ELISA, A DEMONSTRATOR ENVIRONMENT FOR INFORMATION SYSTEMS ARCHITECTURE DESIGN.

Chantal PANEM
CNES French Space Agency
18 Avenue Edouard Belin
31055 Toulouse Cedex- FRANCE
Tel.: (33) 61 28 26 72
email: panem@melies.cnes.fr

ABSTRACT

This papers describes an approach of reusability of software engineering technology in the area of ground space system design. System engineers have lots of needs similar to software developers ones: sharing of a common data base, capitalization of knowledge, definition of a common design process, communication between different technical domains. Moreover system designers need to simulate dynamically their system as earlier as possible. Software development environments, methods and tools now become operational and widely used. Their architecture is based on a unique object base, a set of common management services and they home a family of tools for each life cycle activity. Late 92, CNES decided to develop a demonstrative software environment supporting some system activities. The design of ground space data processing systems was chosen as the application domain. ELISA (Integrated Software Environment for Architectures Specification) was specified as a "demonstrator", i.e. a sufficient basis for demonstrations, evaluation and future operational enhancements. A process with three phases was implemented: system requirements definition, design of system architectures models and selection of physical architectures. Each phase is composed of several activities that can be performed in parallel, with the provision of Commercial Off the Shelves Tools. ELISA has been delivered to CNES in January 94, currently used for demonstrations and evaluations on real projects (e.g. SPOT4 Satellite Control Centre), it is on the way of new evolutions.

Keywords: PCTE (Portable Common Tool Environment), Satellite Control Centre, Ground segment, computer science, data processing, architecture, simulation, queueing networks.

This article starts by a presentation of the rationale for ELISA development, it describes the implemented life cycle, the workbench architecture and ends with first conclusions of the project.

FROM SOFTWARE ENGINEERING TO SYSTEM ENGINEERING...

After several years studying software engineering environments, mainly for the needs of the Hermes program, it appeared that they become operational and that any software project can find rather easily satisfying COTS environments, methods and associated tools.
On the other hand, in the area of system engineering, the lack of an approved, detailed and well-defined common design process, the variety of tools and the poorness of communication between them, increase difficulties when the project size grows.
The idea that the knowledge acquired in software engineering area could help the design of space systems was the starting point for this new orientation of our activities.

A REUSABLE TECHNOLOGY FOR "SYSTEM" DESIGN ?

Before trying to show how the technology was reused, lets explain it in few sentences.
Up-to-date software engineering environments are based on a so-called "integration platform" or "integrated project support environment (IPSE)" or "integration framework", in which a variety of tools are "plugged-in". The framework offers tools integration services, in three degrees: data integration via a repository, which role is to define, store and control all data needed by

heterogeneous tools, control integration for communication between tools (interoperability) and presentation integration for uniform access to tools via the user interface. Some frameworks also provide process integration services for piloting of users activities according to a predefined life cycle. Two kind of services complete the environment to get a full "workbench": "horizontal" services, like documentation, configuration management, project management which are used in all the project phases, and "vertical services" which support individual life cycle activities by means of COTS tools (e.g. IDEF tool).
Such architecture did not seem limited to software engineering applications, but its adequation for system engineering needs had still to be proved.

ELISA, A FIRST STEP TOWARDS A WORKBENCH FOR DATA PROCESSING SYSTEM DESIGN

It was thus decided to develop a "demonstrator", i.e. a demonstrative environment based on an IPSE technology and supporting a coherent and consistent set of system activities. The chosen application domain was the design of data processing systems for ground space segment.

ELISA (Environnement Logiciel Intégré pour la Spécification d'Architectures informatiques/ Integrated Software Environment for Architectures Specification) users requirements specifications were produced in december 92. The objectives were:

- to demonstrate the benefits of software engineering IPSE frameworks in the system area,
- to increase CNES experience on three points: modeling of a design process, interface with a technical knowledge capitalization system and integration facilities,
- to show the interest of specific system tools and moreover of an integrated workbench with respect to isolated tools.

ELISA development was then reduced to the minimum set of services needed for demonstrations and evaluation but as a reusable basis for future enhancements. The constraint was to reuse as far as possible commercial tools and to limit specific developments.

THE ELISA REPOSITORY: a kernel for traceability and reusability.

