SPAW

Software Product Assurance

for Web and communication based

Space Systems

Contract N. 16 811/02/NL/SFe - CCN4

Methods, recommendations and guidelines for web and communication based space systems

Identifier		***TN2***	
Version		***1.2***	
Date of issue		***15/09/06***	
Prepared by	***Name*** ***Company***	***P. Pleczon*** ***EADS ASTRIUM*** ***A. Canals*** ***CS-SI***	***Signature***
Verified by	***Name*** ***Company***	***JM Dussauze*** ***EADS ASTRIUM***	***Signature***

Abstract

The purpose of this document is to propose methods, recommendations and guidelines for the design of web and communication based space systems.

Section 2 exhibits the feedback extracted from space and non space web and communication based projects managed by EADS Astrium and CS. This section analyse each project and highlights their successes and failures and some recommendations built from their experience.

Section 3 is a review of the state of the art of Product Assurance practices on web-based projects in literature and www.

Section 4 proposes a synthesis of some data issued from sections 2 and 3: the recommendations issued from analysed projects, a synthesis on the technology adequacy to systems requirements, an overview of the language PA issues a discussion on the COTS/Open Source components concerns.

Section 5 recommends a set of guidelines for web-based and communication based systems PA.

Section 6 lists open points raised by the study but not covered by the current work.

Section 7 list the references (papers, web-sites, product references…) used to elaborate the study.

DOCUMENT CHANGE LOG

Issue/ Revision	Date	Modification Nb	Modified pages	Observations
1.0	20/06/06		All	Initial version
1.1	19/07/06			Updated after ESTEC comments.
1.2	15/09/06		§3.6.1, 3.6.5, 3.9 added. §3.7 reworded. Guideline 25 reworded. Guideline 28 deleted. §6.1 introduction added. Various rewording and explanations added.	Final update

The “Software Product Assurance for Web and communication based Space Systems” (SPAW) study is an ESA project which is conducted by EADS ASTRIUM with CS. For further information on this study, please contact:

ESTEC, European Space Agency
PO Box 299, NL-2200 AG Noordwijk ZH-The Netherlands
Maria GARCIA-REINALDOS, SPAW ESTEC Technical Officer
Tel : (31) 71 565 3964 Fax : (31) 71 565 4798
Maria.Garcia.Reinaldos@esa.int

EADS ASTRIUM France
31, rue des Cosmonautes, ZI du Palays
31 402 Toulouse Cedex - France
Patrick PLECZON, Project Manager
Tel : (33) 5 6219 6697 Fax : (33) 5 6219 7999
patrick.pleczon@astrium.eads.net

CS Systèmes d'Information
Parc de la Plaine
Rue Brindejonc des Moulinais – BP 15872
F - 31506 TOULOUSE Cedex 5 - FRANCE
Agusti CANALS
Tel : +33 5 61 17 66 15 Fax : +33 5 61 54.13.39
agusti.canals@c-s.fr

Contributors to this document

The people listed below have contributed to this document by providing information on their projects.

Serge BARROS	EADS Astrium
Pierre BORNUAT	CS
Patrice BRISON	CS
Patrick DUC	EADS Astrium
Olivier ETIENNE	CS
Dominique LAPEYRONIE	CS
Philippe ROUZET	EADS Astrium
CS and EADS Astrium Projects teams	

Table of Contents

1. Introduction

1.1. Purpose and scope of the document

The purpose of this document is to propose methods, recommendations and guidelines for the design of web and communication based space systems.

Section 2 exhibits the feedback extracted from space and non space web and communication based projects managed by EADS Astrium and CS. This section analyse each project and highlights their successes and failures and some recommendations built from their experience.

Section 3 is a review of the state of the art of product Assurance practices on web-based projects in literature and www.

Section 4 proposes a synthesis of some data issued from sections 2 and 3: the recommendations issued from analysed projects, a synthesis on the technology adequacy to systems requirements, an overview of the language PA issues a discussion on the COTS/Open Source components concerns.

Section 5 recommends a set of guidelines for web-based and communication based systems PA.

Section 6 lists open points raised by the study but not covered by the current work.

Section 7 list the references (papers, web-sites, product references…) used to elaborate the study.

1.2. Glossary and Acronyms

AD	Applicable Document
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-The-Shelf
CC	Control Centre
CSS	Cascading StyleSheet
DRD	Document Review Discrepancy
ECSS	European Co-operation for Space Standardisation
ESA	European Space Agency
ESTEC	European Space Technology Centre
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HW	HardWare
IT	Information Technology
J2EE	Java 2 Enterprise Edition
Java NIO	Java new I/O
JDBC	Java DataBase Connectivity
JMS	Java Message Service
JMX	Java Management Extensions
JCA	Java Connector Architecture
LoP	List Of Project
MOM	Message Oriented Middleware
MPI	Message Passing Interface
MVC	Model View Controller
NA	Not Applicable
OO	Object Oriented
PA	Project Assurance
PDS	Previously Developed Software
PM	Progress Meeting
RD	Reference Document
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SCC	Satellite Control Centre
SOAP	Simple Object Access Protocol
SOW	Statement of Work
SPAW	Software Product Assurance for Web and communication based Space Systems
TBC	To Be Confirmed
TBD	To Be Defined
TM	Technical Meeting
TN	Technical Note
TOAD	Tool for Oracle Application Developpement
UML	Unified Modelling Language
WBS	Work Breakdown Structure
WP	Work Package
WPD	Work Package Description
WS	Web Service
wrt	with respect to
XP	eXtreme Programming

XSL	eXtensible Stylesheet Language
XSD	XML Schema Definition

1.3. Terms and definitions

.NET	The .NET framework created by Microsoft is a software development platform focused on rapid application development, platform independence and network transparency. .NET is Microsoft's strategic initiative for server and desktop development for the next decade. According to Microsoft, .NET includes many technologies that are designed to facilitate rapid development of Internet and intranet applications.
Browser	A browser, or web browser, is the software used to view web pages and interact with various kinds of Internet resources. The browser interprets the HTML used to format web documents and recreates the page on the screen. There are a variety of web browsers available, the two most common being Microsoft's Internet Explorer and Firefox.
CORBA	Common Object Request Broker Architecture - A standard from The Object Management Group (OMG) for communicating between distributed objects. CORBA provides a way to execute programs written in any language no matter where they reside in the network or what platform they run on. It enables complex systems to be built across an entire enterprise.
CSS	A Cascading Style Sheet (CSS) provides the ability to separate the layout and styles of a web page from the data or information. Styles such as fonts, font sizes, margins, can be specified in one place, then the Web pages feed off this one master list, with the styles cascading throughout the page or an entire site.
Design Pattern	In software engineering, a design pattern is a general repeatable solution to a commonly-occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code; it is a description or template for how to solve a problem that can be used in many different situations. (http://en.wikipedia.org/wiki/Design_pattern_(computing)).
Domain	Domain is used in this document in conjunction with several terms (domain layer, domain process, domain logic, etc.). It describes the part of the application that is specific to the mission logic in opposition to the other parts of the system: presentation, data and technical services. It is similar to the term “business” often used in the computer science terminology (business layer, business logic, business process).
Framework	In software development, a framework (Figure 1) is a defined support structure in which another software project can be organized and developed. A framework may include support programs, code libraries, a scripting language, or other software to help develop and glue together the different components of a software project. Frameworks are designed with the intent of facilitating software development, by allowing designers and programmers to spend more time on meeting software requirements rather than dealing with the more tedious low level details of providing a working system. (http://en.wikipedia.org/wiki/Framework).
IDE	Integrated Development Environment. A GUI workbench for developing code, featuring facilities like symbolic debugging, version control, and data-structure browsing.

J2EE	J2EE (Java 2 Platform, Enterprise Edition) technology and its component-based model simplify enterprise development and deployment. J2EE is a collection of related specifications and corresponding documentation that describe an enterprise-level computing architecture for the Java platform. There are a number of J2EE containers, also known as application servers, such as JBoss, WebLogic, WebSphere, that implement the specification.
JMX	Java Management Extension is a Java API allowing the management of a Java application by external clients.
Layer	In computing, Three-tier is a client-server architecture in which the user interface, functional process logic ("business rules"), data storage and data access are developed and maintained as independent modules, most often on separate platforms. These independent modules are generally named tiers or layers (presentation layer, data layer, domain - often named business - layer). (see Figure 1)
Middleware	Middleware (Figure 1) is computer software that connects software components or applications. It is used most often to support complex, distributed applications. (http://en.wikipedia.org/wiki/Middleware). Most used middleware are CORBA, RMI, SOAP, Message Oriented Middleware.
MOM	MOM (Message Oriented Middleware) software that connects separate systems in a network by carrying and distributing messages between them. MOM infrastructure is typically built around a queuing system that stores messages pending delivery, and keeps track of whether and when each message has been delivered. IBM MQSeries is one of the most known MOM. Standards specifications such as JMS and WS-ReliableMessaging are now enabling standards-based MOM infrastructures.
MVC	MVC (Model View Controller) is a design pattern that defines a separation of concerns in a program where the model defines the internal data structures of the program, the view defines how the model is rendered to the user, and the controller performs the actual actions in the program that affect the model. MVC2 is a term invented by Sun to describe an MVC architecture for Web-based applications in which HTTP requests are passed from the client to a “Controller” servlet which updates the “Model” and then invokes the appropriate “View” renderer.
ORB	Object Request Broker is the software that functions as a broker (or intermediary) between a client request for a service from a distributed object or component and the completion of that request.
Platform	In computing, a platform (Figure 1) describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries. (http://en.wikipedia.org/wiki/Platform_(computing))
RMI	Remote Method Invocation (RMI) provides a means for invoking the methods of remote Java objects.

SOA	A service-oriented architecture (SOA) is a collection of services that communicate with each other. The services are self-contained and do not depend on the context or state of the other services. They work within a distributed systems architecture.
SOAP	SOAP (Simple Object Access Protocol) is a protocol designed to exchange XML structures other a computer network.
Struts	Apache Struts is an open-source framework for developing J2EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt an MVC architecture.
UML	Unified Modelling Language™ (UML) is an industry-standard language for specifying, visualizing, constructing, and documenting the artefacts of software systems standardized by the Object Management Group.
XSD	XML Schema Definition (XSD) is an instance of an XML schema written in the XML Schema language. An XSD defines a type of XML document i.e. the elements and attributes that may appear, their relationship to each other and the data. It can be used with validation software in order to validate whether a particular XML document is of that type

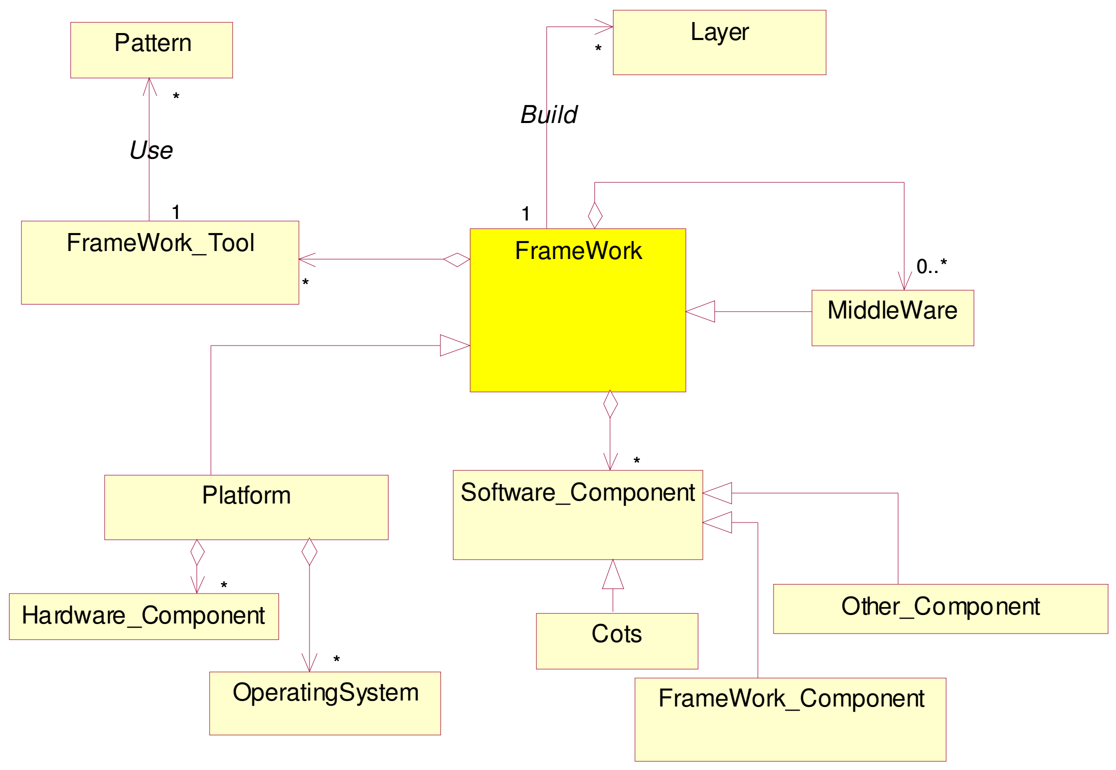


Figure 1: Relationships between Framework, platform, layer and middleware.

Frameworks exist at different levels: a framework can address either part of an application (GUI, communications, data access) or a whole application.

A framework includes software components such as runtime libraries and contains tools (generation tools, application management tools, etc.) and is used to build one or several application layers (presentation, domain, data…).

An installation platform is a framework installed on a specific hardware with a specific operating system.

A middleware can be viewed as a specialized framework used for implementing the communication layer.

1.4. Applicable and Reference Documents

AD1	Software Product Assurance for Web and communication based Space Systems – Statement of Work, TEC-QQ/2005/05-317/MG, issue 2.0
AD2	Technical Proposal for "Software Product Assurance for Web and communication based Space Systems" study, TP/CCSR/106/PP.05, issue 1.1.
AD3	Space Product Assurance — Software Product Assurance, ECSS–Q–80B, Oct 2003, ESA-ESTEC
AD4	Space engineering - Software - Part 1: Principles and requirements, ECSS-E-40 Part 1B, Nov. 2003, ESA-ESTEC
RD1	SPAW List of Projects, SPAW.TN.T.ASTR.0012, issue 1.1.
RD2	SPAW TN1 - Characterisation of typologies and definition of taxonomies, SPAW.TN.T.ASTR.0017, issue 1.1

2. Projects Analysis

This section analyses a set of projects in order to extract some recommendations for web-based and communication based systems.

Analysed projects are those listed in the List of Project (LoP) and in TN1. Some non space projects have been added in order to provide a larger panel of systems.

For each project the constraints and development approach used and the PA methodologies and tools used when they bring relevant information are examined.

2.1. SIPAD-NG

2.1.1. Projects main characteristics description

SIPAD-NG ("Système d'Information, de Préservation et d'Accès aux Données Nouvelle Génération" - "Information, Data Preservation and Access System - New Generation") allows an easy access to scientific data via Intranet or Internet to scientists in various fields.

The SIPAD-NG kernel is composed of a suite of "basic services" offering the standard functions of a data management system:

- ✓ ingestion mechanisms allowing to create a research and consultation catalogue (using complementary data : documents, browse products),
- ✓ data navigation and selection mechanisms allowing selecting the data which meet best the scientist's need.
- ✓ data ordering,
- ✓ Added-Value Services (data transformations), etc…
- ✓ Delivery of the ordered data carried out either by file transfer (ftp) or using a data processing support (CD-ROM,…). The Integration of external systems of media delivery and data restoration.

These "basic services" provide a suite of interface that enable an easy access to external "client applications": web servers, automatic processing software, remote Data Centres.

The SIPAD-NG offers a generic Web Interface, with the ability to be adapted to meet the need of the scientific project.

2.1.1.1. SIPAD-NG Services

The SIPAD-NG architecture is based on a layer of services. This layer is the kernel of the system. Client applications access to these services using the exposed interfaces of this layer.

The services are:

- ✓ **Administration and supervision service:** This service is used to manage (current state, start, stop) and configure all the other services, create user profiles. This service is available via a web GUI.

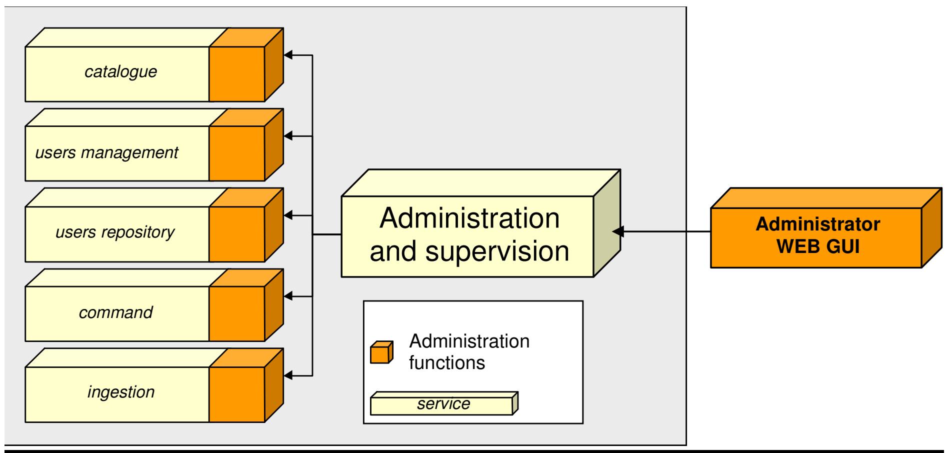


Figure 2 - Administration and supervision Service

- ✓ **Ingestion service**: This service is dedicated to metadata ingestion. This service is available via the administration web GUI and via an intranet application dedicated to data ingestion

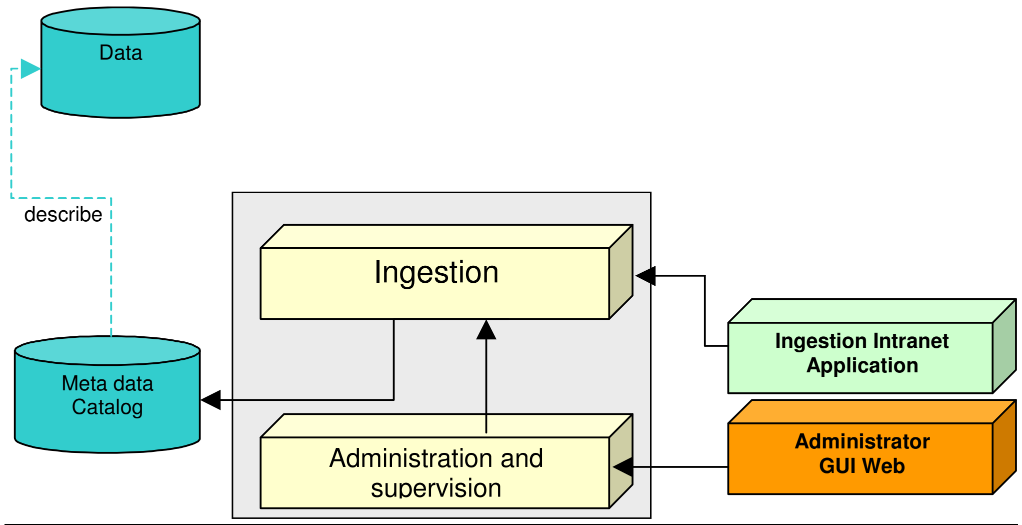

Data
describe
Meta data Catalog
Ingestion
Administration and supervision
Ingestion Intranet Application
Administrator GUI Web

Figure 3 – Ingestion Service

- ✓ **Users' management service**: This service provides functionalities to create/delete users, groups of users and to define their rights. The rights are associated with data model elements.

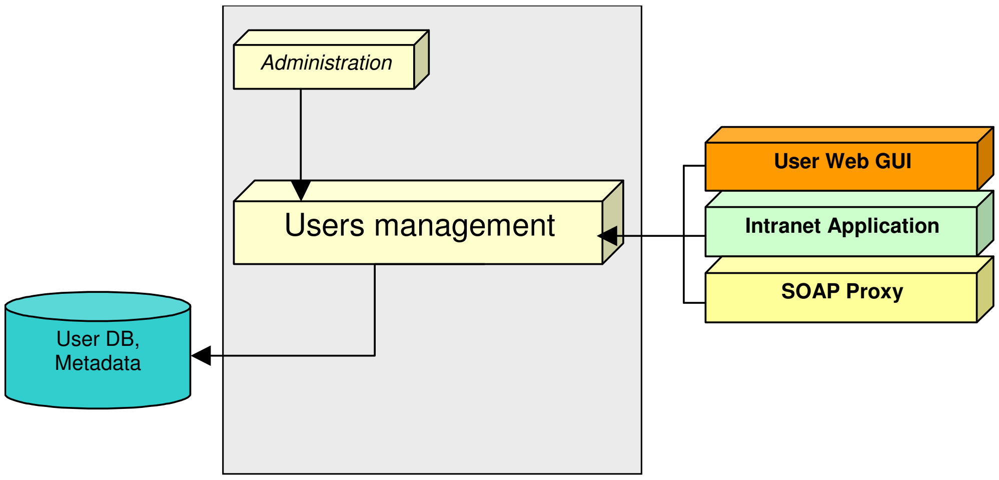


Figure 4 – User management service

- **User space service**: This service manages the physical data management: disks location, data compression, data organization, …

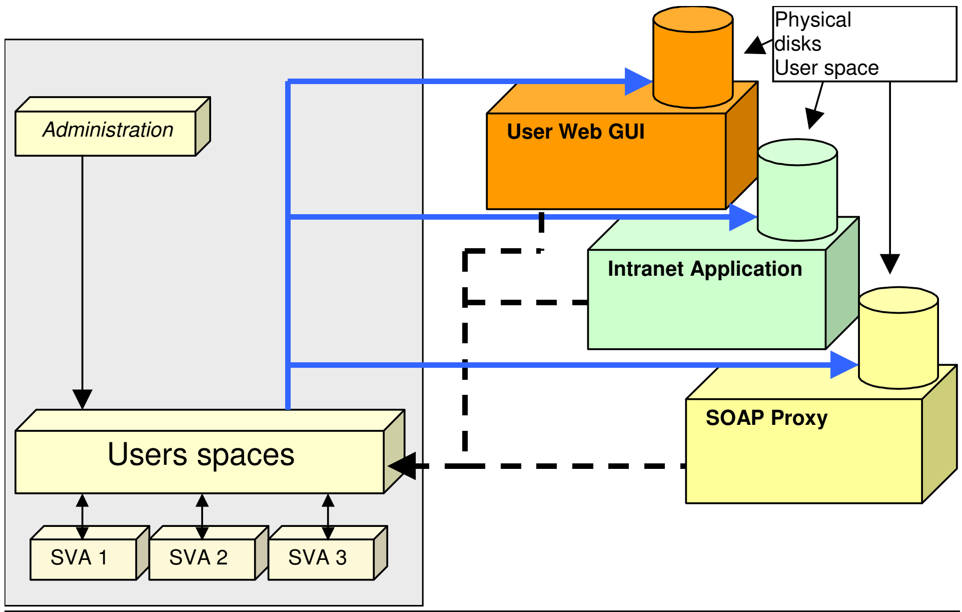


Figure 5 – User space Service

- **Catalog consultation service**: Its interface provides functions to search data from the data model and the rights associated to each element. This service is used by all applications that needs to known user's profiles. The information is stored in the database.

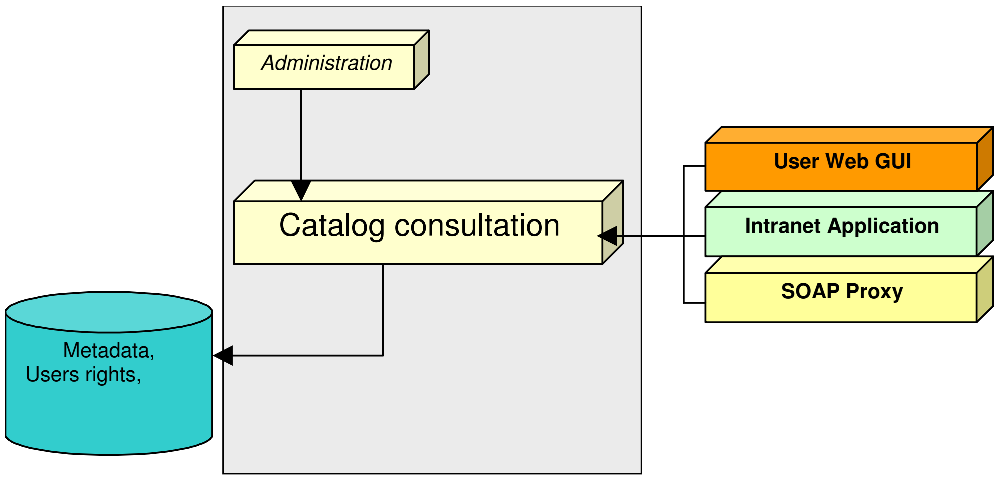


Figure 6 – Catalog Consultation Service

- ✓ **Command management service:** It checks feasibility, data retrieval, processing (SVAB) and commands delivery. It interacts with the storage service (ex: STAF) and also with the delivery services (ex: SEM, FTP). This service can host SVAB plug-ins.

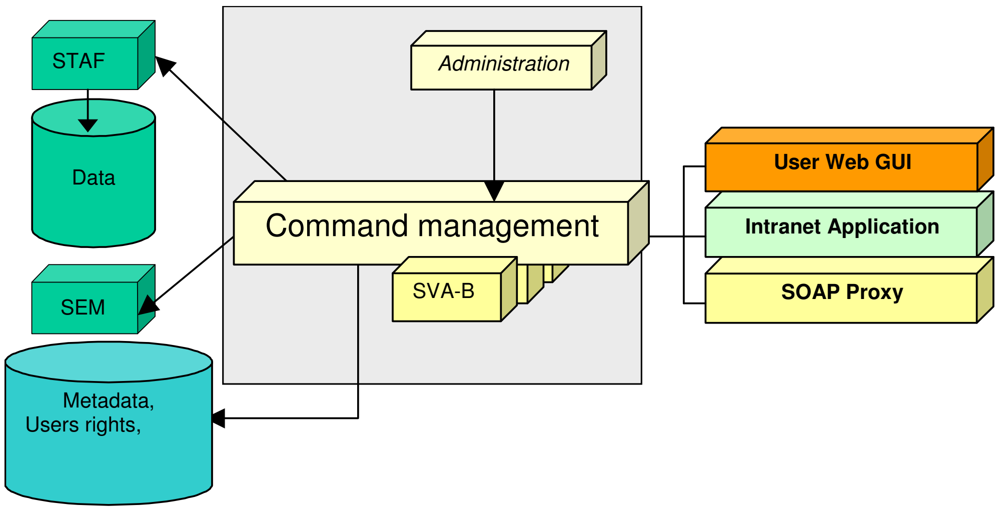


Figure 7 – Commands management Service

2.1.1.2.Client applications of SIPAD-NG

There are three kinds of client applications of the SIPAD-NG services:

- ✓ WEB clients,
- ✓ Internet applications,
- ✓ Intranet applications.

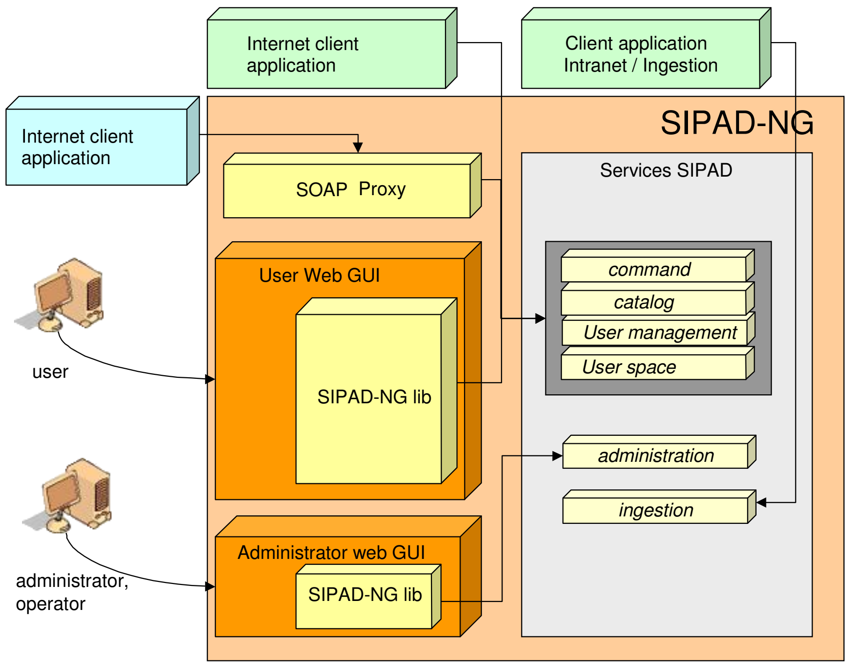


Figure 8 – Client applications of SIPAD-NG services

2.1.1.3.WEB Clients

The only WEB Client is the intranet client application for administration and supervision services. This client can be used from the Internet to work on data and to prepare commands.

2.1.1.4.Intranet clients applications

The ingestion is realized via a specific intranet application directly connected to the ingestion service.

