

PC02-27153

INFOPAE - AN EMERGENCY PLAN DEPLOYMENT SYSTEM

Marcelo Tílio Monteiro de Carvalho

Marco Antonio Casanova

Grupo de Tecnologia em Computação Gráfica / TeCGraf
Departamento de Informática - Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225
Rio de Janeiro, RJ - Brasil - CEP 22453-900

Flávio Torres

Petrobras - Petróleo Brasileiro S.A.
Av. República do Chile, 65
Rio de Janeiro, RJ - Brasil - CEP 20035-900

Ângelo Santos

ABSTRACT

InfoPAE is an automated system designed to improve the response to emergency situations. The system offers sophisticated action plans, easy access to vital information and tight control over the resources allocated to face an emergency. The system is currently in use at Petrobras and its subsidiaries, incorporating the company's technical experience. The system is applicable to pipelines, oils terminals, oil refineries and offshore installations, and it also proved to be a valuable training tool.

1. INTRODUCTION

InfoPAE is an automated system designed to improve the response to emergency situations. The system offers sophisticated action plans, easy access to vital information and tight control over the resources allocated to face an emergency. The system is applicable to pipelines, oils terminals, oil refineries and offshore installations, and it also proved to be a valuable training tool.

InfoPAE works with emergency plans, which are structured collections of actions, coupled with information stored in geographical as well as conventional databases. During an emergency, the team follows a previously stored plan, backed up by its ancillary information. The team registers the actions taken and documents eventual difficulties. Later on, upper level management may use the system to

generate reports that are useful to detect eventual problems with the plan or to assess the performance of the team.

Development of the system begun in 1998 and the first prototype was installed in 1999. In 2000, InfoPAE was in operation at Transpetro and the exploration and production areas of Petrobras. In 2001, the use of the system was expanded to oil refineries and natural gas installations. Current plans include implementing InfoPAE at nearly 80 installations throughout 2002.

A preliminary version of the plan specification language can be found in [1]. The language is similar to other workflow specification languages, such as that adopted in the WIDE Project [2]. FRIEND [3], INCA [4] and MokSAF [5] are examples of emergency management systems. In particular, the MokSAF system supports route planning by combining AI techniques with GIS. Other examples of multi-agent systems with internal planning components are RETSINA [6] and HIPaP [7]. A survey of cooperative multi-agent systems appears in [8].

The paper is organized as follows. Section 2 outlines the architecture of the system. Section 3 describes the system's database. Section 4 introduces the plan editing module. Section 5 describes the plan execution module. Section 6 discusses the visualization module. Finally, Section 7 contains the conclusions and suggests possible extensions to the system.

2. ARCHITECTURE OF THE SYSTEM

The emergency plan deployment system has four basic components: DataPAE, EditaPAE, OperaPAE and VistaPAE.

DataPAE is a database storing data relevant to the emergency plans of interest.

EditaPAE incorporates a plan browser to search for plans in the database, an editor to create new plans and to relate them to persistent data, and a data management interface to maintain the database.

OperaPAE controls the execution of a plan during an emergency and helps access its associated data. It interacts with the human agents to register the progress of actions contained in the plan, who took the action and any observation pertaining to the action. It also helps the upper management staff keep track of the current scenario: the current assessment of the situation (as reported by the emergency teams), what has been accomplished, what remains to be done, etc.

VistaPAE offers a variety of tools to browse the database, including some typical of geographical information systems.

The current implementation of InfoPAE supports both stand-alone and distributed operation via data replication. The system offers automated communication facilities through pagers, e-mail and telephone.

The final implementation will feature a fully distributed architecture where mobile devices communicate with fixed sites to monitor plan execution. Mobile devices will run specialized versions of DataPAE, OperaPAE and VistaPAE, which opens an array of possibilities, including voice output synthesis and other less conventional data rendering techniques.

3. STORING PERSISTENT OBJECTS IN DATAPAE

DataPAE persistently stores data that are valuable to the deployment of emergency plans and that must be readily available during plan execution. This includes conventional and geographical data, such as internal and external contacts that must be aware of the emergencies, descriptions of the resources that are necessary to carry out the actions, and descriptions of the emergency teams. The database also contains facility blueprints, simulation results, and sensitivity maps that are relevant to the execution of the emergency plans.