ELISA is based on a PCTE repository (ECMA and Draft International ISO Standard). The repository is an object management system, which allows to define entities with the Entity-Relationship-Attributes model, to store them in a distributed way and to execute operations on them (calls to external tools). The ELISA data model has been defined in a modular way (thanks to PCTE) leading to an organised network of objects representing pertinent information for the user (functions, requirements, documents, architecture, equipments, simulation scenarios and results, etc..).

In ELISA, links between the objects represent either composition relationships, either trace relationships. Trace links are used to store implementation/ validation relations according to the process model steps: for example a computer linked to a performance requirement can mean that the computer implements it. Traceability allows the user to navigate directly between heterogeneous objects, to assess requirements coverage by architecture trade-offs and to analyse the impact of changes of a customer's requirement, or a system function. Through the trace matrixes produced by ELISA, the list of objects impacted by a modification is available at any step of the process.

The ELISA repository is also used as a "Technical Memory" storage, a place in the data model allows to capitalize information from old projects, feedback from previous studies, catalogues of hardware and software products and also architectural data from previous projects realised with ELISA.

The user can at any time consult and reuse a functional or physical architecture from this technical memory.

The definition of a complete, coherent and efficient data model is one of the tasks that require most reflexion, by the fact that it's the basis for tools integration and invocation. Extension to the data model is easy, but deletion or modification of data types seem more delicate once the environment is used by several projects.

WHICH ACTIVITIES DOES ELISA SUPPORT ?

Large projects are composed of a large number of complex and inter-related tasks. The initial work was to define the reduced life cycle that should implement ELISA, this comprises the definition of the activities that will be supported (WHY), their scheduling (WHEN), the persons who will perform them (WHO), the tools that will be used (HOW) and the products that should be available as inputs and outputs of the tasks (WHAT). This work appeared to be fundamental for the good achievement of the project.

The ELISA process model has three phases:
- System requirements definition,
- System architectural models design,
- Physical architectures assessment.

As it concerns a design process for early phases of a space system, flexibility is the major issue for an efficient assistance to users. The three phases are not purely sequential, but the user can complete them in an iterative way.
The EAST IPSE framework has been chosen, mainly for its ability to define, control and monitor any user defined process model, through the user interface. Different types of users have been defined (customer, architect, administrator and project manager). When a user starts the environment, he can activate tasks that have been assigned to him by the project manager. Starting a new project according to the ELISA process model become a mere operation.

PHASE I: DEFINITION OF SYSTEM REQUIREMENTS

The first task the designer deals with, is the capture of the requirements and constraints of the system. ELISA assists him in performing three activities: formalization of system and functional requirements, functional architecture analysis and definition of the logical sequencing of data processings.

Extract imposed requirements:
The first step is to deduce from input system specifications and customer interviews, the information which will be pertinent and/or constraining for the system architecture. These information are identified as requirements and can relate to several system aspects like performance, security, integrity, sizing, fault-tolerance.
With ELISA, requirements are managed by the LOTUS 123 spreadsheet tool, tables of requirements are created and filled-in by the user. Requirements are formalized by several attributes: an identifier, a textual description, a status (to be defined, hypothesis, computed, stable) and a value that can be the result of a formula computation from other requirements values.
Input specifications and interview notes can be stored in the repository if compatible with FrameMaker format, traceability links can be set towards them in order to keep the origin of design choices.

Analyse the functional architecture of the system:
The second step covers the analysis of the system functions. ELISA assists the user by the integration of the ASA tool supporting the IDEF0 methodology. The designer creates a functional model, edit it and refines the system functions in a hierarchical way, until obtaining a tree where leafs correspond to processes or software pieces. Each function is extracted and accessible in the repository as an object, automatically linked to its father and sons.

Tables of functional requirements can here also be attached to any function of the tree (node or leaf), for enabling the user to add details like performances, input and output data volumes or activation frequency.
ELISA ensures the consistency of the functions tree and the attached requirements tables; if some functions are renamed, moved or deleted in ASA tool, the corresponding objects in the repository are automatically changed; on the reverse, if the user deletes objects in the repository, he will receive inconsistency warnings.