The ingestion is run:

- ✓ Manually: The administrator runs ingestion via the Administrator Web GUI.
- ✓ Automatically: The client application watch a directory and run the ingestion on new data arrival

2.1.1.5.Internet clients applications

The Internet applications accesses to the services "catalog and selection", "command management", "users management" and "user space" via Web Services.

2.1.2. Lessons learned

2.1.2.1. PA methodologies and tools used

2.1.2.1.1. Estimating methods

Project sliced in tasks using a project management tool (GANTT diagrams with MS Project). Activity is reported monthly by the development team. 2nd coast estimation have been calculated using a formula depending on classes identified during preliminary conception.

Tools used: Word + Excel + MS Project

2.1.2.1.2. Team training

Junior developer formed to JAVA language.

2.1.2.1.3. Tools configuration

Realized by development team, for the Database an expert has realized the configuration tuning.

Recom 1 : When the application is data centric, the database configuration is critic and must be managed by an expert (Schema checking, database tuning).

2.1.2.1.4. Life cycles V, V+, Incremental, iterative

From the detailed design:

- ✓ n increments have been identified.

First increment:

- ✓ Prototype to validate the architectural choices (performance).

Other increments:

- ✓ Sets of functionality grouped on a same layer or across different layers.

Recom 2 : In a complex architecture (many layers using different frameworks/tools) the use of a prototype shall be used to validate architectural choices and to start development team training.

2.1.2.1.5. Requirement Analysis

Not specific.

2.1.2.1.6. Architectural Design

Modelling method and tools:

- ✓ Identification of layers,
- ✓ Identifications of components,
- ✓ Repartition of components in each layer,
- ✓ Identification of data flows :
    - Nodes ,
    - Data transformation (HTTP Request -> XML -> Java Object).

Tools used: Word + Rational Rose.

2.1.2.1.7. Detailed design

J2EE Design patterns (Data Access Object, Service Locator, MVC II, Session Facade …) have been used intensively during this phase because one of the requirements of the product was to build an application based on "pluggable" and independent components [J2E 02].

Note: The J2EE tools used by SIPAD_NG are APACHE, TOMCAT with STRUTS, a web services server (Axis) and RMI.

Database model:

- ✓ The database schema has been detailed using Power Designer.

Domain and Data Access Layers:

- ✓ These low layers have been detailed using UML. This part of the application is the most useful to detail because code generation can be made efficiently. The team spent a lot of time to design the core layers because it was the kernel of the application.

Recom 3 : The core layers must be detailed using UML.

Presentation Layer:

- ✓ UML is not adapted to communicate about this aspect. For this layer the transition between screens have been specified using of prototype,
- ✓ The main functionalities have been designed using sequence diagrams in witch Struts components appears in terms of Actions, View and Controllers. This is useful to show a general view of how the services are implemented. It's not interesting to do this for each service because the technical framework impose to do the things always in the same way,
- ✓ The design of COTS specific elements (Mapping XML-Java for the Digesters, Struts Actions chaining, TAG-Libs, …) hasn't been detailed using UML because its syntax and the modeling tools were not adapted for this.

Recom 4 : Use a prototype to validate the presentation layers. UML must be used only to explain high level processes (using sequence diagram for example).

2.1.2.1.8. Testing

Test tools:

- ✓ JUNIT have been used to validate the service layer (Unitary and integration tests). This tool is integrated to the IDE and reduces the regressions when new functionalities are added.

Test coverage:

- ✓ Test coverage matrix exits. Cobertura (http://cobertura.sourceforge.net/) has been used to check tests completeness. It provides a graphical presentation of the code coverage without any cost for the deployment of this solution.

Specific issues:

- ✓ Load testing has been validated using a specific prototype. This aspect have been validated very soon to validate architectural choices,
- ✓ The core functionalities of the data access and of the service layer have been developed independently. The interfaces of these layers have been fully defined and drastic test procedures have been defined to validate the protocols.

Note: *This approach is XP method compliant.*

Recom 5 : When it is possible use automated testing tools.

Recom 6 : When the interface of a component is completely defined, it's a good practice to write JUnit test cases before implementing the component logic.

2.1.2.1.9. Integration/validation

The framework (Digesters / Web services) implies that communications and data exchanges are made using XML. XSD documents have been used to validate these exchanges.

Simplified integration due to incremental development. The development has been made in two steps:

- ✓ First Step : Development of kernel layers (Data access/domain services),
    - For the first step, the integration has been eased because interfaces were fully defined during detailed conception.
- ✓ Second Step : Development of the presentation layer (WS, Struts, …)
    - For the second Step, integration has been made during development because functionalities have been developed using a Use-Case driven development,
    - The same person developed the functionality across all the layers (From user-interface – to the domain layer).

To detect problems earlier, the client was involved in the validation step.

Recom 7 : Validate complex data exchange using data flow descriptor (Ex. : XML validated by XSD).

Recom 8 : In a layered architecture integration must be anticipated by test means adapted at the layer level. The interface of each layer can be validated with automated test or with specific test modules which realizes queries on each service exposed by the layer.

2.1.2.1.10. Verification

Not specific.

2.1.2.1.11. Acceptance

High reactivity: the development team was ready to produce a patch in case of a blocking problem.

2.1.2.1.12. Operation

Not specific.

2.1.2.1.13. Maintenance

Not yet.

2.1.3. Successes and Failures

- ✓ Resources assignment
    - Using a database specialist is a real gain (The database access layer has been essentially develop by this person),

- New developers to the project have been assigned to development of services from user interface to the data layer. This is a good practice because it gives a vision to the whole system architecture.

✓ Incremental development life cycle

- Easiest integration,
- Minimize consequences of technical problems.

✓ Development by layers

- Implies that interfaces are fully defined,
- Strong communication between team members is needed when the design of a layer changes,
- The developers hasn't a global view of the system,
- Developers does "always" the same thing,
- The layer is consistent,
- The developer can't be easily replaced,
- A supervisor/expert is needed for each layer to maintain coherence.

Recom 9 : An expert must supervise the development of technically complex layers.

Recom 10 : In a development by layers, the interfaces must be highly detailed before their implementation occurs. The developer must have means to communicate with managers of layers n+1 and n-1.

✓ Development by functionality (Use case driven)

- This approach has been a real success to implement the presentation layer. Every developer has a global view of the system of how a functionality is implemented across the technical components of the framework of this layer. This reduces integration and performance problems.

Recom 11 : Use case driven approach must be used only when core layers are identified and specified.

Recom 12 : In a use case driven approach, each developers must have a global view of the architecture and have the means to communicate quickly with other use-case developers.

✓ Complex technical framework(RMI, WS, Struts, XML Digesters, …)

- Implies development team with strong OO knowledge,
- All COTS are based on Java => a good knowledge of this language is necessary,
- A specialist of each technology is needed to ensure the consistency of each layer (N-Tier and Web experts),
- Struts (http://www.jakarta.apache.org) is heavy when GUI are complex. Facing the evolution and the enrichment of the user interfaces, the team has reached the limits of light clients and the migration to a rich client has been discussed.

Recom 13 : J2EE projects team must contains senior developers who have an experience in this kind of architecture.

Recom 14 : The complexity of the user interfaces must be evaluated and compared with the Framework capabilities and the client application nature (Rich/light/heavy).

- ✓ Complex and very modular architecture
  - Implies an overhead for each of the new member of the team to understand the architecture and how to use it,
  - Writing the maintenance manual during development phase can help to reduce this overhead.

Recom 15 : The development team shall not change during development phase.

- ✓ Use of prototype
  - Validate architectural choices (performance, compatibility with technical platform),
  - Verify feasibility.
- ✓ Level of quality
  - Detailed design has been made deep enough to start development on strong basis.
- ✓ Use of adapted IDE
  - The use of Eclipse + plug-ins has been a real advantage => integrated environment: test, modeling, debug, …
- ✓ Use of COTS
  - COTS have been massively used (log4J, XML Digester,) they bring functions with a high level of quality and performance.
- ✓ Use of automated test
  - Accelerate integration process: component's interfaces are automatically validated.

Note: *In fact this success is due to the complementary use of XP method and JUNIT tool.*

Recom 16 : Use an IDE integrated to the framework.

2.1.4. Successes and Failures from similar projects

This section describes feedback collected from an other similar project (STILO). The architecture of this project was based on component spread over a local network.

These components are developed with different languages and technologies such as C++ programs, CGI or JAVA. The communication between these components was based on different protocols such as:

- RMI : JAVA Server to JAVA Applet communication.
- HTTP: Java Applet to HTTP Server (retrieve data).
- CGI/HTTP: Java Applet to C++ processes.
- File exchange: inter-C++ processes .
- File exchange: C++ processes to JAVA Server
- JDBC: JAVA Server to Oracle Database

✓ Internationalization

The JAVA environment provide mechanisms to implement internationalized resources. But these facilities are not available in the other development environment such as C++ components or messages stored in DATABASE.

Recom 17 : In an internationalized context, identify all sources of messages: exceptions, GUI, data stored in files or databases.

✓ Time management

The JAVA components were exchanging time tagged messages and the client application send tasks scheduling orders based on execution time.

To provide a consistent current time between these systems, the STILO system provide a time service. This requirement has been lately identified and a specific service has been implemented from scratch with a significant development cost to satisfy it.

Recom 18 : In a distributed architecture when processes are based on time the requirement must be identified and detailed soon in project design.

Basically there are two approaches:

- use system time of each machine synchronized with an external tool;
- develop a specific time service if particular requirements are identified : Specify a future/past date to exploit old data, simulate specific comportments (days changes), ...

✓ Life cycle of distributed objects

In the STILO architecture, the Client application communicates with a server via RMI and the server sends data change notifications to clients applications. In this architecture each component act as a client or as a server depending on interaction kind and must be aware of the availability of its service provider.

To satisfy system availability the team has developed specific monitoring services to inform user of components state (availability, crash, network failure, …) and to try to recover from minor crashes.

Recom 19 : In a distributed architecture based on connected components, the connection loss or degradation must be identified and reaction to these changes must be specified.

✓ Communication between heterogeneous components

The development has faced two major problems due to interfaces between components.

Integration between Java Applet and CGI Clients

The interface between these two components has been identified and detailed but the team has encounter problems using this technology because of parameters transformations : typing and format loss.

Integration between Java Server and C++ processes based on a file

Communications between these components use a single file: one component write in it and the other read from it. This approach is catastrophic for many reasons:

- Structured data is de-structured (on write) and restructured on read (processing time loss + redundant code)
- The file can be in incoherent state (disk full, message partially written)
- The reader component must manage all possible errors in data format due to data transformation
- The file can be replaced (file storage each week) without stopping readers and writers.

Recom 20 : When components communicates using a low level protocol, interfaces must be specified (services provided, arguments and return types, error feedback mechanisms) and the best practice is to provide automatic testing tools to validate protocol and each service.

The communication by file was involved by the reuse of a logging component developed in C++.

Recom 21 : Component reuse must be made accordingly to the context of the application particularly regarding programming languages and frameworks used by other components.

✓ Use of basic framework to satisfy high-level services

One of the major failure on this project was to choose a distributed architecture with heterogeneous components where communication is based on low level protocols (File I/O, sockets, HTTP, …). The integration has been hard and the use of most recent technologies such as webservices, Corba or a full RMI architecture would have been a better approach.

Recom 22 : Distributed components communication must be made using the appropriate framework and integration process must be prepared according to this choice.

2.2. COROLLE

2.2.1. Projects main characteristics description

The ROSETTA probe was launched in March 2004 for a rendez-vous with the Churyumov Gerasimenko comet in August 2014. After a phase of observation of landing site choice in orbit positioning around the comet, the ROSETTA lander will be dropped to land on the comet. Afterwards it will execute a sequence of scientific operations during 5 days. So there will be more than 10 years between the knowledge acquisition and its exploitation, over a few days, by a team currently unknown. Finally, the lander is realized by a consortium of 8 countries (Germany, England, Austria, Finland, France, Hungaria, Ireland, Italy). This schedule illustrates two critical aspects of the ROSETTA project:

- ✓ First, the project will be asleep during the long probe traveling period.
- ✓ Subsequently, the brevity of the study phase requires practically perfect competence on the part of all the players in this phase.

Corolle-Doc software resolves this problem. It is designed to capture maximum knowledge, organize it, and enable future users to quickly access the knowledge they search.

2.2.1.1. Functional Architecture

The functionalities of the COROLLE-DOC system are split in the following components:

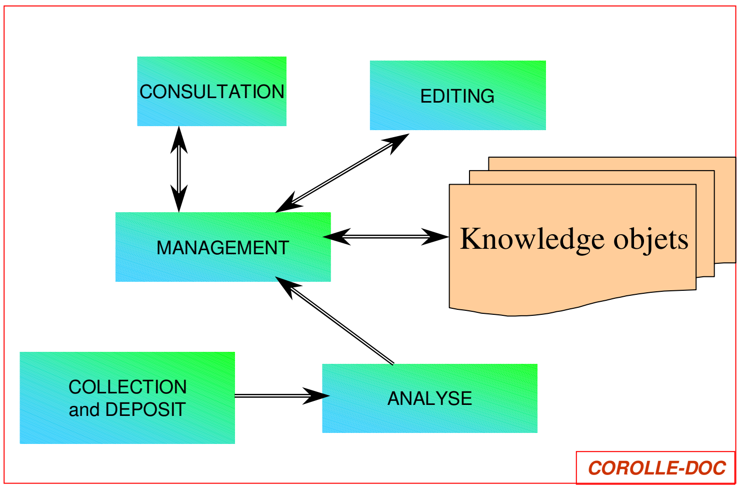


Figure 9 : General architecture of the Corolle system

- ✓ **The collection entity** is used to retrieve raw ROSETTA documents into the data storage. It is based on the internet technologies (HTTP or SMTP). The documents can also be retrieved from the Baghera documentation repository.
- ✓ **The deposit entity** is used to drop off documents directly on the system.

- ✓ **The analysis entity** offers to the administrator all the tools to fill the documentation repository from raw data. The analysis functionality extracts description from raw data and converts it into PDF documents. This functionality can be automatically triggered after each deposit.
- ✓ **The knowledge object** is the engine which retrieves and updates the database. These objects can be documents, "notules" and queries.
- ✓ **The management entity** provides the services to manage the document database: index management, state management, descriptors updating, …
- ✓ **The consultation entity** covers all the functionalities to search, retrieve and export data from document repository according to user's rights.
- ✓ **The editing entity** is used to update, add or delete knowledge objects from the database.

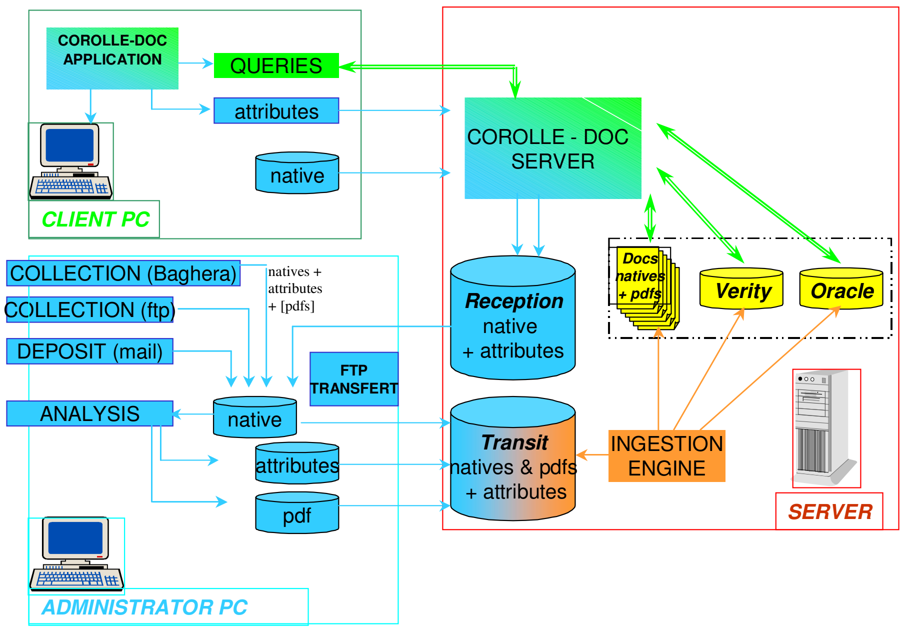


Figure 10 : Functional architecture of the Corolle-Doc system

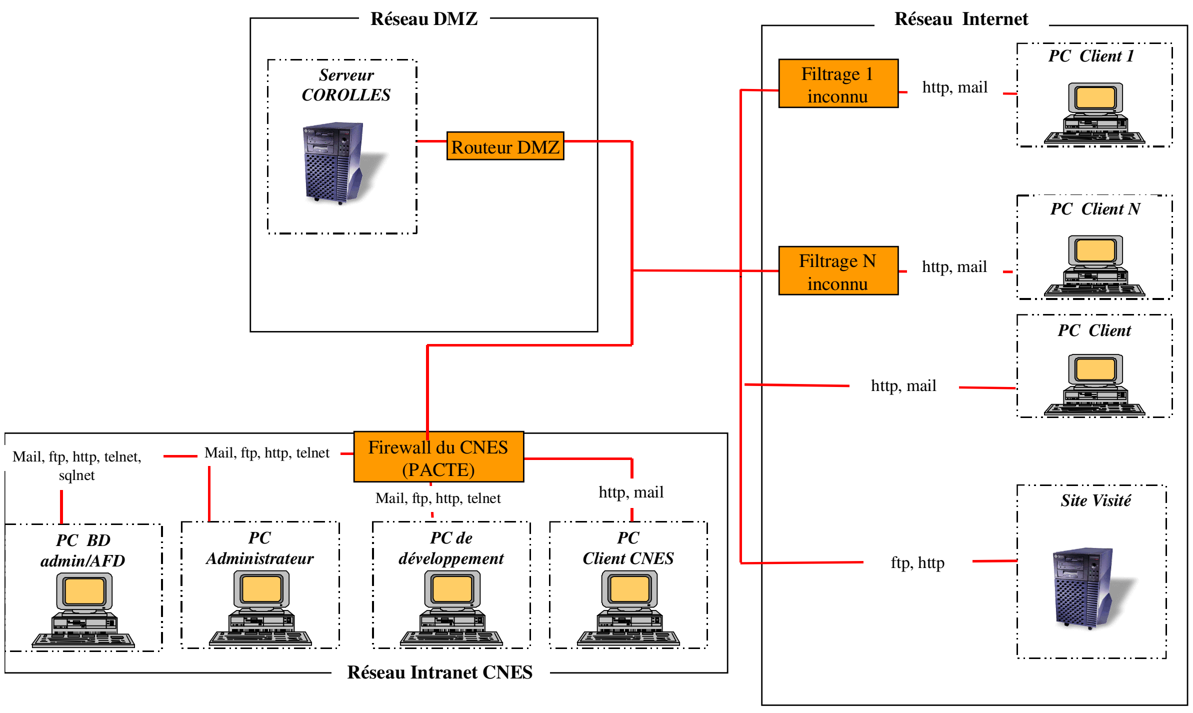
The user accesses to the knowledge database by the client application of COROLLE-DOC. This application is a java applet that runs on the client PC and communicates with the COROLLE-DOC server using the RMI protocol (tunnelling http). The user is allowed to retrieve documents, access to their description, use the full text search (implemented by Verity).

He can also send raw documents and their description to the COROLLE-DOC Server.

The administrator can collect and drop off data from Baghera, mail, ftp and run the analysis function from the administrator PC.

The ingestion engine has the responsibility to populate the knowledge database from PDF documents, their description and verity indexes runs on the application server machine.

2.2.1.2.Physical Architecture



All accesses from the client application (internet or intranet) are made via the Corolle Server. This server runs on a secure DMZ network on the public platform of the CNES.

2.2.2. Lessons learned

2.2.2.1. PA methodologies and tools used

2.2.2.1.1. Estimating methods

Not specific

2.2.2.1.2. Team training

Junior developer formed to JAVA language.

To integrate this kind of project the developer must master the basics concepts of distributed architectures. The Corolle architecture use distributed mechanism based on synchronous and asynchronous RMI calls. The JAVA API hides the underlying mechanisms of network calls and can lead to an incorrect architecture and cause performance or memory leaks problems.

Recom 23 : Supervise junior developer by experts/seniors who have an experience in distributed architecture.

2.2.2.1.3. Tools configuration

Realized by development team.

2.2.2.1.4. Life cycles V, V+, Incremental ,iterative

V cycle.

2.2.2.1.5. Requirement Analysis

Not specific.

2.2.2.1.6. Architectural Design

<u>Modelling method and tools:</u>

- ✓ Functional/logical architecture
- ✓ Identification of use cases,
- ✓ Description of uses-cases with sequence diagrams,
- ✓ Identification of layers,
- ✓ Identifications of components (COTS, …),
- ✓ Repartition of components in each layer.

<u>Physical architecture:</u>

- ✓ Identification of machines (Physical nodes),
- ✓ Communications between machines,
- ✓ Security constraints,
- ✓ Repartition of logical modules of each node,
- ✓ Text + Graphics.

<u>Tools used:</u> Word + Rational Rose

2.2.2.1.7. Detailed design

<u>Detail of each component:</u>

- ✓ Layer in which it is used,
- ✓ Internal structure : UML class diagrams + Text (description/pseudo code) + Graphics,
- ✓ Description of interfaces (services provided + how to use other components interfaces),
- ✓ Exceptions management: Text.

<u>User interfaces:</u>

- ✓ Use of a prototype to validate GUI choices,
- ✓ Description of complex pages structures (panels, sub-panels, …) : Text + graphics,
- ✓ Description of each user action : Text + graphics,
- ✓ Mapping of user actions with other components services: Text.

<u>Tools used:</u> Word + Rational Rose

2.2.2.1.8. Testing

<u>Test tools:</u>

- ✓ None.

<u>Test coverage:</u>

- ✓ Test coverage matrix exits.

<u>Specific issues: load testing, etc.</u>

- ✓ None.

2.2.2.1.9. Integration/validation

Not specific.

2.2.2.1.10. Verification

Not specific.

2.2.2.1.11. Acceptance

Not specific.

2.2.2.1.12. Operation

Not specific.

2.2.2.1.13. Maintenance

Not specific.

2.2.3. Successes and Failures

- ✓ Final target platform not available during development :

- Installation problems detected quite late (SSL, IPFilter, IP Port sharing, DMZ, …),
- System management team not enough available.

Recom 24 : The target platform must be known and available during the development phase.

✓ Migration from intranet to internet :

- Problem with platforms configuration (proxy session timeout),
- Bottleneck on the firewall with data transfer ,
- Success to keep acceptable response time.

✓ Data format exchange :

- Difficulties to exchange internationalized data between heterogeneous platforms (Windows/VB, Unix/JAVA, Oracle/SQL)

✓ Integration of COTS :

- Verity doesn't provide a JAVA API. JNI has been used by adapting existing sample code,
- Acrobat API wasn't enough efficient (in 1999). API viewer hasn't been embedded in the client application.

2.3. ATV-CC

2.3.1. Projects main characteristics description

2.3.1.1.Projects main characteristics description

The architecture of the ATV-CC Monitoring&Control (M&C) relies on three types of hardware components:

- ✓ The Telemetry Telecommand Server (TTS)
- ✓ The Operational Data Base Server (ODBS)
- ✓ The Monitoring & Control Workstation (MCW)

The TTS is a UNIX server that is mainly dedicated to receive the telemetry (TM) flows and to emit telecommands (TC). This server is in charge of acquiring the telemetry packets, and to distribute them towards the ODBS and the MCW (as processed parameters). It also groups all TC that are ready to be emitted and emits them. Furthermore, ATV and ISS processed TM is distributed towards external sub-systems, such as the Flight Dynamics Subsystem (FDS), an external entity (EST) and the DaSS server (Data Services Subsystem, hosted at GSOC).

The ODBS is an operational data server; it mainly contains the DBMS and the Web server used to make data extractions and trend analyses. Therefore, this server does all the storage and archiving tasks, and handles all the data requests. It also ensures the management of all data necessary to the centralized functions (user management, logbook management, etc.).

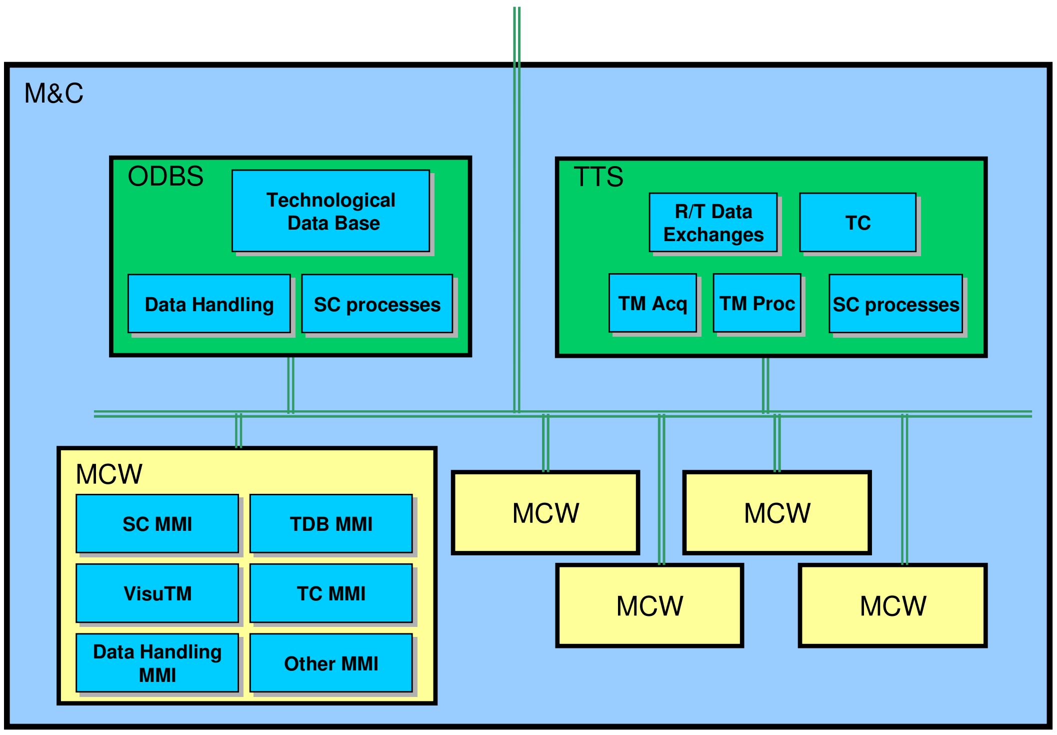
The MCW is a Windows 2000 console that allows any user to interact with the functions available on the ATV-CC M&C. On the MCW, the user can, depending on his profile, prepare TC stacks, send TC, visualize TM flows either in real-time mode or in replay mode (centralized or local), manage the ODB, visualize and analyze data offline, perform his own local monitoring and reload previous mission contexts (please note that not all functions are listed here).

When considering only the nominal instance of the M&C subsystem, it is composed of one TTS server, one ODBS server, and as many MCW as operators consoles.

The following diagrams summarize the distribution of logical components on the different machines of the M&C subsystem. The logical components implemented on the M&C are as follows:

- ✓ SC (System controls). This component groups all functions which deal with system access, user management, sessions management, system supervision.
- ✓ DEX (real-time Data Exchanges). This component ensures all real-time data exchanges between the M&C and external subsystems. This includes ATV and ISS telemetry reception, ATV processed telemetry distribution and telecommands emission.
- ✓ TMAcq (TM Acquisition). This component checks all received telemetry packets and sends them to other components for real-time processing and database storage.
- ✓ TMProc (TM Processing). This components extracts parameters from telemetry packets (raw or physical data) and processed them (calibration, datation, derived parameters calculation, monitoring) and distributes them towards various clients.
- ✓ TC (TeleCommand Processing). This component allows preparing, sending and monitoring telecommands.

- ✓ TDB (Technological Data Base). This component manages all the data necessary for M&C components functioning: operational data (all data for real-time operations) and stored data.
- ✓ Data Handling (Data extractions and analyses). This component permits to extract data stored in the databank (telemetry, telecommand history, logbook).
- ✓ VisuTM (Telemetry visualisation). This component ensures telemetry visualisation and monitoring (monitoring definitions and management, alarms visualisation, processed parameters visualisation).
- ✓ MCW MMI. All user interfaces components which enable servers components management.



ATVCC M&C allows the control and monitoring of the ATV vehicle or its associated simulator. The architecture is based on two configurations, a nominal and a backup one. The nominal configuration is redounded whilst the backup is not. By default, the nominal configuration is used for the current mission. The backup will take over when the nominal configuration one is no longer operational (electrical failure, fire, etc.). The two configurations are located in different geographic areas.

Real-time operations (TM reception and TC emission, on the TTS) are independent from offline operations (data storage, data extractions and analyses, etc. on the ODBS).