DataPAE stores the plans themselves, predefined generic actions and plan frameworks. A generic action may, for example, be associated with a type of resource that is translated into an actual resource when the action is executed, depending on the location of the emergency. A plan framework, as the name indicates, is a framework that describes a generic plan scheme that can be instantiated to generate new plans. The database also persistently maintains the current state of a plan execution, which is an internal object of OperaPAE.

Finally, DataPAE contains metadata that help define generic actions and plan frameworks. This includes action types, internal and external contacts types and resource types.

4. EDITING PLANS WITH EDITAPAE

The EditaPAE module helps the user create emergency plans and manage the objects stored in the underlying database.

Intuitively, an emergency plan congregates:

1. A structured collection of actions to be followed during the emergency.
2. A description of the teams required to execute the actions.
3. A list of the resources, documents and information in general that will help carry out the actions.

In its simplest conception, a plan is just an ordered list of actions. However, in general, a plan will indicate which groups of actions can be executed concurrently, and which groups must be executed serially.

A plan also depends on a characterization of the emergency conditions, such as the type of product and the estimated amount that spilt and environmental conditions (direction of the wind, tide conditions, etc...). Therefore, the plan will contain groups of actions that apply only under certain conditions.

When an emergency occurs, the designated teams must follow the actions previously defined and secure the necessary resources, guided by the documents associated with the actions. The teams must evaluate the current conditions and select the actions that apply under the conditions detected.

In what follows, we will adopt the term "user" to the designate the person who is interacting with the tool, and "team" to designate any group of people that actually executes a sub-plan of the emergency plan.

More precisely, an *emergency plan* contains *actions*, *questions* and *tests*, structured as *sequential* or *parallel groups*. Figure 1 in the Appendix illustrates a plan.

An action, question or test occurring in a plan is called an *elementary sub-plan* of the plan; a sequential or parallel group occurring in a plan is called a (*non-elementary*) *sub-plan* of the plan.

An *action* describes a task that an emergency team must follow. An action may be associated with data, stored in the database, which will be required to carry out the action.

When the system executes the action, it simply displays a text describing the action and offers ways to select the data associated with the action.

Examples of actions are (see Figure 1 in the Appendix):

Action 1:

Description: requests the name, address and telephone of the person that communicates the incident.

Action 2:

Description: sends an emergency team or any internal resource to the location of the incident.

Information: a map of the incident location

The plan editor displays each action on a separate line, preceded by the symbol "O".

A *question* describes a condition that the emergency teams must settle. Associated with the question, there is a text describing the condition, a list of alternative answers and a parameter. Associated with each alternative answer, there is a sub-plan.

When the system processes the question, it displays the text describing the condition, followed by the list of alternative answers, one of which the user must select. The system sets the parameter value to be the alternative answer selected. Processing continues with the sub-plan associated with the answer selected.

Examples of questions are (see Figure 1 in the Appendix):

Question 1:

Text: How was the incident communicated?
Alternatives: Internal Call, External Call

Question 2:

Text: What is the type of the incident?
Alternatives: Spill, Fire, Gas leakage

The plan editor represents the text describing the condition on a separate line, preceded by the question mark "?", and lists the alternative answers also on separate lines, immediately below the condition. If an alternative answer is indeed associated with a sub-plan, its actions are shown indented to the right, just below the alternative answer. As an example, observe how Question 1 above is formatted at the top of Figure 1 of the Appendix.

A *test* emulates a question that was previously defined in the plan, but it does not require user intervention. Thus, a test contains a text, which is always the name of the parameter of the emulated question, and a list of alternative answers, which must be identical to that of the emulated question. The plan editor represents a test in a way very similar to questions, except that the parameter name is preceded by a "T".

Associated with each alternative, there is also a sub-plan, which need not be identical to that of the equivalent alternative of the emulated question.

When processing a test, the system recovers the alternative the user chosen when confronted with the question for the last time (which is the value of the parameter associated with the question). Processing continues with the sub-plan associated with this alternative.

Elementary components can be structured into more complex plans in two ways.

A finite list E_1, E_2, \dots, E_m of sub-plans can be structured as a *sequential group* to indicate that sub-plan E_k can only be executed after E_{k-1} finishes, for each k in $(1, m]$. Alternatively, a finite list E_1, E_2, \dots, E_m of sub-plans can be structured as a

parallel group to indicate that the sub-plans can be concurrently executed, without imposing any specific order.