Define logical sequencing scenarios:
Starting from functions and associated performance and data flows requirements, the designer usually defines a set of data processings and looks for their sequencing and synchronization constraints.
Functional analysis only gives a static view of the system which is not sufficient, the dynamic behavior is represented via "chronograms". Chronograms graphically express duration, start and end dates of each processing execution, in a given time. scale. Several chronograms are necessary to analyse nominal and critic paths of the system. This step allows to highlight possible parallelism, concurrency and synchronization constraints between processings. With the FrameMaker graphical toolbox, it is possible to create, edit as many chronograms as needed (exploitation chronograms, telemetry level n processing sequence..) and to link them to requirement tables and functions.
At any time, one can query the "technical memory" for estimation of some processing duration, by comparison with previous similar projects.

PHASE II: DESIGN OF MODELS OF THE DATA PROCESSING ARCHITECTURE

A key feature in system design is to predict system performances as soon as possible, in order to foresee system evolution ability, according to potential customer requests. System designers perform trade-offs between central processing, distributed, client/server or cluster architectures and have to propose the best one. Alternative solutions are often provided on designers experience basis or on hardware constraints. But few solutions are in fact really studied for a given project.

The objective of this phase is to come up with alternative models of the system hardware and software architecture, which all satisfy the requirements defined in the previous phase.

ELISA gives help in three inter-related activities:
- software architecture modeling,
- hardware architecture modeling and
- overall model validation.

The support to the whole phase relies on an integrated toolset for system modeling and performance evaluation: MODLINE. It is an open environment for modeling discrete event systems, developed and sold by SIMULOG (F). At the time the ELISA project started, no commercial tool with a satisfying high level user interface was available. MODARCH, a new tool has been added to MODLINE on CNES request, starting from an existing mock-up.
System designers are rarely familiar with formal technics (e.g. petri-nets) or with queueing networks, so they need to manipulate "macroscopic" and realistic components. With an ergonomic graphical object oriented interface, MODARCH let them manipulate and compose tasks, processors, networks, storage devices, terminals...
The tool relies on the queueing networks theory (QNAP2/ Queueing Network Analysis Package, from Inria and Bull F).
It must be kept in mind that the objective is not to monitor precisely the performances of a system, but to evaluate roughly the performances and sizing capacities of a future system. Most of its parameters and then results will be known in an approximative way, but in an acceptable margin, depending on the current project phase (A, B or C).

Model software architecture:

The activity consists of defining a model of the software application that answers to the functional architecture of phase I. For simplification purpose and demonstration of data sharing between integrated tools, ELISA implements a single concept: functions; this means that a function, a software, a processing or a task represents the same object in the environment.

From the function tree, the environment automatically extracts the leafs and generates the software tasks each time the user edits the MODARCH architecture.

ELISA again maintains consistency: the functional analysis may evolve, software architecture modifications will automatically follow, for example adding a function in ASA will add a task in MODARCH, but removing a function will generate a warning.

Automation is provided when possible, but the user is still free to work in inconsistent states, ELISA guides him in a predefined way but does not enforce him, at least he is warned.

The initial architectural model is composed of a set of independent tasks (names of leaf functions). The designer refines them by filling attributes (priority, memory) and activation conditions. The behavior of the tasks is defined with QNAP2 language and operations (read/ write in a storage, send message to other task, consume CPU, etc..). Four task types are provided: *sources* which allow to activate tasks by sending requests, *tasks* which execute some code on reception of requests, *in-out tasks* for modeling files, data bases, *exit* tasks for deletion of requests. This ends with an executable tasks network.

Model hardware architecture:

The architect now looks for a hardware configuration that satisfies the software application. Without leaving MODARCH, he selects components in an equipments data base (processors, storage devices, terminals, networks..). Each component is typed (a printer and a screen are of terminal type, the processor type includes workstations and mainframes), and its attributes can be instantiated with user defined values (memory, swap, CPU, rates...). The behavior of the hardware equipments is coded in QNAP2 and is hidden to the user, standard algorithms, systems (Unix) and communication protocols are available.

A major requirement towards MODARCH was the flexibility of the projection of software tasks to hardware equipments. With a simple graphical link, the user affects a task to a processor or to another. No user code is modified if the task is moved or if the storage device to which it sends write operations is attached to a remote machine.

The objective is indeed to analyse several solutions as easily as possible. The MODARCH user interface ensures consistency controls (a task can only be mapped on a processor, a source can be mapped on a terminal, an in-out task can be mapped to a terminal or a storage).