The next paragraphs focus on two important aspects of the ATV-CC M&C architecture:

- ✓ The management of central TM flows (real-time processing of telemetry)
- ✓ The global architecture of the Data Handing (DH) component

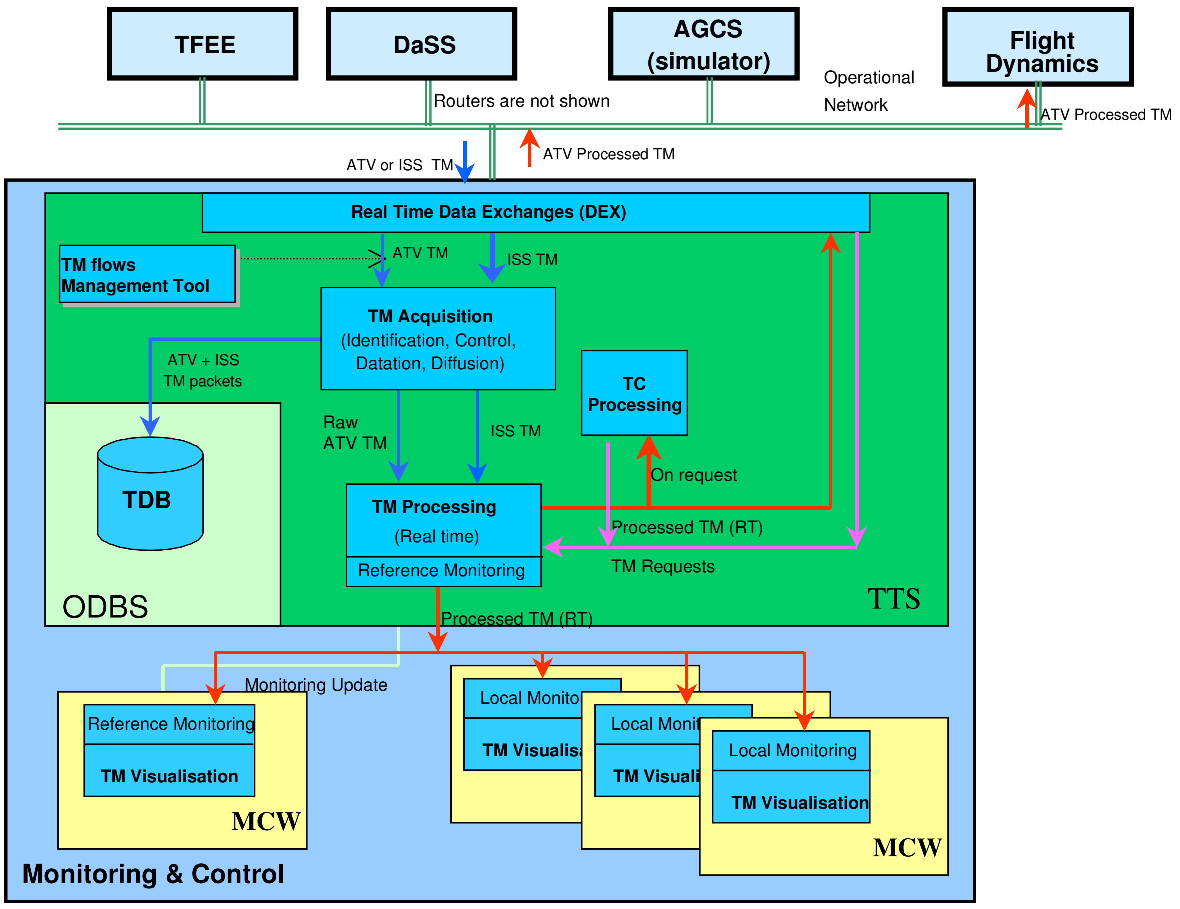
2.3.1.2.General TM flows diagram

This paragraph shows how the TM packets are managed when received by the DEX (real-time data exchanges) layer and when they transit through the different components that deal with them.

This diagram shows the M&C as a whole, inside which the main processes are identified. The physical components (TTS, ODBS, MCW) are not identified on this diagram. Nor are shown the physical elements of the network, such as PGL, IGS nodes, etc.

On this diagram, it is interesting to notice that:

- ✓ Several telemetry sources are received by the TTS server in real time: DaSS, AGCS (ATV simulator), TTFEE (Telemetry and Telecommand Front End Equipment).
- ✓ All external exchanges are realised by the DEX component, which manages all external connections.
- ✓ Under the responsibility of the TM flows management tool (SC component), it ensures the feeding of the TM Acquisition process. The received telemetry can be either raw ATV telemetry or ISS processed telemetry. The acquisition of ISS processed telemetry is made through requests, by the TM flows management tool.
- ✓ Although only one TM Acquisition process is shown, there are two such processes, one for each real-time TM flow. This component ensures the identification, the control and the time computation (time tagging) of entering packets, as well as their distribution. It also ensures the detection of TM holes.
- ✓ Raw ATV TM packets (basic, extended and delayed) and ISS processed TM packets are distributed towards the TDB, but no interaction occurs during real-time operations; the TM packets are stored locally on the TTS, on a dedicated directory. The TDB grabs the TM packets and stores them in the data bank; this operation is not time-related to the storage by the TM Acquisition process. This storage is mainly used for further replays (centralised or local).
- ✓ Raw ATV TM packets (basic and extended only) and ISS processed TM packets are distributed towards the TM processing tasks. Each of these tasks is dedicated to a centralised TM flow, real time or replay. This diagram shows only one TM processing task (dedicated to a real-time flow). However, there are two such processes, which are available to process all centralised TM flows. If two real-time flows are acquired simultaneously, the TM processing task dedicated to centralised replay switches to the second real-time flow.



- ✓ A reference monitoring is applied on the TM Processing tasks. On the TM Visualisation tasks (MCW), a local monitoring can be applied, that, at start, is identical to the reference monitoring, but can be modified in order to change monitoring rules dynamically and locally.
- ✓ Central TMProcessing ensures the extraction, the calibration, derived parameters calculation and reference monitoring. Processed ATV TM is made available to the TC processing task, that registers to the TM Processing task in order to receive only the required parameters (a priori / a posteriori controls, TTAG management, OMP management). Processed data distribution results from requests emitted by the potential clients.
- ✓ Processed ATV and ISS TM are also distributed in real-time to external subsystems (Flight Dynamics, DaSS, etc.) upon requests.
- ✓ Similarly, all processed TM parameters are distributed towards the MCW VisuTM clients, that are connected on the central real time flow. Users may also apply their own local monitoring.
- ✓ Only the “Monitoring” profile user may manually change the reference monitoring.

2.3.1.3.The Data Handling component

<u>Functional description of the Data Handling component:</u>

The DH component mainly serves as an interface to accessing telemetry in an off-line manner, in order to request data extractions or trend analyses. This interface is open to M&C users, and also to external subsystems (via the Data Gateway subsystems). Extractions are aimed at making fine investigations; trend analyses are aimed at giving synthetical views (sampling, statistics, etc.) of the data to the requesters.

The DH component is a web server. Users can create requests and specify extraction criteria and complementary processing. These requests can be reused. The results may be graphic displays, alphanumeric listings or numeric files. Requests are executed on a batch mode or not, and the resulting files are placed on a workspace dedicated to the user. Requests originating from external subsystems are monitored by the DH component so that security rules are enforced.

The DH component uses the following pieces of software:

- ✓ A web navigator located on each workstation (MCW),
- ✓ An HTTP server (Apache), that ensures the communication between the web navigator and the applicative part of the DH component,
- ✓ An applicative part, in charge of analysing the submitted requests, their transmission to extraction processes, the building and reception of result files and HTML pages generation (the MMI part of the DH component).

<u>Global software architecture of the Data Handling component:</u>

The Data Handling component consists of a Web server, which is composed of the three following main parts:

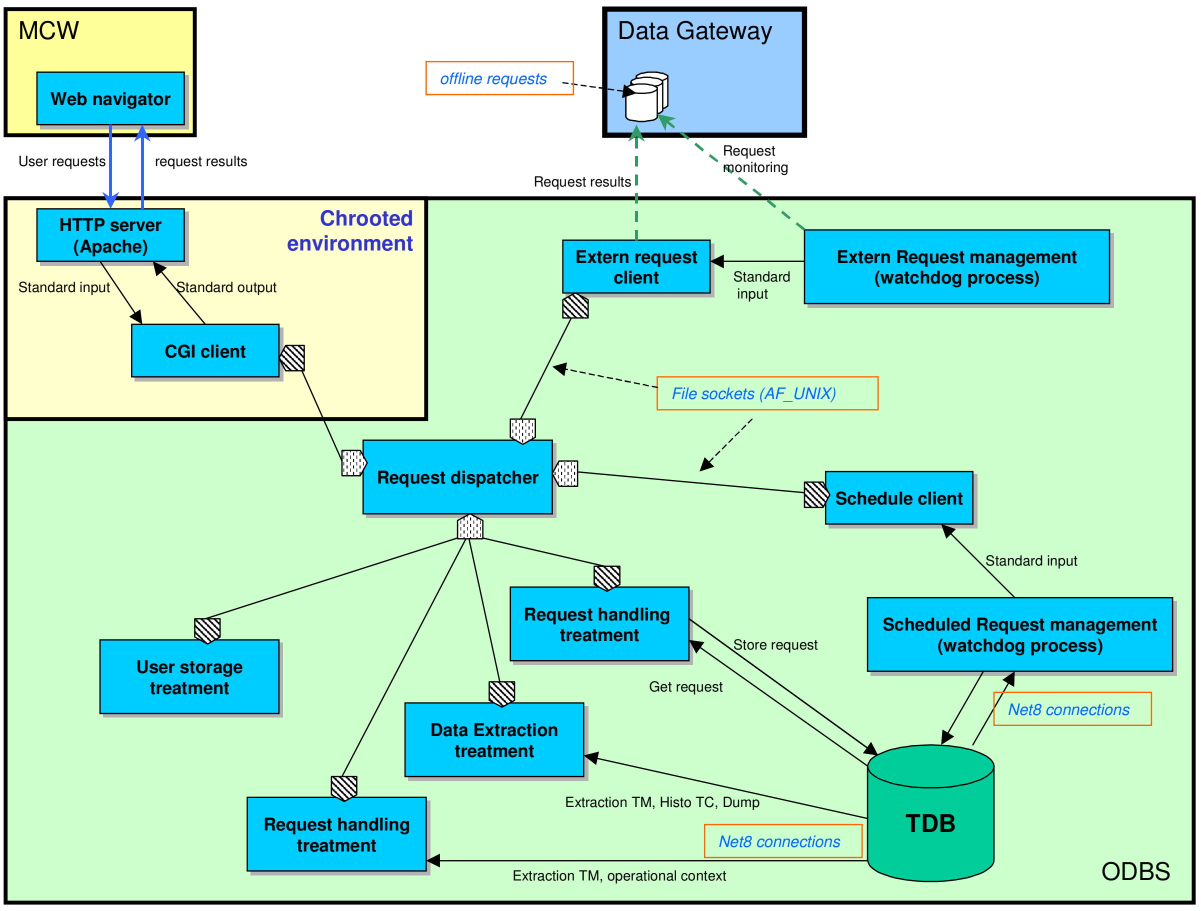
- ✓ On the MCW consoles :
    - a client application, which plays the role of the Web server MMI, which is a standard Web navigator.
- ✓ On the ODBS server :
    - An HTTP server (Apache software), which carries out exchanged data communications between the Web navigator and the applicative part of the Web server; this server more generally implements the HTTP protocol. Please note that no authentication is carried out by this server, since authentication mechanisms are taken into account by the MCW Manager and the SC components.
    - An applicative part, in charge 1) of the analysis of the requests submitted by the user, 2) of their transmission to the data extraction analysis processes, 3) of the production of result files, and 4) of the generation of HTML pages resulting of users interactions.

The applicative part of the Web server is connected to the TDB component (database) to extract the TM packets, which are processed as requested by the user.

It is important to notice that requests are handled in the same way, no matter from where they are issued. The proposed current architecture considers that:

- ✓ For online requests (issued from a Web navigator - for the M&C users), the requests rely on the HTTP protocol to be transmitted to the CGI client.
- ✓ For offline requests, a dedicated watchdog process is in charge of monitoring the deposits of requests on the Data Gateway subsystem. These requests are then inserted in a "CGI-like" format, in order to be processed by a dedicated CGI client, which is close to or even similar to the online CGI client.

The following figure describes the overall processes that compose the Data Handling component. The role of each process is not described here, since it is not the topic of this study.



2.3.2. Lessons learned

Please note that the ATV M&C project is not terminated yet. Therefore, lessons learned from this project are currently not complete. However, some topics have already been partly analyzed and are presented below.

2.3.2.1. Constraints and development approach used

Two main topics are analyzed below: the communication approach (and the M&C use of CORBA) and the architecture of the M&C web-based application (Data Handling component).

<u>Communication approach:</u>

Communications on the M&C subsystem must satisfy different constraints.

Real-time data exchanges, such as telemetry reception, acquisition, processing and visualization rely on TCP and UDP protocols, to guarantee high-level performance. Most offline data exchanges rely on FTP protocol for inter-servers exchanges and on Net8 (Oracle) for data storage.

ATV M&C uses CORBA for two reasons: messaging and interfacing. A first return on experience on ATV M&C concerns CORBA, and the way it has been implemented.

Notification Service:

Messaging relies on the notification service; all applications of the M&C need to send messages and alarms to a central logbook, in order to inform the end users of various events:

- ✓ functional events (telemetry anomalies, alarms on telemetry parameters, telecommands emission status, data flows management, etc.)
- ✓ application events (process supervision, changes in applications status, sessions management, etc.).

The Notification Service extends the existing OMG Event Service. The OMG (Object Management Group – the organization responsible for standardization of CORBA) event service defines three roles: the supplier role (applications generate event data), the consumer role (applications process event data) and the event channel (a dedicated application which encapsulates the semantics of messages queuing and propagation). On the M&C, the SC component manages the event channel.

Among the two general approaches for initiating event communication between suppliers and consumers (push model – a supplier may initiate the transfer of event data towards consumers, and pull model – a consumer may request event data from a supplier), the ATV M&C only uses the push model.

This service is a minimalist form of MOM (Message Oriented Middleware). It does not support auto-descriptive messages and transaction protection. However, it has been chosen since it answers to architectural constraints and needs of the M&C: simple messaging service, event type may be defined in order to group them for filtering purposes, messages may have a priority level (M&C defines two levels: priority level (used for alarms and notification messages) and normal level (used for log messages).

Naming Service:

The Naming Service is the CORBA service which allows objects installed on systems which hosts an ORB to localize other objects. This mechanism helps building powerful and flexible applications interfaces. It is particularly helpful when considering distributed applications.

The architecture of the M&C is not strongly distributed; in the sense that the localization of application processes is known and does not change from time to time. However, this service is adapted to the M&C needs. Moreover the fact that the architecture is not distributed facilitates the resolution of names (binding mechanism), and therefore is highly positive when performances are concerned.

ORB tools:

In order to implement the notification and naming service, the following ORB (Object Request Broker) tools have been chosen:

- ✓ On the TTS and ODBS servers (Sun Solaris, most applications developed using C++): **TAO**. This ORB has been selected for several reasons:
    - High performances. This tool is used on real-time projects (for example manned and unmanned real-time avionics mission computing embedded systems at Boeing/McDonnell Douglas, medical software framework for Siemens Medical Engineering, etc.) with strong constraints, such as those encountered on space control centers. This panel of implementations also serves as an argument for the validation status of TAO.
    - Supports a wide variety of operating systems (including Unix and Linux), which is a good argument for code portability.
    - Associated with the ACE framework (Adaptative Communication Framework), used as external library on the M&C
    - Developed in C++, which helps integrating M&C applications with the ORB.
    - Finally, it is free! The economic argument is part of the choice of a technical solution.
- ✓ On the MCW workstations (Windows 2000, most applications developed using Java): **OpenORB**.
    - TAO is not developed in Java. Since most applications are developed in Java, interfacing with an ORB not developed in Java has been considered as a major drawback (although technically possible).
    - OpenORB has been chosen among other ORB (JacORB, VisiBroker), but no detailed comparative study has been undertaken or consulted to discriminate between these ORB tools, except that :
        - i. OpenORB and TAO seemed to interface correctly (when considering notification and naming services)
        - ii. VisiBroker is not free
        - iii. OpenORB and JacORB seemed at the time of the choice to have comparative performance figures.

<u>Web-based component architecture:</u>

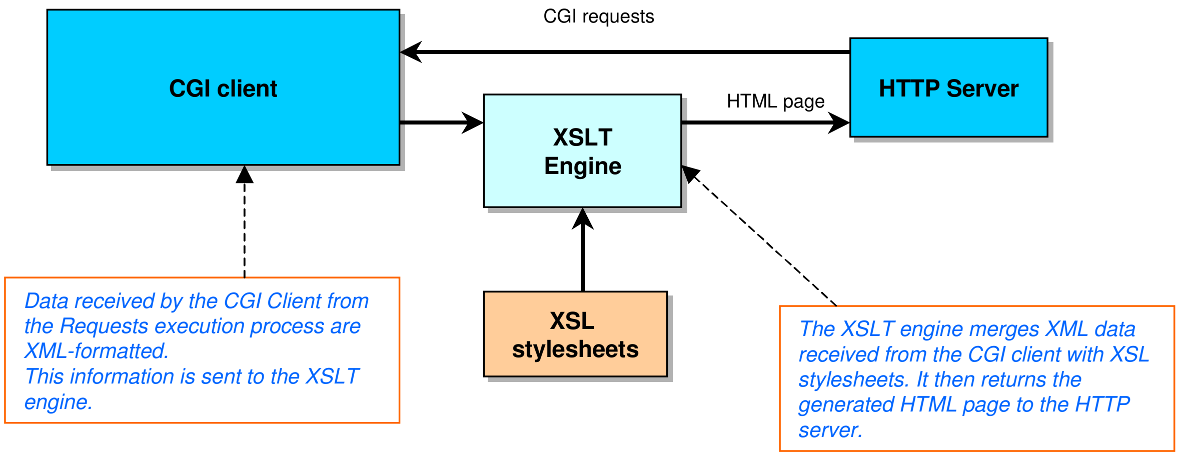
This paragraph focuses on the following topics:

- ✓ how HTML pages are built (using and XML / XLS approach)
- ✓ how security is enforced
- ✓ data extraction performance (due to the high volume of processed data).

How HTML pages are built:

Data which travel between Data Handling processes (from CGI clients to Data extraction processes) are XML-formatted. This makes HTML page generation easier and permits to distinguish data production from data formatting (HTML page).

This generation process relies on XSL style sheets, which contain HTML content (tags, predefined texts, predefined images, etc.) and XSL formatting objects which take care of visual presentation of XML data. The merge between data and presentation tags is realised by an XSLT engine, which dynamically produces HTML pages. These pages are returned to the HTTP server. The foreseen XSLT engine to be used on the Data Handling component is Xalan.


CGI requests
CGI client
HTTP Server
HTML page
XSLT Engine
XSL stylesheets
Data received by the CGI Client from the Requests execution process are XML-formatted.
This information is sent to the XSLT engine.
The XSLT engine merges XML data received from the CGI client with XSL stylesheets. It then returns the generated HTML page to the HTTP server.

Recom 25 : Use of technologies, such as XML / XSL / XSLT, which permits to distinguish between data structuring and data representation, is highly efficient to enhance maintainability of automatically produced HTML pages.

Note: *the direct use of the web navigator (Internet Explorer) as an XSLT engine is confirmed, and the realised tests have confirmed the technical solution. This solution has the advantage of reducing the network traffic, since the XSL pages are stored locally on the MCWs.*

Security:

Security is an important concern on the M&C. Several technical measures are implemented on the Data Handling component to enforce security requirements:

- ✓ CGI clients are executed in a virtual root ("chrooted environment"): only a portion of the hard disk is visible from the CGI applications, including the HTTP server. This ensures that all other applications running on the server hosting the web server are protected from malevolent external attacks. Moreover, database accesses are not possible from this restricted environment, thus protecting all stored data.
- ✓ all communications between Data Handling processes (except for the Apache server) are done using AF_UNIX sockets, i.e. "file sockets" which are protected using UNIX rights, therefore enforcing a strong security level for these communications. This also allows preventing from external illicit attempts to connect directly to Data Handling processes.

Recom 26 : Use of a "chrooted" environment, combined with standard rights access management (HTTP server security, files and directories access rights), is particularly well-suited when security requirements are applicable on a Web server, which happens most of the time.

Data extraction performance:

The main functions of the Data Handling component consists in extracting data stored in the database: telemetry data, telecommands history, logbook messages. For the last two data types, no performance problem is encountered.

However, extractions on telemetry data need to deal with very high volumes: one hour of telemetry approximately occupies 50 Mbytes in the database. Processing telemetry is therefore very time-consuming, memory-consuming and CPU-consuming. This constraint is not always compatible with user expectations.

Adequate indexing of telemetry data helps improving extraction speed, but does not the whole job. Usage of indexing criteria by users to limit the number of extracted data is appropriate, and currently available. But, although it helps, this solution does not guarantee satisfactory results.

Currently, alternate solutions are envisaged but not implemented yet. Solutions to limit data extraction to a strict minimum in order to satisfy functional requirements are studied and should permit to solve the current limitations.

2.3.2.2. PA methodologies and tools used

2.3.2.2.1. Estimating methods

Not specific.

2.3.2.2.2. Team training

Not specific.

2.3.2.2.3. Tools configuration

<u>TAO configuration:</u>

Configuration of the TAO ORB is quite complex and it has not been analyzed thoroughly on the M&C. However, the following choices have been made:

- ✓ The Notify Service allows for multiple worker tasks per dispatching proxy (clients connecting to it)
- ✓ The Notify Service allows for multi-threading dispatching; a configuration is set up to the number of dedicated threads which are used by the Notify Service to dispatch requests. In M&C case, only one thread is allowed at a time.

Since no specific performance problems have occurred so far, no further optimization has been undertaken.

<u>Apache configuration:</u>

Configuration of the HTTP server is not very complex. This tool is widely used, and support on how to configure it is easily found (particularly on the WWW). Moreover, CS has a strong experience on web servers implementation in M&C subsystems.

2.3.2.2.4. Life cycles V, V+, Incremental ,iterative

CS has followed a classical V life cycle in order to design and build the M&C subsystem. However, a prototyping activity has been undertaken from the beginning of the project on MMI, and particularly on the MCW plug-in structure. The MCW (M&C Workstation) is a plug-in structure which enables hosting of various MMI components. This structure proposes common services, such as logbook management, communication management (using CORBA naming services).

This prototyping activity has been very fruitful; it has allowed the end-users to validate this structure and the main ergonomics concepts implemented. Moreover, since several increments have taken place, more and more MMI applications of the M&C have been pre-validated before final delivery.

This prototyping approach is used for almost all requests for changes currently developed. MMI coding converges quicker and validation is rendered more reliable.

Recom 27 : Combining a classical "V life-cycle" with an iterative approach based on prototyping is strongly recommended, in particular when user validation of MMI or early design of complex algorithms is concerned.

2.3.2.2.5. Requirement Analysis

Not specific.

2.3.2.2.6. Architectural Design

Modelling method and tools:

The ATV M&C has been designed using the UML (Unified Modeling Language) formalism and in particular the UML/CS methodology. This methodology has been developed by CS in order to help projects use UML as efficiently as possible, in particular when considering classical V life cycle.

UML/CS is an iterative process based on UML. This process may be viewed as an instance of the Unified Process (proposed by creators of UML). This process consists in five phases: requirements analysis, object analysis, architecture design, object design and physical design. Each of these phases is not described here, but some return on experience may be given on the use of various UML concepts:

- ✓ Use cases: It is recommended to avoid describing use cases which group a set of functions, even if these functions are close to each other. This implies a greater number of use cases, but enables to have a granularity of description which leads more rapidly to efficient design results.
- ✓ Activity diagrams: it is recommended to privilege activity diagrams rather than use case diagrams. These diagrams help show different aspects of the functioning of the system (ex. nominal cases, degraded cases), such as:
  - Succession of use cases execution and triggering
  - Succession of use cases triggering through the description of internal activities (using the concept of « swimlanes »).
- ✓ Sequence diagrams:
  - May be replaced by activity diagrams when describing steps of a key process or when a great number of branches have to be described (in this case, sequence diagrams may include some ambiguities)
  - Activity diagrams permit a non ambiguous representation of successive messages in scenarios.

This return on experience is not intended to be exhaustive, but gives a first insight on modeling activities.

Recom 28 : It is recommended to combine the use of activity diagrams and use cases, since they provide efficient and complementary ways to describe internal activities of any use case of a subsystem.

2.3.2.2.7. Detailed design

Method and tools:

The same remarks may be made as has been said above on architectural design.

2.3.2.2.8. Testing

Test tools:

Two test tools have been used to cover unit testing activities: JTest (edited by ParaSoft), to cover Java unit testing (except all MMI testing), and C++Test (edited by ParaSoft), to cover C++ unit testing.

The JTest and C++Test tools have strong capabilities to perform regression tests. This implies that the initial effort be made to define unit test scenarios to cover different categories of tests, mainly functional and structural testing. However, these efforts to maintain the test scenarios are quite important, in particular when requirements change, which is the case on the ATV-CC M&C projects.

Moreover, these tools are platform specific, which implies several licenses and therefore more expenses for the project. This point is a drawback when considering the platform development of the M&C, which includes Sun Solaris servers and Windows 2000 workstations.

Test coverage:

No requirement is specified on test coverage, neither on unit tests, nor on validation tests. However, the validation strategy has been set up to guarantee that all requirements are tested by all the identified test cases. No specific return on experience has then been made on this particular topic.

Specific issues: load testing, etc.

No specific return on experience has then been made on this particular topic.

2.3.2.2.9. Integration/validation

Not specific.

2.3.2.2.10. Verification

Not specific.

2.3.2.2.11. Acceptance

Not specific.

2.3.2.2.12. Operation

Not specific.

2.3.2.2.13. Maintenance

Not specific.

2.3.3. Successes and Failures

As said previously, the ATV M&C project is not terminated at the time this study is written. Therefore, the return on experience is not complete. However, it is possible to state some preliminary conclusions on the topics studied:

Recom 29 : CORBA represents a very interesting communication middleware which can be used for non-distributed architectures. Highly efficient free tools are available to implement this middleware.

- ✓ **The "minimal" use of CORBA (only two services) is positive so far:**
  - It satisfies M&C architectural needs. Moreover, it is satisfactory when considering performance facts.
  - Although it is somewhat complex to implement initially, these efforts are satisfactory when considering the middleware functions taken into account by an ORB.
  - CS intends to develop its knowledge on CORBA, in order to use other CORBA services, such as life cycle service, persistent state service, concurrency control, security service.
- ✓ **The CGI based architecture is partly adapted to web server in the spatial field:**
  - The architecture is quite simple, and therefore relies on techniques in which CS has a strong expertise. In particular, security concerns are easily implemented as presented above.
  - CGI scripts are not easily adaptable and adding new functions is not always as easy as it could be when considering other technologies.
  - Difficulties are often encountered when generating HTML pages because Web browsers interpret the standards in a different manner. Therefore, it is tough to produce pages which are strictly homogeneous from a web browser to another. This is a drawback when considering that users have different web browsers.
- ✓ **Prototyping is an approach which is complementary to classical development methodologies. This approach has proven to have very positive effects on the development and acceptance process.**

2.4. PROTEUS/MYRIADE

2.4.1. Projects main characteristics description

The MMC architecture was designed based on the system's baseline requirements, the operational concepts and the interface requirements with the others MIGS ground segment components. The architecture was also designed taking into account the system cost-effectiveness, targeting an important cost reduction in terms of development, maintenance and operations cost.