A group can be labeled with a short text to improve legibility of the plan.

Just as for actions, a group may be associated with data in the database. By transitivity, all its sub-plans also become associated with this data.

Finally, a group can be flagged as a *scenario*, although no semantics is associated with this concept in the current implementation.

The plan editor represents a sequential group by placing the symbol ";" on a separate line, followed by the label assigned to the group, if any, and by listing the sub-plans in subsequent lines, indented to the right. The representation of a parallel group is entirely similar, except that the symbol "/" is used instead.

As examples, observe in Figure 1 how the editor represents the sequential group associated with the alternative answer "External Call" and the parallel group labeled (in Portuguese) "Informar a ocorrência ao Coordenador Local".

Finally, the EditaPAE module also offers the user functions to manage the objects in the database, whose description we omit in this paper for the sake of brevity (see [9] for the details).

5. EXECUTING PLANS WITH OPERAPAE

We will again adopt the term "user" to designate the person who is interacting with the tool and "team" to designate any group of people that actually executes a sub-plan of the emergency plan.

When an emergency occurs, the user must invoke the OperaPAE module and choose the appropriate plan. OperaPAE will then guide the teams throughout plan execution, tracing the actions taken, and collecting additional data about plan execution. Figure 2 at the Appendix illustrates the main window of the OperaPAE module.

Once the execution of a plan starts, OperaPAE maintains the list of *current* actions, questions and tests (that is, of those elementary sub-plans) that are ready to execute. Note that this list may contain more than one element if the plan contains parallel groups.

For each current action, the user may access any data element associated with the action. This includes both the data directly associated with the action and the data associated with a group that contains the action.

The user may choose one of the current actions and flag it as executed, thereby transforming into current any elementary sub-plan that depends on it.

The user may also defer the execution of a current action, if the action was flagged as deferrable when the plan was defined. This also transforms into current any elementary sub-plan that depends on the deferred action.

When the user selects a current question, OperaPAE prompts the user to choose one of the alternatives associated with the question. The module stores the alternative in the question's parameter and then it proceeds to analyze the sub-plan associated with the alternative chosen.

OperaPAE automatically executes a test based on the value currently associated with its parameter, that is, the module proceeds to analyze the sub-plan associated with the alternative stored in the sub-plan.

The user also has the option to undo the last action executed. He may also, at any time, change the alternative chosen for a question. In both cases, the module recomputes the list of current elementary sub-plans.

If necessary, the user may register an unpredicted action, which becomes part of the current execution of the plan, but not of the plan itself.

Finally, the user may request a report of the actions already executed, based on the trace OperaPAE maintains. This facility allows post-emergency evaluation of the plan, which may help detect flaws in the plan, or inadequate team behavior.

6. BROWSING DATA WITH VISTAPAE

The VistaPAE module allows the user to browse conventional as well as geographical data related to the location where an emergency occurred. It offers a variety of tools, typical of a geographical information system. We briefly describe in this section just some of the tools.

When the user starts VistaPAE, he is offered the window shown in Figure 3. VistaPAE offers three search interfaces:

- a conventional search interface just to browse the database for conventional data describing, for example, material resources and internal or external contacts;
- a search interface to retrieve geo-referenced data from the database, filtered by type or by name, and to display them using the appropriate tool;
- a point-and-click interface that helps the user retrieve data based on their location over a map or geo-referenced image (see Figure 4).

Finally, VistaPAE features a measurement tool to compute the shortest distance between two objects displayed on a map or a geo-referenced image.

7. CONCLUSIONS

This paper outlined the basic concepts and functionality of InfoPAE, an automated system designed to improve the response to emergency situations. The system allows users to execute comprehensive emergency plans, hyperlinked to conventional as well as geographical data.

We are in the process of extending the plan modeling language to account for the concept of *compensatory tasks* to undo real world activities previously carried out by the human agent. We also plan to introduce the concept of *bypassing* actions that would let the user jump ahead in the plan effectively bypassing a set of actions.

We also plan to extend the system to accommodate simplified plans that would account for incidents that do not require complex procedures and yet must be monitored for various reasons.

ACKNOWLEDGMENTS

We wish to thank the InfoPAE team, especially Eduardo Thadeu Leite Corseuil, Juliana Freire, Fabio Dias and Tatiana Coelho for their many contributions to the ideas expressed in this paper.