Each hardware and software component of a model can be parameterized (e.g. a CPU size, a message length, a task priority), the parameters values will be used for simulation purposes.

At any time, the user can call the other tools, in order to have all the system views in his screen. When he decides to close his model, the repository automatically imports the system components and the mapping links between software and hardware. New tasks will then be created and linked to the processor objects, existing ones (functions) will be updated with MODARCH information.

The user can add trace links between new objects and previous ones. The result is a graph of inter-related objects which allow direct navigation between activities (from a requirement table relating to function A to the description of the computer which runs the task A). The impact of a requirement evolution is thus immediately visible if the user lists the traced objects; the reverse is true, if a hardware equipment evolves, the user can control the impact on traced requirements or functions.

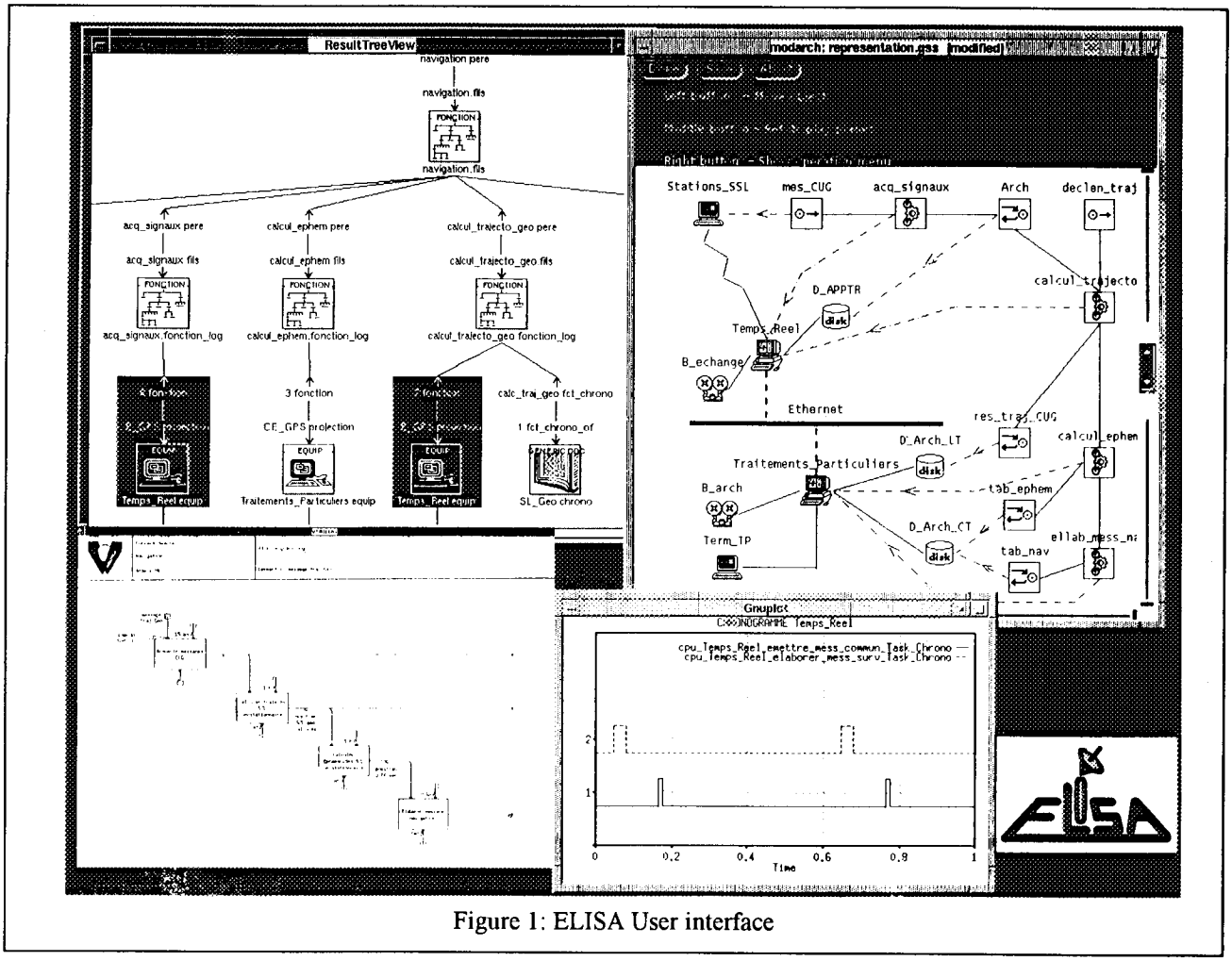

Figure 1: ELISA User interface

Validate the overall model:

The last activity consists of executing the simulation model and analyzing the results. The MODLINE toolset integration in ELISA is helpful.

