummy, 5, source(self))) :-
    display(message_table(dummy, 5, source(self))). //display “End of WAMIT simulation.”

  on-arrive(remote, message_table(dummy, 7, source(self))) :-
    display(message_table(dummy, 7, source(self))). //display “DYNASIM simulation has begun.”

  on-arrive(remote, message_table(dummy, 8, source(self))) :-
    display(message_table(dummy, 8, source(self))). //display the DYNASIM result

  on-arrive(remote, message_table(dummy, 9, source(self))) :-
    display(message_table(dummy, 9, source(self))). //display “A new cycle of simulations will begin.”

  on-arrive(remote, message_table(dummy, 10, source(self))) :-
    display(message_table(dummy, 10, source(self))), //display “Decision Maker, please approve the simulations.”
    read simulations_approval, //read simulations_approval (OK or not_OK) from console
    assert simulations_approval. //assert the simulations_approval in the knowledge base

  when(simulations_approval, not_OK) :-
    write message 11 with “Stop the simulations: higher level required.”,
    send(remote, message_table(source(self), 11, dummy)). //send message 11

  when(simulations_approval, OK) :-
    write message 11 with “Simulations approved.”,
    send(remote, message_table(source(self), 11, dummy)). //send message 11
}

role press //the press company receiving the press release
{
  on-arrive(remote, message_table(dummy, 12, source(self))) :-
    display(message_table(dummy, 12, source(self))). //display “New press release posted.”
}

role technical_teams //the Technical Teams node: role processed by node = post_processing execution node attribute
{
  on-arrive(remote, message_table(dummy, 11, source(self))) :-
    display(message_table(dummy, 11, source(self))). //pos_11_TT determines which nodes will receive message 11
 }
}
```

Table 14 - Role rules for the Decision Maker, the Press and the Technical Teams

In Table 14, we present the role rules for the Decision Maker, the Press, and the Technical Teams group. Once again, the basic rules and formulas are similar to the ones already shown in Tables 11, 12, and 13. In the Decision Maker rules, we have a sequence similar to the one present in the Emergency Pilot role: when message 10 arrives, the term *simulations_approval* has its value read from the console (using the *read* formula) and added to the knowledge database (using the *assert* predicate); then the sequence of commands to be executed is determined by two subsequent *when* rules based on the value of *simulations_approval*. The Press role rule is simply one *on-arrive* fired on arrival of message 12, indicating that a new press release has been made available. Finally the Technical Teams group role is noteworthy because it is the only one of this example associated with a group: when message 11 arrives, the *on-arrive* rule fires and a message is displayed on the console informing that the post-processing module *pos_11_TT* is determining and re-routing message 11 to the appropriate receivers within TT to receive message 11 – in our example, leaf nodes T0 (Emergency Pilot), T1 (SSTAB operator), and T3 (DYNASIM operator), as we can verify in Tables 11, 12, and 13, in the *on-arrive* rules receiving message 11.

In Table 15, we present the complete message attributes table for our first model, including the messages' contents field, to facilitate the understanding. It is important to note that some messages have constant contents, while others have contents that change as the workflow is being processed (in our model, messages 8 and 11).

In this table, we observe two columns corresponding to two of the most important elements that give flexibility to our metamodel: the pre- and the post-processing modules. We can see, for example, that message 9 is sent to three different receivers – T1, T3, and DM1 – with one particular pre- and one particular post-processing module associated to each receiver. The same occurs with messages 2, 3, 4, 5, 7, 8, and 10. Also we can observe that, in the line associated to message 11, DM1 is sending the message to the TT group, which means that *Pos_11_TT* (executed by T0) will determine, in run-time, the TT leaf nodes to receive message 11: T0, T1, and T3.

In order to better explore these pre- and post-processing capabilities of our metamodel, we are considering that T0, T1, DM1, and Press use the Portuguese

language while T3 uses English. This means that we should have automatic translations being executed by the following processing modules:

- *Pre_6_T3*, *Pre_9_T3*, *Pre_10_T3*, *Pre_2_T3*, *Pre_3_T3*, *Pre_4_T3*, and *Pre_5_T3*: Portuguese to English.
- *Pos_7_T0*, *Pos_7_T1*, *Pos_7_DM1*, *Pos_8_T0*, *Pos_8_T1*, and *Pos_8_DM1*: English to Portuguese.

sender	message_id	receiver	edge	message content	pre-processing	post-processing
T0	1	T1	e1	Technician 1, please begin SSTAB simulation.	Pre_1_T1	Pos_1_T1
T0	6	T3	e3	Technician 3, please begin DYNASIM simulation with H=h and P=p.	Pre_6_T3	Pos_6_T3
T0	9	T1	e1	A new simulation cycle will begin.	Pre_9_T1	Pos_9_T1
T0	9	T3	e3	A new simulation cycle will begin.	Pre_9_T3	Pos_9_T3
T0	9	DM1	e7	A new simulation cycle will begin.	Pre_9_DM1	Pos_9_DM1
T0	10	DM1	e7	DM1, please validate the sequence of commands.	Pre_10_DM1	Pos_10_DM1
T0	10	T1	e1	DM1, please validate the sequence of commands.	Pre_10_T1	Pos_10_T1
T0	10	T3	e3	DM1, please validate the sequence of commands.	Pre_10_T3	Pos_10_T3
T0	12	PRESS	e8	New press release posted.	Pre_12_PRESS	Pos_12_PRESS
T1	2	T0	e1	SSTAB simulation has begun.	Pre_2_T0	Pos_2_T0
T1	2	T3	e5	SSTAB simulation has begun.	Pre_2_T3	Pos_2_T3
T1	2	DM1	e7	SSTAB simulation has begun.	Pre_2_DM1	Pos_2_DM1
T1	3	T0	e1	End of SSTAB simulation.	Pre_3_T0	Pos_3_T0
T1	3	T3	e5	End of SSTAB simulation.	Pre_3_T3	Pos_3_T3
T1	3	DM1	e7	End of SSTAB simulation.	Pre_3_DM1	Pos_3_DM1
S2	4	T0	e2	WAMIT simulation has begun.	Pre_4_T0	Pos_4_T0
S2	4	T1	e4	WAMIT simulation has begun.	Pre_4_T1	Pos_4_T1
S2	4	T3	e6	WAMIT simulation has begun.	Pre_4_T3	Pos_4_T3
S2	4	DM1	e7	WAMIT simulation has begun.	Pre_4_DM1	Pos_4_DM1
S2	5	T0	e2	End of WAMIT simulation.	Pre_5_T0	Pos_5_T0
S2	5	T1	e4	End of WAMIT simulation.	Pre_5_T1	Pos_5_T1
S2	5	T3	e6	End of WAMIT simulation.	Pre_5_T3	Pos_5_T3
S2	5	DM1	e7	End of WAMIT simulation.	Pre_5_DM1	Pos_5_DM1
T3	7	T0	e3	DYNASIM simulation has begun.	Pre_7_T0	Pos_7_T0
T3	7	T1	e5	DYNASIM simulation has begun.	Pre_7_T1	Pos_7_T1
T3	7	DM1	e7	DYNASIM simulation has begun.	Pre_7_DM1	Pos_7_DM1
T3	8	T0	e3	DYNASIM result.	Pre_8_T0	Pos_8_T0
T3	8	T1	e5	DYNASIM result.	Pre_8_T1	Pos_8_T1
T3	8	DM1	e7	DYNASIM result.	Pre_8_DM1	Pos_8_DM1
DM1	11	TT	e7	Decision made.	Pre_11_TT	Pos_11_TT

Table 15 - Message attributes table for the first model for the disaster application

With the role rules and the message attributes table just described, we are able to keep the collaborative session data, such as name of the executor, date, and time of each action performed, in one database. This is a very useful feature, helping the specialists while elaborating an investigation report about the accident, as well as serving for training purposes.

5.1.1.1. An HLA-Compliant Prototype for the First Model for the Disaster Management Collaborative Application

We now reach the last step in developing our disaster management collaborative application, which is to map our model into an implementation-level architecture. In this first subsection, we present an HLA-compliant prototype for our first model (Figure 20).

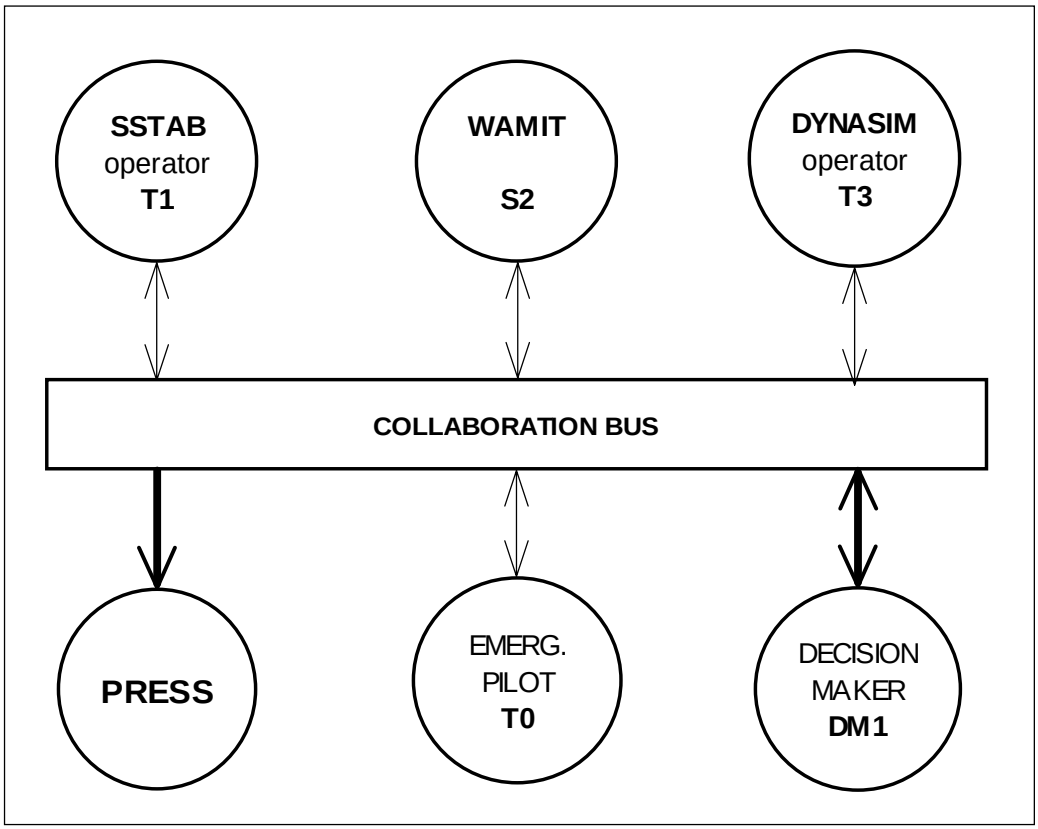

Figure 20 - HLA-compliant prototype for the first model for the disaster application

Observing our Activity-Centred model of Figure 19, we see that, since all the nodes are connected, all participant members can constitute a single

Federation, which we call *integrated_simulation* according to the conceptual collaboration of Tables 11, 12, 13, and 14.

We then associate a Federate with each participant member of the *integrated_simulation* Federation, namely T0, T1, T3, and DM1 (persons), S2 (software agent), TT (group), and Press (company). Each Federate code is a Java program built based on the workflow rules written in a logic-based program. To enhance flexibility, the main method of each Federate is the one with name *process_role*, which receives as parameter the role to be played by the Federate, coded in a separate Java module. Using this strategy, we can code the workflow rules associated with a specific role directly into a separate module dedicated to this role.

In terms of the network part of our Activity-Centred model, in fact a tree, we have implemented it using linked lists in Java language. One interesting point we should consider here is the implementation of the *send* and the *receive* algorithms used by each Federate.

We first remember that the HLA RTI chosen, XRTI, has as its communication model a client-server topology, with no point-to-point communication and simply sending the messages through a channel of the collaboration bus with no specific receiver. So, in order to implement some coordination rules specifically related to particular receivers, we had to think about a way of indicating receivers in messages.

Associated with each message, XRTI has a byte array called *userSuppliedTag*. Our prototype has seven participants (six leaf nodes and one group node, TT) and so we use one byte of this array to represent each one of the message receivers, with each byte array position corresponding to one participant.

We can now explain each one of the algorithms:

- *Send*: this algorithm can be implemented in two basic forms. In the first one, we directly implement the conceptual algorithm described in Chapter 3, sending the same basic message to each different receiver associated with it in the message attributes table, using the *userSuppliedTag* to represent to which receiver the message is being sent (apart from also registering the *message_id* in it). This allows flexibility not only in terms of a different *post-processing module* associated with each pair (*message_id*, *receiver*) present in the

message table, but also in terms of *pre-processing modules*, since we will be able to execute a different one associated with each receiver. This seems to be reasonable regarding our application with few nodes. If we find that it would be more interesting to lose some flexibility and enhance the network performance, we can adopt another strategy. Before sending the message through the channel, we could process every line of the message attributes table associated to this particular pair (*sender*, *message_id*) and represent all the receivers found in the *userSuppliedTag* byte array. Then we send through the channel only one single message associated with this *message_id*, with all of its receivers indicated in the *userSuppliedTag*. In this way, we gain in network performance by decreasing its load and lose some flexibility for not being able to execute the pre-processing modules prescribed for each message associated with each receiver (bear in mind that we do not lose all the flexibility, since we still are able to execute the *post-processing* at the receiver side). Probably more interesting than using these two strategies would be to implement a mix of them: before sending the message through the channel, we still process every line of the message attributes table, but at this time, instead of only grouping messages using the pair (*sender*, *message_id*), we also verify if they use the same *pre-processing module*, in which case we combine the messages with the same attributes *sender*, *message_id*, and *pre_processing* into a single one via the *userSuppliedTag*. Using this strategy, we maintain all our original flexibility and reduce the network load to the possible minimum.

- *Receive*: it simply verifies through the *userSuppliedTag* byte array if the message is destined to the present Federate and then uses the pair (*message_id*, *receiver*) to access the message attributes table and discover the *post-processing module* to be run. An interesting case is the one in which the receiver represented in the *userSuppliedTag* is a group node (TT in our prototype). Recalling Chapter 3, we already know that, for group nodes, there is an attribute indicating which leaf node (in our implementation, a Federate) will execute the *receive*

algorithm and then the *post-processing module* associated with the pair (*message_id*, *group_node*). This *post-processing module* will necessarily determine which receivers (leaf nodes) will receive the message.

Some screenshots of this integrated simulation collaborative session using the first version of our HLA-compliant prototype can be seen in the Appendix.

Beyond a message passing application (a restriction imposed by the real simulators available), we thought about forms of enhancing the collaboration. One simple feature that greatly improved awareness information was the addition of a video capture tool in each of the two interactive simulators, transmitting frames periodically through the bus to the other participants. In this way, they not only receive messages about beginning and ending a particular simulation, but also receive intermediate frames with the simulation evolution.

We finish this subsection highlighting two points of the XRTI implementation:

- The Management Object Model (MOM): each set of object model tables has object and/or interaction class structure tables that depict inheritance relationships between classes. For each class, there is a flag indicating if a Federate can publish, subscribe, publish and subscribe, or neither publish nor subscribe instances of the class. There are also tables describing object attributes and interaction parameters, such as name, data type, update type, update condition, ownership transfer capability, publication and subscription capability, available dimensions, transportation, and order types of each attribute of every object class.
- The *mergeFDD* method: FDD means Federation Object Model Document Data. XRTI implements a method called *mergeFDD* that allows Federates to add contents of other FDDs to the current FOM during a Federation execution. This encourages the use of lightweight, composable models. So, under XRTI, Federates can specify an FDD containing only the MOM (a mandatory component of every FOM) as the initial FDD, then merge smaller FDDs into the FOM as needed.

5.1.1.2. An InfoGrid Prototype for the First Model for the Disaster Management Collaborative Application

We now present in Table 16 an InfoGrid prototype for our first model for the disaster management collaborative application.

The InfoGrid prototype is described in terms of sequential time events, with the first one responsible for creating the project which will contain our model. Each time event is initiated by a term indicating the time in which the event occurs: t_0, t_1, t_2, t_3, etc. Also, when another event occurs immediately after the occurrence of a previous one, we denote this by adding Δ to the subscript, such as $t_{3+\Delta}$ and $t_{5+\Delta}$, occurring, respectively, immediately after t_3 and t_5 in our example. For each time event, we define a sequence of commands that are executed. The whole set of events and commands defines the collaborative application.

t_0: T0 creates a project
T0 authorises T1 and T3 to access the project
T0 sends persistent notifications
T0 writes data in the project (ex.: P-34 model)
t_1: T1 enters
T1 receives past and persistent notifications //edge e1 asynchronous
T1 remotely accesses the model in the project and executes SSTAB
T1 sends message to all notifying the beginning of SSTAB simulation
t_2: T3 receives notification by e-mail
t_3: T1 writes the SSTAB results in the project
T1 sends message to all notifying the end of SSTAB simulation
T1 sets parameters and activates S2
$t_{3+\Delta}$: all the participants receive notification of the activation of S2 simulation (automatic send by the system)
t_4: T3 enters
T3 receives notifications
t_5: all the participants receive notification of the end of S2 simulation (automatic send by the system)
$t_{5+\Delta}$: T3 accesses S2 result and starts DYNASIM
T3 sends message to all notifying the beginning of DYNASIM simulation
t_6: T3 writes the DYNASIM results in the project
T3 sends message to all notifying the end of DYNASIM simulation
t_7: T0 analyses DYNASIM results
T0 sends message to DM asking for his approval
t_8: DM receives notification by e-mail
DM enters
DM analyses the results
DM notifies T0 about his approval
t_9: T0 extracts results to be published by Press
T0 sends the results to be published to Press

Table 16 - InfoGrid prototype for the first model for the disaster application

An interesting aspect of the InfoGrid prototype is that it allows both persistent and volatile notifications. Also all the notifications can be sent by e-mail, which can be the case for non-critical messages.

5.1.2. A Second Model for the Disaster Management Collaborative Application

We derived our first model for the disaster management application considering the actual simulators available today at Petrobras. Now imagine that we could configure our model ideally, with no implementation restrictions. A possible conceptual architecture would be one with all the participants separated from the simulations' engines and being capable of activating them remotely from other sites and to display their outputs using a local user interface program. We would then have the Task Force group again co-located in an especial room with all the required facilities, but at this time activating remotely the simulators available at other sites of the company. Again, since in our model TF is the only team inside TT, we are not explicitly representing it (if it had particular in- and out-policies, it should also be a node of our model). The Decision Maker continues to play his role of making the decision remotely to the Task Force.

The model derived from our Activity-Centred metamodel and correspondent to this new scenario is the one shown in Figure 21.

We can see that it was not difficult to accommodate the previous model to this new approach. All we had to do was define new nodes corresponding to each new simulation engine (S1 and S3) and define the interactions among them and with the other nodes. It is interesting to observe that in this second model we have as many agents (S1, S2, and S3) as persons (T0, T1, and T3).

5.1.2.1. An HLA-Compliant Prototype for the Second Model for the Disaster Management Collaborative Application

We now present the HLA-compliant prototype implementing the second model for the disaster management application (Figure 22).

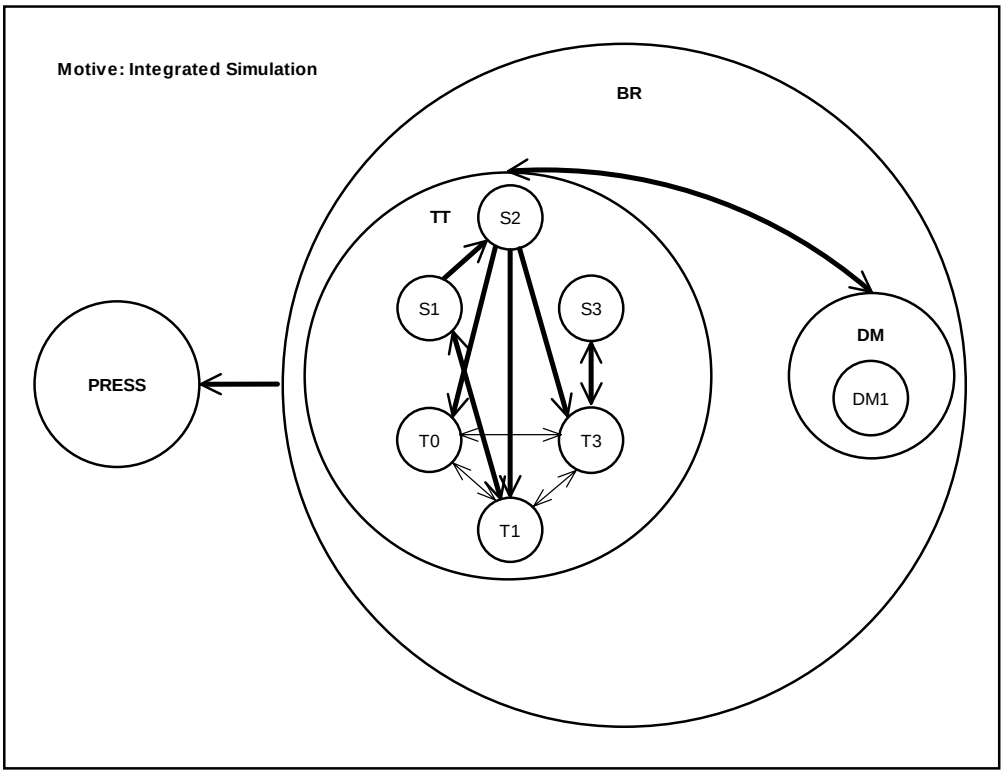

Figure 21 - Second model for the disaster management collaborative application, focussing on the integrated simulation

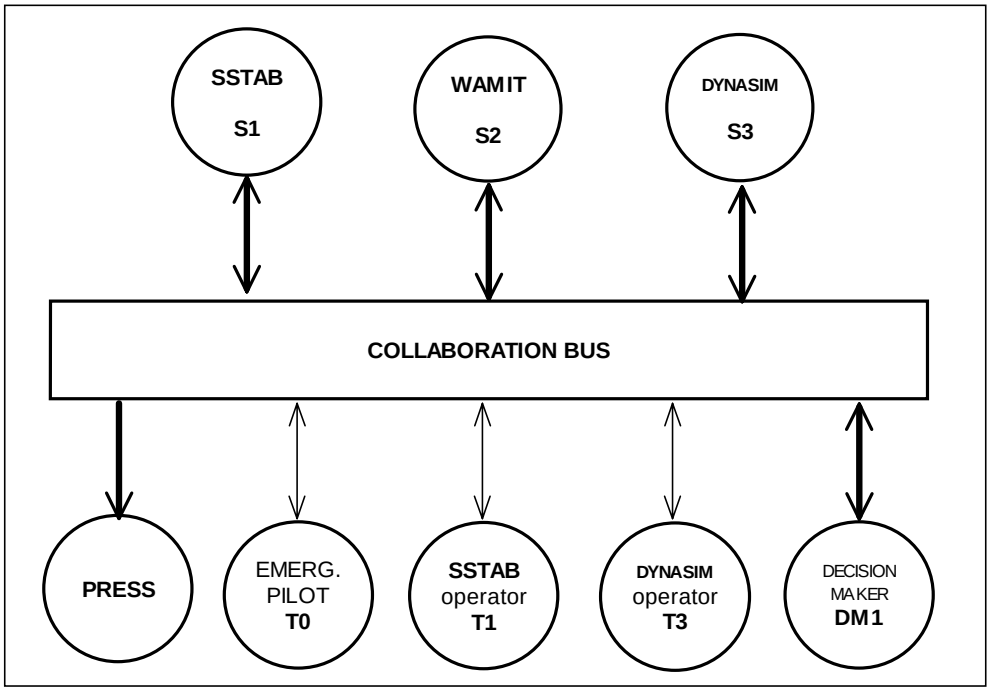

Figure 22 - HLA-compliant prototype for the second model for the disaster application

What differs in this second HLA prototype when compared to the first one is that now we have two new Federates, each one corresponding to each of the

new simulation engines: SSTAB and DYNASIM. SSTAB and DYNASIM operators still communicate with the other participants and now also with their respective simulation engines. The remaining nodes retain their previous behaviours.

5.2. CAD Visualisation in Virtual Environments

In order to validate the generality of the metamodel, we now delineate a model for another application, namely CAD visualisation in virtual environments.

We again consider a case study of the Brazilian oil & gas company Petrobras. Suppose that one Exploration & Production (E&P) Unit is developing a project and that during a specific phase of the project they need to update and validate a CAD model within another E&P Unit and with the General Manager of the E&P Department. The typical workflow for this application would be as follows:

- The technician from E&P Unit 1 sends the CAD model to E&P Unit 2 so that it can be updated considering the unit's particular aspects. A first question that arises here is that E&P Unit 1 uses one CAD software (which we are going to name A) while E&P Unit 2 uses both this CAD software and another one from a second vendor (which we are going to name B), depending on the age of the CAD model (they use software A for the old models and software B for the new ones). This means that the application may have to convert models from software A to software B.
- After having updated the CAD model, the technician from E&P Unit 2 (using software A or B) then returns the model to the technician from E&P Unit 1. We can also have here the need to convert models, depending on the software being used at each side.
- The technician from E&P Unit 1 now sends the updated model to be validated by the General Manager, who is working in a Virtual Reality environment. This means that once again we need model conversion, now from a CAD environment to a Virtual Reality environment.

- Finally the General Manager sends back a message to the technician from E&P Unit 1 telling whether he has or has not approved the updated model.

In an emergency condition, the CAD visualisation system should be used by specialists to discuss the simulation results before they show them to the decision makers.

5.2.1. A Model for CAD Visualisation in Virtual Environments

We are now going to derive a complete Activity-Centred model for this new application.

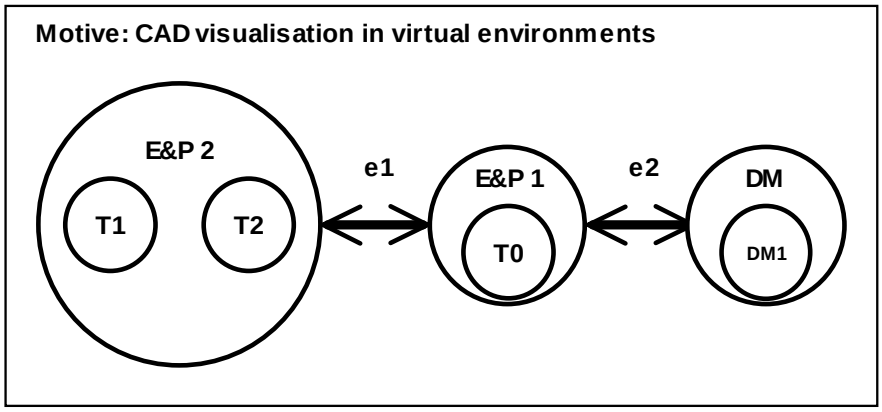

Figure 23 - Model for CAD visualisation in virtual environments

First we define the network component as shown in Figure 23. There we can see three main nodes: node E&P 1 leading the collaborative application and, remotely located to it, node E&P 2 and node Decision Maker (DM).

Inside node E&P 1 we have one E&P technician (T0) running CAD software A.

Inside node E&P 2 we have two E&P technicians, T1 running software A for old CAD models and T2 running software B for new CAD models.

Finally inside node DM, we have General Manager DM1 (Decision Maker 1) using a Virtual Reality software in a virtual environment.

Having defined the network component of our model, we are now going to present the role rules and the message attributes table defining the application workflow.

The role rules for the E&P technician 0, the E&P Unit 2, the E&P technician 1, and the E&P technician 2 are shown in Table 17. The rules defining the role of E&P technician 0 are similar to those of the examples previously shown. The E&P Unit 2 group role has an *on-arrive* rule that fires upon receiving message 1, when we inform on the console that the post-processing module *pos_1_EP2* is determining and re-routing the message to technicians from E&P Unit 2. In this example, we do not know in advance which of the two technicians from E&P Unit 2 will interact with the technician from E&P Unit 1. In spite of that, our model is still capable of accommodating this new situation, with post-processing module *pos_1_EP2* being responsible for determining which of the two technicians from E&P Unit 2 will participate in the collaborative session, depending on the date of the CAD model.

The role rules for the decision maker are shown in Table 18. The noteworthy aspect of this role is that we have an *on-arrive* rule firing a sequence of commands on arrive of message 3. This sequence first displays a message requesting the model validation, then displays the updated CAD model, and waits until the decision maker validates or not the model displayed.

It is important to note that, in both Tables, messages 1, 2, and 3 go with the model as an attached file.

Finally in Table 19 we show the message attributes table for this model. We first note that, for simplification, we are replacing E&P Unit 2 with EP2. Second, only *message_id* together with *receiver* were not sufficient to designate our pre- and post-processing module names – we had also to consider the *sender*. This occurs because, although in fact not interacting simultaneously during the same collaborative session, T1 and T2 are both represented in the message attributes table as senders of messages with the same *message_id* (2). Simply also considering the *sender* while building the pre- and post-processing module names, our metamodel proved to be capable of handling this new condition.