REFERENCES

- [1] Carvalho, M.T., Freire, J., and Casanova, M.A., 2001, "The Architecture of an Emergency Plan Deployment System", Proceedings, III Workshop Brasileiro de GeoInformática, Instituto Militar de Engenharia, Rio de Janeiro, Brazil.
- [2] Chan, D., Vonk, J., Sánchez, G., Grefen, P., and Apers, P., 1998, "A Conceptual Workflow Specification Language", Proceedings, ACM Symposium on Applied Computing, Atlanta, USA.
- [3] Bruegge, B., O'Toole, K. and Rothenberger, D., 1994, "Design Considerations for an Accident Management System", Proceedings, Conference on Cooperative Information Systems, pp. 90-100.
- [4] Iba, W., and Gervasio, M., 1999, "Adapting to User Preferences in Crisis Response", Proceedings, Intelligent User Interfaces, pp. 87-90.
- [5] Lenox, T., Payne, T., Hahn, S., Lewis, M., and Sycara, K., 1999, "MokSAF: How should we support teamwork in human-agent teams?", Technical Report CMU-RI-TR-99-32, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
- [6] Paolucci, M., Kalp, D., Pannu, A., Shehory, O., and Sycara, K., 1999, "A Planning Component for RETSINA Agents", Agent Theories, Architectures, and Languages.
- [7] Paolucci, M., Shehory, O., and Sycara, K., 2000, "Interleaving planning and execution in a multiagent team planning environment", Technical Report CMU-RI-TR-00-01, Robotics Institute, Carnegie Mellon Univ., Pittsburgh, PA.
- [8] Lesser, V., 1999, "Cooperative Multiagent Systems: A Personal View of the State of the Art", Knowledge and Data Engineering, **11**:1, pp.133-142.
- [9] TecGraf, 2002, "Sistema de Informação para Apoio a Planos de Ação de Emergência. Manual de Operação e Edição", Laboratório de Computação Gráfica - TecGraf, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil.