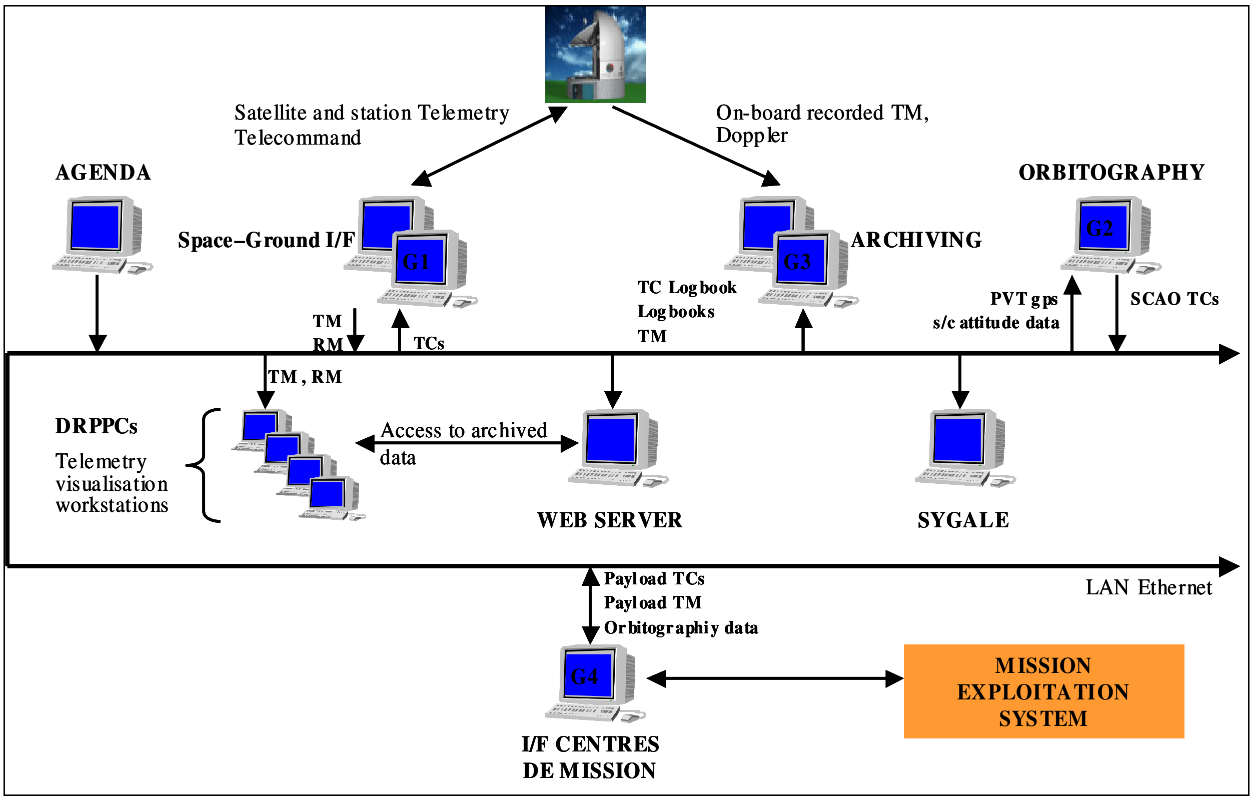
The MCC is based on a modular architecture split in several software components each one regrouping a logical set of features. This modularity enabled the parallel development of the different software components and thus optimising the project's development planning. Simple and well-defined interfaces between the different components have also ensured the success of the integration and validation phase eased it in a considerable manner.

Myriade main software components, which will be presented in the current section, are listed bellow:

- ✓ G1: Real-time TM/TC processing
- ✓ G2: Flight dynamics processing
- ✓ **G3: Off-line SCC data processing and archiving**
- ✓ G4: Mission exploitation system interface
- ✓ DRPPC: Real-time telemetry visualisation
- ✓ **WWW Server: User entry point to access archived data**
- ✓ Agenda: SCC operations automation
- ✓ Sigale: SCC alarm management

The architecture is heavily based on the JASON Control Centre (JCC), the strong reuse of JCC software components has enabled an important reduction of development cost as well as associated maintenance costs.

The closeness between the JCC and MCC, in terms of physical and software architecture and operations, enable the CNES to optimise the operations teams and thus reduce substantially the associated costs. In the next figure the Myriade Control Centre architecture is exposed:



The Myriade control centre is fully CCSDS compliant. The MCC is based on SUN SOLARIS, PC WINDOWS-NT and PC LINUX platforms. The interfaces between the different software components are based in an IP network, favoring the ASCII file interface and using FTP protocol. MMC has been developed using C, C++, Web Object, SL-GMS, Javascript, HTML.

Hereafter are described the 2 systems involved in the web application concerned by the project:

- ✓ G3 –data archiving software component
- ✓ SWWW –Data handling server

G3 –data archiving software component

The G3 - data archiving software component, is one of the core sub-systems of the MMC. In this section their main features and capabilities are presented.

Satellite monitoring, which is carried out off-line after a satellite pass and based on the HKTMR. The HKTMR packets are processed and the parameters extracted. Each parameter is then limit checked. After the parameters control, an overall status of the satellite is obtained and eventually alarms are detected and written in the logbook.

Station booking: after the orbit determination of all the satellites, which is performed by the control centre, G3 receives from the "orbit and attitude" component all the calculated orbital and terrestrial events. Based on this information, G3 creates the satellites pass plans, retrieves the ephemerid file for each pass and send the whole information to the different stations (TTCET and TETX).

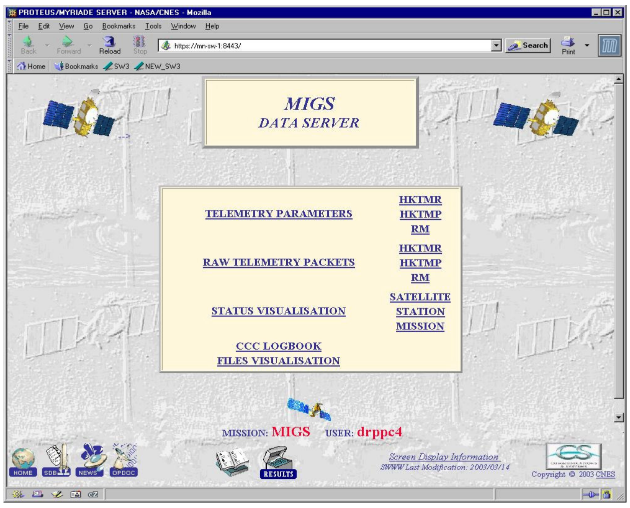
Archived data consultation: The archived telemetry, system logbook, TC logbook, can be accessed either by dedicated control centre workstation or by a web server. G3 receives the requests, which define the data type (telemetry parameters, raw TM, logbooks…) and the selection criteria (satellite, timeframe, parameters list…). G3 processes these requests and supplies the results in files that are made available to the user.

The G3 - data archiving component is also in charge of:

- ✓ Station retrieval of on-board recorded TM (HKTM-R) and ranging measurements,
- ✓ G1 retrieval of real-time satellite telemetry acquired during a satellite pass (HKTM-P) as well as the stations (TTCET et TETX) telemetry (RM),
- ✓ Control Centre data retrieval: system logbook, TC logbook, orbit & attitude data…,
- ✓ Raw telemetry (HKTMP et HKTMR), station telemetry (RM) and control centre data archiving,
- ✓ Retrieval, from the HKTMR, of all the data required by the "orbit and attitude" component (GPS data – speed and positioning, attitude data),
- ✓ Retrieval, from the HKTMR, of the memory dumps requested by the TCs.

<u>SWWW –Data handling server</u>

The Data handling component of the Myriade control centre is a web server allowing the user a wide range of features for retrieving and analysis of the system archived data. For each mission and during the entire satellite life cycle the SWWW allows:



- ✓ To create and manage the user's requests for accessing the MCC archived data (TM and RM parameter values, TM or RM raw packets, system logbook, TC logbook…),
- ✓ To visualise/display the data requests results delivered by the G3 component,
- ✓ To visualise/display the satellites an stations synthetic states,
- ✓ To visualise the information stored in the satellite database (parameters list, decommutation plans, telecommand formats, monitoring functions…),
- ✓ To access the project's operational documentation.

2.4.2. Lessons learned

2.4.2.1. Constraints and development approach used

The main constraints concerning the web server were:

- ✓ The web server must be acceded by

Operators from the Control Center room,

Experts during the launch phase, from the Experts room, Authorized user, from any other room (Example: CNES agent office ...).

- ✓ The web server must be compliant to CNES security standard.

2.4.2.2. PA methodologies and tools used

2.4.2.2.1. Estimating methods

Not specific

2.4.2.2.2. Team training

Not specific, only an OBJECT WEB tool training was organized at the beginning of the project.

Recom 30 : A specific training must be organised at the beginning of the project for each used technology.

2.4.2.2.3. Tools configuration

The default OBJECT WEB tool configuration was used.

2.4.2.2.4. Life cycles V, V+, Incremental ,iterative

V Cycle with MMI prototyping.

2.4.2.2.5. Requirement Analysis

Not specific

2.4.2.2.6. Architectural Design

A top down and textual approach was used. Design and development are based on modularity principles and well-defined interfaces, the control centre is extremely scalable for very reasonable costs, which implies reducing maintenance costs and thus a significant decrease on exploitation costs.

2.4.2.2.7. Detailed design

Not specific

2.4.2.2.8. Testing

Not specific

2.4.2.2.9. Integration/validation

Not specific

2.4.2.2.10. Verification

Not specific

2.4.2.2.11. Acceptance

This project was organized with different plat-forms:

- ✓ A MYRIADE complete platform used by CS team to validate the system at CS site,
- ✓ A MYRIADE complete platform used by client team to validate the system at client site.

These 2 validations are made before the delivery/installation.

2.4.2.2.12. Operation

During the launch phases the server can be used simultaneous by 40 users with good performances.

2.4.2.2.13. Maintenance

Not specific, only a comment: “very small numbers of maintenance interventions during operational phase”.

2.4.3. Successes and Failures

- ✓ Technology used is now obsolete (difficulties to maintain the Object Web competences) but the application is nominal and compliant with the initial specifications,
- ✓ Performance

Reduce archived data access delays, significant real-time displays capabilities, fully automatic operations capabilities.

Note: *This first use of a web server in a CNES control centre is a real success.*

Recom 31 : An operational Control Center must include a web server to provide an easy access to the data

- ✓ Object Web is a choice made at the beginning of the project (1998), the criteria for this choice were:

Do not use the CGI (Object Web provides an alternative solution),

Easy development of dynamic HTML pages (Object Web provide mechanisms)

Note:

Since, the server core has been replaced by an Apache server (the objective of this change is to improve the maintenance phase), we the configuration is now:

- ✓ *Apache and Object Web Core, both running on the server*
- ✓ *Object Web Core and the Client applications (written in Objective C a proprietary language of Object Web), both running also on the server*

Important: *The full Object Web solution is equivalent of a mixed Object Web/Apache solution*

- ✓ The constraint: “web server must be acceded by Operators from the Control Center room” has been reached.
- ✓ The constraint: “web server must be acceded by Experts during the launch phase, from the Experts room”, has been reached, but by a specialized communication line, instead of a public secure access.

- ✓ The constraint: “web server must be acceded by Authorized user, from any other room (Example: CNES agent office ...)” and “web server must be compliant to the security CNES standard” has not been reached.

Note: *This is due to the Object Web and objective C language usage. Using this tool and language is not compatible with the security CNES standard*

One can also note as an important thing, that:

- ✓ to buy, upgrade and maintain Object Web represents an important cost,
- ✓ the Object Web framework (example: to generate dynamic pages) is interesting but limited, the advantage to use it is minimal,

- ✓ the European distributor is missing now, and the future contacts will be made directly to the USA companies.

IMPORTANT: For this reasons the new PLEIADES Control Centre Web server is developed with the same specification but with a PHP architecture instead of Object Web.

Recom 32 : Use open source technologies instead of proprietary solutions o develop this kind of application.

2.5. RSB

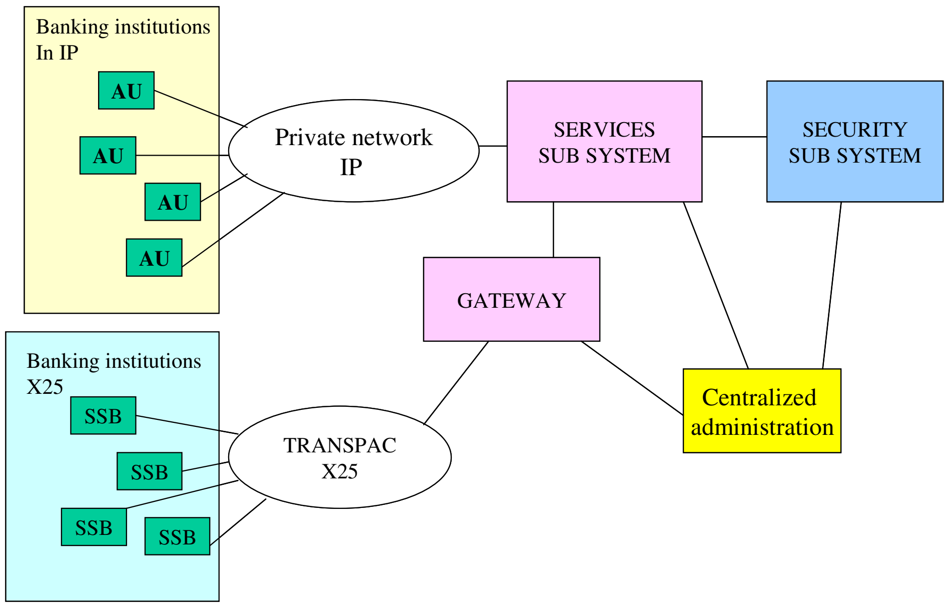
2.5.1. Projects main characteristics description

The purpose of the RSB network is to provide authorisation services for credit card operations when these operations require treatments from two banking institutions. The different kinds of operations concerned by this project are:

- ✓ Payment operations with credit card
- ✓ Cash withdrawal from ATM
- ✓ Credit cards management operations (sending/receiving opposition lists, information exchanges between bank institutions …)

Note: Security management by exchanges of cryptographic keys between the bank institutions and the RSB network.

2.5.1.1.General project architecture



The following relationships must be offered by the system:

- ✓ An IP requester with an IP server
- ✓ A X25 requester with an IP server
- ✓ An IP requester with a X25 server
- ✓ A X25 requester with a x25 server

Notes:

- ✓ User configuration files and system versioning managed at centralized administration level.

- ✓ Duplicate subsystems to prevents service stop

2.5.2. Lessons learned

2.5.2.1. Constraints and development approach used

- ✓ EAL4 Certification (Security subsystem),
- ✓ User configuration update (must be made without stopping the system and without connection lost with the banking institutions),
- ✓ Transaction Performance (Acquire request, send this request, acquire answer, send answer plus the security controls –signatures-) must be made with a limited time (< 1 second),
- ✓ Number of transactions (4 billion) must be move to 10 billion in a close future (only with HW adding).

2.5.2.2. PA methodologies and tools used

2.5.2.2.1. Estimating methods

Not specific.

2.5.2.2.2. Team training

The system must be STUR – "Spécifications Techniques d'Utilisation du Réseau" - (the client API description) compliant; the developer team was trained to STUR specification plus UML, C and C++.

2.5.2.2.3. Tools configuration

VISUAL AGE, CLEARCASE and PVCS are used with the default provided configuration.

2.5.2.2.4. Life cycles V, V+, Incremental ,iterative

Rapid Application Development (RAD) is used for the Administration Sub System; for the others parts of the system incremental cycle is used.

2.5.2.2.5. Requirement Analysis

Not specific.

2.5.2.2.6. Architectural Design

The project use MIL STD498 norm.

- ✓ At system level: production of SSS – "System Subsystem Specification" -, SSDD – " System Subsystem Design Description" - and validation tests documents,
- ✓ At sub system level: production of SSS, SSDD (with a detailed architectural design by SSDD –with the first modules description-), and integration tests documents.

All the approach is "requirements based", with a very complete traceability from the requirements to SSS System, SSDD System, SSS Sub System, SSDD Sub System, detailed architectural design and tests.

2.5.2.2.7. Detailed design

At this stage, a detailed description for each module is provided.

2.5.2.2.8. Testing

Test tools:

- ✓ electronic banking protocols (application based) simulators

<u>Test coverage:</u>

- ✓ Use of DOORS tool (One Requirement = One test)

<u>Specific issues: load testing, etc.</u>

- ✓ Not specific.

2.5.2.2.9. Integration/validation

Not specific.

2.5.2.2.10. Verification

This project was organized with different teams:

- ✓ Development team,
- ✓ Administration team: verification and delivery/installation (2 versions by year).

2.5.2.2.11. Acceptance

This project was organized with different platforms:

- ✓ A RSB complete platform used by CS team to validate the system at CS site,
- ✓ A RSB complete platform used by client team to validate the system at client site.

These 2 validations are made before the delivery.

2.5.2.2.12. Operation

The operational computers are in hid rooms (without human presence). They are only acceded by the administration system. The Administration system software produces metrics used by himself for an automatic protection. The administration system is very SECURED.

2.5.2.2.13. Maintenance

The maintenance must accept easily additional or updated functionalities. Indeed the electronic banking protocols are updated each year and these evolutions must be planed.

2.5.3. Successes and Failures

- ✓ Transfer to the new system was made without problems, and in transparent manner for the user, (the technical approach uses gateways to hide the actual systems – legacy one or new one).

Recom 33 : Use gateways for a progressive and secure migration of an operational distributed system.

Note: *Transfer from X25 to TCPIP (better performances with TCPIP, but the error processing becomes ineffective – the errors are detected, but later than X25 -). To guarantee the same reliability the team has produced additional code added to TCPIP.*

- ✓ The MIL STD498 usage was very efficient (recommended documents, with recommended templates …),

Note: *UML was used on the SSDD document (System scenarios definition).*

- ✓ Good usage of the electronic banking protocols defined by ISO8583 norm,
- ✓ Good usage of the KANEST simulator tool (user simulator STUR compliant) to made performances measures (this is a very specific tool),

- ✓ Specific synchronizing TCP/IP protocols, was developed,
- ✓ Choice of the best subsystem instance to process the transaction,

Note: *this system is build by about 1000 000 of C/C++ instructions.*

2.6. PAC-LUX

2.6.1. Projects main characteristics description

The PAC LUX tool is a software (developed for the Agriculture Ministry of the Grand Duché of Luxembourg) dedicated to the graphical PAC (Politique Agricole Commune) declarations processing; this software includes the next main services:

- ✓ PDF documents production,
- ✓ Support for administrative control,
- ✓ Support for control board activities,
- ✓ Agro-environmental objects management.

Note:
The principal PAC LUX constraints are: (1) the system must implement a new geographical reference for 2006 declarations: "the agricultural land parcel"; (2) The system must respect the functional and technical constraints of the CIE (Centre Informatique de l'Etat) and must collaborate with the existing AIACS (Integrated Control System).

The used technologies are:

- ✓ Architecture & executing platform J2EE

IBM WEBSPHERE Application Server

- ✓ Operational platform

IBM mainframe and SUN/SOLARIS & LINUX servers,

- ✓ Development language

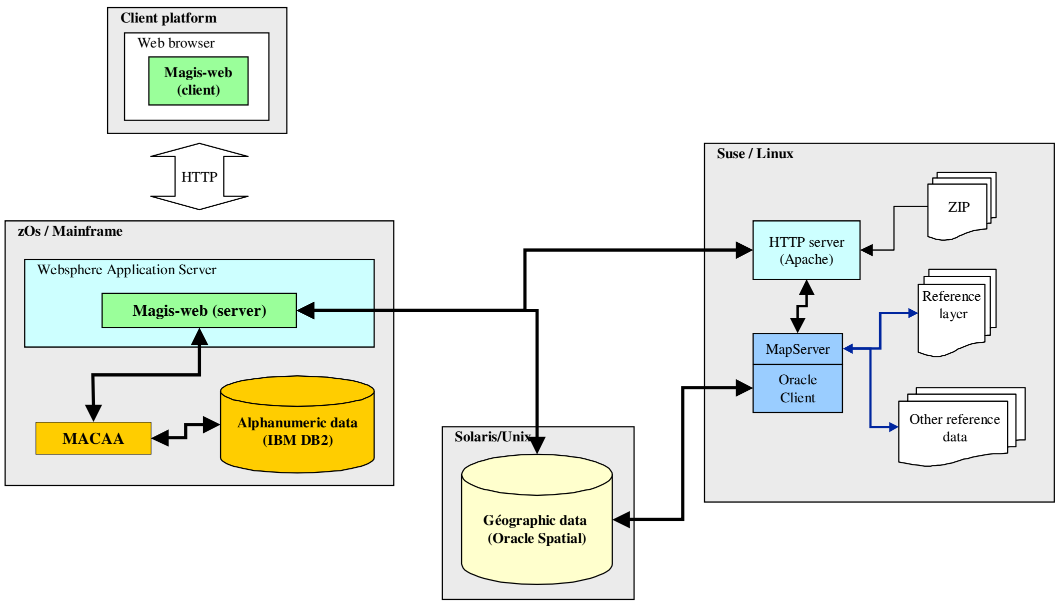
JAVA,

- ✓ Geographical data base

ORACLE Spatial,

- ✓ Map server

MAPSERVER



<u>Client constraints:</u>

The target platform of the MAGIS application is a websphere server running under zOS.

The client has imposed the utilization of its toolkit to manage logs, exceptions and configuration. He had also made some technical choices such as prohibiting the use of other EJB than stateless EJB.

2.6.2. Lessons learned

2.6.2.1. Constraints and development approach used

The members of the team in charge of communication with the client have developed a model using HTML to discuss with him. This model was maintained during development phase.

The development process was based on use-cases. From the UML model, each developer

- ✓ created JSP pages (HTML + javascript + Struts tags)
- ✓ developed Struts actions,
- ✓ defined EJB interfaces and develops domain classes (java and PL/SQL)

This organization requires good team communication (four developers). A supervisor has to insure the coherence of the resulting code.

2.6.2.2. PA methodologies and tools used

2.6.2.2.1. Estimating methods

Not specific.

2.6.2.2.2. Team training

- ✓ Websphere training and Websphere administration training,
- ✓ Oracle training,
- ✓ Frequent recourse to specialists: ORACLE, Java, Mapserver, Kaboum...
- ✓ SIPAD team experience feedback (Struts, J2EE).

Recom 34 : Identify experts in each technology at the beginning of the project.

2.6.2.2.3. Tools configuration

Realized by development team, and Client team support for dataSource setup in Websphere environment.

2.6.2.2.4. Life cycles V, V+, Incremental , Iterative

3 increments have been defined:

- ✓ Increments 1 & 2 (console applications): classical cycle V.

Development of a prototype (consultation use-case) for rise in competence and integration test on the target platform.

- ✓ Version 3: iterative development based on reusable packages and HTML model. New functionalities where added progressively.

2.6.2.2.5. Requirement Analysis

Not specific.

2.6.2.2.6. Architectural Design

- ✓ Identification of layers,
- ✓ Identification of components (repartition by layers).

2.6.2.2.7. Detailed design

- ✓ UML: Rose/Soda + Word + Struts diagram generator integrated in WSAD.
- ✓ Power Designer.

2.6.2.2.8. Testing

- ✓ A test project was created for each layer :
    - Console application for the test of the classes in charge of database access (PL/SQL functions).
- ✓ Web project in Websphere for EJB testing,

Recom 35 : Provide ways to test quickly a complex layer/component.

2.6.2.2.9. Integration/validation

A prototype has been developed and deployed on the target platform during the detailed conception phase. This action was made to reduce integration problems involved by the lack of target platform on the development site. Integration has also been simplified by the use case driven development.

Recom 36 : Anticipate installation/integration problems if the target platform is not available.

2.6.2.2.10. Verification
Not specific.

2.6.2.2.11. Acceptance
Not specific.

2.6.2.2.12. Operation
Not specific.

2.6.2.2.13. Maintenance
Not yet.

2.6.3. Successes and Failures

✓ Development by functionality (Use case driven) versus development by layer

- Development was organized by functionality: each developer is in charge of a use-case from user interface to PL/SQL functions. This choice implies a good communication in the team. Every member must know several development languages.
- To avoid redundancies and insure that the design choices are respected, the project manager has a view of the whole development process and a member of team is designated to be the reference for the user interface layer, another for the database access layer.
- This organization was a success with a small team (4 members). The maximum number of developers for this organization could be evaluated to six.

Recom 37 : Use a use-case driven approach with small teams located on the same site (communication is vital).

✓ Distributed application benefits

- One of the major risks identified was the access to the data stored in the client DB2 database and the safety constraints it involved. Distributed architecture made it possible to circumvent the problem: a single parameter setting made it possible either to question base MACAA, or to use a copy of the data stored in an Oracle base. This solution has permitted to respect the strong constraints related to the production of pre-printed documents.
- To improve the production of pre-printed documents, the application magis-editions was launched on height computers in parallel which addressed to two instances of mapServer.

✓ Database access : PL/SQL

- The choice of PL/SQL functions has permitted to improve the performances of database access by a database specialist without any modification of the java code.

Recom 38 : Reduce the use of SQL code in the domain layer's code.

✓ Test strategy by layer

- When the application was integrated in the client framework, the possibility of testing successively the layers was a great benefit.

- ✓ Experts intervention
  - The help of experts in different areas has been really appreciated: database, design choices, java libraries… The availability of experts in the client team was a good point for the success of the project.

- ✓ Technologies
  - The integrated Websphere environment made it easy to develop J2EE application and to define the web server settings. The CVS plugging for Websphere was not used because of identified bugs (the fact that this point was not analyzed further may be considered as a failure).
  - Java offers possibility of getting free libraries (iText for PDF documents generation).
  - No Struts tag-lib was developed. Java coding in the JSP pages could not be avoided.

Recom 39 : Choose the programming languages accordingly to open-source components availability.

- ✓ Javascript versus actions
  - The team had had an important discussion about the use of javascript for domain code development. On one hand, javascript permitted a fluent user interface, that is the reason why this solution was adopted. On the other hand, java code would have been much more comprehensible and maintainable.

Recom 40 : Reduce the use of dynamically generated code (Javascript code generated by servlets).

Recom 41 : Reduce the implementation of complex domain logic in the presentation layer.

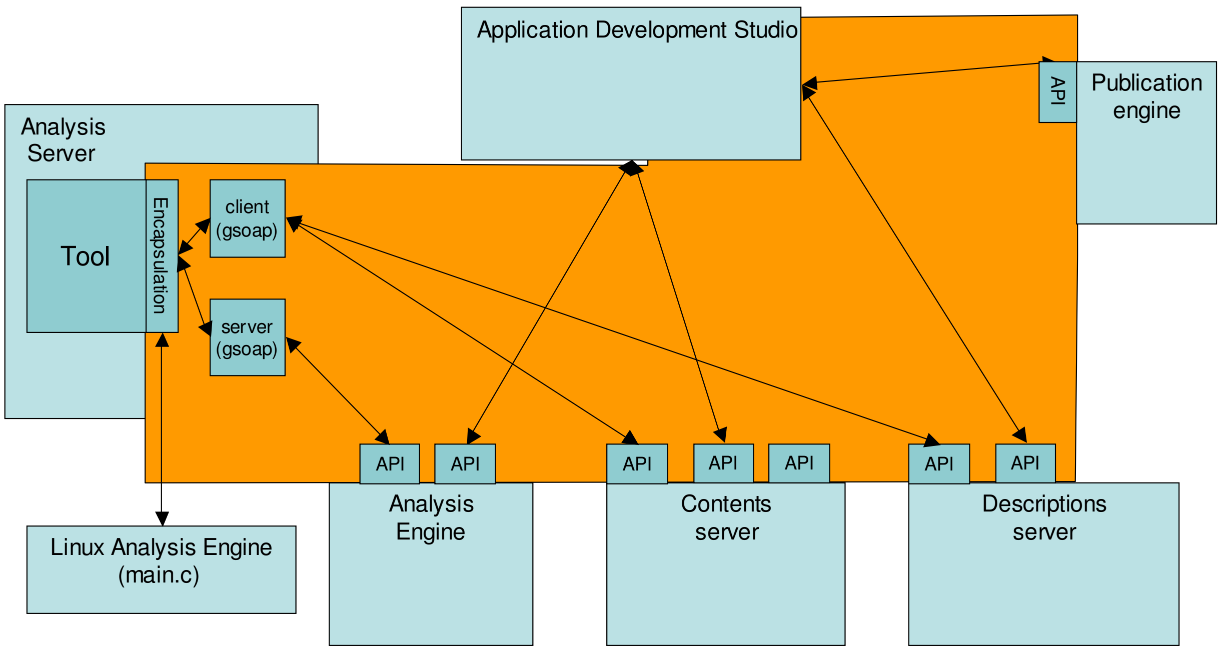
2.7. FERIA

2.7.1. Architecture overview

FERIA stands for Framework for Experimentation and Industrial Realisation of Multimedia Applications. This framework is composed by:

- A set of dedicated servers (content, description, analysis, publication)
- A set of .NET classes
- A set of tools to accelerate the use of these components with Visual Studio .NET 2003.

Each component of the framework provides services that can be used to generate an application based on them. When the application is defined, it is exported via the publication engine as a web-site or a stand-alone application.



The architecture of FERIA is highly configurable, the components exists in two states: an independent server accessed via the network or a standalone executable deployed locally on a machine.

This particularity has been chosen to provide different configurations of the framework based on functional user needs and on capacity of the target platform (Single PC or computers on LAN).

2.7.2. subsystems breakdown

Application Development Studio

This component is a set of tools that ease the access to others components of the framework. It also provides additional services like user/profiles management and access to the administration API of each component of the framework.

Analysis engine

This engine is used to order and parameterize analysis tools execution. It is based on analysis schemas.

Analysis server:

This component encapsulates an analysis tool which realizes processing on multimedia content. These processes can be : scenes detection, faces detection, voice recognition, … It produces a description of the multimedia content.

Content server

This component provides an access to a repository used to store multi-media content (Video or Sound).

Its API provides services to add new content (encoding / adding) and to retrieve content (Loading, streaming, …)

Description server

This component manages all requirements relative to media description, it can store all metadata relative to a multimedia content: sub-titles, scenes, time-slices, participants, indexed content,…. This component also provides services to retrieve the description in various formats, manual edit and validates them or to query these descriptions.

Publication engine

This component is used to generate an independent product based on the services of the framework's components. The product can be a DVD, a website, a program that runs on set-top-box (receivers of satellite TV).