```
collaboration CAD_VR_visualisation
{
 collaboration_bus {
            channel(remote).
                    }

 role E&P_Unit_1_technician_0 //technician responsible for coordinating the activities
 {
  on-init(CAD_VR_visualisation) :-
   send(remote, message_table(source(self), 1, dummy)). //send "E&P Unit 2 technician, please update the CAD model."

  on-arrive(local, message_table(dummy, 2, source(self))) :-
   display(message_table(dummy, 2, source(self))), //display "CAD model has been updated."
   display the updated CAD model,
   read CAD_model_seen_button, //read CAD_model_seen_button from console
   send(remote, message_table(source(self), 3, dummy)). //send "Decision Maker, please validate the model."

  on-arrive(remote, message_table(dummy, 4, source(self))) :-
   display(message_table(dummy, 4, source(self))). //display the decision made
 }

 role E&P_Unit_2 //unit responsible for routing the CAD model to technician 1 or  2
 {
  on-arrive(local, message_table(dummy, 1, source(self))) :-
   display(message_table(dummy, 1, source(self))). //pos_1_EP2 determines which technician will receive message 1
 }

 role E&P_Unit_2_technician_1 //technician responsible for updating the CAD model using software A
 {
  on-arrive(local, message_table(dummy, 1, source(self))) :-
   display(message_table(dummy, 1, source(self))), //display "E&P_Unit_2 technician, please update the CAD model."
   read CAD_model_updated_button, //read CAD_model_updated_button from console
   send(remote, message_table(source(self), 2, dummy)). //send "CAD model has been updated."
 }
 role E&P_Unit_2_technician_2 //technician responsible for updating the CAD model using software B
 {
  on-arrive(local, message_table(dummy, 1, source(self))) :-
   display(message_table(dummy, 1, source(self))), //display "E&P_Unit_2 technician, please update the CAD model."
   read CAD_model_updated_button, //read CAD_model_updated_button from console
   send(remote, message_table(source(self), 2, dummy)). //send "CAD model has been updated."
 }
```

Table 17 - Role rules for E&P technicians 0, 1, and 2, and for E&P Unit 2

The last aspect to be considered about this application is the appropriate selection of processing modules (pre- or post-) to execute the conversion of the engineering models. We have to consider each particular message to select the appropriate processing module to it:

```
role decision_maker //manager responsible for making the decision
{
  on-arrive(remote, message_table(dummy, 3, source(self))) :-
    display(message_table(dummy, 3, source(self))), //display "Decision Maker, please validate the model."
    display the updated CAD model,
    read model_validated, //read model_validated (OK or not_OK) from console
    assert model_validated. //assert model_validated in the knowledge base

  when(model_validated, not_OK) :-
    write message 4 with "The CAD model has not been validated.",
    send(remote, message_table(source(self), 4, dummy)). //send message 4

  when(model_validated, OK) :-
    write message 4 with "CAD model has been validated.",
    send(remote, message_table(source(self), 4, dummy)). //send message 4
 }
}
```

Table 18 - Role rules for the decision maker

sender	message_id	receiver	edge	pre-processing	post-processing
T0	1	EP2	e1	Pre_1_EP2	Pos_1_EP2
T0	3	DM1	e2	Pre_3_DM1	Pos_3_DM1
T1	2	T0	e1	Pre1_2_T0	Pos1_2_T0
T2	2	T0	e1	Pre2_2_T0	Pos2_2_T0
DM1	4	T0	e2	Pre_4_T0	Pos_4_T0

Table 19 - Message attributes table for the CAD visualisation application

- Message 1: sent from T0 to E&P Unit 2. As T0 does not know in advance which software E&P Unit 2 will use, he simply sends the message with the model through the channel, without any conversion, and lets E&P Unit 2 determine if any conversion will occur. At the E&P Unit 2 side, the post-processing module *Pos_1_EP2* will determine to which of the two E&P Unit 2 technicians the message should be delivered, based on the date of the CAD model (this determines which software will be used).
- Message 2: this message is sent by one of the two E&P Unit 2 technicians to T0. If it is T1 who is participating in the collaborative session (using software A), there will not be any conversion while sending message 2. If it is T2 who is participating (using software B), then in this case the pre-processing module *Pre2_2_T0* should convert the model from format B to format A, guaranteeing that it arrives at the T0 side already converted. This is important because,

using *dummy* as the sender in the *on-arrive* rules, we will not be capable of determining precisely in this case which line of the table corresponds to the message being received and so we will not be able to use the post-processing modules. Another alternative would be using *source(sender)* instead of *dummy* in the *on-arrive* rules (the *sender* information has to be included in the message), in order to be able to determine exactly the line of the message attributes table corresponding to an incoming message – in this case, we could use either the pre- or the post-processing module to make the conversion of formats.

- Message 3: this is sent from T0 to DM1. In principle, we could use either the pre- or the post-processing module to convert the CAD model to the Virtual Reality environment model. But it seems more natural to allow the Decision Maker to use his post-processing module to make this conversion since he is the one who better knows the conditions and resources of the Virtual Reality environment.

The objective of this second example was to validate the generality of the metamodel, applying it to another scenario, not related to emergencies. This new scenario brings at least two new interesting situations that our metamodel can accomplish. The first one is the fact that we have a group node (E&P Unit 2) containing two leaf nodes which do not communicate between themselves – the only communication with them comes through their parent node. The second one is the fact that we have two pre-defined participants in the network component and in the role rules – E&P Unit 2 Technicians 1 and 2 – which do not participate in the same collaborative session: there is a run-time selection based on an attribute of part of the message (the date of the CAD model transmitted with the message as an attached file). This causes the appearance of two lines in the message attributes table having the same *message_id* with two different senders, apparently failing to determine which of the lines is related to the message being received. However, by simply using the optional *sender* argument of the *on-arrive* rule with the sender information included in the message, we can deal with this situation.

6
Conclusions and Future Work

In this work we have proposed a multiple-perspective metamodel – the Activity-Centred metamodel – which mixes the Place-Centred and the People-Centred perspectives in the degree required by the designer. It employs not a technology-driven but a human-centred approach – or better, as it also emphasises group and organisation level issues, a socially-centred approach (Neale et al., 2004) or a mix of both (Gutwin & Greenberg, 1998). It allows experimenting and deriving models in two orthogonal dimensions: in relation to the number of group levels and breaking down a particular group level. Associating pre- and post-communication processing to each of these levels, we could accommodate policy and privacy rules of organisations, even allowing inter-organisational work.

In our metamodel, we have identified elements associated with each of the three components of the 3C collaboration model: the communication component is represented by the edges; the coordination component is represented by the edge specialisation elements, the role rules, and the message attributes table; and finally the cooperation component is formed by the different abstraction levels which represent the whole activity being performed, including all the applications being executed in the leaf nodes and the artefacts being produced or manipulated. The leaf nodes can indistinctly be persons or software agents.

The metamodel allows flexibility in many dimensions. Separating high-level abstraction features from low-level implementation features allows the designer and the application developer to concentrate on their particular domain of expertise. The use of edges allows synchronous and asynchronous work. Separating the computational program and the coordination program (the role rules) allows programmers to concentrate on coordination issues with high-level abstraction. The role rules, written in a logic-based language, specify coordination policies and participants' roles, accommodating both tightly- and loosely-coupled work. With the participants divided in a number of fixed roles, the number of participants assuming these roles can be fluid and large.

More directly related to particular characteristics of its elements, the metamodel also provides a set of other flexibility features. It is customisable in the sense that it allows associating pre- and post-communication processing modules with each message sent. It allows parametric run-time changes such as changing the names of pre- and post-communication processing modules in the message attributes table, or even the possibility of changing the pre- and post-communication codes before they have been loaded during a collaborative session. The roles are coded in modules separated from the computational program, in consonance with the principle of reusable collaborative software components. Finally, it also provides some extensibility, having the capability of adding new lines in the message attributes table, associating a specific message with new receivers, as well as loading new post-processing modules in run-time.