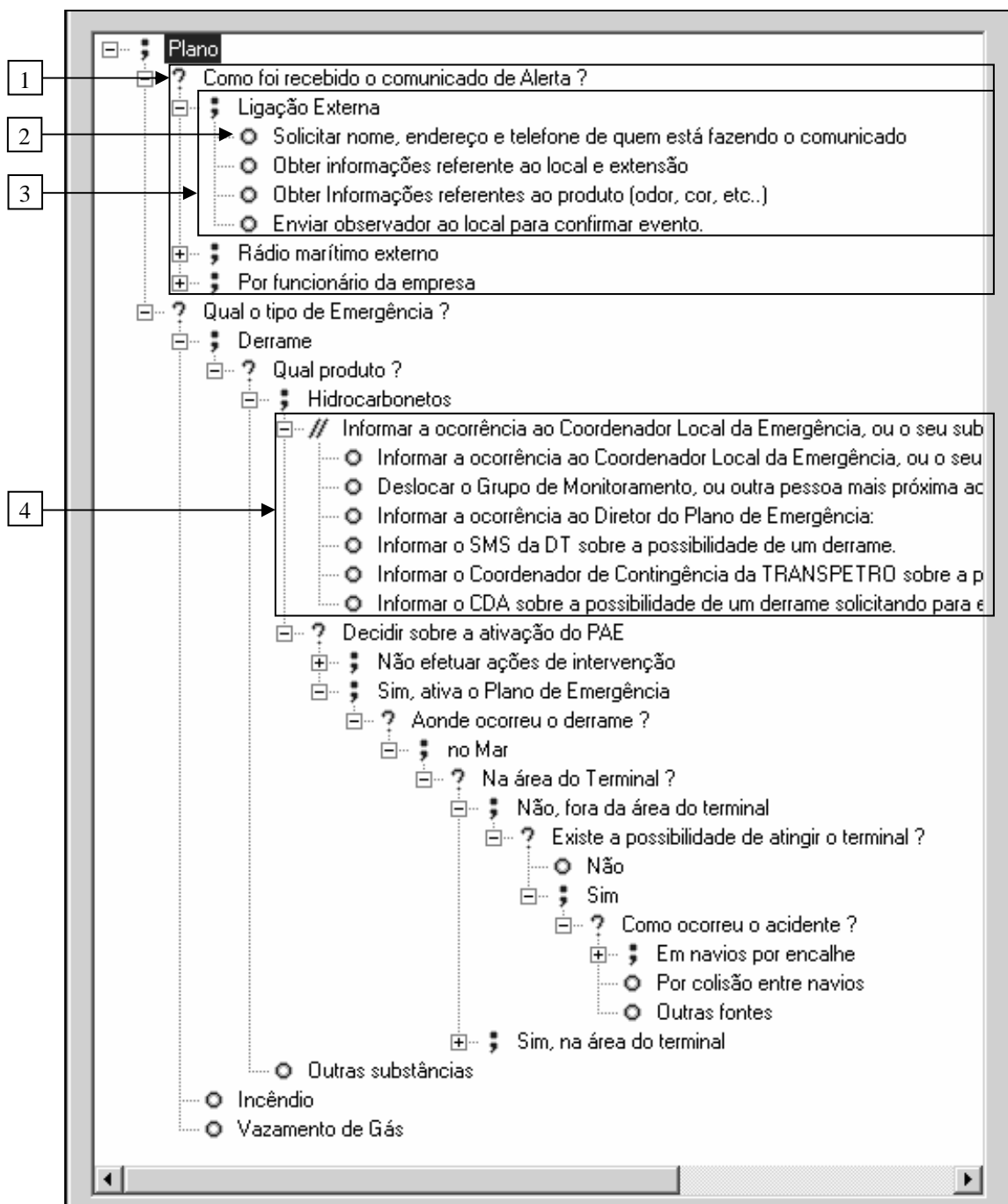


Figure 1 - EditaPAE main window.

1. Question with text “*Como foi recebido o comunicado de alerta*” (“How was the incident communicated”) and three alternative answers, “*Ligação externa*” (“External call”), “*Rádio marítimo externo*” (“External radio communication”) and “*Por funcionário da empresa*” (“By an employee of the company”).
2. Action with text “*Solicitar nome, endereço e telefone de quem está fazendo o comunicado*” (“Request the name, address and telephone of the person that communicated the incident”).
3. A sequential group composed of four actions.
4. A parallel group, composed of six actions, and labeled “*Informar a ocorrência ao Coordenador Local...*” (“Inform the incident to the Local Coordinator...”).