2.7.2.1.COTS environment

X-HIVE: XML Database

Tomcat + AXIS: Webservice server to encapsulate X-HIVE

IIS: Microsoft Web server

.NET: Webservice framework for windows platforms

GSOAP: Webservice framework for Linux platforms

Windows Media service: Tools provided to manage multimedia data on windows environments

2.7.2.2.Access to functions

Webservices

2.7.2.3.Windows environment

Linux (Redhat 9.x)

Windows XP

Windows Server 2003

2.7.3. Lessons learned

2.7.3.1. Constraints and development approach used

The services implemented by the components must work in standalone and in remote mode.

- The standalone version is an independent tool that works with a GUI or with command-line parameters
- The remote version is a server that exposes all its services via a network connexion.

The components are numerous, developed with heterogeneous languages and deployed on heterogeneous platforms.

These constraints have been satisfied using a webservice layer over these components.

2.7.3.2. PA methodologies and tools used

2.7.3.2.1. Estimating methods

Not specific.

2.7.3.2.2. Team training

Webservices development with .NET

2.7.3.2.3. Tools configuration

Realized by development team.

2.7.3.2.4. Life cycles V, V+, Incremental ,iterative

Iterative

2.7.3.2.5. Requirement Analysis

Not specific.

2.7.3.2.6. Architectural Design

The architecture has been imposed by the client.

FERIA architecture has been developed to integrate existing components on an existing architecture. The architecture was imposed by the data location and the data format.

2.7.3.2.7. Detailed design

Identification of components (Location, services provided, …).

The number of components has led the team to enforce the definition of component's interfaces. With this approach, the development team have been dispatched by component.

A first prototype has been used to validate design choices and to validate the use of GSOAP for communications between Windows and Linux platforms via webservices.

A second prototype has been used to present the GUI to the client.

2.7.3.2.8. Testing

Test tools

Simulators based on components interfaces have been developed for this purpose.

2.7.3.2.9. Integration/validation

This step has been eased by the strong interfaces design.

2.7.3.2.10. Verification

Not specific

2.7.3.2.11. Acceptance

Not specific.

2.7.3.2.12. Operation

Not specific.

2.7.3.2.13. Maintenance

Not specific.

2.7.4. Successes and Failures

Components interface:

Interfaces of each component have been identified and detailed. This work has drastically reduced the problems of integration and allowed to write realistic simulators for each component.

Recom 42: In an environment based on numerous communicating components the interfaces definition is a critical point that must be seriously managed during detailed design.

Use of webservices

The use of webservices have been a real success to share data provided by heterogeneous components spread over heterogeneous platforms.

Use of COTS

The GSOAP COTS and multi media data visualization COTS have been evaluated and tested using a very simple prototype. The team has lost a lot of time due to the poor maturity and the lack of documentation on this product when they tried to use it with real data.

Recom 43: Critical COTS must be accurately evaluated with one or more prototypes based on the same kind of use than the final application.

Webservices with .NET

The .NET framework is well integrated and provides adapted tools to works in a webservices architecture : a lot of work is automated by graphical tools. Webservices have been developed faster under this environment that on any others (Tomcat/Axis and GSOAP).

2.8. Astrium CC

2.8.1. Projects main characteristics description

2.8.1.1.SCC environment

Astrium provides complete Satellite Ground Control Systems (SGCS) with full redundancy, including equipments to fully control and monitor one or several satellites.

The SGCS consists generally of two redundant control systems to be installed at a Primary Control Facility (PCF) and a Secondary Control Facility (SCF) located in a different location. Both PCF and SCF infrastructure are identical and are able to operate satellites whether using PCF or SCF ground stations.

The SGCS main functions are:

- Satellite monitoring and control
- Station configuration, monitoring and control (includes ranging processing)
- Satellite simulation

Part of the ground system is the Satellite Control Centre (SCC), dedicated to the monitoring, control and data archive management for all satellites.

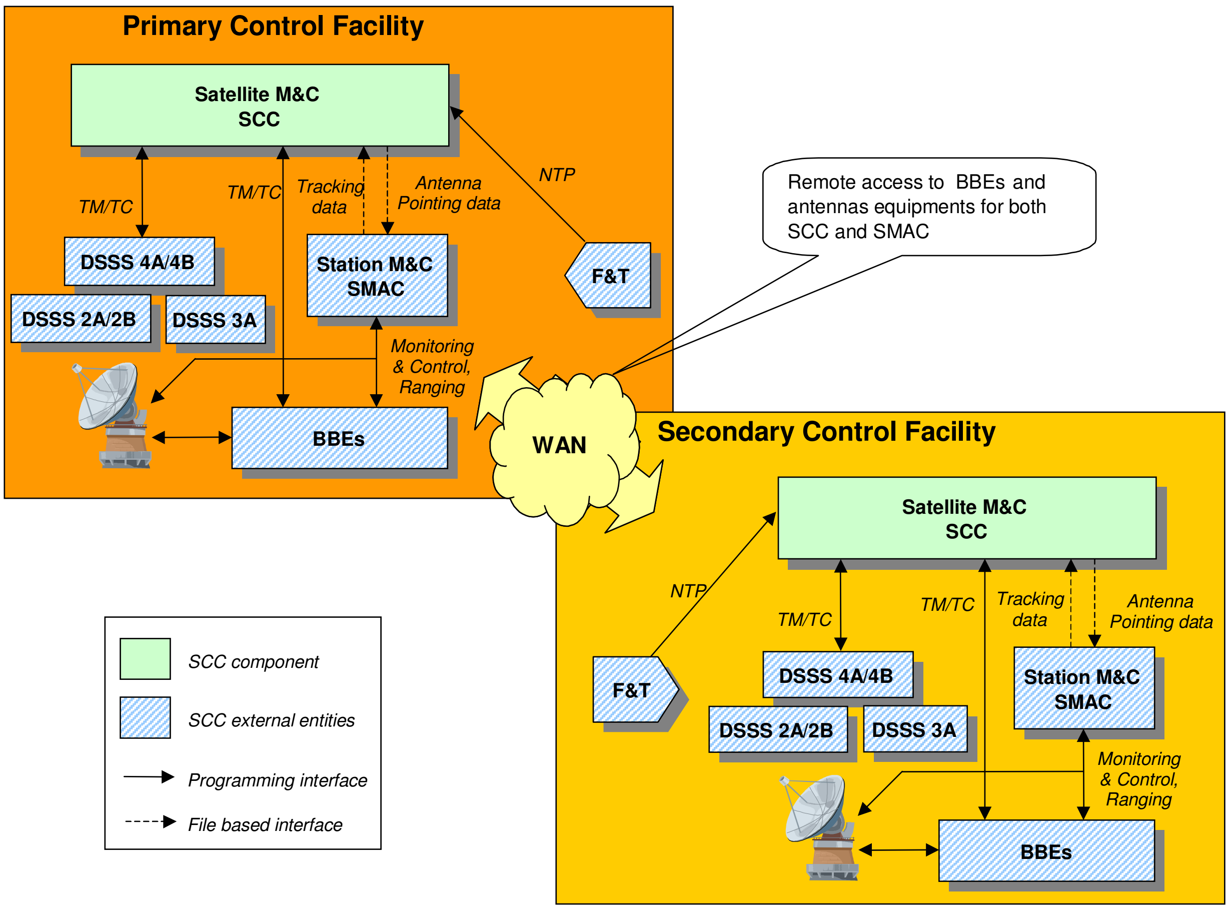
The SCC core functions are:

- System database management
- Flight Operation Procedures (FOP) preparation and execution
- Telemetry and command processing

- History filling and retrieval
- Flight dynamics processing

The technical solution is mainly based on the OPSWARE product line, developed by EADS Astrium and on the hifly^{TM} product from GMV, S.A (based on ESA SCOS 2000).

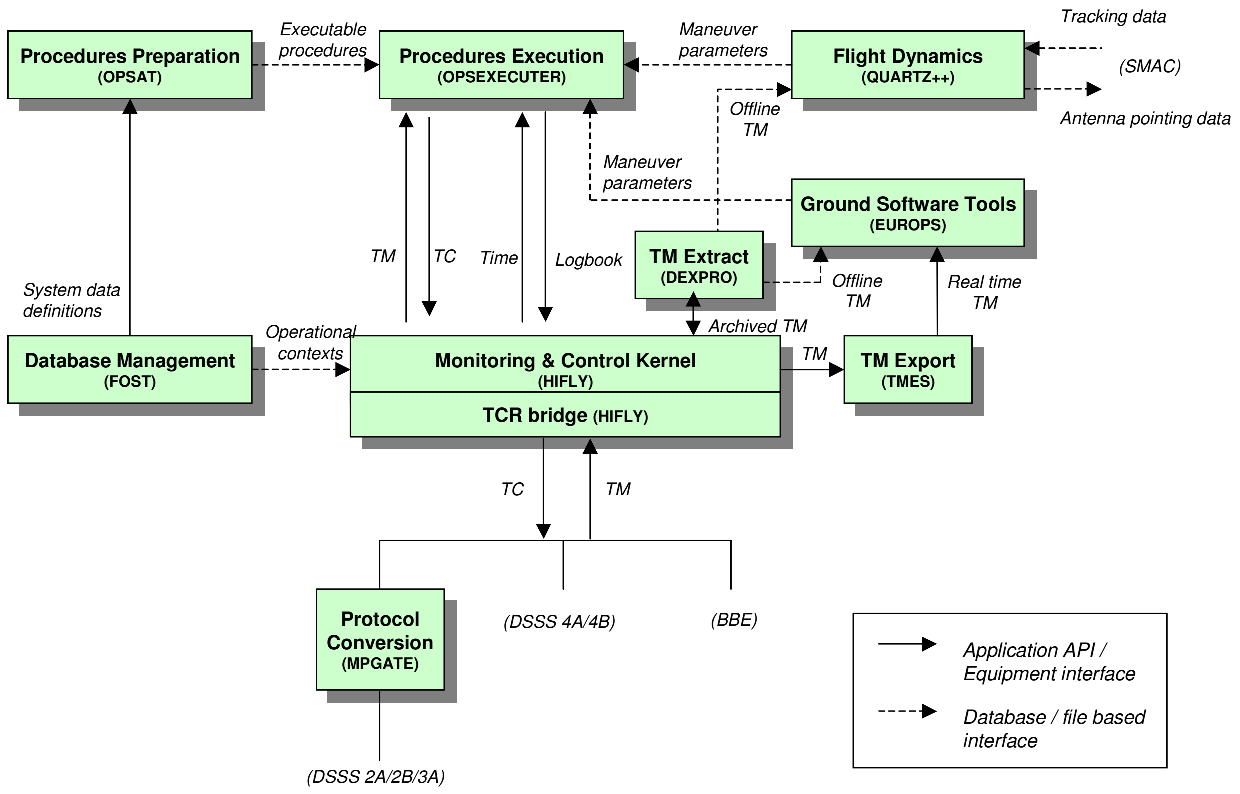
The following figure shows the SCC in its operating environment. SCC external entities are mentioned for a better understanding, but are not detailed in this document.



2.8.1.2.SCC software architecture

The SCC is built based on the existing EADS Astrium OPSWARE products, and GMV hifly^{TM} product. These core products are already flight proven in various environments and are customized as necessary to meet the customers' requirements.

2.8.1.3.Architecture overview



2.8.1.3.1. Subsystems breakdown

The components involved in the above described architecture can be classified in three categories:

- online functions, involved in real-time processing phases
- offline functions, involved in preparation or post-processing phases
- Flight dynamics, as a specific standalone component not directly interfacing the SCC, but exchanging data necessary to real-time operations such as maneuvers parameters.

Monitoring & Control "online" functions	Monitoring & Control Kernel (HIFLY)	Core TM/TC processing, archiving and SCC centralized services
	Procedures Execution (OPSEXECUTER)	Automated execution of operation procedures
	Real-time Telemetry Exportation (TMES)	Remote access to real-time telemetry information
	Multi-Protocol Gateway (MPGATE)	TM/TC protocol conversion
Monitoring & Control "offline" functions	Database Management (FOST)	System database management and operational contexts generation
	Procedures Preparation (OPSAT)	Procedures edition, verification and formatting
	Data Extraction & Processing (DEXPRO)	Archived data retrieval and trend processing
	E2000+ Operations Applications (EUROPS)	Operational data processing
Flight Dynamics	Flight Dynamics (QUARTZ++)	Orbital operations computation

Hereafter is a brief description of the functional scope of each component. A more detailed presentation is available in the following paragraphs.

- hifly^{TM} product
  - Commands processing, including commands formatting, queuing and transmission via base band equipment, and related processing verifications via analysis of returned telemetry parameters values
  - Real-time telemetry processing and monitoring, including direct, derived and dwell parameters, alarms management and related user oriented warning features, replay capability
  - Online database management for TM/TC processing data and mimics definitions
  - Archive function, for both short term and long term archiving for all type of data produced by the SCC
  - Centralized services, including time management, events/logbook management, users management, configuration and supervision of SCC processes, along with a fully integrated fleet management environment/desktop enabling a single access point to all SCC functions and data.

- OPSWARE products suite
  - FOST

System database management product, allowing full definition of TM/TC, derived parameters and monitoring laws. Online context files are generated and transmitted to hifly^{TM}.

- OPSAT

  Procedures edition and generation product, allowing editing of FOP, checking syntax and consistency against system data and allowing procedures management and documentation generation. It is based on a well defined operations language (PIL).

- OPSEXECUTER

  Procedures preparation and execution product, providing a fully automated execution capability for FOP, with strong information report features. Automation can be overridden by the operator each time it is needed.

- QUARTZ ++

  Flight dynamics product, providing orbit propagation and prediction, maneuvers management, attitude computation and orbital prediction for station keeping operations.

- TMES

  Real-time telemetry information (values and status) distribution to SCC internal or external components via TCP/IP interface, independent of TM/TC task provided services (no direct interaction with hifly^{TM}).

- DEXPRO

  Data analysis component, providing extraction capabilities on archived data (TM, TC history, events) with basic trending processing such as min/max/average.

- MPGATE

  Multi-protocol adaptation component, responsible for adapting SCC TM/TC protocol with DSSS 2A/2B and DSSS 3A TM/TC protocol.

- EUROPS

  Eurostar 2000+ operations support applications framework, providing access to various operations-oriented applications such as CIU or SPY telemetry dump analysis, using archived historical data or real-time telemetry.

2.8.1.3.2. COTS environment

The list below described the COTS (related to this study) used by the various products and the necessary operating system depending on the hardware platforms within the SCC infrastructure.

Products	**Products / Hardware**
COTS	
ACE (Communication library)	OPSEXECUTER, TMES
TAO (CORBA ORB, based on AEC)	OPSEXECUTER, TMES
OmniORB (CORBA ORB)	hifly^{TM}
Java JRE	OPSEXECUTER (GUI)
Elements Presenter (C/C++ GUI builder)	OPSAT, FOST
Ilog Views (C++ GUI builder)	hifly^{TM}
Ilog Views C++ GUI builder)	QUARTZ++
Oracle Standard Edition	OPSAT, FOST,
HP Insight Manager	Computer Monitoring
Operating Systems	
Solaris	OPSWARE workstation QUARTZ++ workstation
Linux SuSE	Monitoring & Control servers Monitoring & Control workstations
Windows	EUROPS workstation MPGATE workstation
Windows Server	NAS

2.8.1.4.Security management

2.8.1.4.1. Access to computers

The access to each server/workstation is submitted to a login/password. In addition of the root (Unix) or administrator (Windows) account dedicated to hardware administration or COTS installation, each type of server/workstation has two different user accounts:

- an SCC administration account for software installation and administration
- an SCC use account for using the different functions, preventing users to corrupt software configuration

2.8.1.4.2. Access to functions

In addition of the privileged access to servers or workstations, some applications have an additional user privilege management enabling to grant different users with different privileges, whether via a specific login window or a workstation configuration:

- FOST (View/Modify)
- OPSAT (View/Modify/Approve)
- HIFLY (configurable roles with access to specific functions)
- OPSEXECUTER (View/Control/Administrate)
- COMPUTER MONITORING (Use/Configure)

2.8.1.4.3. Windows environment

Even if the SCC network is not connected to the external company environment, in order to prevent Windows workstation equipped with floppy drivers to be infected via imported data, Norton Antivirus is installed on EUROPS workstations.

2.8.2. Lessons learned

2.8.2.1. Constraints and development approach used

<u>APIs</u>

OpsWare provide APIs that can be used to integrate OpsWare in various Control Centres using various third party services (TM/TC, logbook, right management…).

These APIs are C++ oriented (as OpsWare servers are in C++) and based on the use of dynamic libraries.

On the other side, some third party tools provide pure CORBA interfaces.

These are two different approaches that have pros and cons.

- OpsWare C++ APIs does not imply to use CORBA. If a third party service is for instance socket oriented, OpsWare APIs can be easily adapted. If the service offers only a CORBA API, it is quite easy to implement a library using this API.

However OpsWare client APIs are only available in C++ (even if these API are in fact CORBA based). This makes the API easy to use in a C++ client application (no need to know CORBA) but limits the use of this API to C++ clients.

- Providing a CORBA API implies the need to use explicitly CORBA in the clients (and then to know its complex programming model).

ORB interoperability

CORBA based third party services are dependant of their underlying ORB, i.e. they are compiled and linked with one ORB among TAO, OmniORB, Orbix, Visibroker, etc.

Our own applications are compiled and linked with our selected ORB (TAO for Astrium).

Therefore interoperability issues may arise.

Fortunately, the test made with the various ORB show a good maturity of ORB interoperability.

ORB standardization

Another aspect of the ORBs is the possibility to replace a given ORB by another ORB for performance purpose, license cost reasons and so on. This requires that ORBs conform to the standard and that developers restrict their ORB use to standard functions.

During the life of OpsWare the ORB (ObjectBroker, Orbix, TAO) has been changed twice, see Successes and failures.

2.8.2.2. PA methodologies and tools used

2.8.2.2.1. Estimating methods

Not specific.

2.8.2.2.2. Team training

It shall be noticed that CORBA aspects requires a specific training or a coaching of developers by a CORBA expert.

This is required at two levels:

- On the architectural level, where the design of the interface between the components (IDL) requires a strong background in distributed architectures in order to eliminate the risk of an inadequate design leading to poor performances.
- On the development level, as CORBA requires a lot of knowledge, particularly when using C++, (for which CORBA requires the knowledge of specific and rather complex memory management rules for instance).

Recom 44: Developers shall be trained for CORBA and the target language CORBA mapping.

Recom 45: Design of a CORBA based architecture shall be managed or controlled by a CORBA expert.

2.8.2.2.3. Tools configuration

TAO

TAO (ORB) is highly configurable. Some configuration parameters have side effect on the server behaviour, and particularly on how requests are managed. Therefore some configurations can introduce unexpected behaviours (incorrect processing of a request wrt request sequence for instance).

Recom 46 : Be aware of ORB configuration characteristics and of potential side effects on processing correctness.

2.8.2.2.4. Life cycles V, V+, Incremental ,iterative

Not specific.

2.8.2.2.5. Requirement Analysis

Not specific.

2.8.2.2.6. Architectural Design

See §2.8.2.2.2.

The design of CORBA-based system may benefit of the use of specific Design Patterns [MOW 97].

Recom 47 : Use CORBA Design Patterns when designing a CORBA based architecture.

2.8.2.2.7. Detailed design

Not specific.

2.8.2.2.8. Testing

Test tools

Rational Purify was used to detect memory leaks in C++ processes.

Borland OptimizeIt was used to detect memory leaks in Java processes.

The need for testing memory leaks in C++ is quite obvious; however developers often think that Java ensures 0 leak programs because of the garbage collector. This is of course a heresy because objects with alive pointers are not cleaned-up and then memory leaks can also occur in Java. Therefore using a memory leak detection tool for Java is highly recommended for applications that are intended to be run several hours or days without being restarted.

Recom 48 : Use a memory leak detection tool even for Java applications if they are supposed to be used several hours or days without being restarted.

Test coverage

Code coverage was achieved using the following tools:

- Purecoverage (Rational IBM) for C++ applications.
- JCoverage (Open Source) for Java applications

Specific issues: load testing, etc.

Not Applicable

2.8.2.2.9. Integration/validation

Constraints: debugging non ASCII interfaces, propagation of bugs between processes.

The use of application management functions (traces, trace level management, access to applications internal data while the application is running) have great benefits for integration purpose and also to investigate problems when applications are in operations.

Recom 49 : Use application management tools in order to allow monitoring of application state and internal data while they are running (trace management, access to internal data).

2.8.2.2.10. Verification

No information.

2.8.2.2.11. Acceptance

No information.

2.8.2.2.12. Operation

See Recom 49.

2.8.2.2.13. Maintenance

Because of ORB evolutions and also because it was chosen to change the ORB from a COTS to an Open Sources solution, OpsWare applications ORB has evolved several times in ten years.

See Recom 49.

2.8.3. Successes and Failures

Use of CORBA for Control Centres Applications

CORBA has been used for several years in OpsWare and to interface third party control centres products with many benefits.

- The performance level is correct.
- The interoperability between ORBs works well (at least for the studied applications).
- CORBA simplifies applications interfacing.

See §2.8.2 for other feedback on CORBA use.

ORB standardization

Orbs do not always comply with standard for every feature (often because products are evolving in parallel with standards evolutions). They generally provide features that are not part of the standard but that have great benefits for the developers. Different behaviours on some specific features such as communications failures (various types of exceptions raised by the different ORBs for the same problem) have been detected.

Therefore, porting an application from an ORB to another can have some cost.

CORBA languages independence

Thanks to CORBA, OpsWare legacy C++ GUIs (built on top of expensive GUI libraries) have been successfully upgraded with new Java GUIs.

OpsWare integration with various control centres third party products

OpsWare was successfully integrated with three different TM/TC products thanks to their C++ or CORBA APIs. This was done by writing plug-in libraries bridging OpsWare APIs with the third party products APIs.

TAO configuration

See §2.8.2.2.3.

Management of CORBA process redundancy.

As explained in [RIS 03], CORBA, today, does not help for redundancy implementations:

- CORBA does not include fault tolerance mechanisms.
- CORBA generally hides the low level communication mechanisms and does not provide interface for fault detection so it makes often difficult fault detection implementation (some ORBs provide non standard API for this).
- Because of this, it is often necessary to implement (and reimplement) detection mechanisms (such as pinging clients via the callback pattern).
- CORBA FT (Fault Tolerant) describes some descriptions of fault tolerance mechanisms but does not provide ready to use implementations.

Recom 50: Do not assume that CORBA will be able to solve your redundancy needs. Anticipate budget for dealing with redundancies.

2.9. Phase 3 - AFP

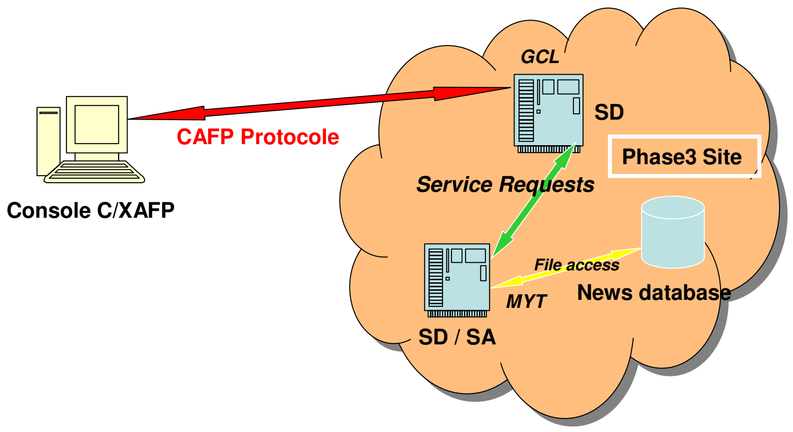
2.9.1. Projects main characteristics description

Phase3 is the Agence France-Presse (AFP) system used by reporters to enter their news and to send them to recipients inside or outside the AFP. News contents are stored for later consultation.

A Phase site hosts several machines : two machines in hot redundancy, acting as a database server, two machines in hot redundancy acting as a front-end with the outside of the site, a machine dedicated to the system supervision and several other machines used to allow reporters connections through their terminals.

There are 3 sites in the world: Paris, Washington, Hong-Kong.

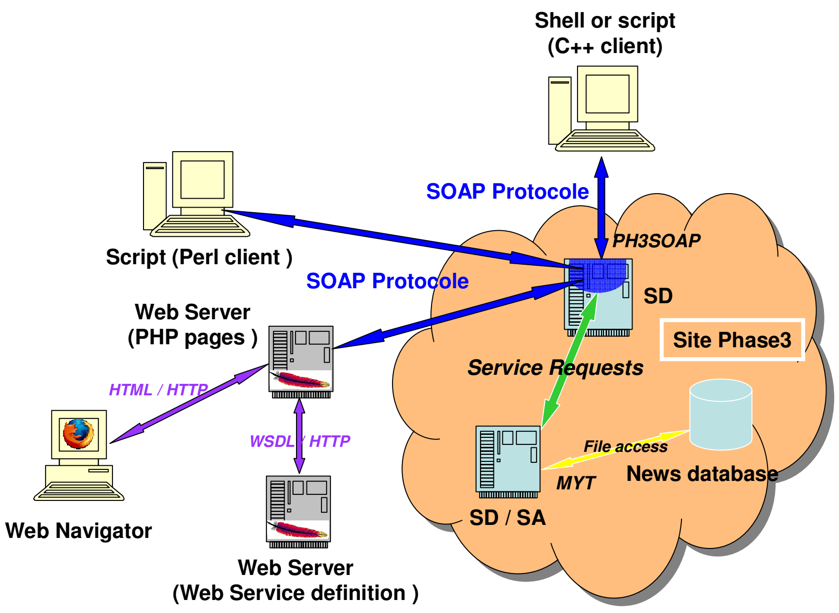
Phase3 development started at the beginning of the 90', it is then based on old technologies (C language, UNIX IPC, X/Motif, proprietary protocols). The figure below shows the initial architecture where only dedicated protocols are used.



- **CAFP** (**AFP console**): reporter console, i.e. software news production and consultation.
- **CAFP Protocol**: legacy dedicated protocol between a CAFP and Phase3 system.
- **SA** (**Archive Server**): server acting as a news database management system. There are in fact two SA per sites (1 nominal and 1 backup).
- **SC** (**Communication Server**): server acting as a gateway with the external world, mainly for news exchanges. There are fact two SS per sites (1 nominal and 1 backup).
- **SD** (**Distributed Server**): server dedicated to CAFP connections. There are several SD per site in order to balance the load. Each SD hosts a news database updated from the central database hosted by the nominal SA.

- **MYT** (**Mean Term**): Phase3 application (set of processes) the role of which is the management of the news. This application is run on SA and SD.
- **DS** (**Service Request**): request exchanged by Phase 3 processes.

Evolutions have been requested by the customer and new technologies have been used around Phase3 core in order to provide new features and particularly to open the system to standard protocols in order to allow for instance the automated production of some news and to allow the use of a navigator on a standard PC instead of a dedicated terminal with preinstalled specific software. The resulting new architecture is described in the figure hereafter.



2.9.2. Lessons learned

2.9.2.1. Constraints and development approach used

The web developments were performed iteratively using a mock-up/prototype with strong interactions with the customer.

2.9.2.2. PA methodologies and tools used

2.9.2.2.1. Estimating methods

Based on, prototyping activities.

2.9.2.2.2. Team training

Self training.

2.9.2.2.3. Tools configuration

The Apache server is highly configurable via text files.

2.9.2.2.4. Life cycles V, V+, Incremental ,iterative

The life cycle for the web-based part of phase 3 was iterative and based on the development of a mock-up.

The mock-up was really useful to assess the security and performance characteristics.

2.9.2.2.5. Requirement Analysis

Direct interaction with customer with prototypes.

2.9.2.2.6. Architectural Design

No specific method nor tool.

2.9.2.2.7. Detailed design

Not specific.

2.9.2.2.8. Security

An external client access the system via a VPN (Virtual Private Network) protected by password. Once the VPN is opened, the user shall enter another user/password to log in the Phase 3 system.

2.9.2.2.9. Testing

Manual testing.

2.9.2.2.10. Integration/validation

No information.

2.9.2.2.11. Verification

Not applicable.

2.9.2.2.12. Acceptance

No information.

2.9.2.2.13. Operation

No information.

2.9.2.2.14. Maintenance

Not specific.

2.9.3. Successes and Failures

<u>Using dedicated protocols</u>

The initial system used a dedicated protocol based on binary data. This protocol was very efficient but it also had some drawbacks:

- Binary messages are hard to monitor without specific (and dedicated) tools. So this can be an issue when there is a need of problem investigation.

- The protocol required a dedicated compiler (similar to an IDL compiler), that has to be maintained.
- The protocol required dedicated libraries.
- The efficiency of the protocol was interesting on 10 Mb network but its interest was limited with the new network generations (100 Mb, 1 Gb).
- A dedicated protocol is, by definition, only usable by dedicated clients.
- A dedicated protocol does not benefit of third party tools.

Recom 51: Avoid dedicated protocols whenever possible.

Web oriented technologies

Web oriented technologies allows the easy creation of GUIs offering a good compromise between the users friendliness and the development cost. As a matter of fact, designing an HTML form is quite easy. Furthermore, the GUI can be accessed from any PC on the network via the navigator. Finally, a web server like Apache is quite easy to install and configure and allows the use of various script languages (such as PHP) to develop dynamic pages.