Simulation scenarios (experiment plans) allow to define variation laws for the parameters values of the above model.

Via a single operation, MODLINE automatically checks, generates and compiles the code, executes and provides results. With a graphical results analyzer, the user can edit curvs or resources occupation chronograms. An animation tool allows to follow messages exchanges dynamically and then helps debugging of the model.

Each data is stored in the repository: scenarios, results, analysis drawings, links to its parent objects are set by the environment. Through the repository the user can compare output analysis chronograms and those issued in phase I, this assessment may help him choosing an architecture type or another.

The process is iterative, according to resources utilization analysis, the user can change his models until obtaining the better optimized one.

Figure 1 shows the ELISA user interface with views on the repository, ASA and MODARCH models and a chronogram.

PHASE III: ASSESSMENT OF PHYSICAL ARCHITECTURES

In the last phase, the designer chooses a real physical architecture after studying several implementations for each model (HP, SUN, IBM?). ELISA supports four activities:

- Selection of real equipments,
- Validation of physical architecture,
- Cost computation,
- Choice of physical architecture.

The tools are the same as above: MODARCH/ MODLINE, LOTUS 123 and Frame Maker.

Select real equipments. The architect selects existing physical equipments for each component of the chosen model. Specially, he chooses, via the MODARCH component base, a given workstation (e.g. SUN Sparc 10) or a given disk (e.g. Sundisk), and all their characteristics are updated. He may also refine the tasks behavior if necessary (e.g. Oracle for a data base model in phase II). When leaving the MODARCH tool, the repository is updated with new objects and links towards the origin model and functions.

Validate physical architecture. The validation of the physical architecture can be directly performed by using the scenarios of the source model. The user has to verify that the physical architecture still satisfies the system requirements.

Compute architecture cost. The objective is to provide an overall cost for the physical alternatives. By affecting a cost to all the components in MODARCH and with a LOTUS 123 integration, the user gets a table of costs he can complete and sum.

Propose an architecture. The end step is to give a proposition to the customer. With all information issued by previous steps (performances evaluation, requirements coverage, functional decomposition and costs estimation), the user can edit a Decision Justification Document with FrameMaker.

HORIZONTAL SERVICES

Documentation with Frame Maker.
Documentation is a very time consuming task, specially in early project phases. ELISA supports the user in composing and editing documents containing information produced by the tools. The "specification document" can be assembled in a semi-automated way: the IDEF0 graphics, the spreadsheet requirement tables, the chronogram drawings are imported in a synthetic document, from a predefined template. The user can yet complete and polish it before printing.

Configuration management.
The ELISA framework allows the user to manage versions of objects and to generate full or user-defined configurations of his project. Snapshots of his project will allow to stabilize versions of his work.

Administration tasks.
The administrator is responsible of the environment evolutions, of the repository management (save and restore operations), of feeding the technical memory, and of users and projects management.

ELISA ARCHITECTURE

The ELISA architecture is compliant with the one defined in the ECMA reference model [ECMA 91] and can be represented as in Figure 2.

ELISA runs on a Sun SparcStation 2, SunOS 4.1.3, Motif.
The PCTE repository is the Emeraude implementation.
ELISA has been developed by CISI S.A.