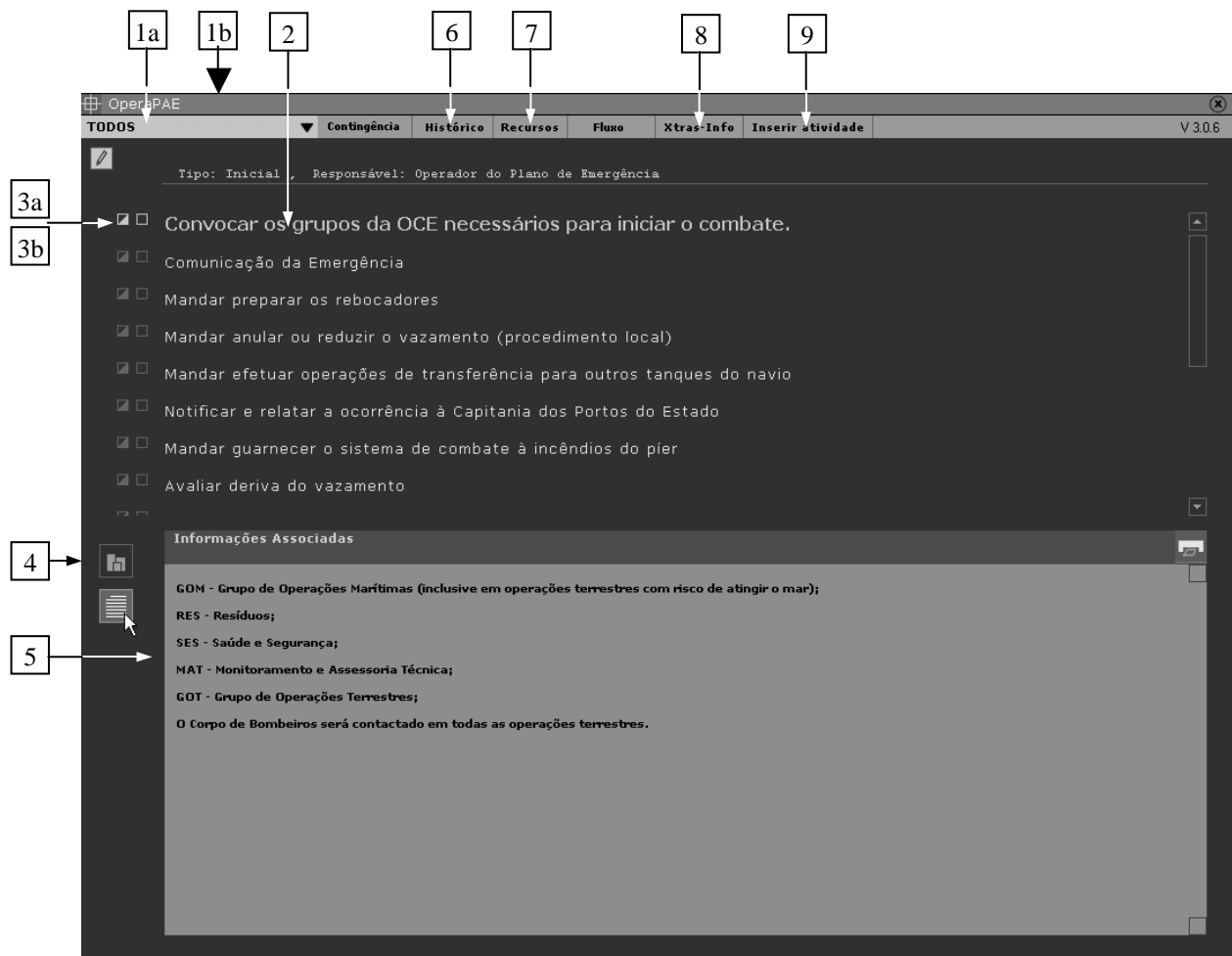


Figure 2 - OperaPAE main window.

1. The user may select, from the set of current actions, the subset of all actions of a given type:
 - a. Pulldown menu used to choose the desired action type.
 - b. Panel showing the list of actions of the selected type.
2. The user may pick up an action from those selected.
3. The user clicks on:
 - a. the button on the left to inform that he started the action, and
 - b. the button on the right to indicate that he finished the action.
4. List of predefined icons representing the information types associated with the action.
5. Panel showing information associated with the action.
6. Report listing all actions that were already started.
7. Report listing all resources associated with current actions.
8. Report listing all documents associated with current actions.
9. Insertion of an anticipated action (only for the current execution of the plan).