The metamodel seems to have elements to be sufficiently generic to accommodate a range of applications as well as being specialisable for a particular domain. We also conclude that the metamodel offers conditions to develop a collaborative workspace: it allows sharing materials among a specific group and provides appropriate behaviours and awareness information to support cooperation.

The metamodel allows storing all the elements of a collaborative session, such as the applications being executed in the leaf nodes, the artefacts being produced or manipulated, and the name of the executor of the application, as well as its date and time. This may be used at the conceptual level as a knowledge base for post-auditing or reconfiguration of the model in future sessions, as well as at the real-world level, helping specialists while elaborating an investigation report about the accident, or for training purposes.

Another important characteristic of this work is that it was motivated by and was conducted in real-world settings, namely an oil & gas offshore structure disaster scenario. This seems to contribute to the CSCW field, since a review of CSCW evaluation studies concluded that less than half of them were conducted in real-world settings (Pinelle, 2000). We have conducted a survey on the main commercial emergency management systems available, concluding that they still lack full and suitable integration of simulators with high-performance visualisation systems. The decision of using HLA (High Level Architecture) as a

platform for our prototype favours the inclusion of more complex simulations, even those with high-performance visualisation requirements.

We believe that, considering the generality of the metamodel, we may support other application domains, but only the ones related to the oil & gas area, in particular the emergency scenario ones, were thoroughly tested.

After deriving from our metamodel an adequate model for the disaster scenario, we have implemented our first prototype, as a proof-of-concept of our metamodel, using an HLA run-time infrastructure, namely the XRTI. It comprises all the flexibility requirements demanded as well as is open-source and freely distributable. It has proven particularly suitable to accommodate a varying number of participants (in our model associated with roles) since HLA allows Federates to freely join or resign from Federations at any time during the collaborative session.

We have also discussed how the prototype could be implemented using another implementation platform, InfoGrid. Still for the disaster management application, we have presented a second model and its prototype showing that we can derive different models for the same application. Finally, to validate the generality of the metamodel, we have also delineated a model for another application, namely CAD visualisation in virtual environments.

In terms of contributions for Petrobras, this work has provided:

- Improved understanding of ICT requirements from the oil & gas industry for supporting disaster management involving distributed expert teams.
- Creation of a decision-making environment which integrates simulators, visualisation centres, and various experts for emergency situations.
- Evaluation results of the prototypes within real industrial settings using experts involved in real disaster management situations.

6.1. Future Work

We have a lot of interesting future work related to the present one that could be developed in many dimensions.

First we consider our metamodel. Although providing the degree of flexibility mentioned above, there is still a lot of work to do in order to make our metamodel a fully flexible and evolving collaborative architecture:

- First, it would be interesting to include a latecomer-joining mechanism such as Chung & Dewan's (2004) logger, capable of replaying recorded output messages, or an image copy method, since it is usual to have people joining and leaving collaborative sessions.
- Second, particularly in the situation of an emergency scenario being considered, it would be very important to include an Expertise Recommender system, such as the one proposed by McDonald & Ackerman (2000), since in a crisis situation it is fundamental to locate the expertise necessary to solve the problem in the lesser possible time.
- Third, again directly related to the particular case of emergency scenario, it seems that it would be interesting to include some kind of redundancy, in one or more of the dimensions considered by Tjora (2004): redundancy of functions, communicative redundancy, competence redundancy or technological redundancy.
- Fourth, it should be investigated how the number of lines of the message attributes table escalates, considering many aspects such as the quantity of nodes and edges, the quantity of different messages, the quantity of variations of a same message, and the quantity of group nodes (which re-route messages).
- Finally, we should investigate how to promote our metamodel from a customisable category to an adaptable category (Dourish, 1998), upgrading from the capability of adjusting parametric controls to the capability of reconfiguring its behaviour according to immediate patterns of use. We can think about implementing this in two phases. In a first phase, we would monitor the current activities and interactions among users to derive a diagnosis, which would serve as the base for possibly adjusting the model to the found pattern, and then use the adjusted model in subsequent collaborative sessions. In

a second phase, we would use some kind of learning mechanism to make these adjustments in run-time.

The second dimension to be considered is that of tools to help the configuration and implementation of models:

- Since we have identified the importance of being capable to rapidly reconfigure the model, especially in emergency situations, it would be very interesting to have some kind of diagram editor to configure and reconfigure models as well as automatically updating the database containing the network models.
- Also it seems to be very useful to have automatic mappers of our role rules to the implementation programming languages being used.

A third dimension to be considered is the issue of validating the metamodel. Validation is always a very complicated problem while developing a large system. We suggest validating the metamodel according to different aspects:

- First, for a particular scenario and using the semi-formal representation constituted by the network model of our metamodel, the users should validate the particular model being employed, verifying, for example, if all the important groups are represented, if the groups are placed in their right positions, if there are no missing edges, and if the representation of local and remote nodes is correct.
- Second, the users should verify all the role rules to see if they really reflect what is happening in the real world and if there is no missing rule.
- Third, the users should verify if all the important pre- and post-processing modules are represented in the message attributes table. Also, the users should verify if the messages being sent to group nodes are the appropriate ones.
- In a second dimension of validation, we could implement a prototype for a particular application using different implementation platforms to validate the metamodel, as we have done with the emergency scenario application using HLA and InfoGrid.