Apache

Apache is a well known web server. It is well supported on HP-UX (Phase 3 HW platform), but it is also known for its tremendous robustness. In addition, it can be easily configured via simple text files.

PHP

PHP is executed on the server and is in charge of the processing that were, at the beginning of the WWW, devoted to CGI scripts.

The integration of PHP in Apache is straightforward. Its syntax is quite easy (it looks like the C language) and is then easily learned. It shall be noticed that last evolution of PHP language is the introduction of object orientation, the downside of this is that the language is then more complex to learn. PHP offers a huge number of libraries.

XHTML

The World Wide Web Consortium (W3C) published an evolution of HTML named XHTML. It is a XML derived language using strict tags (HTML was supporting approximate use of close tags). XHTML was chosen for phase 3 because it has been decided to use XML format as much as possible for Phase 3 files and because of its stricter format:

- XHTML elements must be properly nested.
- XHTML documents must be well-formed.
- All XHTML elements must be closed.

Thanks to its XML heritage, XHTML files can be validated by schema validation tools.

Recom 52: Use XHTML instead of HTML.

Style Sheets

Cascading Style Sheets (CSS) are used to group web pages presentation information. This avoid to put recurrent presentation information in every (X)HTML page of the project.

The main benefit of this is that the look of the application is centralized in one place and can be updated easily (less update time, less errors).

Recom 53: Avoid complex HTML/XHTML files, instead use CSS to manage the presentation of their content.

2.10. Military Mission Centre for observation satellite

2.10.1. Projects main characteristics description

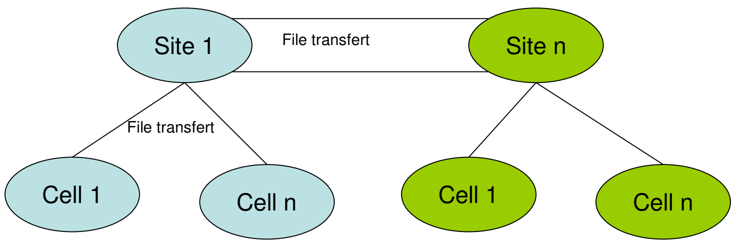
This system is used by several EU armies. It allows the definition of snapshot needs that are used to program the satellite mission and the exploitation of received and processed data by end-users.

This system is widely distributed across EU countries and involves communications between several sites in Europe. Each country hosts a main site and several cells that communicate with the main site, main sites are linked together.

The image telemetry is received, processed and image products are stored by each main site.

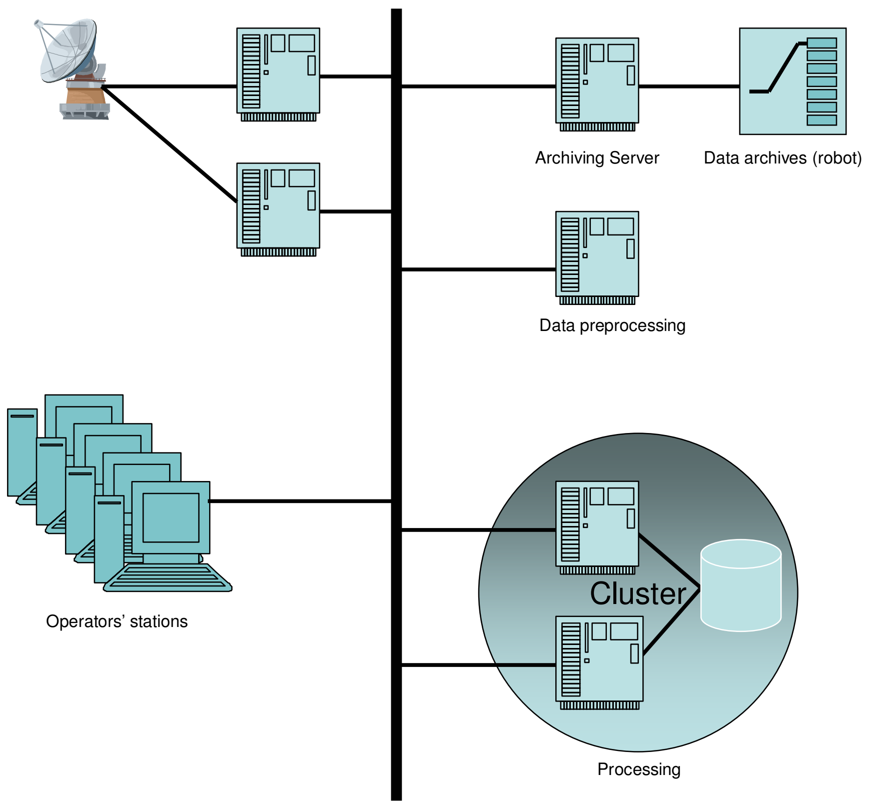
This project relies on different types of communications (FTP, CORBA, SNMP) and involves security aspects such as segregation of sites.

Between sites, the communication is based on file transfers (FTP). Files are encrypted with specific devices and firewalls are used to protect each site. Between sites, there are two separate physical links.



The figure below describes each site. The communications between processes are mainly based on CORBA.

A cluster (using floating IP addresses) is used for the processing server in order to improve the availability.



2.10.2. Lessons learned

2.10.2.1. PA methodologies and tools used

2.10.2.1.1. Estimating methods

Not specific.

2.10.2.1.2. Team training

CORBA requires specific training.

2.10.2.1.3. Tools configuration

NA

2.10.2.1.4. Life cycles V, V+, Incremental ,iterative

Not specific.

2.10.2.1.5. Requirement Analysis

Not specific.

2.10.2.1.6. Architectural Design

UML.

LDS was used to model the system dynamics.

2.10.2.1.7. Detailed design

Not specific.

2.10.2.1.8. Testing

Not specific.

2.10.2.1.9. Integration/validation

Using CORBA raised several integration issues see §2.10.3.

2.10.2.1.10. Verification

Not specific.

2.10.2.1.11. Acceptance

Not specific.

2.10.2.1.12. Operation

Not specific.

2.10.2.1.13. Maintenance

Not specific.

2.10.3. Successes and Failures

<u>CORBA</u>

Using CORBA for communication purpose removes the need to implement dedicated protocols on top of TCP/IP and the management of low level primitives. However, CORBA may introduce several drawbacks that can obliterate its benefits.

1. CORBA is (in general) not very appropriate to deal with changing interfaces. When generated stubs are used (that is the most general case), modifying a CORBA interface requires the recompilation of the server and of every clients.
2. CORBA does not support natively redundancy mechanisms and, for instance, the object references scheme (IOR) is not compatible with the floating IP mechanism and generally, relocating a CORBA process requires relaunching the clients.
3. Invoking a service requires that the target is up and ready, i.e. there is no “store and forward” mechanism in opposition of messaging systems. When the number of interconnected processes and the number of machine is very high, the probability that everything is up and ready at any time decreases. For the same reason, starting up the system is tricky as the start-up sequence have to be coordinated.

CORBA creates strong coupling of applications, this can be a benefit in some case but this can be a disadvantage when there are many components linked together and for integrating large systems.

There are several ways to avoid such strong coupling such as using the Dynamic Invocation Interface or defining IDL interfaces with parameters based on data that are interpreted by the server (for instance formatted strings, attribute/value pairs or XML passed in strings). The flexibility brought by such solutions is gained against the strong typing and verification mechanisms of the static interface.

3. State of the art in Literature and www

3.1. Introduction

Web-based applications were initially devoted to business application on the internet, often developed by startup companies whose main goal was to deliver products as quickly as possible in order to be visible on the Internet, and if feasible, the first one.

Thus, Web projects were not, initially, product assurance oriented but time to market oriented.

The table below, extracted from [REI 00] provides a comparison of the development approaches between "traditional" systems and Web systems.

Characteristic	**Traditional**	**Web**
Primary Aim	Build products at minimum cost	Bring products to market quickly
Typical Size	Medium to large 10 to 100+ eng.	Small 3 to 10 eng
Timeline	12 – 18 months	3 – 6 months
Technology Used	OOT, CASE tools, generators, C++, etc.	CBSE, frameworks, java, multi-media, etc
Process	CMM-based	Ad Hoc, death marches
Products	Code-based systems, done in-house, mostly new, many external interfaces, often complex	Object-based systems, multi-media, done out-house, many reusable parts, few external interfaces, often simple
Development staff	Software engineers	Graphics designers, software engineers, etc.
Estimating technology	Size: SLOC or fp Resources: models or WBS estimate	Size: ??? (web objects) Resources: ad hoc or WBS estimate

Currently, web-based applications are used in every area, even for quite critical applications; therefore product assurance for web-based applications should evolve quickly. But the fact is that today, product assurance of web-based application is not the first concern of the web application development community, far beyond the pure technological aspects that, on the contrary, occupy the front of the scene.

Nevertheless, in the next sections, a review of the current state of the art of web applications product assurance is proposed.

It shall also be noticed, that the NASA faced the same issues about product assurance for web-based applications and performed a 3 years study entitled "Software Assurance Of Web-Based Applications" (SAWBA) [KUR 04] that produced interesting documents [KUR 02] and particularly a Guide Book containing recommendations and check lists [KUR 03].

IEEE provides recommendations for web site engineering, management and life cycle [IEE 03]. This standard proposes some basic best practices but is mainly focused on web pages content (html tags usage, presentation recommendation, etc.).

3.2. Estimating methods

For traditional systems, the metrics to estimate projects and control their progress are well known (even if somewhat controversial): lines of codes, number of classes/methods, functions, etc. Web-based systems are generally hybrid systems containing different types of languages and documents.

Reifer in [REI 00] compares the metrics used in traditional system and web-based systems (table below) and proposes an adaptation of the COCOMO II estimating model called WebMo that describes a new sizing metric called Web Objects.

Traditional	Web
• Source lines of code • Function points • Function point extensions (3D, predictive object points, etc.) • UML metrics	• Multi-media files – Graphics - Text – Audio - Motion – Pictures (JPEG, etc.) • Web components – Applets, Agents, etc. • Fine-grained building blocks – DCOM, CORBA, etc. • Coarse-grained components – Shopping carts, etc.

Table 1: Sizing metrics.

[REI 02-a] explains the conventions proposed by Reifer for web objects counting.

Web Objects extend traditional function points [MIC] to take the following four additional types of objects into account because they require additional effort to incorporate them into web applications:

- Multi-media files – size predictors developed to take the effort required to incorporate audio, video and images into applications. Such effort includes the work involved in creating web pages; creating video for the web (MPEG-1&2 files); creating publishable documents for the web; and creating, editing and enhancing complex images for both clients and servers.
- Web building blocks – size predictors developed to take the effort required to develop web-enabled fine-grained component and building block libraries and any wrapper code required to either instantiate or integrate them. Such predictors do not count the standard libraries that come as part of the web environment and typically include both Windows and Java components. Instead, they take only the additional active (ActiveX, applets, agents, guards, etc.), fine-grained static (COM, DCOM, OLE, etc.) and course-grain reusable (shopping carts, buttons, logos, etc.) building blocks that are acquired or developed to incorporate into web applications into account for both the client and server.
- Scripts – size predictors developed to take the effort required to link html/xml data and generate reports automatically; query ODBC-compliant databases via prompts; integrate and animate applications via predefined logic (via GIF); and direct dynamic web content per customizable pallets, masks, windows and commands (streaming video, real-time 3D, special effects, motion, guided workflow, batch capture, etc.) for both clients and servers.
- Links (xml, html and query language lines) – size predictors developed to take the effort required to link applications, integrate them together dynamically and bind them to the database and other applications in a persistent manner.

Basically, to count web objects, Reifer proposes to evaluate the following nine components of a web system based upon user requirements and page layouts:

- Internal Logical Files – logical, persistent entities maintained by the web application to store information of interest.
- Multi-Media Files – physical, persistent entities used by the web application to generate output in multi-media format.
- Web Building Blocks – logical, persistent entities used to build the web applications and automate their functionality.
- Scripts – logical, persistent entities used by the web application to link internal files and building blocks together in predefined patterns.
- Links – logical, persistent entities maintained by the web application to find links of interest to external applications
- External Interface Files – logical, persistent entities that are referenced by the web application, but are maintained by another software application.

- External Inputs – logical, elementary domain processes that cross into the application boundary to maintain the data on an Internal Logical File, access a Multi-Media File, invoke a Script, access a Link or ensure compliance with user requirements.
- External Outputs – logical, elementary domain processes that result in data leaving the application boundary to meet a user requirements (e.g., reports, screens).
- External Queries – logical, elementary domain processes that consist of a data “trigger” followed by a retrieval of data that leaves the application boundary (e.g., browsing of data).

It is interesting to notice that Ruhe et al [RUH 03] shown from experimentations that WebMo outperforms other technics for web-based projects estimations.

3.3. Requirement Analysis

[ESC 04] proposes a state of the art of requirements engineering in methodologies used for the development of Web applications. For the authors, web applications require a more extensive and detailed requirements engineering process due to the number of stakeholders involved and due to the diversity of the requirements including among others requirements on the navigation and on the domain processes as well as Web usability. The conclusion of this paper states that there is still a great research potential in the field of requirements engineering for Web applications and that the majority of the currently existing Web methodologies focuses on design aspects and not on requirements engineering.

According to Lowe and Eklund a major problem that occurs in Web Engineering projects is that the users get to know how to express their requirements very late in the process, i.e. after design artifacts have appeared [EKL 02]. Prototyping the presentation layer with static pages can then be a good solution at least to define the GUI organization and the navigation requirements.

3.4. Configuration Management

Web application content involves many types of data that are explicitly or implicitly linked together:

- Html, CSS, XSL files,
- code in various languages,
- documents (pdf for instance),
- graphics,
- audio/video.

The challenge of web-application configuration management is to keep the consistency of the network of various elements that are linked together.

F. Vitali in [VIT 99] discusses the issues of hypermedia versioning : his conclusion is that “the issues arising from complex project management, support for multi-authored documents, and collaboration in hypermedia are similar to the corresponding ones in many other fields”, however hypermedia versioning differs from other fields “because of the management of explicit relationships among the resources being managed”. As a matter of fact the major issue is to keep the links integrity, particularly when links are refencing external documents that can be changed by their authors.

However for web-based applications that are not internet public applications, this issue is less significant.

Versioning tools can be classified into two categories of tools: code versionning tools (such as CVS, Rational Clearcase, etc.) that have been used for a long time for software development, and web authoring tools configuration management, that can be either proprietary are that can be based on WebDAV.

WebDAV, that stands for "Web-based Distributed Authoring and Versioning" is a set of extensions to the HTTP protocol: retrieving and update of resource properties, lock management, copy/move of resources, server-side search; it allows users to collaboratively edit and manage files on remote web servers [WHI a][WHI 99]. In spite of its name, the WebDAV protocol only provides facilities for remote collaborative authoring of documents, and does not provide any versioning capability. The DeltaV protocol is an extension to WebDAV that provides versioning management [WHI b]. WebDAV is supported by most of authoring tools and is implemented by several servers (such as Apache throught its mod_dav module).

As complex web-based application mix web pages and software code, it seems better to use a single tool in order to manage the different application files consistently. For space projects, traditional configuration tools such as Clearcase can be used.

3.5. Design

Several specific methodologies and techniques have been elaborated, mainly during the nineties, for hypermedia applications. New methods have been proposed in 2000-2001 based on UML that fit more web applications.

Below is a list of (some) of these methods.

Relationship Management Methodology (RMM)	Isakowitz et al. 1995 [ISA 95]
Object-Oriented Hypermedia Design Methodology (OOHDM)	Schwabe & Rossi, 1995 [SCH 96]
Enhanced Object-Relationship Model (EORM)	Lange, 1994
World Wide Web Design Technique (W3DT / SHDT)	Bichler & Nusser, 1996
Extended World Wide Web Design Technique (eW3DT)	Scharl, 1999
Scenario-based Object-Oriented Hypermedia Design Methodology (SODHM)	Lee et al. 1998 [LEE 98]
View-Based Hypermedia Design Methodology (VHDM)	Lee et al. 1999
OO/pattern	Thomson & Al, 1998 [THO 98]
WebArchitect	Takahashi & Ling, 1997 [TAK 97]
Web Site Design Method (WSDM)	De Troyer & Leune, 1997 [DET 97]
WebML (Web Modeling Language)	Ceri & Al, 2000 [CER 00]
UML based Web Engineering Methodology, UWE,	Koch, 2001 [KOC 02]

OO-Hmethod	Gomez & Al, 2000 [GOM 00]

However, as described in [HAL 05], facing the economical context and the challenges of the "new economy", most of web developments do not use any method. Furthermore there are several other reasons for this, and among them:

- Lack of documentation: most of these methods come from the university world and some "exist" only in conference papers.
- Lack of operational tools : methods are often supported by prototypes but not industrial tools;
- Lack of training courses : technologies are often part of education but methodologies are infrequently taken into account.

It shall be noticed that Koch et al. showed in [KOC 02] that UML is powerful enough to cover the requirements that arise when modeling Web applications. They showed especially how to model, links, pages, navigation, etc. This approach could have benefits but it is actually interesting and usable if the model can be used to generate the code (based on the MDA approach as proposed in [MEL 05] for instance).

Because of the current state of exiting methods, one can suggest that prototyping a site with static pages can be a good choice to help:

- Final users' requirements capture.
- Defining the design of the presentation layer of the web-based system.

3.6. Architectures

Architectures of web-based systems are evolving. Today, the two main directions of evolutions are SOA and AJAX. In addition JMS adds a MOM facility to Java based systems that complement the RPC based middleware (CORBA, RMI).

3.6.1. Low level protocols and middleware

Middleware such as CORBA, RMI, SOAP, MOM are built on top of low level protocols (generally TCP or sometimes UDP, SOAP relies mostly on HTTP – itself built on top of TCP – but can also use SMTP).

Middleware avoid the need to define an application specific protocol on top of a low level protocol, reducing the need to develop and test low added value code and allowing the use of standard tools.

The figure below extracted from [TCP] provides the characteristics of the two basic protocols that are used for building higher level protocols.

Characteristic / Description	**UDP**	**TCP**
General Description	Simple, high-speed, low-functionality “wrapper” that interfaces applications to the network layer and does little else.	Full-featured protocol that allows applications to send data reliably without worrying about network layer issues.
Protocol Connection Setup	Connectionless; data is sent without setup.	Connection-oriented; connection must be established prior to transmission.
Data Interface To Application	Message-based; data is sent in discrete packages by the application.	Stream-based; data is sent by the application with no particular structure.
Reliability and Acknowledgments	Unreliable, best-effort delivery without acknowledgments.	Reliable delivery of messages; all data is acknowledged.
Retransmissions	Not performed. Application must detect loss of data and retransmit if needed.	Delivery of all data is managed, and lost data is retransmitted automatically.
Features Provided to Manage Flow of Data	None	Flow control using sliding windows; window size adjustment heuristics; congestion avoidance algorithms.
Overhead	Very low	Low, but higher than UDP
Transmission Speed	Very high	High, but not as high as UDP
Data Quantity Suitability	Small to moderate amounts of data (up to a few hundred bytes)	Small to very large amounts of data (up to gigabytes)
Types of Applications That Use The Protocol	Applications where data delivery speed matters more than completeness, where small amounts of data are sent; or where multicast/broadcast are used.	Most protocols and applications sending data that must be received reliably, including most file and message transfer protocols.
Some Protocols	TFTP (Trivial File Transfer Protocol), SNMP (Simple Network Management Protocol),	FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), HTTP, POP (Post Office Protocol),

Table 2: UDP/TCP comparison.

3.6.2. SOA (Service oriented Architecture)

Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership areas [OAS 06].

A service-oriented architecture is a collection of services that communicate with each other. The services are self-contained and do not depend on the context or state of the other service. They work within a distributed systems architecture. SOA is the reuse of the old idea of RPC (Remote Call Procedure), but set in the new context of the www and (generally) based on XML.

SOAs comprise loosely-coupled, interoperable services. These services interoperate based on a formal definition (or contract) which is independent from the underlying platform and programming language (e.g., WSDL). The interface definition encapsulates the vendor and language-specific implementation. A SOA is independent of development technology (such as Java and .NET).

SOA is the new “buzz”: it is everywhere in IT papers and on the web. However it is not obvious to guess if SOA will be useful for space systems and how they will bring benefits, and the answer to this cannot be elaborated in the frame of the current study ; but SOA is probably well suited to non-real time sub-systems (such as an image acquisition requests application for an observation satellite) and not appropriate for real time applications. But as discussed in [ARJ 04] SOA requires additional activities and artifacts that are not found in traditional Object-Oriented Analysis and Design (OOAD).

For [ZIM 04], experience from early SOA implementation projects suggests that existing development processes and notations such as OOAD, EA (Enterprise Architecture), and BPM (Business Process Modeling) only cover part of the requirements needed to support the SOA paradigm. While the SOA approach reinforces well-established, general software architecture principles such as information hiding, modularization, and separation of concerns, it also adds additional themes such as service choreography, service repositories, and the service bus middleware pattern, which require explicit attention during modeling.

SOAP is the predominant communication protocol for SOA. SOAP is today regarded as the silver bullet of middleware, however, several authors [JON 02][SCH 01] are considering that SOAP is not so marvelous and cannot replace every other middleware for every types of application. At first sight simplicity of SOAP is often counterbalanced by the complexity of the XML encoding of the messages and the poor readability of WSDL. In addition, SOAP toolkits are not always well-documented and even if they are evolving, do not have the maturity of CORBA implementations.

3.6.3. AJAX

AJAX is another IT “buzz”. AJAX stands for Asynchronous JavaScript And XML. It is a Web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire Web page does not have to be reloaded each time the user makes a change. This is meant to increase the Web page's interactivity, speed, and usability.

AJAX is not a new “technology” but a technique using a combination of existing technologies: XHTML, CSS, XML, JavaScript.

AJAX supports the concept of "rich thin client" and is the basis of Google applications such as GMail, Google Earth, Google Suggest.

3.6.4. MOM and JMS

Message-oriented middleware [CAR 05] is software that resides in both portions of a client/server architecture and typically supports asynchronous calls between the client and server applications. Message queues provide temporary storage when the destination program is busy or not connected. MOM reduces the involvement of application developers with the complexity of the master-slave nature of the client/server mechanism.

MOM increases the flexibility of an architecture by enabling applications to exchange messages with other programs without having to know what platform or processor the other application resides on within the network. The aforementioned messages can contain formatted data, requests for action, or both. Nominally, MOM systems provide a message queue between interoperating processes, so if the destination process is busy, the message is held in a temporary storage location until it can be processed. MOM is typically asynchronous and peer-to-peer, but most implementations support synchronous message passing as well.

MOM is most appropriate for event-driven applications. When an event occurs, the client application hands off to the messaging middleware application the responsibility of notifying a server that some action needs to be taken.

Asynchronous and synchronous mechanisms each have strengths and weaknesses that should be considered when designing any specific application. The asynchronous mechanism of MOM, unlike Remote Procedure Call (RPC) , which uses a synchronous, blocking mechanism, does not guard against overloading a network. As such, a negative aspect of MOM is that a client process can continue to transfer data to a server that is not keeping pace. Message-oriented middleware's use of message queues, however, tends to be more flexible than RPC-based systems, because most implementations of MOM can default to synchronous and fall back to asynchronous communication if a server becomes unavailable.

MOM is typically implemented as a proprietary product, which means MOM implementations are nominally incompatible with other MOM implementations. However JMS (Java Message Service) can be considered as a Java standard MOM.

[LAR 02] establishes the following pros (+) and cons (-) of JMS.

(+) JMS is a set of standards created by multiple vendor implementations; therefore, it avoids the vendor lock-in problem.

(+) JMS allows for abstraction (via a generic API) between client and server; one can change a database schema or platform without changing the application layer.

(+)/(-) JMS can help a system scale (a pro). The con is that any system that scales with JMS can scale without it.

(-) JMS is complicated. It's an entirely new layer with a new set of servers. Software rollout management, server monitoring, and security are just a few of the nontrivial problems associated with a JMS rollout. Costs should also be considered.

(-) Vendors do not always interpret and therefore implement standards exactly the same way, so differences exist between various implementations.

(-) With JMS, one need more system checks and balances. It not only introduces a new layer, it also introduces asynchronous data distribution and acknowledgement, which has the added complexity of asynchronous notification.

(-) No message reporting/updating/monitoring queues without custom software.

(-) Security, JMS changes security requirements and/or introduces potential security holes

(-) Processing order. JMS is best for systems that allow independent processing order

3.6.5. Connection Oriented vs. Connectionless

Connection-Oriented means that when devices communicate, they perform handshaking to set up an end-to-end connection, for example, TCP is connection oriented. During the connection, mechanisms can be used to ensure that peers are aware of the connection status.

Connectionless means that no effort is made to set up a dedicated end-to-end connection. Connectionless communication is usually achieved by transmitting information in one direction, from source to destination without checking to see if the destination is still there, or if it is prepared to receive the information. UDP and SNMP are connectionless protocols.

Things are in fact more complicated, for instance HTTP relies on TCP, that is connection-oriented. However, a new TCP connection is established for each HTTP request, then the connection is dropped, therefore, HTTP is connectionless.

If it is important within an application for a client to be continuously aware of its server status than connection-oriented protocols are better.

3.6.6. Notion of coupling in architecture based on middleware

It is well accepted that different types of distributed architectures require different levels of coupling. For example, in client-server and three-tier architectures the application components are generally tightly coupled between them and with the underlying communication middleware. Meanwhile, in off-line transaction processing, grid computing and mobile application architectures, the degree of coupling between application components and with the underlying middleware needs to be minimised along different dimensions. In the literature, terms such as synchronous, asynchronous, blocking, non-blocking, directed, and non-directed are generally used to refer to the degree of coupling required by a given architecture or provided by a given middleware.

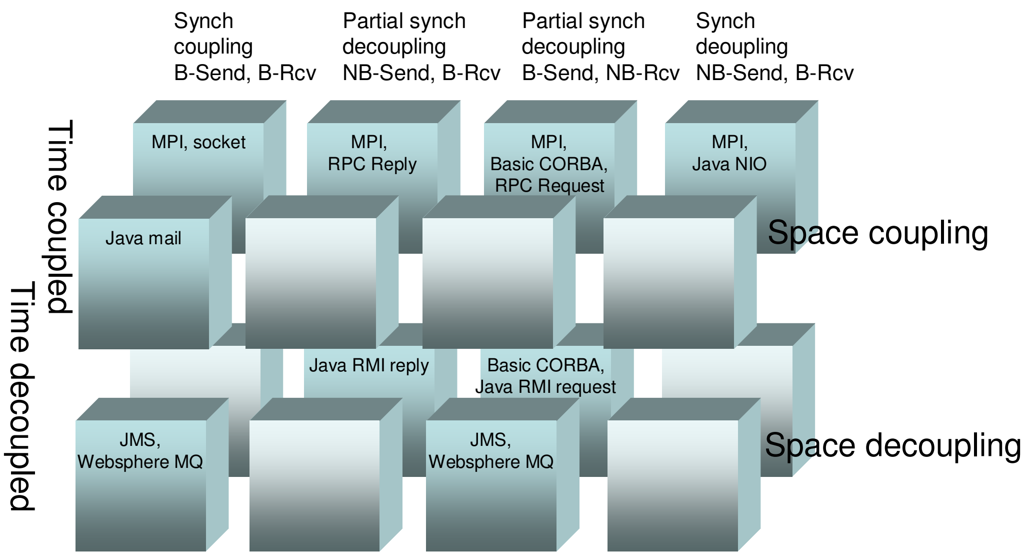
[LAC 05] proposes a middleware classification according to 3 axes:

<u>Synchronization coupling:</u> defines if client and server communications are blocking or not blocking for either the sender (B-Send or NB-Send) and the receiver (B-RCv or NB-Rcv). Non blocking means that the end-point can easily interleaves processing with communication.

<u>Time coupling</u>: defines if both end-points have to be operating at the same time to communicate. RPC based middleware (basic CORBA and RMI) are time coupled, while Message Oriented Middleware (MOM) [CAR 05] are not, as they rely on a third participant allowing store-and-forward mechanisms.

Space coupling: defines if the sender needs to know the receiver direct address to send the message to or not. For instance CORBA is able to hide the location of a server object to the client.

The figure below shows the classification proposed by [LAC 05] for several communications means.


Synch coupling B-Send, B-Rcv
Partial synch decoupling NB-Send, B-Rcv
Partial synch decoupling B-Send, NB-Rcv
Synch deoupling NB-Send, B-Rcv
Time coupled
Time decoupled
Space coupling
Space decoupling
MPI, socket
MPI, RPC Reply
MPI, Basic CORBA, RPC Request
MPI, Java NIO
Java mail
Java RMI reply
Basic CORBA, Java RMI request
JMS, Websphere MQ
JMS, Websphere MQ

The analysis of the Military Mission Centre described in section 2.10 showed that a time decoupled middleware would have been better than a time coupled solution (basic CORBA). However this does not mean that CORBA was a bad solution - as a time decoupled solution could have been implemented over CORBA - but that the use of basic CORBA, which is time coupled, was not the best choice. The ready to use time decoupled systems are the use of e-mail and Message Oriented Middleware.