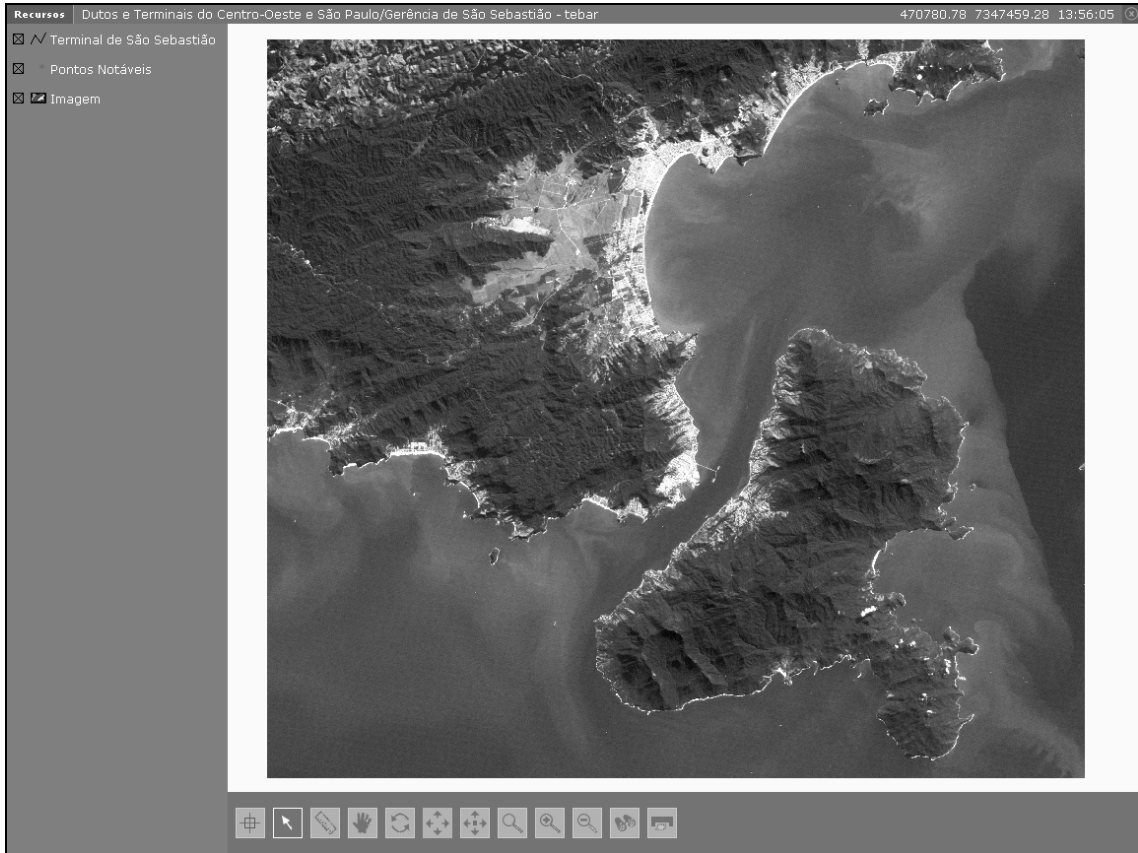


Figure 3 - VistaPAE main window.

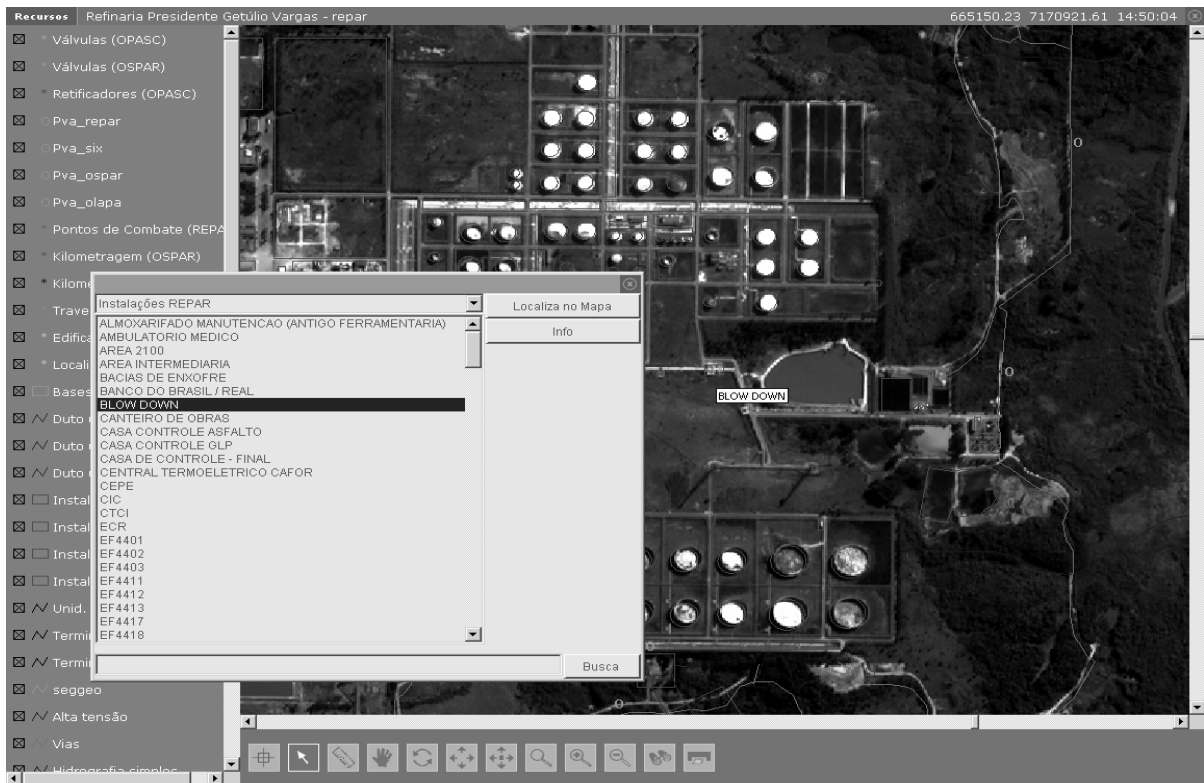


Figure 4 – VistaPAE - Locating objects over a geo-referenced image.