- In a third dimension of validation, we could experiment different models for the same application, to verify how flexible the metamodel is, similarly to what has been presented in Subsections 5.1.1 and 5.1.2.
- Finally, in a fourth dimension of validation, to validate the generality of the metamodel, we should derive models for different types of applications. The CAD visualisation application (Section 5.2) is an example of this kind of validation.

A fourth dimension to be considered in terms of future work is the one related to the HLA-compliant prototype. Although already including some features in our first prototype such as a video capture tool and an automatic language translator for small messages, there are still many new features that could be added to enhance our prototype. One of those could be a video teleconferencing facility allowing communication among the participants in real time.

A fifth dimension is to fully develop the InfoGrid prototype so that we can compare it with the HLA-compliant prototype, identifying what are the advantages of each one of them.

The sixth dimension is related to developing new applications to use and validate our metamodel under different conditions. One application that should be further developed is the CAD visualisation in virtual environments described in Chapter 5. Particularly in a multifunctional room or in a virtual reality environment, CAD interaction would be critical. Another application to be investigated in the near future is again one related to the oil & gas area, namely the Reservoir Dimensioning application.

Finally, the seventh dimension of future work is the one related to enhancing the collaborative workspace for the disaster management application. This could be done in two main directions:

- In order to simulate the operator's actions, the system should work with two different scenes, each of them with the possibility of choosing between the first- and the third-person point of view:
  - The first scene would be the normal scene of the interactive simulators (SSTAB and DYNASIM), with the users visualising

the unit movements in the virtual world based on the stability conditions.

- The second virtual scene would be the unit control room, so that the specialists could simulate unit reactions based on the activation of buttons.

- In addition to these two virtual scenes, the system should have at least one and possibly two projections of the real world:
  - The most important one would be the image of the unit control room, captured by a video camera. With this, the users could observe and help the operator's actions in these extreme conditions.
  - A second possible one would be the outside image of the unit, again captured by a video camera possibly in a ship or in a helicopter, so that the users could observe the real situation of the unit.

7
References

ARMSTRONG, C. R.; GANNON, D.; GEIST, A.; KEAHEY, K.; KOHN, S.; MCINNES, L.; PARKER, S.; SMOLENSKI, B. Towards a Common Component Architecture for High-Performance Scientific Computing. In: 8TH IEEE International Symposium on High Performance Distributed Computing – HPDC 1999, 1999. **Proceedings of 8th IEEE International Symposium on High Performance Distributed Computing – HPDC 1999**, 1999.

ARSENAULT, L. et al. DIVERSE: a Software Toolkit to Integrate Distributed Simulations with Heterogeneous Virtual Environments. Technical Report TR-01-10, Computer Science, Virginia Tech, 2001.

ATHANASOPOULOS, G.; LIASKOVITIS, P.; PILIOURA, T.; SOTIROPOULOU, A.; TSALGATIDOU, A.; YPODIMATOPOULOS, P. D8: Models and specification primitives for building dependable P2P Systems/Applications. University of Athens, Information Societies Technology (IST) Programme, IST-2001-32708, P2P Architect, June 2003, 106 p.

BENFORD, S.; BULLOCK, A.; COOK, N.; HARVEY, P.; INGRAM, R.; LEE, O. K. A spatial model of cooperation for virtual worlds. In: INFORMATIQUE'93: INTERFACE TO REAL & VIRTUAL WORLDS, 1993, University of Nottingham. **Proceedings of INFORMATIQUE'93**, 1993, p. 445-456.

BIERBAUM, A.; JUST, C. Software Tools for Application Development. In: SIGGRAPH'98 CONFERENCE, July 19-24, 1998, Orlando, Florida, USA. **SIGGRAPH'98 Course 14**, 1998, p. 3-1, 3-45.

BIERBAUM, A.; JUST, C.; HARTLING, P.; MEINERT, K.; BAKER, A.; CRUZ-NEIRA, C. VRJuggler: A Virtual Platform for Virtual Reality Application Development. In: IEEE VR2001, 2001, Yokohama. **Proceedings of IEEE VR2001**, 2001.

BOOZ ALLEN HAMILTON. Automated Resource Management System (ARMS), System Requirements Document (SRD). Federal Emergency Management Agency, Contact No.: GS-35F-0306J, FEMA BPA # EMV-2001-BP-0147, Task Order 11, USA, 2003.

BOS, N.; SHAMI, N. S.; OLSON, J. S.; CHESHIN, A.; NAN, D. N. In-group/Out-group Effects in Distributed Teams: An Experimental Simulation. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 429-436.

BRIGNULL, H.; IZADI, S.; FITZPATRICK, G.; ROGERS, Y.; RODDEN, T. The Introduction of a Shared Interactive Surface into a Communal Space. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 49-58.

BUHR, R. J. A.; CASSELMAN, R. S. **Use CASE Maps for Object-Oriented Systems**. Prentice-Hall, 1996.

CAPPS, M.; MCGREGOR, D.; BRUTZMAN, D.; ZYDA, M. Npsnet-v a new beginning for dynamically extensible virtual environments. **IEEE Computer Graphics and Applications**, 2000, p. 12-15.

CHUNG, G.; DEWAN, P. Towards Dynamic Collaboration Architectures. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 1-10.

COELHO, L. C. G.; JORDANI, C. G.; OLIVEIRA, M. C.; MASETTI, I. Q. Equilibrium, Ballast Control and Free-Surface Effect Computations Using The Sstab System. In: 8TH INTERNATIONAL CONFERENCE ON STABILITY OF SHIPS AND OCEAN VEHICLES – STAB, 2003. **Proceedings of the 8th International Conference on Stability of Ships and Ocean Vehicles - Stab**, 2003, p. 377-388.

COELHO, L. C. G.; NISHIMOTO, K.; MASETTI, I. Q. Dynamic Simulation of Anchoring Systems Using Computer Graphics. In: 20TH INTERNATIONAL CONFERENCE ON OFFSHORE MECHANICS & ARTIC ENGINEERING – OMAE 2001, June 3-8, 2001, Rio de Janeiro, RJ, Brazil. **Proceedings of the 20th International Conference on Mechanics & Artic Engineering – OMAE**, 2001.

COHEN, A. L.; CASH, D.; MULLER, M. J. Designing to Support Adversarial Collaboration. In: CSCW'00, December 2-6, 2000, Philadelphia, PA, USA. **Proceedings of CSCW'00**, 2000, p. 31-39.

COMMON COMPONENT ARCHITECTURE FORUM, THE. Available in: <http://www.cca-forum.org>. Access on: February 25th, 2006.

CORTÉS, M.; MISHRA, P. DCWPL: A Programming Language For Describing Collaborative Work. In: CSCW'96, 1996, Cambridge, MA, USA. **Proceedings of CSCW'96**, 1996, p. 21-29.

COSTA, T. V. M. **Modelos de aprendizado organizacional de Chris Argyris: Uma investigação em segurança operacional e conservação do meio ambiente (SCA) com base em acidentes da indústria do petróleo**. 2004. 159 p. Tese de Doutorado – Coordenação dos Programas de Pós-Graduação de Engenharia, Engenharia de Produção, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2004.

COVER – COVISE VIRTUAL ENVIRONMENT. Available in: <http://www.hlrs.de/organization/vis/covise/features/cover/>. Access on: March, 2nd, 2006.

CSBASE. Available in: <http://www.tecgraf.puc-rio.br/csbase/>. Access on: February 25th, 2006.

DAHMANN, J.; WEATHERLY, R.; KUHL, F. **Creating Computer Simulation Systems: An Introduction to the High Level Architecture**. Upper Saddle River, NJ: Prentice-Hall, 1999.