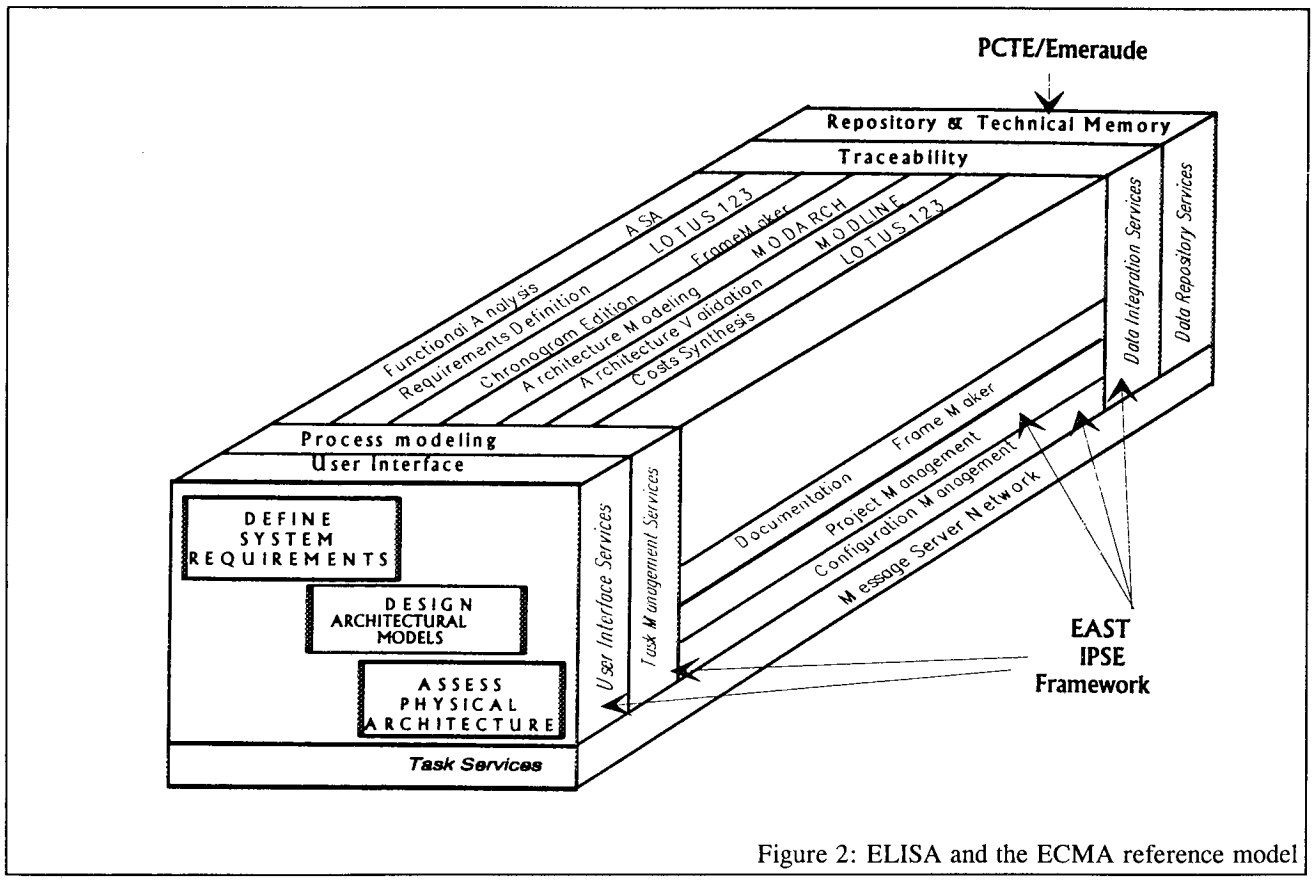

Figure 2: ELISA and the ECMA reference model

FIRST CONCLUSIONS

ELISA has been delivered in January 94. Architects have been trained to the demonstrator. Some demonstrations, at CNES and externally, have shown the public interest for the subject for the ELISA solution. Studies on system engineering environments currently raise in the european space and confirm our opinion.

The project allowed to complete our experience in tools integration, mainly in the impact of real data sharing between tools and the induced severe consistency checks.

The benefits of a system simulation tool are clear for the users. Moreover, with the ELISA environment, one can measure and better understand the benefits of integration like traceability, transparency of tools invocation, common services , and specially assistance to the generation of documentation.

ELISA has been delivered with an integrated real test case based on the CE-GPS project. It is currently being calibrated by Matra Marconi Space on the operational SPOT4 Satellite Control Center.

Some evolutions are going-on: porting on ECMA PCTE, moving to an operational environment, adding demonstrative features (multi-platform communications), integration of new tools version and process model enhancement.

References

[ECMA 91] ECMA, "A reference model for frameworks of software engineering environments", ECMA report n° TR/55.