3.7. Verification and Testing

[KUR 03a] proposes QA verification activities (audits, formal inspections, testing, safety/security). The PA tools he proposed are quite the same tools used for other types of software development:

- Verifications
  - Adherence to Style Guides/Coding Standards.
  - Audits and use of Checklists.
  - Formal Inspections.
  - Static analysis.

  The proposed items are reused in chapter 5.6.
- testing

Web-systems specificities imply testing specificities [KUR 03]:

- Many different languages are used.

- The browser is a generic GUI that "generates" dynamically the graphical interface from html, xml, css.
- Data used dynamically by the GUI (browser) such as html, xml, css data can be generated dynamically by the server.

Two main categories of tests shall be performed:

- Functional testing, i.e. test against functional requirements as in other types of software.
- Load testing, which is a specific type of performance testing consisting of testing the system with a large number of clients with "typical" behaviours.

3.7.1. Static analysis

Static analysis uses tools (see §3.7.4) that verify whether:

- Files contain broken links and navigational problems, i.e. the analysis aims to ensure that every link in html pages is actually attached to a web page.
- HTML, CSS, JavaScript, and ASP/VBScript code complies with W3C standards and industry guidelines. HTML syntax errors for instance are difficult to detect as there is no compiler, errors are only detected at run time when the browser receives the page, therefore using an HTML verifier helps removing errors early.
- XML is well-formed and valid: XML files are generally linked to a schema (XML Schema Definition or XSD) which describes the "grammar" of a particular XML file (for instance a telemetry file can be described in that way by an XSD and an instance of XML telemetry file can be checked against the schema).
- HTML and XML files contain typos and misspelled words.

3.7.2. Functional testing

Static analysis can provide good results, but preventing bugs from showing up in delivered software requires functional testing (that can possibly be automated) [PAR 04]. While static analysis finds problems by inspecting source code, functional testing finds problems by analysing how click paths would operate in a browser. Functional testing involves verifying whether designated click paths execute correctly. Functional testing exposes problems such as:

- JavaScript runtime errors.
- PHP runtime errors.
- Pop-up windows, page changes, and other effects that do not work as expected.
- Frames that do not load correctly in the context in which they are used.
- Valid pages that are blank or incomplete.
- Server-side program crashes and exceptions.

- Server errors and failures.
- Unexpected page content changes.
- Unexpected click path flow changes.

Functional testing activities include:

- Black box testing – performs tests to verify that the expected outputs or outcomes result from a given set of inputs. Testing from the user's point of view.
- Web box testing – performs tests to verify that the dynamically generated output i.e. reports, query results, etc. are correct. It is also used to test exception handlers and other programming failures.

  This concept is interesting, it has been introduced in Kurtz's NASA study [KUR 02] however web box testing is not defined with more details (how to implement and perform web-box testing is not explained) and the only information found on this topic are those of Kurtz and Parasoft company. Parasoft claims to be the only tool provider that support web-box testing (in its webking tool [PAR]).

Functional testing can be done at any level of the system: unit testing, component testing, integration testing.

3.7.3. Load testing

Load testing involves exercising the application with virtual users and verifying whether it supports the expected traffic levels (requests and responses, these later can contain huge data: data extracted from a database, pictures, etc). It is typically used to verify whether the application will perform adequately under the expected loads and surges, determine system capacity, and identify the source of any bottlenecks.

Load testing can be performed by using specific tools developed during the project, however several COTS exist to do this such as (among tenth of such tools) AdventNet QEngine [ADV], Parasoft WebKing [PAR], Apache JMeter [JME 05]. These tools can generally also be used for automating functional testing.

3.7.4. Test tools

[SOF] and [WST 06] are web sites referencing test tools covering the following aspects:

- Link and HTML Test Tools : HTML, XML, CSS, etc. syntax checking and verification that links refer to existing pages) (e.g. LinkScan, HTML Tidy).
- Functional Test Tools : used for dynamic testing. These tools can be based on scripts or can have some capture/replay ability (e.g eValid Functional Test, AppPerfect FunctionalTest, Parasoft WebKing).
- Performance Test Tools : used to stress the application by simulating several users (e.g. eValid Load Test, AppPerfect LoadTest, Openload).
- Performance Test Services (an external web-site used to test the application performance by generating internet traffic, e.g. SiteStress, WebStress).

[SOF] and [WST 06] provide a list of 10 to 20 tools per category with their main characteristics, however selecting test tools for a given project requires to study them on real test cases in order to assess their efficiency, their usability and their adequation to the project.

3.7.5. Code coverage

Code coverage is an analysis method that determines which parts of the software have been executed (covered) by the test case suite and which parts have not been executed and therefore may require additional attention.

Code coverage tools are widely used for traditional languages (C, C++, ADA…). Java is also well supported either by commercial tools or Open Sources ones (e.g. JCoverage, Emma).

Coverage tools for other languages used in web-based systems are less usual. However, some tools are available for PHP [SEM 05], JavaScript [SOF 06] and Perl [DEV 00].

3.8. Safety and Mission Critical Applications

Applications based on web and communication technologies that are identified as critical applications shall follow the same methods and the same PA activities than for other types of critical applications shall be applied. Especially, ECSS Q80 §6.2.2 (Software dependability and safety analysis) is applicable.

3.9. Security

Confidence in applications security can be gained through actions that may be taken during the process of development, evaluation and operation.

ISO 15408, "Evaluation criteria for IT security" more known as "Common Criteria" [COM] provides the tools for evaluation of IT security. It is based on the alignment and development of a number of source criteria: the existing European (ITSEC), US (TCSEC) and Canadian (CTCPEC) criteria.

The Common Criteria (CC) presents requirements for the IT security of a product or system under the distinct categories of functional requirements and assurance requirements.

- The CC functional requirements define desired security behavior and security functionalities that a product can implement.
- Assurance requirements are the basis for gaining confidence that the claimed security measures are implemented correctly and are effective.

The CC is a flexible approach that allows the selection (generally by the product customer or user) of the security functionalities and of an evaluation assurance level.

EAL (Evaluation Assurance Levels) is the scale of predefined assurance level of the Common Criteria. The CC provides seven levels:

- EAL1 - functionally tested
- EAL2 - structurally tested
- EAL3 - methodically tested and checked
- EAL4 - methodically designed, tested and reviewed
- EAL5 – semi-formally designed and tested
- EAL6 – semi-formally verified design and tested
- EAL7 - formally verified design and tested.

4. Synthesis

4.1. Recommendations issued from analyzed projects

The list bellows summarizes the raw recommendations derived from analyzed projects. Recommendations are grouped by issues.

4.1.1. Management

4.1.1.1.Team

Recom 34 : Identify experts in each technology at the beginning of the project.

Recom 1 : When the application is data centric (i.e. the complexity of the application is more in the database schema than for instance in algorithms), the database configuration is critic and must be managed by an expert (Schema checking, database tuning).

Recom 9 : An expert must supervise the development of technically complex layers.

Recom 45: Design of a CORBA based architecture shall be managed or controlled by a CORBA expert.

Recom 13 : J2EE projects team must contains senior developers who have an experience in this kind of architecture.

Recom 23 : Supervise junior developer by experts/seniors who have an experience in distributed architecture.

4.1.1.2.Developement organization

Recom 15 : The development team shall not change during development phase.

Recom 30 : A specific training must be organised at the beginning of the project for each used technology.

The recommendation above is covered by ECSS-Q-80B §5.8.3.2.

Recom 44: Developers shall be trained for CORBA and the target language CORBA mapping.

Development can be organized either

- *by layers: e.g. one sub-team is in charge of the presentation layer, another one is in charge of the domain layer and another one is in charge of the data layer. Developers can have specialized skills but they have to manage the interactions with the other layers.*
- *or by use case: sub-sets of use cases are allocated to developers that shall work on all layers), developers shall have skills for each layer.*

Recom 37 : Use a use-case driven approach with small teams located on the same site (communication is vital).

Recom 10 : In a development by layers, the interfaces must be highly detailed before their implementation occurs. The developer must have means to communicate with managers of layers n+1 and n-1.

Recom 11 : Use case driven approach must be used only when core layers are identified and specified.

Recom 12 : In a use case driven approach, each developers must have a global view of the architecture and have the means to communicate quickly with other use-case developers.

4.1.1.3.Life Cycle

Recom 27 : Combining a classical "V life-cycle" with an iterative approach based on prototyping is strongly recommended, in particular when user validation of GUI or early design of complex algorithms is concerned.the requirement must be identified and detailed soon in project design.

Recom 19 : In a distributed architecture based on connected components, the connection loss or degradation must be identified and reaction to these changes must be specified.

Recom 21 : Component reuse must be made accordingly to the context of the application particularly regarding programming languages and frameworks used by other components.

Recom 22 : Distributed components communication must be made using the appropriate middleware or framework (e.g. J2EE application server) and integration process must be prepared according to this choice.

Recom 22 : It is recommended to combine the use of activity diagrams and use cases, since they provide efficient and complementary ways to describe internal activities of any use case of a subsystem, more precisely, activity diagrams provide detailed information on the dynamic part of the use case.

Recom 42: In an environment based on numerous communicating components the interfaces definition is a critical point that must be seriously managed during detailed design.

The recommendation above is covered by ECSS-E-40Part 1B §5.3.4, §5.4.3.8, §5.5.2.2.

4.1.1.4.Application layers

Recom 2 : In a complex architecture the use of a prototype shall be used to validate architectural choices and to start development team training.

Recom 3 : The core layers (i.e. domain layer and data layer) must be detailed using UML.

Recom 4 : Use a prototype to validate the presentation layers. UML must be used only to explain high level processes (using sequence diagram for example).

Recom 41 : Reduce the implementation of complex domain logic in the presentation layer.

Recom 38 : Reduce the use of SQL code in the domain layer's code.

Recom 40 : Reduce the use of dynamically generated code (Javascript code generated by servlets).

4.1.1.5.Technology selection

Recom 51: Avoid dedicated protocols (protol specifically developed for a given application) whenever possible. As a matter of fact, standards are supported by tools, tested libraries, and are less difficult to maintain as the risk of "knowledge loss" is limited**.**

Recom 25 : Use of technologies, such as XML / XSL / XSLT, which permits to distinguish between data structuring and data representation, is highly efficient to enhance maintainability of automatically produced HTML pages.

Recom 52: Use XHTML instead of HTML.

Recom 53: Avoid complex HTML/XHTML files, instead use CSS to manage the presentation of their content. Cascading Style Sheets (CSS) are used to group web pages presentation information. This avoid to put recurrent presentation information in every (X)HTML page of the project. The main benefit of this is that the look of the application is centralized in one place and can be updated easily (less update time, less errors).

Recom 29 : CORBA represents a very interesting communication middleware which can be used for non-distributed architectures. Highly efficient free tools are available to implement this middleware.

> Recom 47 : Use CORBA Design Patterns when designing a CORBA based architecture.
>
> Recom 50: Do not assume that CORBA will be able to solve your redundancy needs. Anticipate budget for dealing with redundancies.

Recom 14 : The complexity of the user interfaces must be evaluated and compared with the development framework capabilities and the client application nature (Rich/light/heavy).

Recom 39 : Choose the programming languages accordingly to open-source components availability.

Recom 43: Critical COTS must be accurately evaluated with one or more prototypes based on the same kind of use than the final application

The recommendation above is linked to and extents ECSS-Q-80 §6.2.7.

4.1.2. Software design and implementation

Recom 16 : Use an IDE integrated to the application framework (i.e. able to communicate with the application server).

The recommendation above extents ECSS-Q-80 §5.8.

Recom 17 : In an internationalized context, identify all sources of messages: exceptions, GUI, data stored in files or databases.

Recom 24 : The target platform must be known and available during the development step

Recom 38 : Reduce the use of dynamically generated code (Javascript code generated by servlets).

Recom 20 : When components communicates using a low level protocol, interfaces must be specified (services provided, arguments and return types, error feedback mechanisms) and the best practice is to provide automatic testing tools to validate protocol and each service.

4.1.3. Verification and Testing

Recom 5 : When it is possible use automated testing tools.

Recom 6 : When the interface of a component is completely defined, it's a good practice to write JUnit test cases before implementing the component logic.

Recom 7 : Validate complex data exchange using data flow descriptor (Ex. : XML validated by XSD).

Recom 8 : In a layered architecture integration must be anticipated by test means adapted at the layer level. The interface of each layer can be validated with automated test or with specific test modules which realizes queries on each service exposed by the layer.

Recom 48 : Use a memory leak detection tool even for Java applications if they are supposed to be used several hours or days without being restarted.

Recom 35 : Provide ways to test quickly a complex layer/component : dedicated test project, simulators, etc.

Recom 26 : Use of a "chrooted" environment, combined with standard rights access management (HTTP server security, files and directories access rights), is particularly well-suited when security requirements are applicable on a Web server, which happens most of the time.

Note: Using a "chrooted environment"consist in using the "chroot" Unix command to limit the visibility of a program (typically a web server) on only a portion of the hard disk. This ensures that all other applications running on the server hosting the web server are protected from some types of external attacks.

Recom 31 : An operational Control Center must include a web server in order to provide an easy access to the archived data from various locations and by the various type of specialists.

Recom 32 : When it is not possible to built a custom application due to project constraints (cost, delay, etc.) to develop web-based applications, and COTS or Open Source products shall therefore be selected, use open source technologies or COTS based on standards and avoid proprietary non-standard products.

Recom 37 : Use a use-case driven approach with small teams located on the same site. As a matter fact, all developers work potentially on all layers, so they have to manage potential developpement interferences; therefore communication within the team is vital.

4.1.4. Software Delivery/Installation and Operation

Recom 36 : Anticipate installation/integration problems if the target platform is not available.

Recom 33 : Use gateways for a progressive and secure migration of an operational distributed system.

Recom 46 : Be aware of ORB configuration characteristics and of potential side effects on processing correctness.

Recom 49 : Use application management tools in order to allow monitoring of application state and internal data while they are running (trace management, access to internal data). The use of application management functions (traces, trace level management, access to applications internal data while the application is running) have great benefits for integration purpose and also to investigate problems when applications are in operations.

4.2. Adequacy of technology vs system specific requirements

The table below is a complement to the table proposed in TN1 [RD.2]. Middleware are assessed wrt to the key criteria defined in TN1.

Requirement/Middleware	HTTP	SOAP	RMI	CORBA	MOM/JMS
Performance *Real time data flow*	inadequate	inadequate	inadequate	Correct with specific implementation (real time CORBA)	inadequate
Performance *Near Real time data flow*	inadequate	inadequate	Correct (RMI over IIOP)	correct	poor
Performance *Non real time data flow*	Correct	Correct	Very good	Very good	correct
Connection reliability	poor	poor	correct	correct	No real connection between client and server as the messaging API decouples the client from the server.
Availability					
Callback (server->client invocation)	inadequate	Inadequate/poor	Yes	Yes	Yes
Scalability	Load balancing is supported by some servers.	Load balancing is supported by some servers.	No built-in mechanisms.	No built-in mechanisms. A load balancing service exist (not analysed here)	Yes by adding JMS servers.
Accessibility	Very good	Very good	Correct (HTTP	Poor. Require HTTP	Poor/Correct HTTP

			tunnelling)	tunnelling (ORB dependant implementation)	tunnelling (JMS server features).
Usability	Light client	Light client	Rich Client	Rich Client	Rich Client
Security	HTTPS (HTTP over SSL)	Over HTTPS	Can be used over SSL.	Can be used over SSL.	Can be used over SSL.
Testability	ASCII based interface	ASCII based interface	Binary based interface, require a specific tool.	Binary based interface, require a specific tool.	No information
Ease of Integration/Modularity	Depend on data exchanged, low to medium coupling. Time coupled Space coupled	Medium coupling of components Time coupled Space coupled	High coupling of components Time coupled. Space decoupled	High coupling of components Time coupled. Space decoupled	Low coupling of components. Time decoupled. Space decoupled
Maintainability (expressed by complexity)	Simple	Medium	Complex	Complex++	Complex
Delivery/Installation	Require an HTTP server	Require an HTTP/SOAP server	Require rich client deployment	Require rich client deployment	Require a JMS server (server + database).
Manageability	Not linked to the middleware choice. However Java based systems are manageable via the JMX API.				

4.3. Languages

The table below describes the main programming languages used for web-based applications implementations and some associated "PA" tools.

Language	**Editor**	**Debugger**	**Unit Testing**	**Code coverage**	**Comment**
Java	Eclipse	Eclipse	JUNIT	Several: e.g. JCoverage, Emma	
PHP	Eclipse (plug-in)	Eclipse (plug-in)	PHPUnit	PHP Test Coverage [SEM 05]	C like language with object extensions in the last release.
Javascript	Eclispe WST, MyEclipse AJAX Tools, JSEclipse	MyEclipse AJAX Tools	JSUnit [JSU]	JavaScript Coverage Validator [SOF 06]	Java like language. Often used in html files and mixed with html.
Perl				Devel::Cover	Awkward Syntax
XSLT	XML Spy, Eclipse (several plug-ins)	e.g. XML Spy	NA	NA	Awkward Syntax. Difficult to read.

Defining the best language for web-based applications is an endless quest as each developer who is specialized in a given language is absolutely sure that his/her language is the best one. But developers have generally a functional view of languages (PERL is the best language for text processing, etc.), and there is little useful information on these languages from a quality assurance perspective in the literature.

From the architecture perspective, Perl and PHP are equivalent (i.e. they are used as "CGI" languages). Java requires an additional server (servlet server e.g. Tomcat). Javascript is generally used on the client side.

Java is supported by a large number of tools (memory checking, performance checking, code coverage, unit testing, editors, debuggers, etc., commercial and open source).

Perl and PHP tools are more limited but code coverage tool, unit testing tool, editors and debugger exist.

Perl is often presented as a powerful language. However its syntax is far from obvious and it can be considered that Perl is more difficult to learn than PHP and Perl application are probably more difficult to maintain.

PHP is quite easy to learn by most programmers (C or Java programmers).

Java seems to be more appropriate for "quality oriented" projects. However, as said before, using Java require a more complex infrastructure.

Java and JavaScript share a number of vocabulary and syntax constructions, but the languages are intended for very different purposes. Today JavaScript is essentially used on the client side mixed with html. Therefore JavaScript is the language to implement quite complex behaviours in the presentation layer for light clients.

XSLT is used to process XML files. XSLT is powerful but quite complex to write and to read. There are a small number of XSLT debuggers and no code coverage tools for xslt has been found during the study. Therefore complex XSLT files can be difficult to write and maintain.

As an intermediate conclusion on language for web application the following can be stated:

- Perl shall be used as less as possible, and PHP or Java shall be preferred.
- Java seems to be more suitable to a PA approach than PHP. So for complex systems use Java instead of PHP and then limit the use of PHP to simple functions.
- JavaScript tools are quite limited but JavaScript is useful on the client side and cannot be avoided all the time. Furthermore the AJAX paradigm is based on an intensive use of JavaScript (one can then expect that this will have an impact on good tools production).
- XSLT is also useful on the client side as it can be processed by the navigator. Complex XSLT files shall be avoided (hard to write, hard to maintain).

4.4. COTS and Open Sources

COTS raise several issues [NAY 05], OS have some benefits but do not solve every issues:

Issues	**COTS**	**OS**
Obsolescence	Vendor policy	Depends on the community (how many people are using the OS).
Version Control	Vendor policy	Often several versions per year.
Support	By vendor but not always actually efficient.	Community, Commercial. Poor documentation for some OS.
Testing Issues (regression testing)	No information	Some OS provide testing information.
Robustness of Vendor's testing is Unknown	Robustness of Vendor's testing is Unknown	Some OS provide testing information.
Structural coverage	Inability to perform adequate structural coverage	Theoretically feasible.
Maintenance	Vendor policy.	Depends on community.
Training	Vendor	Commercial for some OS.
Undisclosed Problems	Commercial policy.	Problems can be found by anybody!
Data availability (e.g., source code, test, validation, etc.)	No.	Yes.
Development Process is known	No.	Sometime.
Security	No “open” information.	Code source available.
Maturity		
Lack of knowledge in determining the best product for your needs, i.e. at the time of the selection people do not have enough information on the various products to make a good choice.		

Table 3: Issues raised by COTS and Open Sources Components.

Security Issues with COTS:

- COTS products are inherently susceptible to intrusion
- COTS developers are:
    - Outside the control of the developing and contracting organizations!
    - COTS development personnel in all likelihood, do not possess a security clearance!

- Many COTS products are developed in designated countries which may be sympathetic and possibly even supportive of terrorist organizations!
- Outside organizations know more about your vulnerabilities than you do and can take advantage of them!
- Time bombs can be placed within code that is virtually impossible to detect without the source code, etc.!

Security Issues with OS:

- Outside organizations know the code.
- Are you able to analyse the code?

The problematic of dealing with COTS or Open Source components is not specific for web-based applications. It seems that the issues are the same than for more traditional systems. However the fact is that COTS and even more Open Sources are widely used for web-based systems, and one cannot even think of rewriting components for (at least):

- HTTP server,
- Servlet server,
- Application server
- LDAP server
- Web browser
- XML parsers
- PHP or Javascript libraries
- Search engine

Therefore using COTS or Open Sources components for web-based applications is almost mandatory.

The table below provides for each type of component (application type) the COTS/Open Sources that are the most used. It shall be noticed that things in this field can change in some months/years.

Application type	**Name**		**Comment**
	COTS	**OS**	
HTTP server		Apache	Most popular web server in use, serving as the reference platform against which other web servers are designed and judged
	Microsoft Internet Information Services (IIS)		Set of Internet-based services for servers using Microsoft Windows. It is the world's second most used web server in terms of overall websites but is perhaps the most widely used web

			server for corporate websites
Servlet server		Tomcat	Web container developed at the Apache Software Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.
		Jetty	Jetty is a pure Java based HTTP Server and Servlet Container. Jetty is released as an open source project under the Apache 2.0 License. Due to Jetty's small size, Jetty is suitable for providing web services in an embedded Java application.
Application server		JBoss	Many companies choose the JBoss Open Source J2EE application server because of its extreme ease of deployment and because it runs on almost every operating system platform out there. It provides a strong support for standards, and its modular design makes it ideal for embedding within other applications.
	BEA WebLogic		WebLogic has been one of the premier enterprise-class J2EE application servers on the market for several years now. However, the product is facing more competition as corporations become increasingly comfortable with open-source alternatives.

	IBM WebSphere		J2EE application server capable of scaling from a single mobile installation to complex, distributed enterprise systems
	Microsoft Internet Information Services (IIS)		.NET application server. Microsoft application server for Microsoft systems (Windows). Nevertheless, .NET would be more used than J2EE world wide.
LDAP server		OpenLDAP	
Web browser	Microsoft Internet Explorer (IE).		About 85% of usage share in Jan. 2006. [WIK 06]
		Mozilla Firefox	About 12% of usage share in Jan. 2006. [WIK 06]

Table 4: Most used COTS/Open source components for web-based applications.

5. Guidelines for PA

5.1. Introduction

This section proposes guidelines for product assurance for each process of software engineering.

Only activities and guidelines that are specific to web and communication based technologies are described; they are to be considered as a complement to basic software engineering best practices.

For each process of the life cycle, specific items to check (introduced in the text by "☐") are proposed and for most of them guidelines to implement or support the checking are provided:

☐ Item to be checked.

GUIDELINE X: Guideline to implement/support item checking.

Motivation: the reason to use this proposed guideline (if relevant).

5.2. Management

The main activity is the production of a software development plan including the life cycle description, activities description, milestones and outputs, the techniques to be used, and the risks identification.

The management process also includes the organization and handling of all the joint technical reviews, the definition of procedures for the management of the system interface, and the technical budget and margin management [AD4].

Check that:

☐ An estimate of the size of the software to be developed for the web-based application has been made and the estimation method is documented.

☐ An appropriate method and models for estimating activity duration have been identified.

GUIDELINE 1: Define the indicators to be used for the project estimate and work progress control taking into account the web applications specificities.

These indicators can be based on web objects for instance or the WebMO method can be used.

Motivation: It is very important to check that every aspect of web-applications have been taken into account (html pages, graphics, animated/video items, presentation layer code, domain layer code, data layer code, database).

This guideline above is linked to ECSS-Q80 §6.2.5.2.

GUIDELINE 2: The role of experts is essential for web-based applications:

- **Identify experts in each technology at the beginning of the project.**

- **Design of a CORBA based architecture shall be managed or controlled by a CORBA expert.**
- **An expert shall supervise junior developers implementing a distributed architecture.**
- **J2EE projects team must contain senior developers who are experienced in this technology.**
- **An expert shall manage the development of technically complex layers.**
- **When the application is data centric, the database configuration is critic and must be managed by an expert (Schema checking, database tuning).**
- **Check that the system architect has the adequate background regarding the technology he/she has to consider.**
- **An expert shall review the tasks duration.**

GUIDELINE 3: A specific training must be organised at the beginning of the project for each used technology.

E.g.: Developers shall be trained for CORBA and the target language CORBA mapping.

The guideline above is linked to ECSS-Q-80B §5.8.3.2.

☐ Facilities and tools are or will be available for the development, testing and administration of the web-app.

☐ An appropriate life cycle has been selected for the project.

GUIDELINE 4: Combining a classical "V life-cycle" with an iterative approach based on prototyping is strongly recommended, in particular when user validation of MMI or early design of complex algorithms is concerned.

This guideline is related to ECSS-E-40Part 1B §5.3.2.2 and §4.2 and ECSS-Q-80 §6.1.

5.3. Requirements Analysis

This process aims at producing the following items [AD4]:

- Functional and performance specifications, including hardware characteristics, and environmental conditions under which the software item executes, including budgets requirements.
- Software product quality requirements.

- Security specifications, including those related to factors which can compromise sensitive information.
- Human factors engineering (ergonomics) specifications, including those related to manual operations, human equipment interactions, constraints on personnel, and areas requiring concentrated human attention, that are sensitive to human errors and training.
- Data definition and database requirements.
- External interfaces.

Check that:

☐ All the stakeholders have been identified.

Note: Web-based applications often imply several types of users with different needs and different privileges.

☐ Analyze the different stakeholders' needs to determine the requirements the application must satisfy.

☐ Analyze the available technologies to determine which ones will be used in the application to meet those needs.

☐ Define the GUI organization/layout and the navigation requirements.

GUIDELINE 5: The use of prototypes of the presentation layer is recommended to support GUI organization and navigation requirements definition.

Forecast direct interactions with customers with prototypes and prototype the presentation layer with static pages to define the GUI organization and the navigation requirements.

This guideline is linked to ECSS-E-40Part 1B §5.2.2.3 and §5.4.2.6.

☐ Define security and privacy requirements.

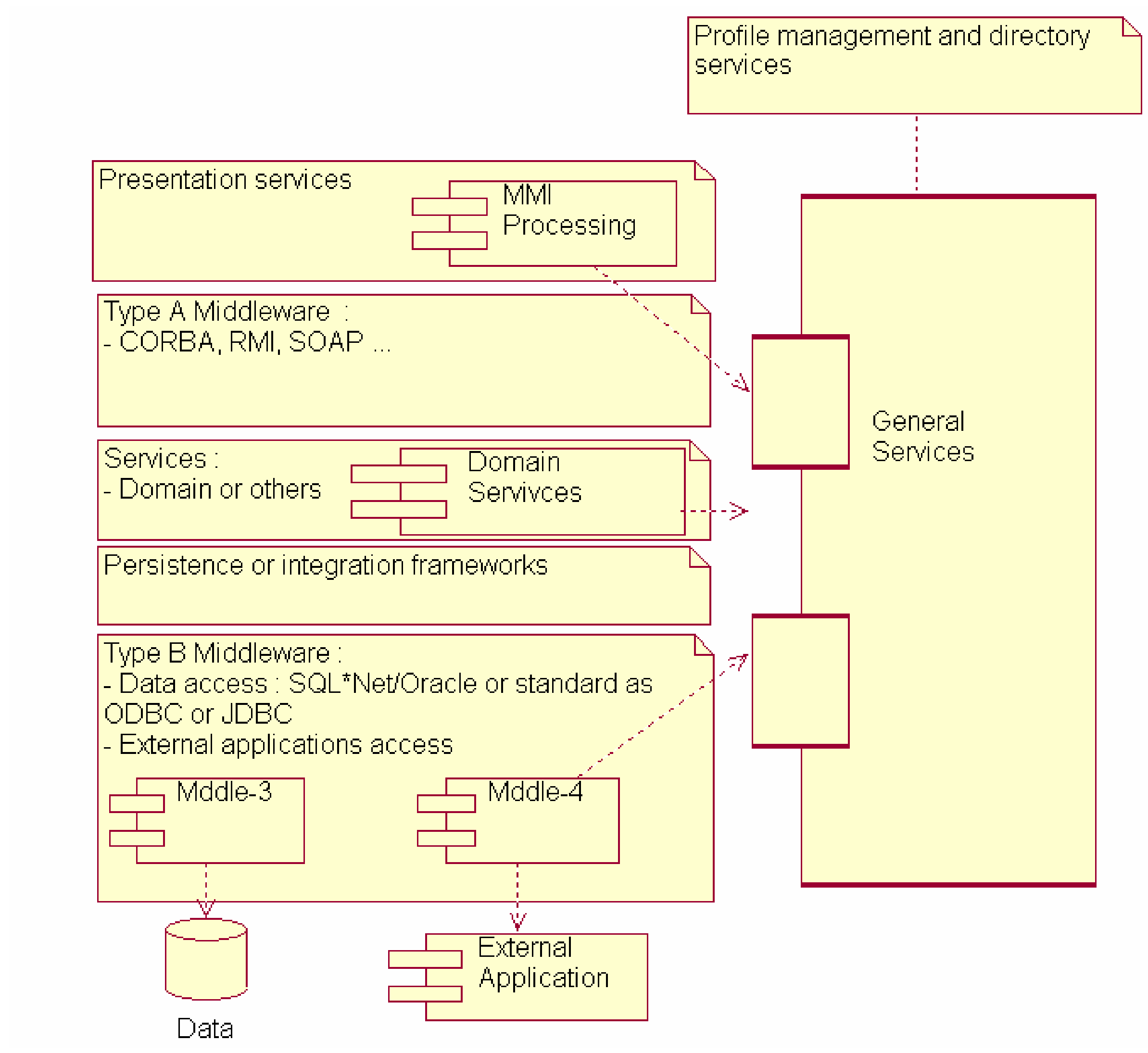
Note: privacy requirements address the need to protect the privacy of the users according to the law.