DAVID, J. M. N.; BORGES, M. R. S. Selectivity of Awareness Components in Asynchronous CSCW Environments. In: 7TH INTERNATIONAL WORKSHOP ON GROUPWARE – CRIWG'2001, 2001. **Proceedings of the 7th International Workshop on Groupware – CRIWG'2001**, 2001, p. 115-124.

DEFENSE MODELING AND SIMULATION OFFICE HIGH LEVEL ARQUITECTURE Web Site, THE. Available in: <https://www.dmso.mil/public/transition/hla/>. Access on: March, 2nd, 2006.

DEWAN, P. Architectures for Collaborative Applications. In: Beaudouin-Lafon (Ed.). **Computer Supported Cooperative Work**, John Wiley & Sons Ltd., 1999, p. 169-194.

DONGARRA, J. J.; HEMPEL, R.; HEY, A. J. G.; WALKER, D.W. A proposal for a user-level, message passing interface in a distributed memory environment. Technical Report TM-12231, Oak Ridge National Laboratory, 1993.

DOURISH, P. Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications. **ACM Transactions on Computer-Human Interaction**, v. 5, n. 2, p. 109-155, 1998.

DOURISH, P.; BELLOTTI, V. Awareness and Coordination in Shared Workspaces. In: CSCW'92, November, 1992. **Proceedings of CSCW'92**, 1992, p. 107-114.

DOURISH, P.; BLY, S. Portholes: Supporting Awareness in a Distributed Work Group. In: CHI'92, May 3-7, 1992. **Proceedings of CHI'92**, 1992, p. 541-547.

DYBVIG, R. K. **The Scheme Programming Language: ANSI Scheme**. 2.ed. Englewood Cliffs, NJ 07632, USA: P T R Prentice-Hall, 1996.

EDWARDS, W. K. Policies and Roles in Collaborative Applications. In: CSCW'96, 1996, Cambridge, MA, USA. **Proceedings of CSCW'96**, 1996, p. 11-20.

ELLIS, C. A.; GIBBS, S. J.; REIN, G. L. Groupware – some Issues and Experiences. **Communications of the ACM**, v. 34 , n. 1, p. 38-58, 1991.

FAGG, G. E.; MOORE, K.; DONGARRA, J. J.; GEIST, A. Scalable Network Information Processing Environment (SNIPE). In: INTERNATIONAL CONFERENCE FOR HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS, November 15-21, 1997, San Jose, CA, USA. **Proceedings of International Conference for High Performance Computing and Communications**, 1997.

FOSTER, I.; KESSELMAN, C. Globus: A Metacomputing Infrastructure Toolkit. **International Journal of Supercomputer Applications**, v. 11, n. 2, p. 115-128, 1997.

FOSTER, I.; KESSELMAN, C. (Ed.). **The GRID: Blueprint for a New Computing Infrastructure**. Morgan Kaufmann, 1998.

FUKS, H.; RAPOSO, A. B.; GEROSA, M. A.; LUCENA, C. J. P. Applying the 3C Model to Groupware Development. **International Journal of Cooperative Information Systems (IJCIS)**, World Scientific, v. 14, n. 2-3, p. 299-328, 2005.

FURUTA, R.; STOTTS, P. D. Interpreted Collaboration Protocols and their use in Groupware Prototyping. In: CSCW'94, 1994, Chapel Hill, NC, USA. **Proceedings of CSCW'94**, 1994, p. 121-131.

FUSSELL, S. R.; KRAUT, R. E.; SIEGEL, J. Coordination of Communication: Effects of Shared Visual Context on Collaborative Work. In: CSCW'00,

December 2-6, 2000, Philadelphia, PA, USA. **Proceedings of CSCW'00**, 2000, p. 21-30.

GODEFROID, P.; HERBSLEB, J. D.; JAGADEESAN, L. J.; LI, D. Ensuring Privacy in Presence Awareness Systems: An Automated Verification Approach. In: CSCW'00, December 2-6, 2000, Philadelphia, PA, USA. **Proceedings of CSCW'00**, 2000, p. 59-68.

GREENHALGH, C. Understanding the Network Requirements of Collaborative Virtual Environments. In: **___Collaborative Virtual Environments**. London: Springer, 2001, p. 55-74.

GREENHALGH, C.; PUBRICK, J.; SNOWDON, D. Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring. In: ACM CONFERENCE ON COLLABORATIVE VIRTUAL ENVIRONMENTS (CVE2000), 2000, San Francisco, USA. **Proceedings of ACM Conference on Collaborative Virtual Environments (CVE2000)**, 2000, p. 119-127.

GREENSPAN, S.; GOLDBERG, D.; WEISMER, D.; BASSO, A. Interpersonal Trust and Common Ground in Electronically Mediated Communication. In: CSCW'00, December 2-6, 2000, Philadelphia, PA, USA. **Proceedings of CSCW'00**, 2000, p. 251-260.

GRIMSHAW, A. S.; WULF W. A. The Legion Vision of a Worldwide Virtual Computer. **Communications of the ACM**, v. 40, n. 1, 1997.

GROSS, T.; PRINZ, W. Modelling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. **Computer Supported Cooperative Work**, Kluwer Academic Publishers, Netherlands, v. 13, n. 3-4, p. 283-303, August 2004.

GUTWIN, C.; GREENBERG, S. Design for Individuals, Design for Groups: Tradeoffs Between Power and Workspace Awareness. In: CSCW'98, 1998, Seattle, Washington, USA. **Proceedings of CSCW'98**, 1998, p. 207-216.

HANDEL, M.; HERBSLEB, J. D. What is Chat Doing in the Workplace? In: CSCW'02, November 16-20, 2002, New Orleans, Louisiana, USA. **Proceedings of CSCW'02**, 2002, p. 1-10.

HART, S. V. Crisis Information Management Software (CIMS) Feature Comparison Report. U.S. Department of Justice, Office of Justice Programs, National Institute of Justice, NCJ 197065, USA, 2002.

HAYNES, S. R.; PURAO, S.; SKATTEBO, A. L. Situating Evaluation in Scenarios of Use. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 92-101.

HERBSLEB, J. D.; GRINTER, R. E. Splitting the Organization and Integrating the Code: Conway's Law Revisited In: ICSE'99, 1999, Los Angeles, CA, USA. **Proceedings of ICSE'99**, 1999, p. 85-95.

HERBSLEB, J. D.; MOCKUS, A.; FINHOLT, T.A.; GRINTER, R.E. Distance, Dependencies, and Delay in a Global Collaboration. In: CSCW'00, December 2-6, 2000, Philadelphia, PA, USA. **Proceedings of CSCW'00**, 2000, p. 319-328.

HUBBOLD, R.; COOK, J.; KEATES, M.; GIBSON, S.; HOWARD, T.; MURTA, A.; WEST, A.; PETTIFER, S. GNU/MAVERIK: A micro-kernel for

large-scale virtual environments. **Presence: Teleoperators and Virtual Environments** 10, p. 22-34, 2001. ISSN 1054-7460.

IEEE - INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC. Available in: <http://www.ieee.org>. Access on: March, 3rd, 2006.

ISHII, H.; KOBAYAHSI, M.; ARITA, K. Iterative Design of Seamless Collaboration Media. **Communications of the ACM**, v. 37, n. 8, p. 83-97, August 1994.

JOHNSEN, S. O.; BJØRKLI, C.; STEIRO, T.; FARTUM, H.; HAUKENES, H.; SKRIVER, J. CRIOP®: A scenario method for Crisis Intervention and Operability analysis. Report No. STF38 A03424, SINTEF, Norway, 2004.

JONES, Q.; GRANDHI, S. A.; TERVEEN, L.; WHITTAKER, S. People-to-People-to-Geographical-Places: The P3 Framework for Location-Based Community Systems. **Computer Supported Cooperative Work**, Kluwer Academic Publishers, Netherlands, v. 13, n. 3-4, p. 249-282, August 2004.

JUNGHYUN, K.; SUNGIL, K.; UNGYEON, Y.; NAMGYU, K. COVRA-CAD: A CORBA based Virtual Reality Architecture for CAD. In: INTERNATIONAL CONFERENCE ON VIRTUAL SYSTEMS AND MULTIMEDIA, 1998. **Proceedings of International Conference on Virtual Systems and Multimedia**, 1998.

KAPOLKA, A. **The Extensible Run-Time Infrastructure (XRTI): An Experimental Implementation of Proposed Improvements to the High Level Architecture**. 2003. 133 p. Master's thesis – Naval Postgraduate School, Monterey, CA, USA, 2003.

LAUCHE, K. Collaboration Among Designers: Analysing an Activity for System Development. **Computer Supported Cooperative Work**, Kluwer Academic Publishers, Netherlands, v. 14, n. 3, p. 253-282, June 2005.

LAURILLAU, Y.; NIGAY, L. Clover Architecture for Groupware. In: CSCW'02, November 16-20, 2002, New Orleans, Louisiana, USA. **Proceedings of CSCW'02**, 2002, p. 236-245.

LEONTJEV, A. N. **Activity, Consciousness and Personality**. Englewood Cliffs, NJ, USA: Prentice-Hall, 1978.

LI, D.; MUNTZ, R. COCA: Collaboration Objects Coordination Architecture. In: CSCW'98, 1998, Seattle, Washington, USA. **Proceedings of CSCW'98**, 1998, p. 179-188.

LIMA, M. J.; MELCOP, T.; CERQUEIRA, R.; CASSINO, C.; SILVESTRE, B.; NERY, M.; URURAHY, C. CSGrid: Um Sistema para Integração de Aplicações em Grades Computacionais. In: SBRC'2005, SALÃO DE FERRAMENTAS, May 2005, Fortaleza, Ceará, Brasil.

LOCKE, J. An Introduction to the Internet Networking Environment and SIMNET/DIS. Computer Science Department, Naval Postgraduate School, USA, 1993.

MACEDONIA, M. R.; BRUTZMAN, D. P.; ZYDA, M. J.; PRATT, D. R.; BARHAM, P. T.; FALBY, J.; LOCKE, J. NPSNET: A Multi-Player 3D Virtual Environment over the Internet. In: ACM 1995 SYMPOSIUM ON

INTERACTIVE 3D GRAPHICS, 1995, Monterey, California, USA. **Proceedings of the ACM 1995 Symposium on Interactive 3D Graphics**, 1995.

MACEDONIA, M. R.; ZYDA, M. J. A Taxonomy for Networked Virtual Environments. **IEEE Mutimedia**, v. 4, n. 1, p. 48-56, 1997.

MACEDONIA, M. R.; ZYDA, M. J.; PRATT, D. R.; BRUTZMAN, D. P.; BARHAM, P. T. Exploiting Reality with Multicast groups: A Network Architecture for Large-Scale Virtual Environments. **IEEE Computer Graphics and Applications**, v. 15, n. 5, p. 38-45, 1995.

MARSH, J.; PETTIFER, S.; WEST, A. J. A technique for maintaining continuity of perception in networked virtual environments. In: UKVRSIG'99, 1999, Salford, UK. **Proceedings of the UKVRSIG'99**, 1999.

MASTAGLIO, T. W.; CALLAHAN, R. A Large-Scale Complex Virtual Environment for Team Training. **Computer**, v. 28, n. 7, 1995.

MCDONALD, D. W.; ACKERMAN, M. S. Expertise Recommender: A Flexible Recommendation System and Architecture. In: CSCW'00, December 2-6, 2000, Philadelphia, PA, USA. **Proceedings of CSCW'00**, 2000, p. 231-240.

MCLEOD INSTITUTE OF SIMULATION SCIENCES. California State University, Chico, California, USA. Available in: <http://www.ecst.csuchico.edu/~mcleod/>. Access on: March, 3rd, 2006.

MELO, R. N. Um ambiente de ferramentas integradas para desenvolvimento de sistemas interativos. In: I Simpósio Brasileiro de Engenharia de Software, October 1987. **Anais do I Simpósio Brasileiro de Engenharia de Software**, 1987, p. 159-169.

MILNER, R. **Communication and Concurrency**. Prentice-Hall International, International Series on Computer Science, 1989.

MORRIS, M. R.; MORRIS, D.; WINOGRAD, T. Individual Audio Channels with Single Display Groupware: Effects on Communication and Task Strategy. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004a, p. 242-251.

MORRIS, M. R.; RYALL, K.; SHEN, C.; FORLINES, C.; VERNIER, F. Beyond “Social Protocols”: Multi-User Coordination Policies for Co-located Groupware. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004b, p. 262-265.

MORRISON, J. The VR-Link Networked Virtual Environment Software Infrastructure. **Presence: Teleoperators and Virtual Environments**. Spring, 1995.

MPI-2. Extensions to the Message-Passing Interface. Technical Report, University of Tennessee, Knoxville, Tennessee, USA. Available in: <http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html>. Access on: March, 8th, 2006.

MPRI SHIP ANALYTICS. **L-3 CRISIS Brochure**. North Stonington, Connecticut, USA, 2003.

MURATA, T. Petri Nets: properties, analysis and applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541-580, 1989.

NARDI, B. A.; WHITTAKER, S.; BRADNER, E. Interaction and Outeraction: Instant Messaging in Action. In: CSCW'00, December 2-6, 2000, Philadelphia, PA, USA. **Proceedings of CSCW'00**, 2000, p. 79-88.

NEALE, D. C.; CARROLL, J. M.; ROSSON, M. B. Evaluating Computer-Supported Cooperative Work: Models and Frameworks. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 112-121.

OMG – Object Management Group. Available in: <http://www.omg.org>. Access on: March, 20th, 2006.

OMG – Object Management Group. The Common Object Request Broker: Architecture and Specification. Revision 2.0. OMG Document, 1995.

PAEK, T.; AGRAWALA, M.; BASU, S.; DRUCKER, S.; KRISTJANSSON, T.; LOGAN, R.; TOYAMA, K.; WILSON, A. Toward Universal Mobile Interaction for Shared Displays. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 266-269.

PALEN, L. Social, Individual & Technological Issues for Groupware Calendar Systems. In: CHI'99, 1999, Pittsburgh, PA, USA. **Proceedings of CHI'99**, 1999, p. 17-24.

PANKOKE-BABATZ, U.; SYRI, A. Collaborative Workspaces for Time Deferred Electronic Cooperation. In: GROUP'97, 1997, Phoenix, Arizona, USA. **Proceedings of GROUP'97**, 1997, p. 187-196.

PARK, K.; CHO, Y.; KRISHNAPRASAD, N.; SCHARVER, C.; LEWIS, M.; LEIGH, J.; JOHNSON, A. CAVERNSoft G2: A toolkit for High Performance Tele-Immersive Collaboration. In: ACM SYMPOSIUM ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY 2000 (VRST'00), 2000, Seoul, Korea. **Proceedings of the ACM Symposium on Virtual Reality Software and Technology 2000 (VRST'00)**, 2000, p. 8-15.

PARKER, S. G. **The SCIRun problem solving environment and computational steering software system**. PhD Thesis, 1999.

PETROBRAS, REVISTA. Rio de Janeiro, ano 10, n. 96, p. 10-13, jan. 2004.

PETROBRAS MAGAZINE. Rio de Janeiro, v. 7, n. 33, p. 30-35, 2001.

PETTERSSON, M.; RANDALL, D.; HELGESON, B. Ambiguities, Awareness and Economy: A Study of Emergency Service Work. **Computer Supported Cooperative Work**, Kluwer Academic Publishers, Netherlands, v. 13, n. 2, p. 125-154, April 2004.

PETTIFER, S.; COOK, J.; MARSH, J.; WEST, A. J. DEVA3: Architecture for a Large Scale Virtual Reality System. In: ACM SYMPOSIUM ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY 2000 (VRST'00), 2000, Seoul, Korea. **Proceedings of the ACM Symposium on Virtual Reality Software and Technology 2000 (VRST'00)**, 2000.

PINELLE, D. A Survey of Groupware Evaluations in CSCW Proceedings. Technical Report HCI-TR-2000-01, Computer Science Department, University of Saskatchewan, Canada, 2000.

RANTZAU, D.; FRANK, K.; LANG, U.; RAINER, D.; WOSSNER, U. COVISE in the CUBE: An Environment for Analysing Large and Complex Simulation

Data. In: 2ND WORKSHOP ON IMMERSIVE PROJECTION TECHNOLOGY (IPT'98), 1998, Ames, Iowa, USA. **Proceedings of the 2nd Workshop on Immersive Projection Technology (IPT'98)**, 1998.

RAPOSO, A. B.; FUKS, H. Defining Task Interdependencies and Coordination Mechanisms for Collaborative Systems. In: A. M. Pinna-Dery, K. Schmidt and P. Zaraté (Eds.). **Cooperative Systems Design: A Challenge of the Mobility Age** (Frontiers in Artificial Intelligence and Applications, v. 74). Amsterdam: IOS Press, 2002. p. 88-103.

RIBAK, A.; JACOVI, M.; SOROKA, V. "Ask Before You Search" Peer Support and Community Building with ReachOut. In: CSCW'02, November 16-20, 2002, New Orleans, Louisiana, USA. **Proceedings of CSCW'02**, 2002, p. 126-135.

ROUSSEV, V.; DEWAN, P.; JAIN, V. Composable Collaboration Infrastructures Based on Programming Patterns. In: CSCW'00, December 2-6, 2000, Philadelphia, PA, USA. **Proceedings of CSCW'00**, 2000, p. 117-126.

RUSSO, E. E. R. **Um sistema de interpretação de diálogos gráficos**. 1988. 235 p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, March 1988.

RUSSO, E. E. R.; RAPOSO, A. B.; FERNANDO, T.; GATTASS, M. Workspace Challenges for the Oil & Gas Exploration & Production Industry. In: 4TH CONFERENCE OF CONSTRUCTION APPLICATIONS OF VIRTUAL REALITY - CONVR 2004, September 14-15, 2004, Lisboa, Portugal. **Proceedings of the 4th Conference of Construction Applications of Virtual Reality - CONVR 2004**, 2004, p. 145-150.

SACHS, P. Transforming Work: Collaboration, Learning and Design, **Communications of the ACM**, v. 38, n. 9, p. 36-44, September 1995.

SANTOS, A. G. **Sistema de gerência de interface do usuário: Especificação e interpretação de diálogos**. 1986. 93 p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, August 1986.

SCHMIDT, D. The adaptive communication environment: Object-orientated network programming components for developing client/server applications. Technical report, 11th and 12th Sun Users Group, 1994.

SCHMIDT, D. C. Our research on high-performance and real-time corba. Available in: <http://www.cs.wustl.edu/~schmidt/corba-research-overview.html>. Access on: February, 25th, 2006.

SCHUCKMANN, C.; KIRCHNER, L.; SCHÜMMER, J.; HAAKE, J. M. Designing object-oriented synchronous groupware with COAST. In: CSCW'96, 1996, Cambridge, MA, USA. **Proceedings of CSCW'96**, 1996, p. 30-38.

SETLOCK, L. D.; FUSSELL, S. R.; NEUWIRTH, C. Taking It Out of Context: Collaborating within and across Cultures in Face-to-Face Settings and via Instant Messaging. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 604-613.

SILICON GRAPHICS. Opengl scene graph / opengl++. Technical report, Silicon Graphics, 1997.

SINGHAL, S. K. **Effective Remote Modeling in Large-Scale Distributed Simulation and Visualization Environments**. PhD Thesis, Department of Computer Science, Stanford University, 1996.

SINGHAL, S.; ZYDA. M. **Networked Virtual Environments: Design and Implementation**. New York: ACM Press, 1999.

STEVENS, G.; WULF, V. A New Dimension in Access Control: Studying Maintenance Engineering across Organizational Boundaries. In: CSCW'02, November 16-20, 2002, New Orleans, Louisiana, USA. **Proceedings of CSCW'02**, 2002, p. 196-205.

SWEDISH INSTITUTE OF COMPUTER SCIENCE (SICS): DIVE – A Toolkit for Distributed VR Applications. Available in: <http://www.sics.se/dce/dive>. Access on: March, 2nd, 2006.

SWI-PROLOG. Available in: <http://www.swi-prolog.org>. Access on: March, 3rd, 2006.

TJORA, A. Maintaining Redundancy in the Coordination of Medical Emergencies. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 132-141.

TRAMBEREND, H. Avocado: A Distributed Virtual Reality Framework. In: IEEE VIRTUAL REALITY'99, 1999. **Proceedings of IEEE Virtual Reality'99**, 1999.

TUIKKA, T. Remote Concept Design from An Activity Theory Perspective. In: CSCW'02, November 16-20, 2002, New Orleans, Louisiana, USA. **Proceedings of CSCW'02**, 2002, p. 186-195.

UML – UNIFIED MODELING LANGUAGE. OMG Documentation. Available in: <http://www.omg.org/technology/documents/formal/uml.htm>. Access on: March, 20th, 2006.

WAMIT. Available in: <http://www.wamit.com>. Access on: March, 9th, 2006.

WATSEN, K.; ZYDA, M. Bamboo – A Portable System for Dynamically Extensible, Real-time, Networked Virtual Environments. In: VRAIS'98, 1998. **Proceedings of VRAIS'98**, 1998, p. 260-267.

WEST, A.; HUBBOLD, R. System Challenges for Collaborative Virtual Environments. In: **___Collaborative Virtual Environments**. London: Springer, 2001. p. 44-54.

XCAT. The Indiana University, Extreme Lab implementation of the Common Component Architecture (CCA). Available in: <http://www.extreme.indiana.edu/xcat/>. Access on: February, 25th, 2006.

YANG, Y.; LI, D. Separating Data and Control: Support for Adaptable Consistency Protocols in Collaborative Systems. In: CSCW'04, November 6-10, 2004, Chicago, Illinois, USA. **Proceedings of CSCW'04**, 2004, p. 11-20.

YOUNGMAN, N. DIS Frequently Asked Questions. Available in: <http://www.crg.cs.nott.ac.uk/~dxl/DIS/dis.faq>. Access on: March, 2nd, 2006.

Appendix: Screenshots of the HLA-Compliant Prototype

In this Appendix, we present some screenshots of the execution of our HLA-compliant prototype.

In Figure 24, we can see the collaborative session being initiated. It has seven tiled windows representing respectively, from top to bottom: *(i)* at the left side, the Executive server, the Emergency Pilot T0, the SSTAB operator T1, and the Press P1; *(ii)* at the right side, the software agent S2 executing the WAMIT simulator, the DYNASIM operator T3, and the Decision Maker DM1. It should be noted here that each of these participants could be using different machines, but they are using the same machine at this moment only to show the collaborative session on the same screen. Figures 25 to 45 show, in chronological order, a sequence of screenshots of the prototype execution.

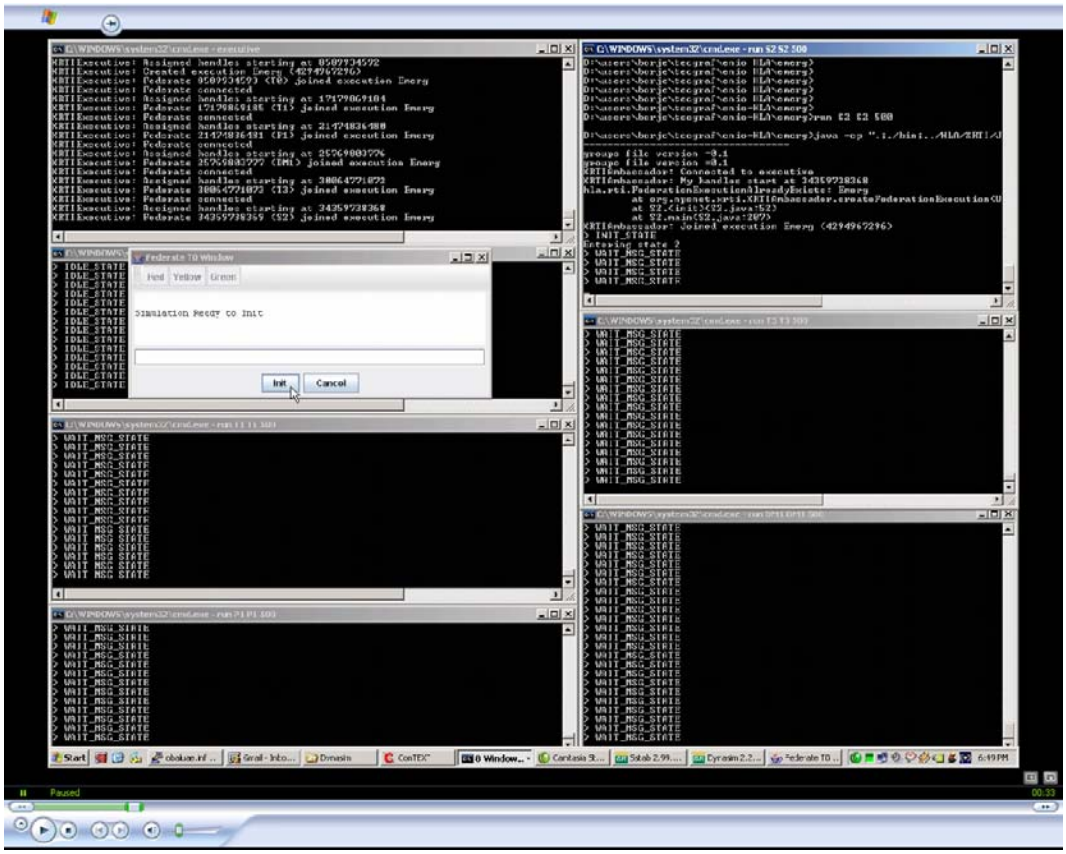

Figure 24 - Collaborative session being initiated

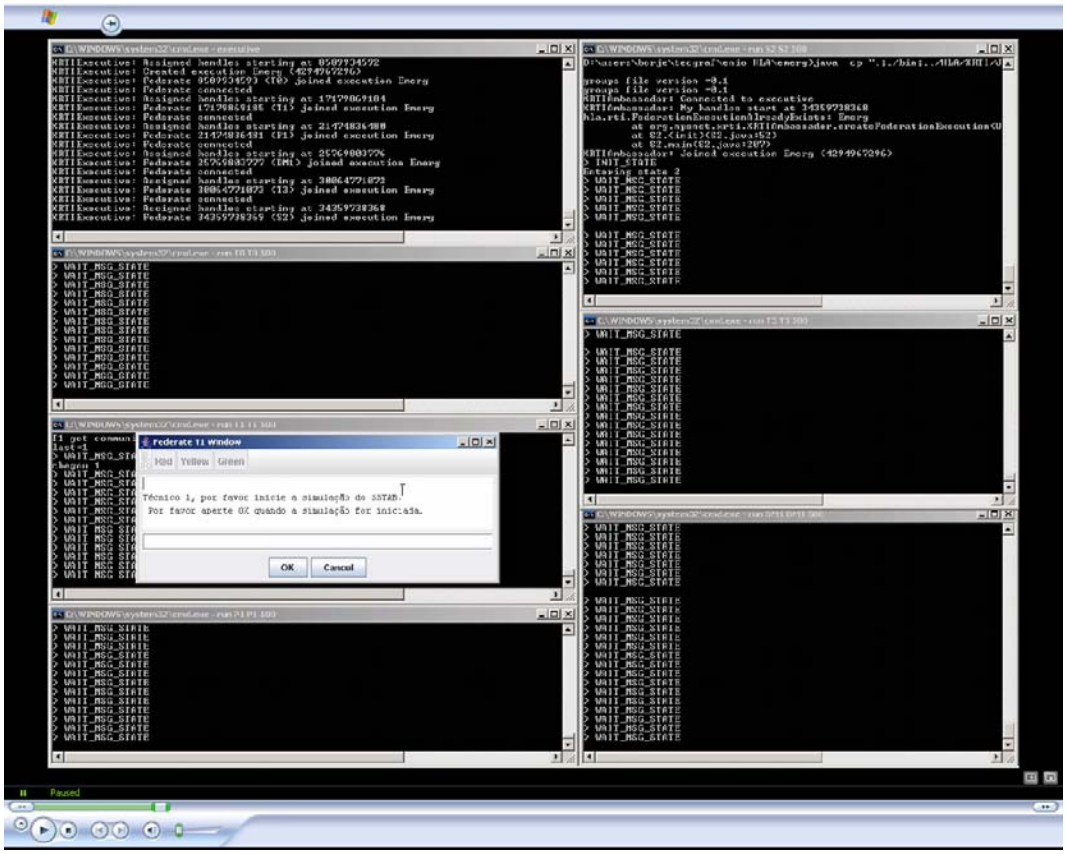

Figure 25 - The SSTAB operator T1 receives a message to initiate the SSTAB simulation

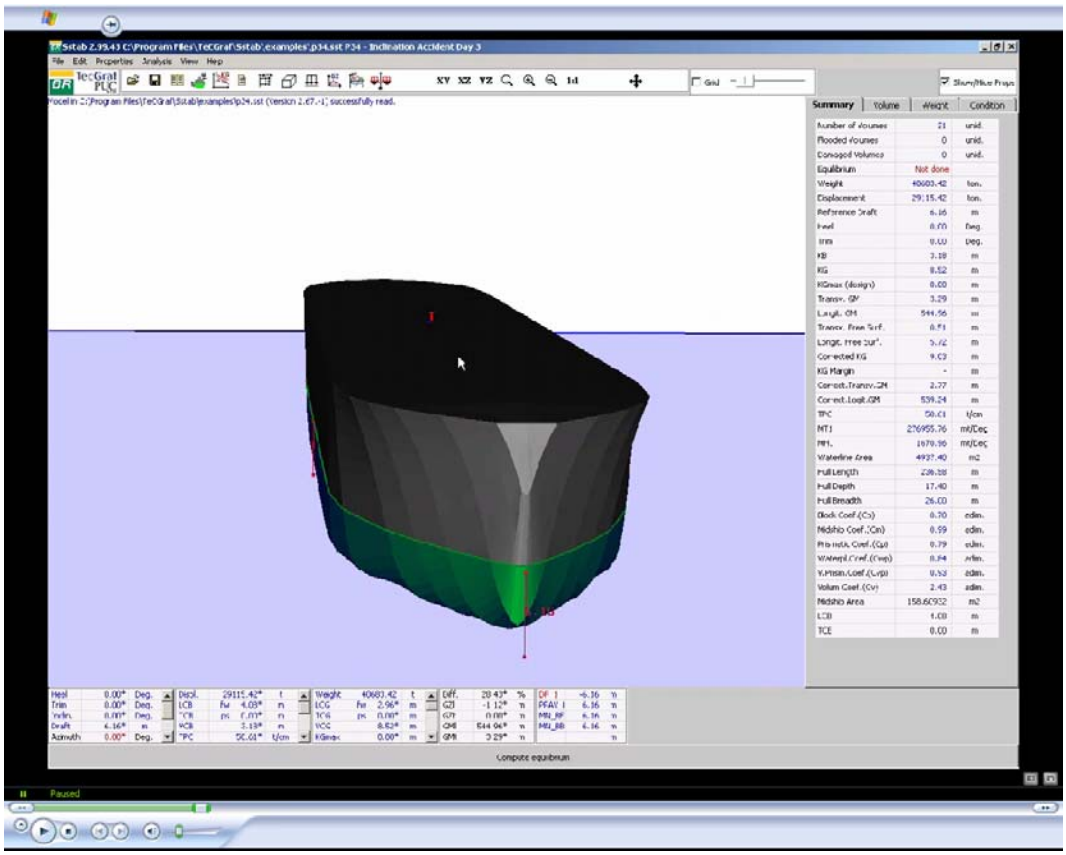

Figure 26 - The SSTAB operator T1 initiates the SSTAB simulation

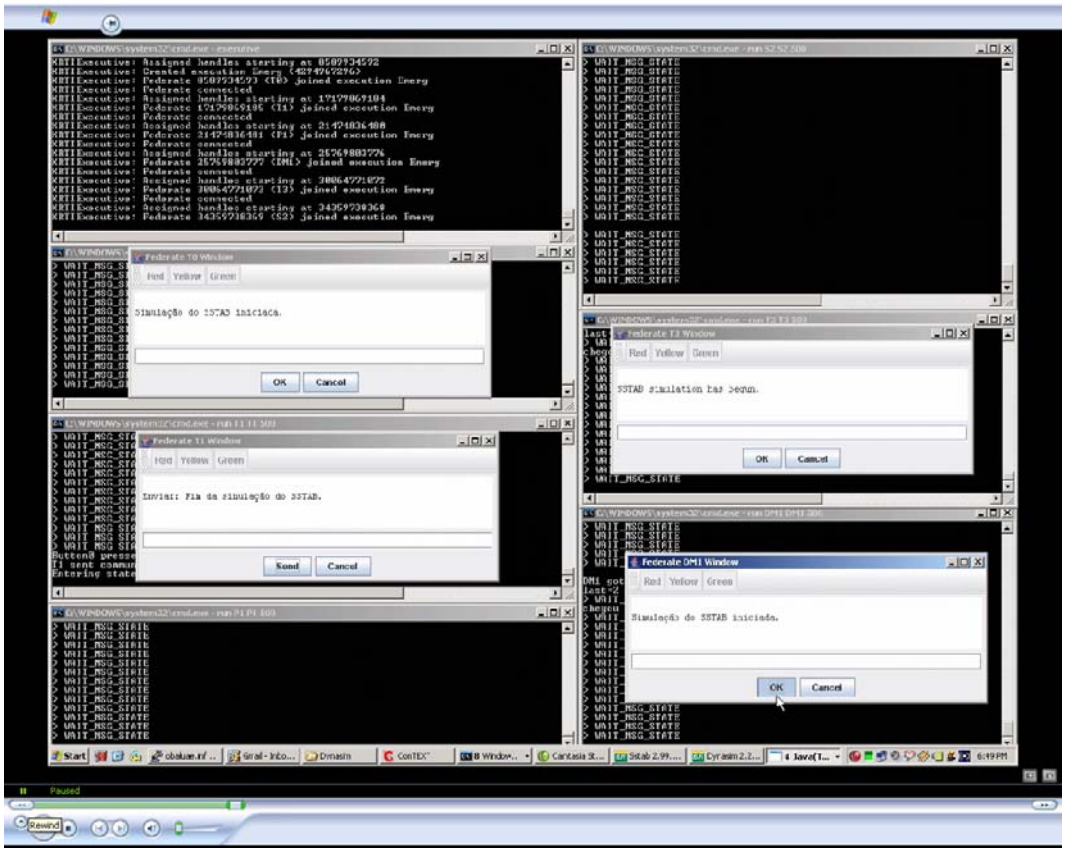

Figure 27 - Federates T0, T3, and DM1 receive a message from T1 telling that he has begun the SSTAB simulation

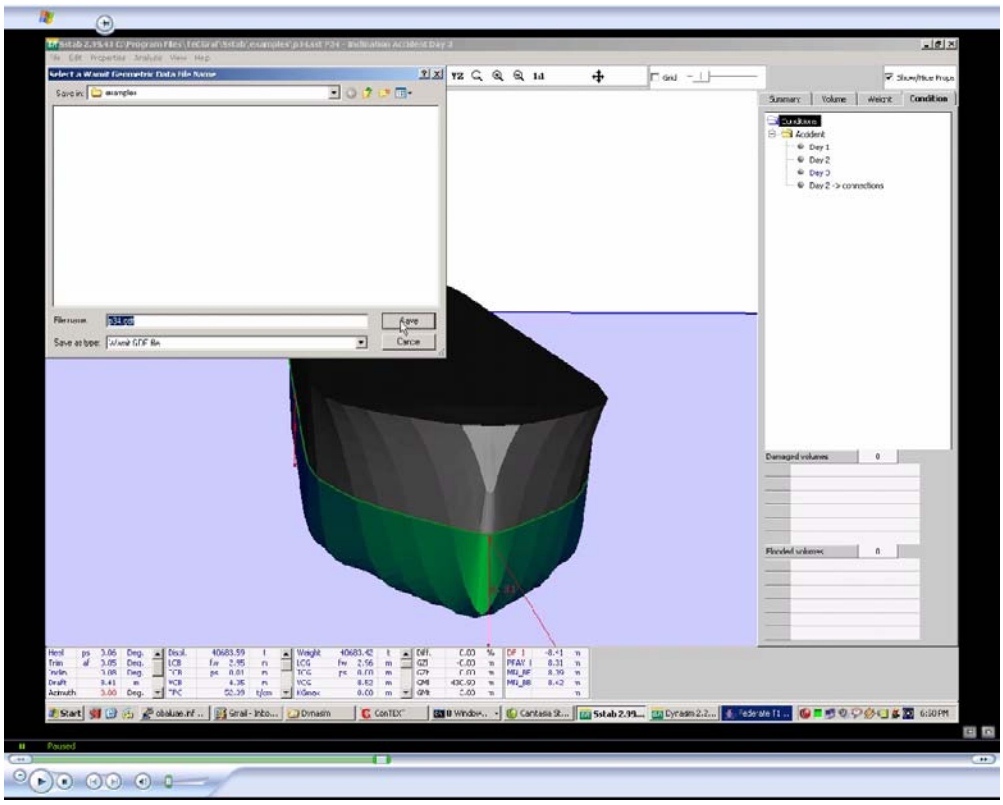

Figure 28 - T1 exports a WAMIT geometric data file and terminates the SSTAB simulation

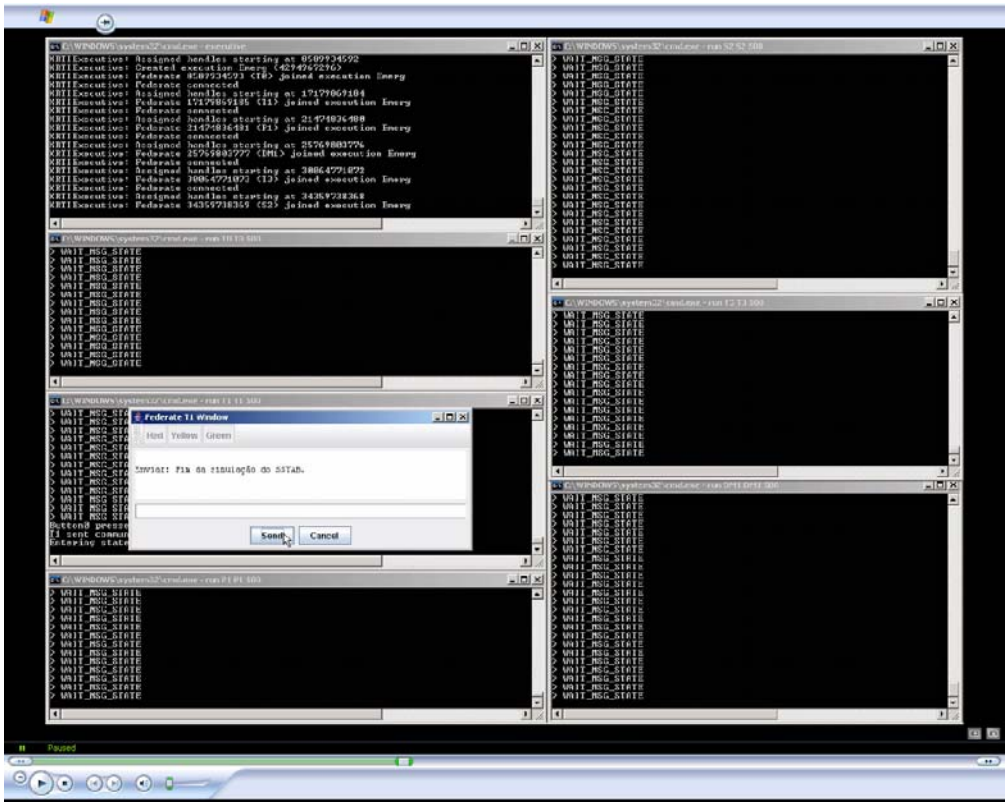

Figure 29 - T1 sends a message informing the end of SSTAB simulation, with S2 automatically activating the WAMIT simulator

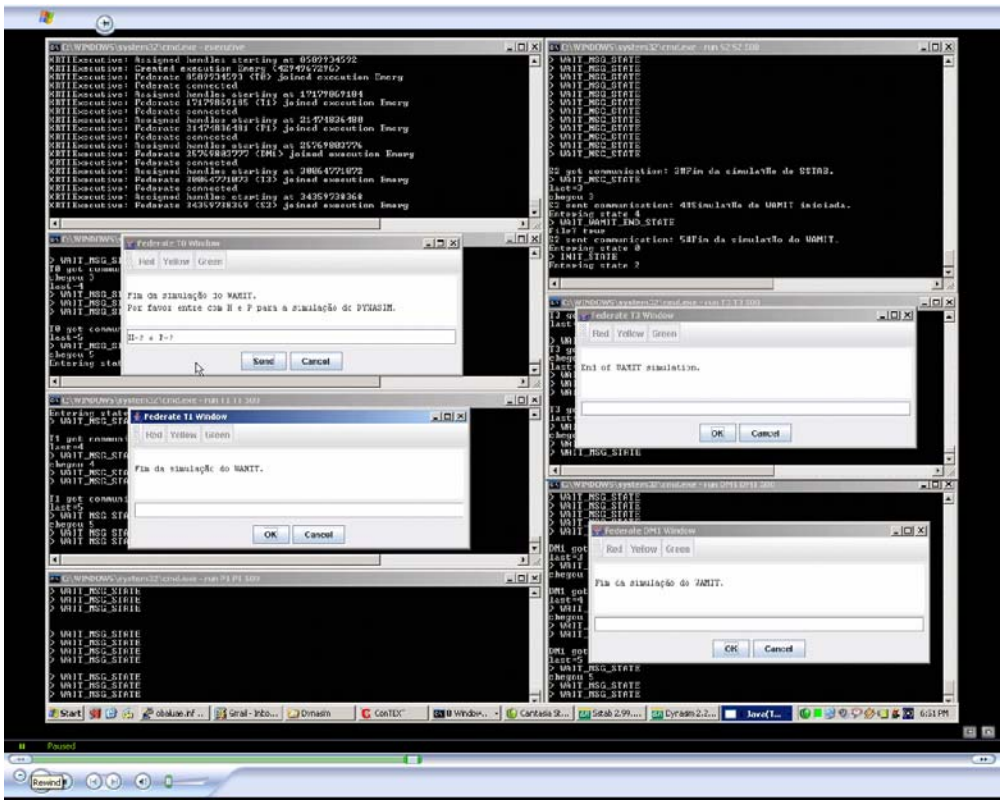

Figure 30 - S2 sends automatically to federates T0, T1, T3, and DM1 a message informing the end of the WAMIT simulation

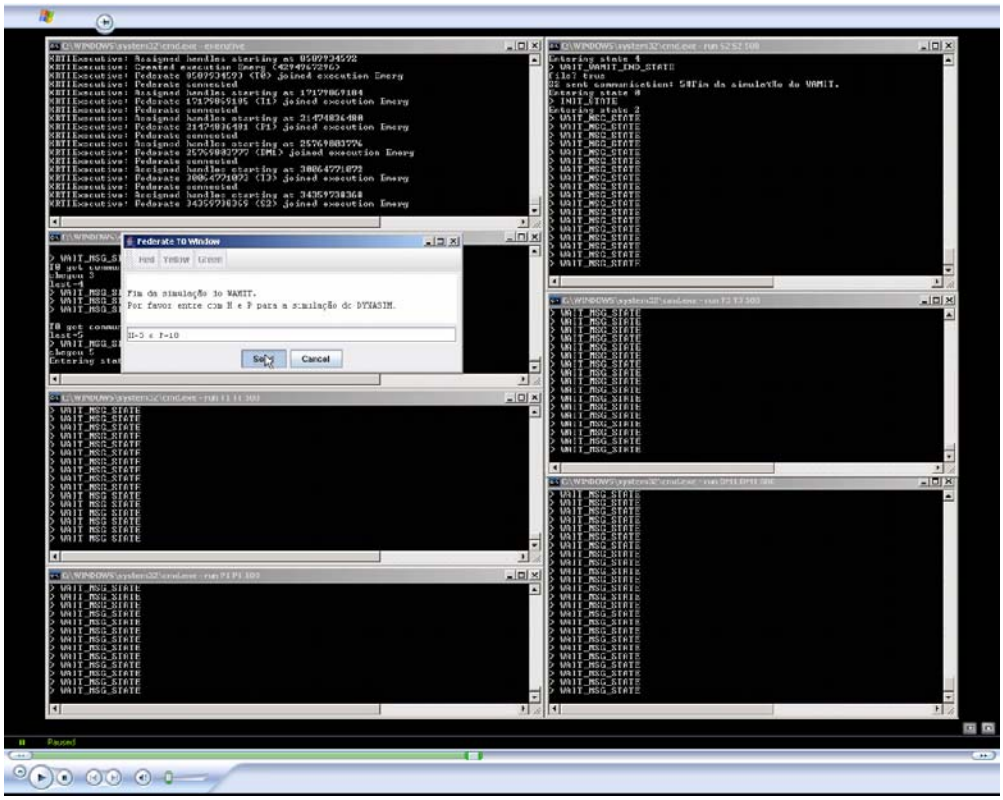

Figure 31 - The Emergency Pilot T0 sends the environmental data (H=5 and P=10) to the DYNASIM operator T3

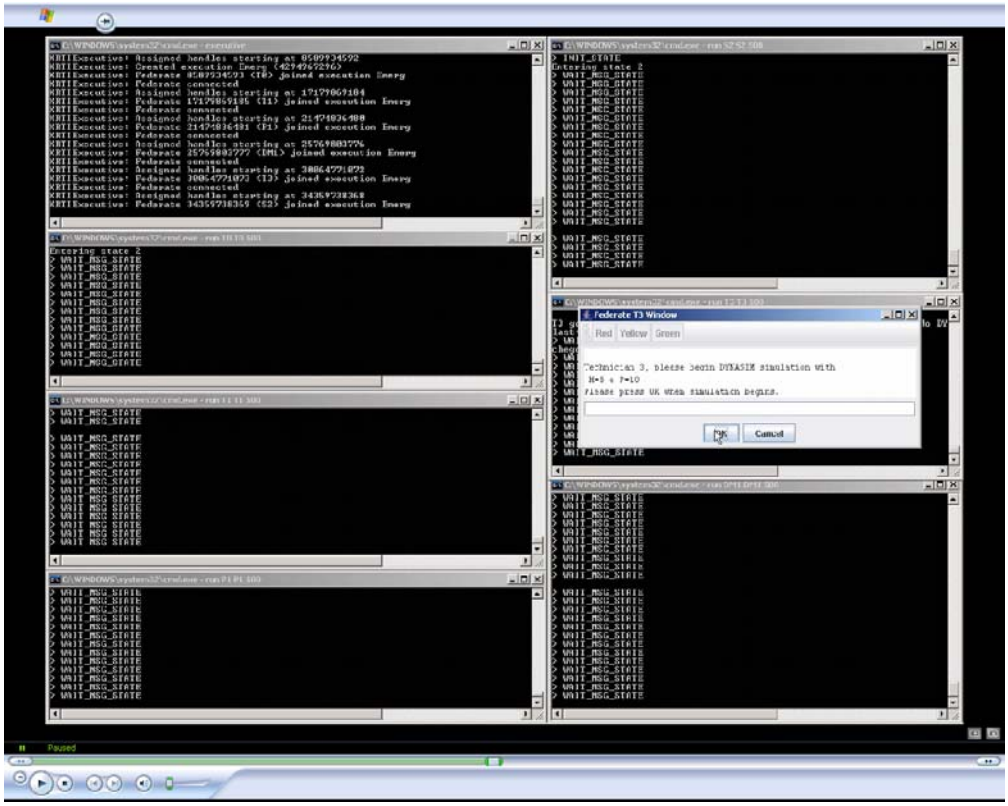

Figure 32 - The DYNASIM operator T3 receives the environmental data (H=5 and P=10) from T0

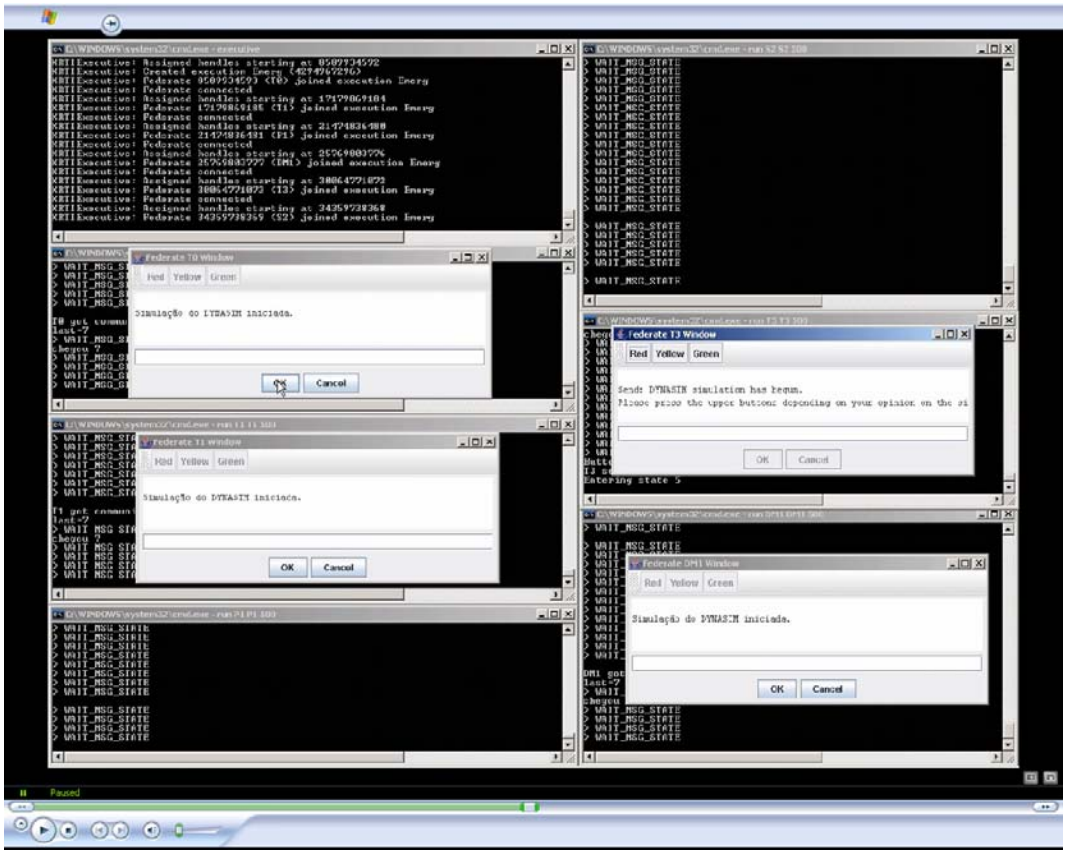

Figure 33 - The DYNASIM operator T3 sends a message informing that he will begin the DYNASIM simulation

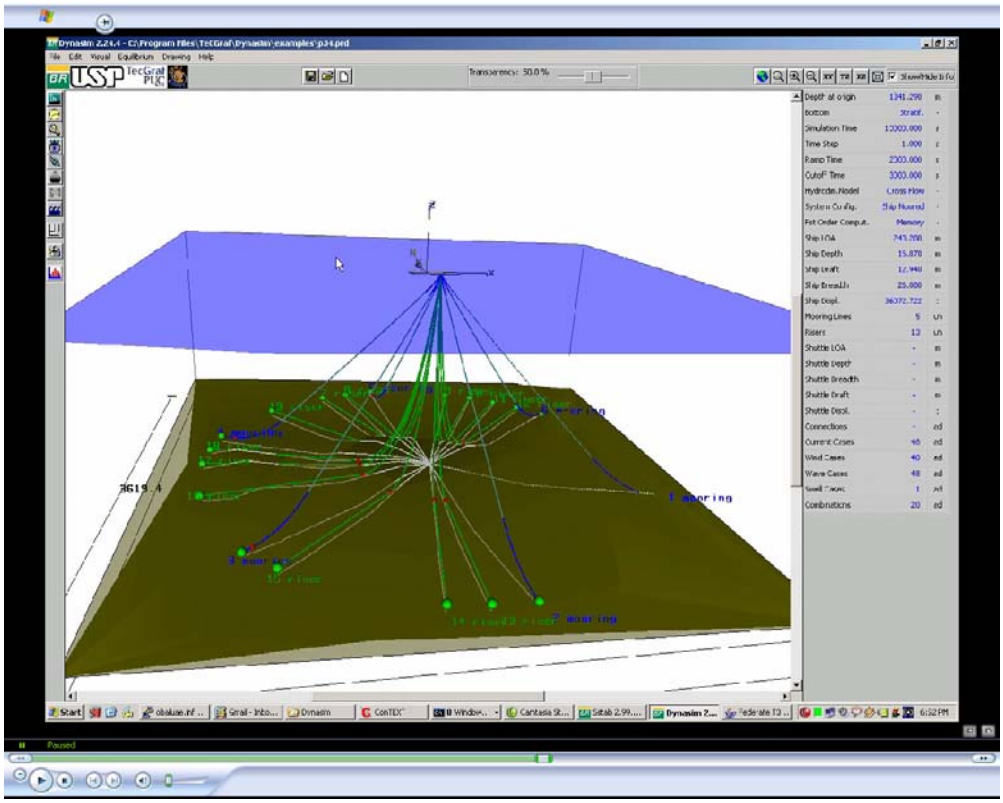

Figure 34 - The DYNASIM simulation is initiated

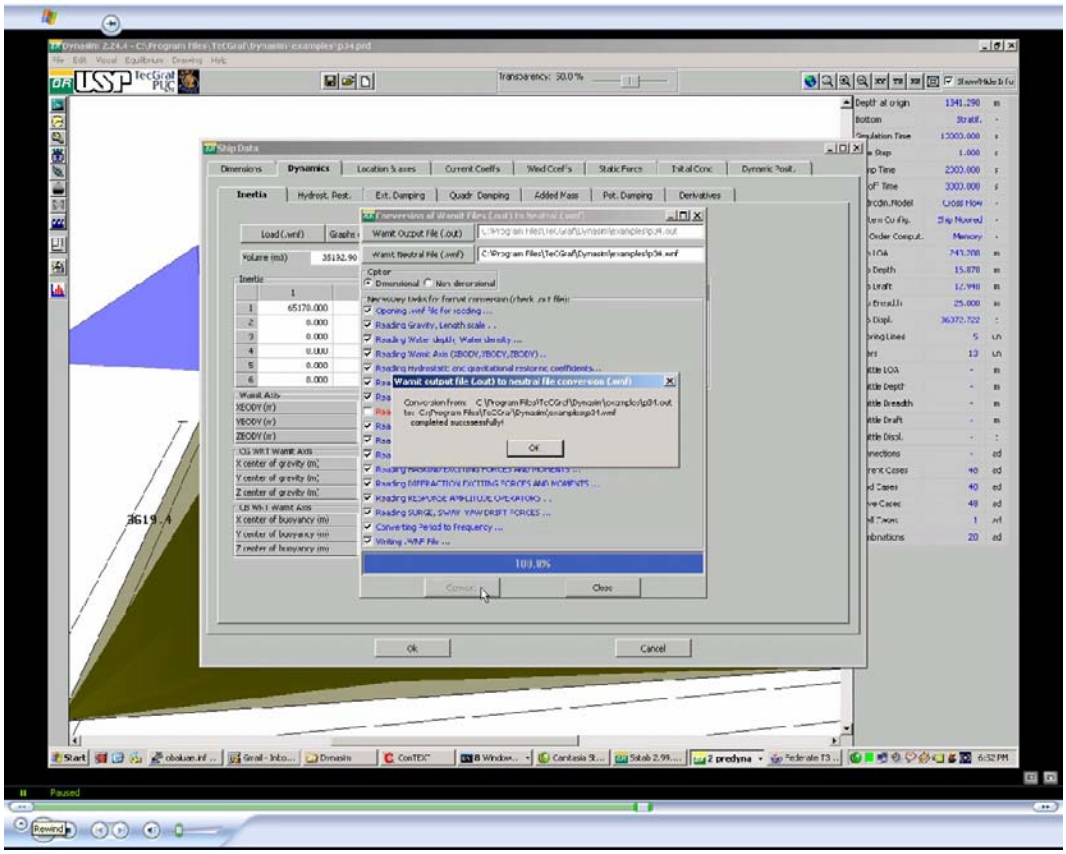

Figure 35 - DYNASIM reads and converts the WAMIT output file to a WAMIT neutral file

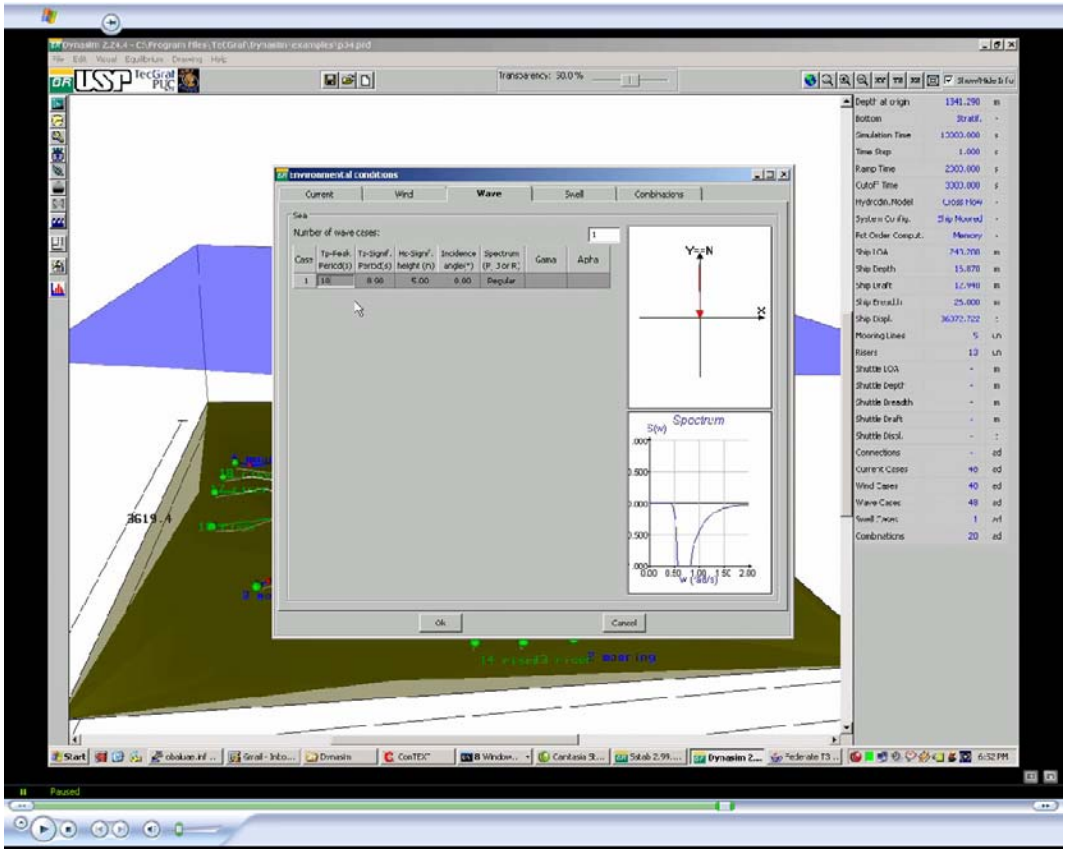

Figure 36 - T3 enters the environmental data (H=5 and P=10) into DYNASIM

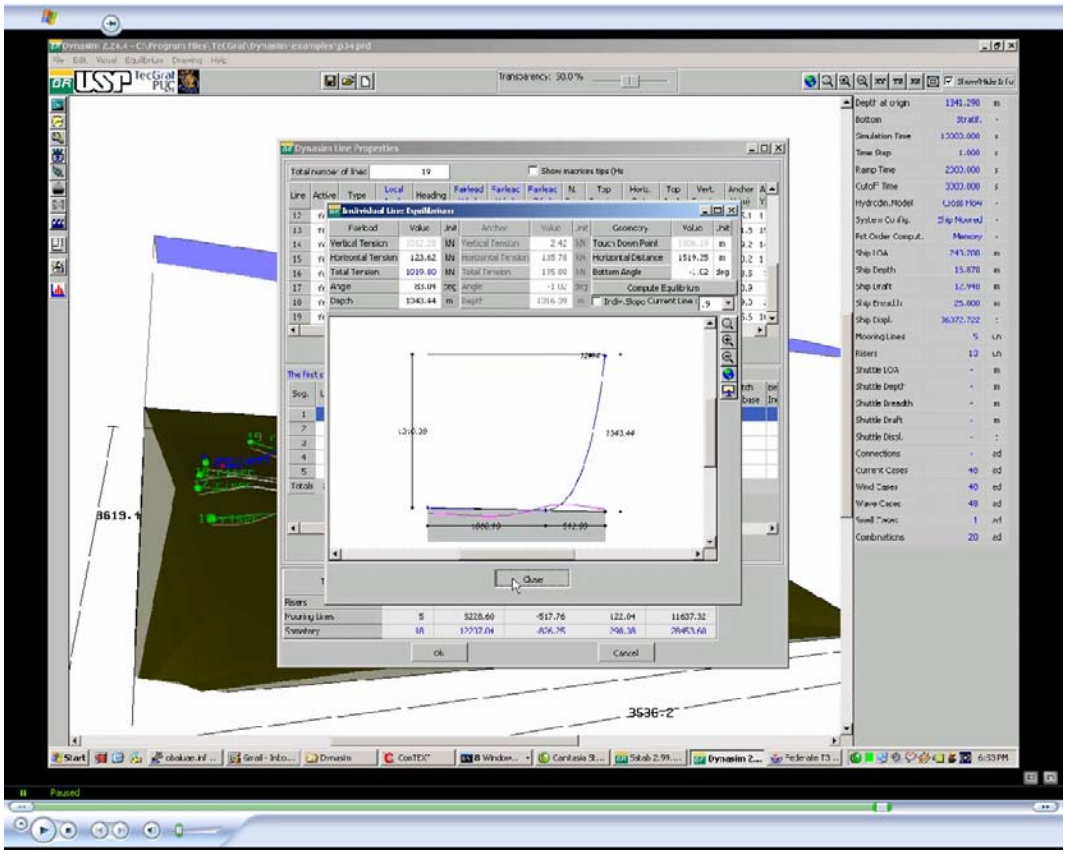

Figure 37 - T3 ends the DYNASIM simulation

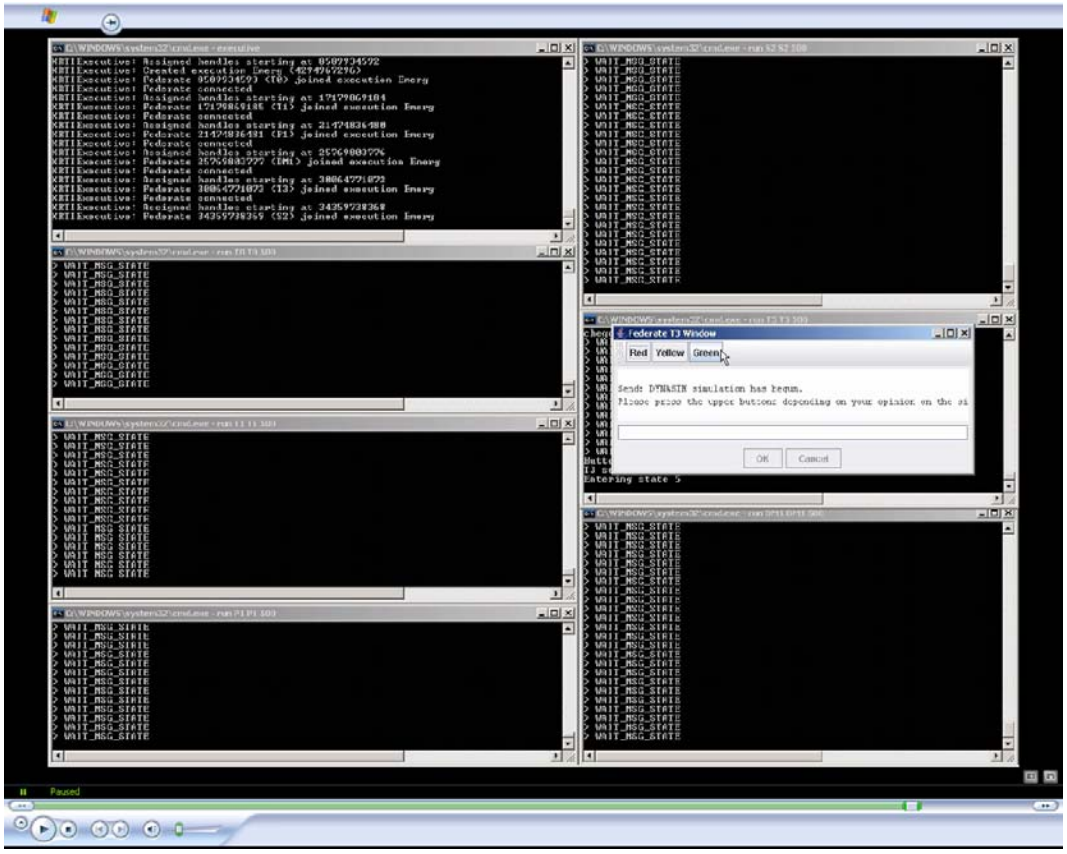

Figure 38 - T3 sends green signal for the DYNASIM simulation

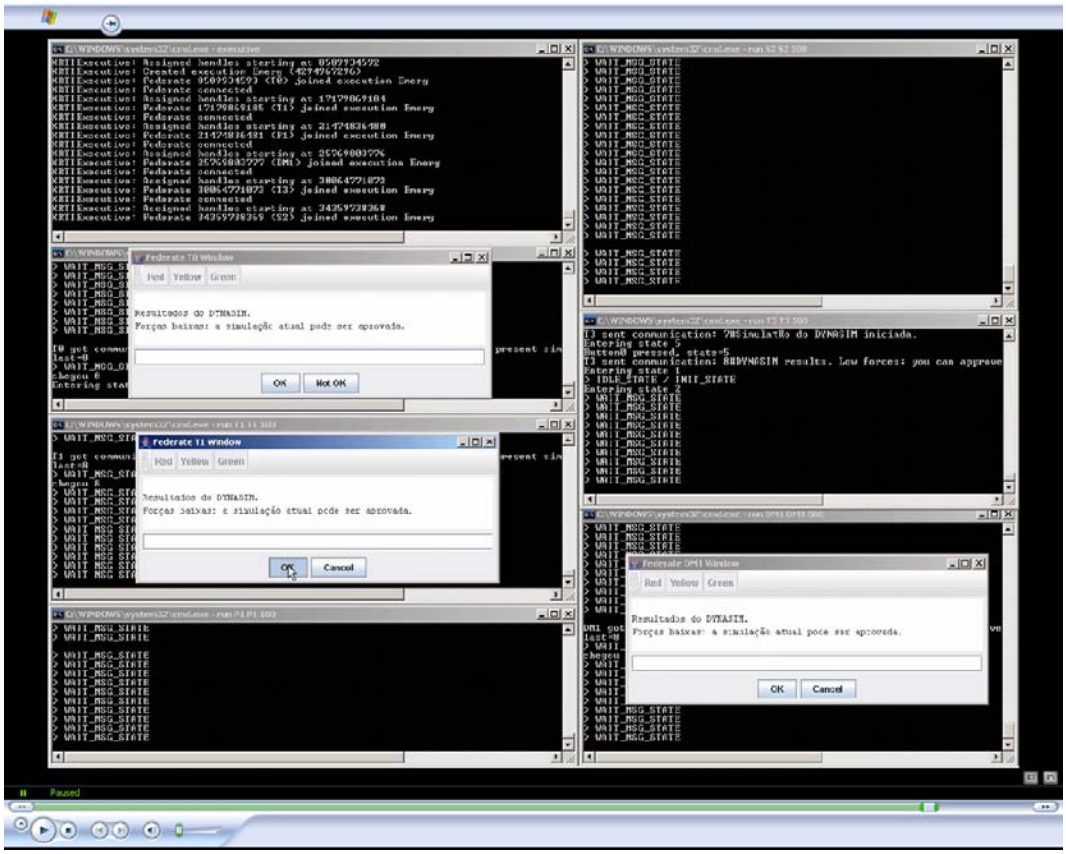

Figure 39 - Federates T0, T1, and DM1 receive the message informing the result of the DYNASIM simulation

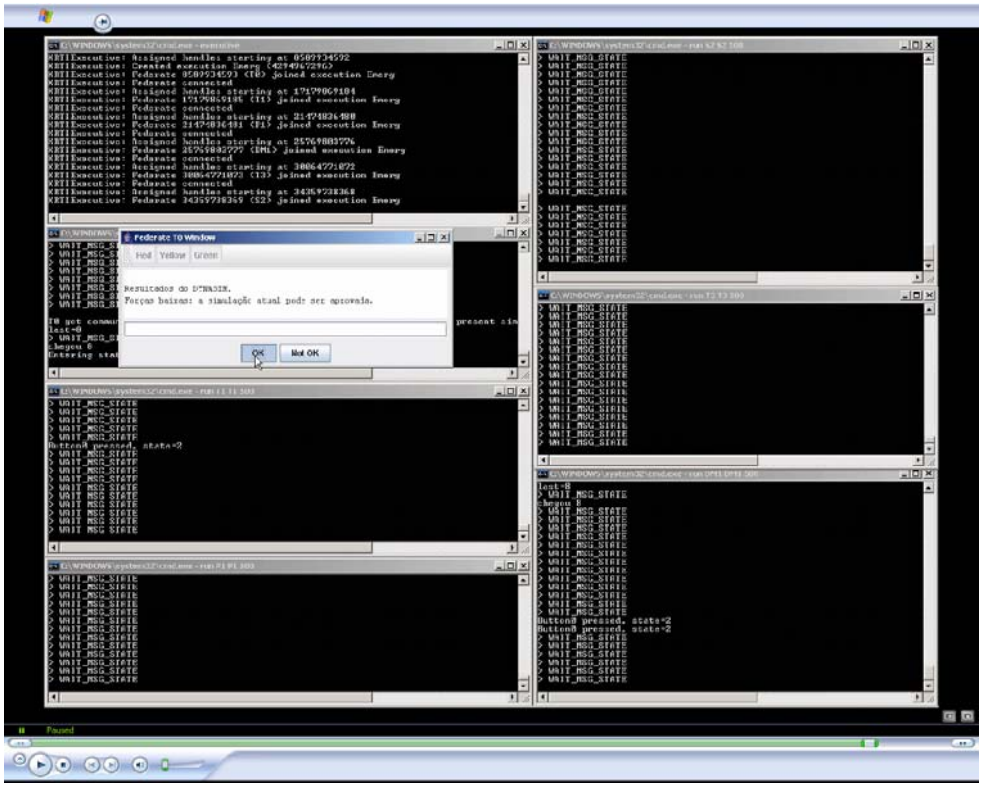

Figure 40 - The Emergency Pilot T0 approves the results of the simulations

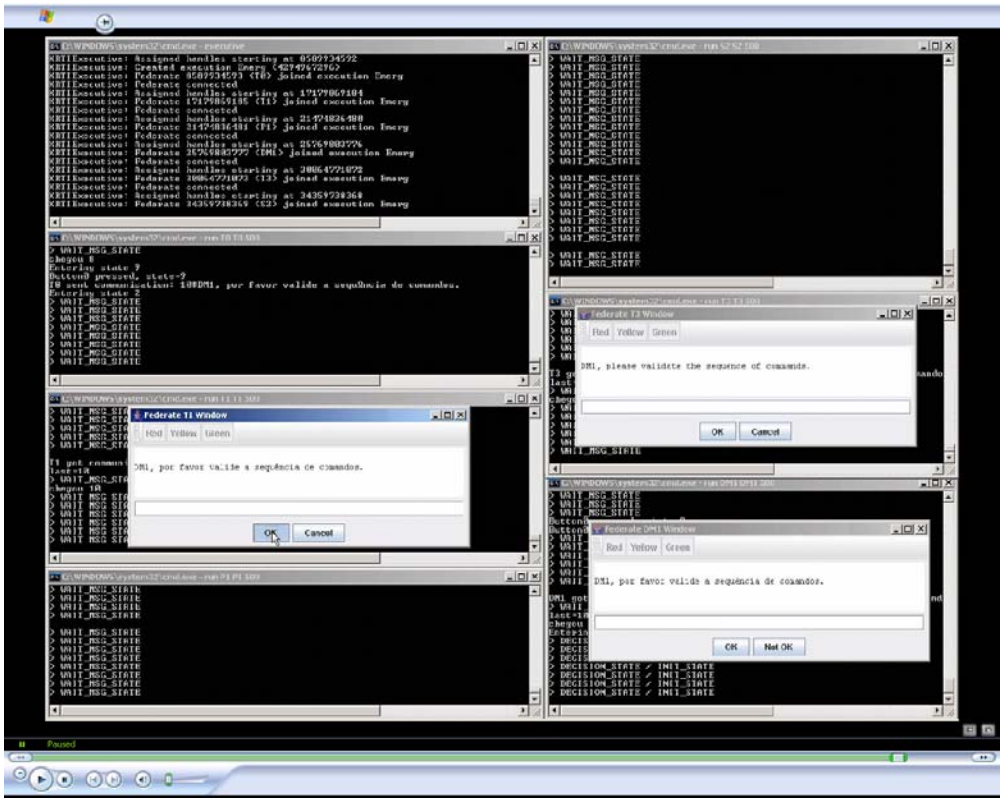

Figure 41 - The Decision Maker DM1 (and federates T1 and T3) receives from T0 a message asking him to validate the sequence of commands to be executed

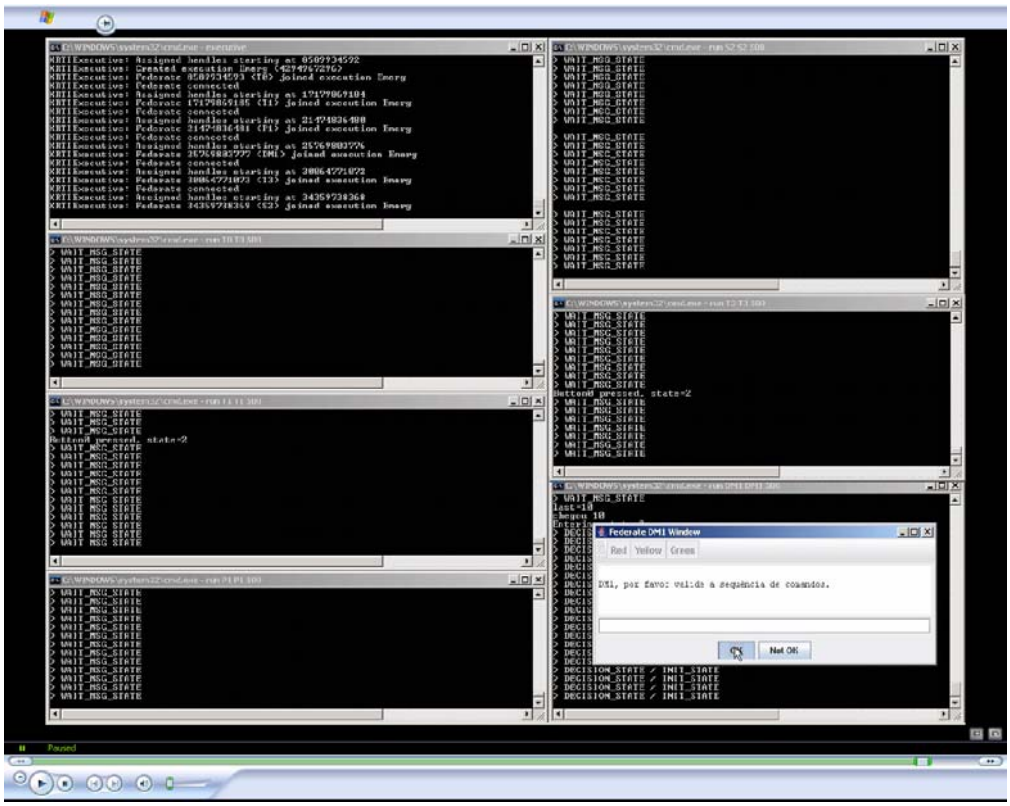

Figure 42 - The Decision Maker DM1 validates the sequence of commands to be executed

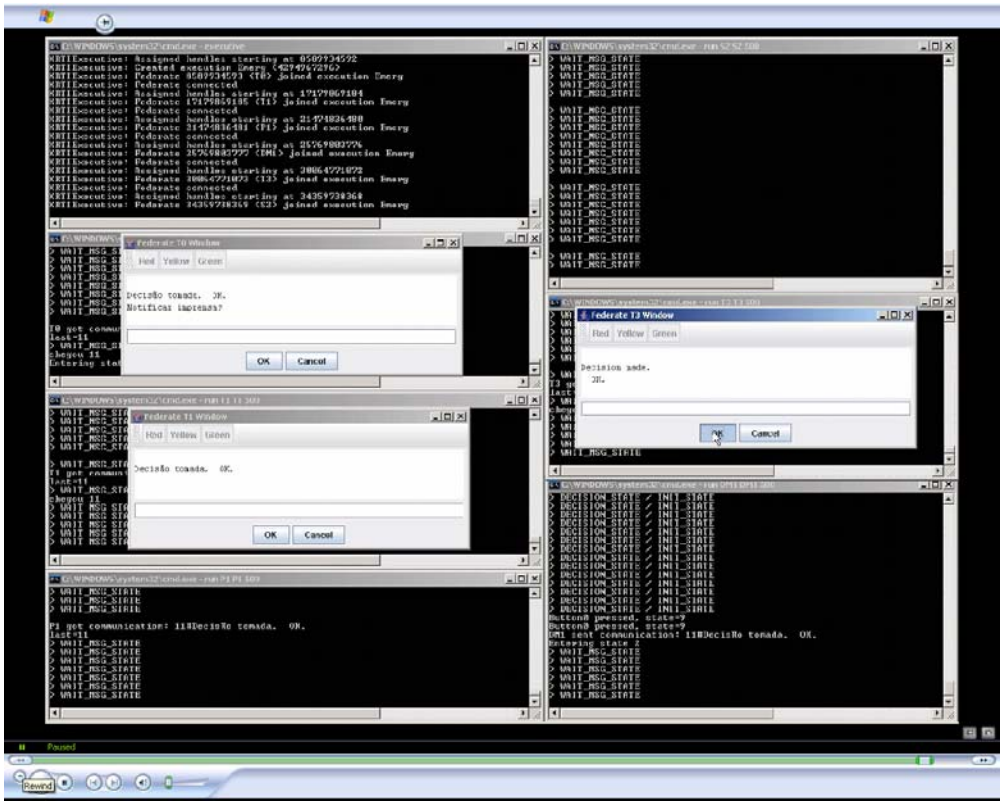

Figure 43 - Federates T0, T1, and T3 receive a message from DM1 telling that he has validated the sequence of commands to be executed

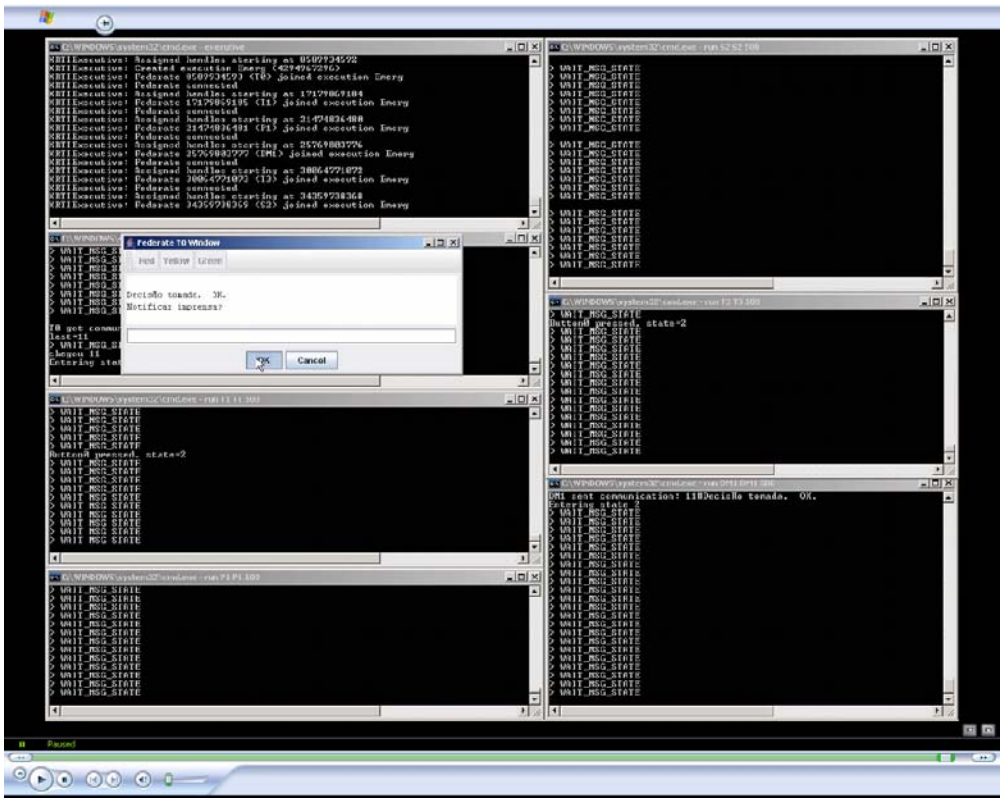

Figure 44 - The Emergency Pilot T0 notifies the Press about the decision made, sending a press release

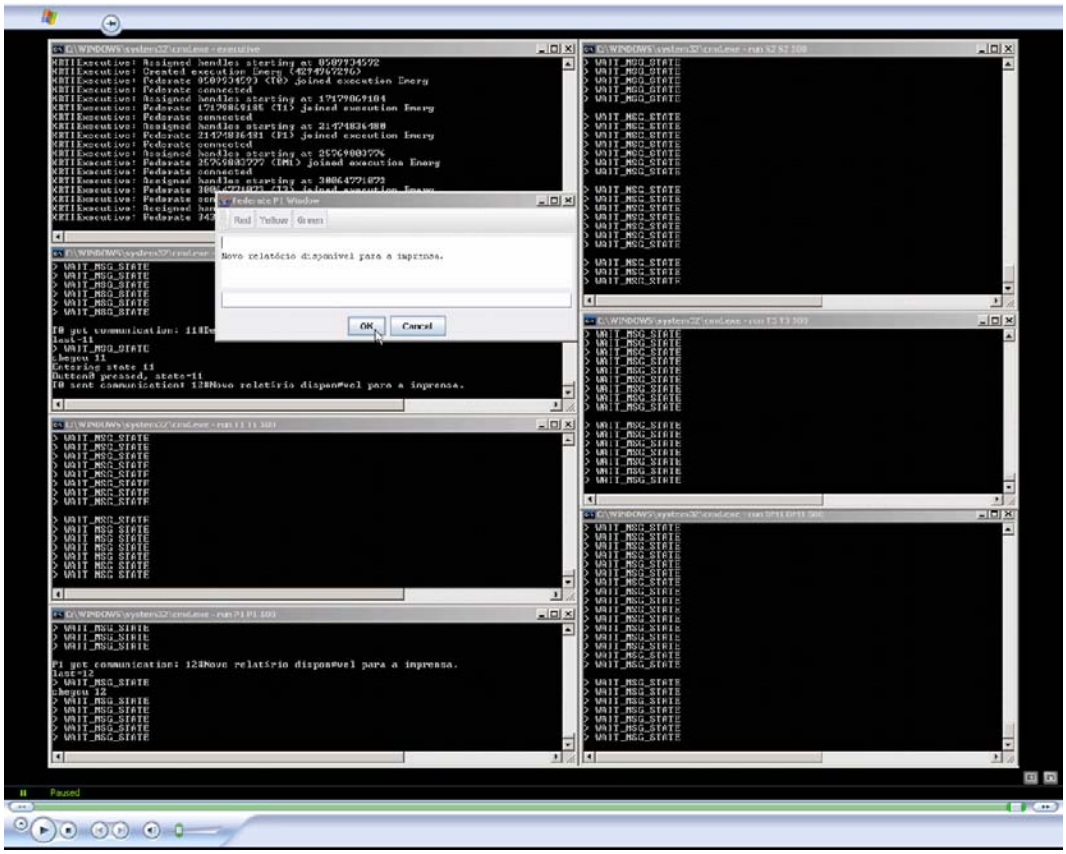

Figure 45 - The Press receives a message from T0 informing that a new press release is available