☐ Clearly define all external interfaces, i.e. browsers (one or several different browsers?), servers, to the system.

5.4. Architecture and design

This process aims at transforming the requirements for the software item into an architecture that describes its top-level structure and identifies the software components. Each software component is then refined into lower levels containing software units that can be coded, compiled, and tested.

The figure hereafter provides the representation of a generic web-based architecture.



Check that:

☐ Every application layers are described (presentation, domain, data, and service).

GUIDELINE 6: The core layers (domain, service, data) shall be detailed using UML.

GUIDELINE 7: It is recommended to combine the use of activity diagrams and use cases, since they provide efficient and complementary ways to describe internal activities of any use case of a subsystem.

GUIDELINE 8: Use a prototype to validate the presentation layer. ***It is recommended to work in integrated teams (Users and developers), via a recurrent "week meeting" during the complete prototyping phase.***

Motivation: UML must be used only to explain high level processes such as navigation (using sequence diagram for example).

This guideline is linked to ECSS-E-40Part 1B §5.2.2.3 and §5.4.2.6.

☐ Complex algorithms and performance requirements are validated using prototypes.

☐ Complete the GUI style guide for the application (layout, navigation principles).

☐ Every different language that will be used has been identified.

GUIDELINE 9: Languages to be taken into account are server side languages (e.g. Java, C++, PHP, Perl, XML) and also client side languages (such as XSL or Javascript).

It shall be noticed that some languages can be "hidden" for instance Javascript code can be "embedded" in html pages. They shall also be taken into account.

☐ A coding style guide has been generated for every different language used.

GUIDELINE 10: Coding standards:

- **May be several standards or combined.**
- **Shall be tailored to each project, environment and requirements.**
- **Needs to be enforced:**
  - **by tools (checker),**
  - **by manual verification.**

Style guides can be derived from existing style guides, for instance [CON 06] [PER 01] for PHP or [DOJ] for JavaScript.

Motivation:

- Reduces the opportunity for making errors.
- Ensures browser compatibility**.**

This guideline is linked to ECSS-Q-80Part §6.3.3.1.

☐ Guidelines for distribution aspects (e.g. rules for distributed objects usage) are available.

GUIDELINE 11: Use Design-Patterns when they are available for the selected technology (design Patterns exist for architectures based on CORBA [MOW 97], J2EE[J2E 02]).

Motivation: design patterns are well-known and proven design solutions to common design problems.

☐ An analysis of alternatives technologies and platforms has been completed.

GUIDELINE 12: The selected application framework shall be chosen with respect to the application functions complexity.

- **PHP is a good choice for applications with simple functions.**
- **For complex systems use J2EE or .NET.**

Motivation: J2EE and .NET are supported by several development environment and PA tools. PHP is simpler to learn but more difficult to maintain for large and/or complex systems.

GUIDELINE 13: Analyze the coupling mode (space, time, synchronization) required by the system and select technologies and architecture that fit the best to the identified coupling mode.

E.g. if time decoupling is required do not define an architecture based on a time coupled technology or add a time decoupling component in the architecture.

☐ All software units, i.e. screens, components, scripts, database tables, etc. have been identified.

GUIDELINE 14: Web-based applications are often composed of several small interconnected items. For instance, small components are often required (e.g. javascript functions, PHP scripts) to link other elements together.

Identifying each component allows a better cost estimation and control.

☐ Standards are used whenever they exist.

GUIDELINE 15: Avoid dedicated protocols whenever possible; use standards as much as possible.

Motivation: standards are supported by tools, tested libraries, and are less difficult to maintain as the risk of "knowledge loss" is limited.

☐ Application layers are loosely coupled.

GUIDELINE 16: The domain layer shall never return directly data (file) containing presentation information. For instance, the application layer shall preferably return XML files instead of HTML files.

Motivation: keeping a clear separation between the presentation and the rest of the application allow a better maintainability and a better evolutivity.

☐ Application components that are expected to run continuously (H24) provide monitoring functions.

GUIDELINE 17: Use application management tools in order to allow monitoring of application state and internal data while they are running (trace management, access to internal data).

Motivation: this feature is helpful for integration/validation, operations and maintenance as it provides a good way to anticipate and investigate problems.

5.5. Software design and implementation

The software design and implementation engineering process consists of the following activities:

- design of software items;
- coding and individual testing of web pages, forms, menus, scripts, domain layers, data layers.
- generating the artwork (page layout, graphics, etc.).
- components integration

Note: testing is addressed in next section.

Check that:

☐ An IDE has been selected and that it is correctly integrated to the framework (e.g. to J2EE application servers).

GUIDELINE 18: The selected IDE shall allow the integration of the following tools:

- **Compiler**
- **Debugger**
- **Interaction with the framework (deployment, debugging).**

In addition the integration of the following tools in the IDE can enhance the team efficiency:

- **Profiler (memory checker, performance analysis)**
- **Coding rules checker**

Note: The Eclipse open Source IDE provides many valuable features and many commercial or open source plug-ins are available.

The guideline above extents ECSS-Q-80 §5.8.

☐ A configuration process is implemented even for web entities (html, xml, php, pictures, etc.), and items configuration is correctly applied.

☐ The coding style guides and the usage guidelines are actually used.

GUIDELINE 19: Checking tools shall be used to help verifying the conformance of code to coding style guides. It is recommended to use, whenever possible, tools that check the coding rules during the edition process (by highlighting non conformances) instead of tools that are used to check the code from time to time. This improves the coding process as the editor directly ensures that written code is conformant to the standard

The guideline above extents ECSS-E-40 1B §5.8.3.4.

GUIDELINE 20: Use XHTML instead of HTML.

Motivation: XHTML use a more rigorous notation and can be checked more efficiently by tools.

GUIDELINE 21: Use CSS whenever possible in association with (X)HTML files.

Motivation: CSS allow the centralization of common presentation data facilitating the maintenance and the evolutions of the GUI.

GUIDELINE 22: Perl shall be used as less as possible, PHP or Java shall be preferred.

Motivation: Perl is hard to learn and to maintain.

GUIDELINE 23: Complex XSLT files shall be avoided.

Motivation: Complex XSLT files are hard to write and hard to maintain.

☐ Web-pages design and layout conform to well-established recommendations (e.g. IEEE Std 2001-2002 [IEE 03]).

5.6. Software verification and testing

Check that:

☐ Methods have been identified and described.

GUIDELINE 24: It is recommended to generate the test cases (if possible in an automatic way) from a

model. In general, the use cases views completed by the associated sequence diagrams can be used.

Motivation: This approach guarantees a good traceability between the domain model and the needed tests.

GUIDELINE 25: Web-based application shall be:

Verified by:

- **Adherence to Style Guides/Coding Standards checking.**
- **Audits and use of Checklists.**
- **Formal Inspections.**
- **Static analysis to check whether:**
  - **Files contain broken links and navigational problems, i.e. the analysis aims to ensure that every link in html pages is actually attached to a web page.**
  - **HTML, CSS, JavaScript, and ASP/VBScript code complies with W3C standards and industry guidelines.**
  - **XML is well-formed and valid.**
  - **HTML and XML files contain typos and misspelled words**

Tested using functional testing and load testing.

GUIDELINE 26: Application and data Layers shall be tested independently from the presentation layer (black box and white box testing).

GUIDELINE 27: The presentation layer shall be tested using web-box testing.

Web-box testing: Performs tests to verify that the dynamically generated output i.e. reports, query results, etc. are correct. It is also used to test exception handlers and other programming failures.[1]

Motivation: Web-box testing allows checking that dynamic pages are correctly generated.

☐ Inspection and verification are defined and performed.

GUIDELINE 28: Formal Inspections shall be performed in order to verify the following features:

- **spelling, grammar**
- **anchors (links)**

[1] As presented in "Software Assurance of Web-based Applications" document

- **adherence to coding standards**
- **error detection and prevention routines**
- **implementation of**
  - **Requirements in design**
  - **Design in code**

The guideline above extents ECSS-Q-80 §6.2.6.

GUIDELINE 29: Test verification shall include:

- **Witnesses of selected tests**
  - **Verify component functionality and integration.**
  - **Verify that dynamically generated screens contain correct information and conform to style guide conventions**
- **Monitor problem reports**
- **Verify all testing is completed satisfactorily**
- **Witness regression testing, if necessary**
  - **Review change to ensure additional errors have not been introduced**
  - **Verify all required tests are performed**

The guideline above extents ECSS-Q-80 §6.2.6.

☐ Tools have been identified and their configuration described

GUIDELINE 30: Code coverage tools shall be used whenever possible.

Motivation: If coverage tools do not exists or are not considered being relevant, additional test verification shall be planned.

GUIDELINE 31: Unit testing tools shall be used to automate unit tests.

Motivation: These tools are essential for the component testing (i.e. Junit library (http://www.junit.org).

Best practices of unit testing shall be promoted, for instance, for object languages (C++, Java), it is recommended to develop one test class (a class used for test purpose only) for each critical class (particularly for the domain classes).

GUIDELINE 32: Use a memory leak detection tool even for Java applications if they are supposed to be used several hours or days without being restarted.

Motivation: Java memory leaks exist! The garbage collector is not able to clean-up data that are referenced by forgotten relationships (lists, hash tables, etc.).

Note*: For the future and as a perspective, the "UML testing profile" provided by OMG to create a model of test is recommended.*

5.7. Software Delivery/Installation and Operation

When the application is ready for release, it is deployed on the operational platform and made available for use. The system must then be maintained up in order to meet users' requested availability. Anomalies shall be analyzed and corrected.

Check that:

- ☐ Installation means (script, installation tools, etc.) have been identified and correctly set-up.
- ☐ Every configuration file is identify and described.

GUIDELINE 33: Anticipate the management of the configuration files, document file location and content.

It can be valuable to develop a central configuration script or tool.

Motivation: The use of various COTS or Open Source tools in the final application may request the configuration of several files in different format that can be located in different directories. In case of bad configuration, it is often difficult to find where the problem is.

- ☐ The application includes management tools in order to allow monitoring of application state and internal data while the system is running (trace management, access to internal data).

GUIDELINE 34: Monitoring functions are required to manage the system and to provide observability:

- **Resources usage (memory, CPU, disk).**
- **Error log.**
- **Security log.**

- **Internal state/data exploration.**

When anomalies are identified, the monitoring functions are generally useful to help establishing the diagnosis.

5.8. Safety, reliability and security issues

Web-based applications involve many third party products or Open Source components. This is a critical issue for safety critical or mission critical applications.

In addition, applications are more and more accessible from the "external" world, typically through Internet. This involves new security issues that have to be taken into consideration in the design.

Secutity issues are handled through the application of dedicated approaches such as the Common Criteria [COM] (see 3.9).

Guidelines that are specific to web and communication based technologies and that have to be used as a complement to the "classical" standards are provided below.

Check that:

☐ COTS/Open Source components have been assessed wrt safety/security issues.

The guidelines below are linked to and extent ECSS-Q-80 §6.2.7.

GUIDELINE 35: Review/provide input to safety/security issues. Especially, define the COTS/ Open Source Components configuration recommendations, for instance when a web server is required review and define each configuration parameter (port used, directory access, etc.).

GUIDELINE 36: In order to prevent safety/security issues, remove every unused item from the COTS/Open Source Components configurations such as: unused services, samples, tests.

Motivation: unused items can sometime be used by hackers as backdoors.

GUIDELINE 37: When downloading Open Source components, always verify the integrity of the downloaded files using the signatures corresponding to their release provided on the components web site (e.g. MD5) if available. Signatures shall be detailed in the system configuration documentation.

Motivation: Checking the signatures ensures that the downloaded version is an "official" one and has not been trapped by someone.

- ☐ Safety, reliability and security objectives are well defined and that verifications levels (e.g. various level of code coverage can be required according to the criticality level : statement coverage, decision coverage, condition coverage, etc.) are defined and applied.

GUIDELINE 38: Ensure by verification that controls are implemented to reduce/eliminate security/safety issues.

GUIDELINE 39: Monitor development and testing of controls implemented to reduce/eliminate security/safety issues.

GUIDELINE 40: Ensure by verification that COTS/Open Source Components are configured according to the recommendations (as defined in guideline 36).

5.9. Dealing with COTS and Open Sources components

Even if the use of COTS or Open Source components is generalised, one can consider that the issue of dealing with COTS or Open Source components is not specific for web-based applications (§4.4), the same rules can be applied for web-based systems than for traditional systems [CAR].

See §5.8 for consideration of COTS/Open Source components impact on safety and security.

5.10. Relevance of ECSS Q80 to these guidelines

ECSS Q80 is a standard defining a set of software product assurance requirements to be used for the development and maintenance of software for space systems.

This standard is intended to be tailored to match the requirements of a particular profile and circumstances of a specific project.

The guidelines described in this technical note are focused on web and communication based systems involving a level of quality that requires good product assurance practices.

The proposed guidelines are grouped in categories that are addressed by general product assurance approaches and bring implementation details of basic PA practices (that are requested in ECSS Q80).

Therefore, ECSS Q80 is relevant to these guidelines and these guidelines shall be considered as a refinement of ECSS Q80 requirements for web and communication based systems specificities. It shall be noticed however that the security topic is not really handled by ECSS 80B. From our point of view this point shall be included as future systems will be probably more “open” and then more subject to security issues.

6. Open points

This chapter gathers some points that seem important to deal with but that are out of the scope of the current study or that require specific analysis.

6.1. Technologies not covered

The technologies covered in this document are those that have already been used in space projects or that seems mature enough to be used in a close future. This document describes their relevance with respect to applications requirements and in the next step of this study we will propose a selection of applicable technologies for each project typology.

Nevertheless other technologies for web and communication based application exist and new one are emerging each year.

Obviously, Microsoft is one of the major actors for web-based systems through its IIS platform and its .NET technology. It shall be noticed that .Net seems to be more used worldwide than J2EE and that Microsoft development platform seems to be more productive than J2EE development platforms. However, Microsoft solutions are proprietary and Windows oriented. It has been chosen in this study to focus the work on open standards that are supported by several vendors and by open sources solutions even if some elements about .NET are provided.

The XML family contains many derived languages that can be useful such as SVG (Scalable Vector Graphics, SVG is a language for describing two-dimensional graphics in XML). The study does not cover each of them.

Other technologies have not been studied such as Macromedia Flash (now Adobe Flash). Flash requires the use of the specific player (Adobe Flash Player) that is used to interpret flash files containing vector graphics, rasters, audio and video data and a scripting language. A flash file can be either loaded as a part of a web page (animation for instance) or can contain an application with navigation capabilities.

Another type of solution that is not handled here is the peer to peer technology.

6.2. Future trends

6.2.1. Technologies

The topics listed below seem promising, their progress and use have probably to be monitored.

- Web 2.0 is an emerging term for a perceived transition of the world wide web from a set of web sites to service oriented applications that are expected to replace desktop applications.
- The use of workflow based technologies (such as BPEL) to define and implement domain processes and services collaborations will probably grow in the next years.

6.2.2. Model Driven Engineering[2]

The "*Object Management Group*" (OMG: http://www.omg.org/) was created in 1989 to provide frameworks and standards for the integration of object-oriented applications, mainly the CORBA and technologies.

The growing diversity of techniques and platforms used in software developments, as well as the emergence of non-CORBA based middleware, like EJB from Sun and .NET from Microsoft, have taken the pre-eminent role away from CORBA. Then, OMG refocused its strategy and standards to support the MDA approach.

MDA addresses the complete life cycle: analysis, design, and programming aspects, providing an interoperability framework for defining modular and interconnected systems, and with the will to offer more flexibility in system integration and system evolution. A system design is organized around a set of models and a series of transformations between models, into a layered architecture.

Central to MDA is the principle of defining different models at different levels of abstraction and linking them together to form an implementation. MDA separates the conceptual elements of an application from the representation of these elements on particular implementations technologies. For that purpose a distinction is made between "*Platform Independent Models*" (PIMs), representing the conceptual design of the application, and "*Platform Specific Models*" (PSMs) which are more solution oriented.

Transformations between models are key elements in the MDA approach: vertical transformations between PIMs and PSMs to express realizations, or horizontal transformations between PSMs for integration features.

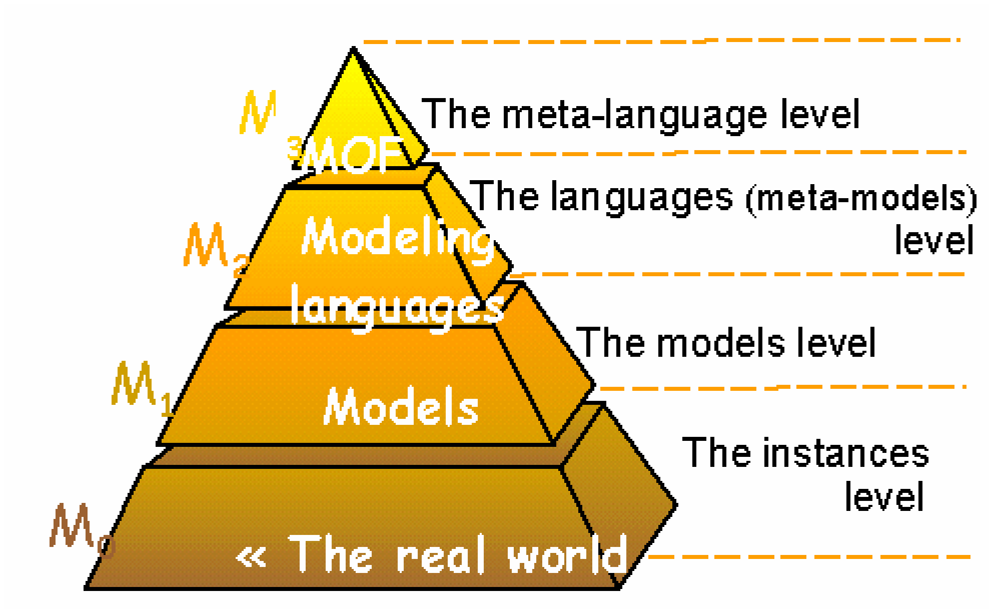
Underlying these model representations and transformations is the notion of "*meta-model*". The ability to express and transform models requires a rigorous definition of the (textual or graphical) notations supporting these models. Therefore, the notations to express models must themselves be described into models, which are called "*meta-models*". For example, the UML meta-model formally describes the notation of the different UML diagrams, giving so an unambiguous and precious common base to all tool providers and users.

The importance of meta-models has been recognized by OMG who proposed the "*Meta Object Facility*" (MOF). The MOF provides a meta-modeling hierarchy as well as a standard language for expressing meta-models. It proposes a structure of data in four different levels, so called the "*four-layer meta-model architecture*":

- the bottom M0 level, or *Application level* gives the data values i.e. the extension of an application,
- the M1 level defines the model i.e. the intention for an application,
- the M2 allows to define a schema, or language, for the application model: it is the meta-model level,
- the M3 level, or *MOF level,* proposes general concepts, defining the MOF meta-language, i.e. a schema for a meta-model: it is the meta-language level.

The *four-layer meta-model architecture* is now widely accepted and was proposed (before the OMG MDA-MOF proposals) in other standards like the ISO/IEC standard for IRDS ("*Information Resource Dictionary System"*) and CDIF ("*CASE Data Interchange Format*") from EIA (Electronics Industries Association).

[2] This chapter is an extract of the paper « The TOPCASED project: a Toolkit in Open source for Critical Aeronautic SystEms Design » presented at ERST'2006.



The four-layer meta-model architecture.

The M3 meta-modeling language used can be Ecore that is provided by the EMF ("*Eclipse modeling Framework*") project which is strongly related to Essential MOF 2.0 as specified by the OMG. Several M2 modeling language editors have been developed (ECORE, UML2, SAM, AADL) and others will follow (SYSML, SPEM, ...). The M1 level then corresponds to specific system models (an UML use case, activity or class diagram) and the M0 level to instances of the models (a real execution of the system).

6.2.3. The MDA shorted perspectives

Today one can say that MDA (MDE, MDD...) is the great adventure for the next ten years. The PSM (Platform Specific Model) concept will be very used in the distributed architectures to prepare the code generation to a J2EE, CORBA … or .NET environment. With this approach a domain application (a PIM or Platform Independent Model) will be independent of any platform and will run anywhere.

MDA/MDE will be the translation from the modelling approach initiated by UML and others as OMT to the "Model Engineering", the TRUE way, that needs reproducible design processes to transform models and to improve the development (time, and performances).

7. Bibliography

[ADV] AdventNet QEngine, http://www.adventnet.com/products/qengine/index.html

[ARS 04] Ali Arsanjani, Service-oriented modeling and architecture, IBM web site, Nov. 2004, http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/

[BEN 02] Michael Benedikt, Juliana Freire , Patrice Godefroid, VeriWeb: Automatically Testing Dynamic Web Sites, WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA, http://www2002.org/CDROM/alternate/654/

[CAR] Carnegie-Mellon Software Engineering Institute, COTS-Based Systems (CBS) Initiative, Web site, http://www.sei.cmu.edu/cbs/

[CAR 05] Carnegie-Mellon Software Engineering Institute, Message-Oriented Middleware, Web site, Last Modified Dec. 2005, http://www.sei.cmu.edu/str/descriptions/momt_body.html

[CER 00] Ceri S., Fraternali P., Bongio A. (2000). Web Modeling Language (WebML): a modeling language for designing web sites. Ninth World Wide Web Conference.

[COM] Common Criteria portal, http://www.commoncriteriaportal.org/

[CON 06] Daniel Convissor, Martin Jansen, Alexander Merz, PEAR Manual, Chapter 4. Coding Standards, 2006, http://pear.php.net/manual/en/standards.php

[COO 05] IT Projects Estimation, References and Resources, Paul Coombs, web site, last update 2005, http://www.itprojectestimation.com/estrefs.htm

[DET 97] De Troyer O., Leune C. (1997). WSDM - A user-centered design method for Web sites. 7th International World Wide Web Conference.

[DEV 00] Devel::Coverage, Perl module to perform coverage analysis, 2000, http://search.cpan.org/dist/Devel-Coverage/

[DOJ] Dojo (Open Source JavaScript toolkit), JavaScript Style Guide, web site, http://dojotoolkit.org/js_style_guide.html

[EKL 02] John Eklund, Using Partial Designs to Elicit Requirements in Web Development – a Survey of Commercial Practice, AUSWEB 02, 2002, http://ausweb.scu.edu.au/aw02/papers/edited/eklund/paper.html

[ESC 04] Maria José Escalona and Nora Koch. Requeriments Engineering for Web Applications: A Comparative Study. Journal of Web Engineering, Rinton Press, Vol. 2, No. 3, February 2004, 192-212, http://www.pst.informatik.uni-muenchen.de/personen/kochn/KochEscalonaJWE-rev.pdf.

[EWE 03] eWeek.com, Survey Questions Java App Reliability, Nov. 2003, http://www.eweek.com/article2/0,1759,1388603,00.asp

[EWE 04] eWeek.com, JBoss AS 4.0, web site, Oct. 2004, http://www.eweek.com/article2/0,1759,1668400,00.asp

[EWE 05] eWeek.com, Web Logic Server 9.0, web site, Dec. 2005, http://www.eweek.com/article2/0,1895,1900483,00.asp

[GOK 02] Aniruddha Gokhale, Bharat Kumar, Arnaud Sahuguet, Reinventing the Wheel? CORBA vs. Web Services, WWW2002, THE ELEVENTH INTERNATIONAL WORLD WIDE WEB CONFERENCE, May 2002, http://www2002.org/CDROM/alternate/395/

[GOM 00] Gomez J. , Cachero C., and Pastor O. (2000). Extending a Conceptual Modelling Approach to Web Application Design, 12th International Conference CAiSE 2000, Stockholm, Sweden, June 5-9.

[HAL 05] Gilles Halin, De la conception d'hypermédia à la conception d'application Web, Sciences et Technologies de l´Information et de la Communication pour l´Éducation et la Formation, Volume 12, 2005.

[IEE 03] IEEE Recommended Practice for the Internet-Web Site Engineering, Web Site Management, and Web Site Life Cycle, IEEE Std 2001™-2002, IEEE Computer Society, March 2003.

[ISA 95] Isakowitz T., Stohr E. A. et Balasubramanian P. (1995). A methodology for the design of structured hypermedia applications. Communications of ACM, vol 38(8): p. 34-44.

[J2E 02] J2EE Patterns Catalog, Java Blueprints, 2002, http://java.sun.com/blueprints/patterns/catalog.html

[JEE 03] Martti Jeenicke, Wolf-Gideon Bleek Ralf Klischewski, Revealing Web User Requirements through e-Prototyping, The Twelfth International World Wide Web Conference 20-24 May 2003, Budapest, HUNGARY , http://www2003.org/cdrom/papers/poster/p303/p303-jeenicke.html

[JME 05] Apache JMeter, web site, http://jakarta.apache.org/jmeter/

[JON 02] Irmen de Jong (and others), Web Services/SOAP and CORBA, April 27, 2002, http://www.xs4all.nl/~irmen/comp/CORBA_vs_SOAP.html

[JSU] JSUnit home page, http://www.edwardh.com/jsunit/

[KOC 02] Nora Koch and Andreas Kraus. The expressive Power of UML-based Web Engineering. In Second International Workshop on Web-oriented Software Technology (IWWOST02), D. Schwabe, O. Pastor, G. Rossi, and L. Olsina, editors, CYTED, 105-119, June 2002, http://www.pst.informatik.uni-muenchen.de/personen/kochn/IWWOST02-koch-kraus.PDF

[KUR 02] Tim Kurz, Testing and QA of web-based applications, PSQT/PSPT 2002 South Conference, march 2002, http://smad-ext.grc.nasa.gov/rmo/sawba/archives/SAWbAPSQTPSTT.ppt

[KUR 03] Tim Kurtz, Best Practices for Software Assurance of Web-based Applications, Guide Book, Oct. 2003, on Software Assurance Research Program Results Web Site, http://sarpresults.ivv.nasa.gov/ViewResearch/165/89.jsp.

[KUR 03a] Tim Kurz, Software Assurance of Web-based Applications, 3rd Annual NASA Office of Safety and Mission Assurance Software Assurance Symposium (OSMA SAS '03), http://sas.ivv.nasa.gov/conclusion2003/14

[KUR 04] Tim Kurtz, Software Assurance of Web Based Applications, SARP results site, Oct. 2001-Sept. 2004, http://sarpresults.ivv.nasa.gov/ViewResearch/251/89.jsp

[LAC 05] Lachlan Aldred, Wil M.P. van der Aalst, Marlon Dumas, Arthur H.M. ter Hofstede, On the Notion of Coupling in Communication Middleware , 2005 Symposium on Distributed Objects and Applications, Cyprus, 2005, http://sky.fit.qut.edu.au/~aldredl/integration/coupling/couplingNotion.html

[LAN 02] Michael Lang, Hypermedia Systems Development: Do We Really Need New Methods?, In Cohen, E. & Boyd, E. (eds), Proceedings of the Informing Science + IT Education Conference, Cork, Ireland, June 19-21 2002, pp. 883-891. InformingScience.org. ISBN 1535-0703. http://www.is.nuigalway.ie/mlang/research/Lang148Hyper.pdf

[LAR 02] Thomas Laramee , Should you go with JMS? Why JMS isn't always the best solution for distributed system development, JavaWorld, Oct. 2002, http://www.javaworld.com/javaworld/jw-10-2002/jw-1025-jms.html

[LEE 98] Lee H., Lee C., Yoo C. (1998). A scenario-based object-oriented methodology for developing hypermedia information systems. 31st Annual Conference on Systems Science.

[MEL 02] Santiago Meliá, Andreas Kraus and Nora Koch. MDA Transformations Applied to Web Application Development. In 5th International Conference on Web Engineering (ICWE 2005), Sydney, Australia,David Lowe and Martin Gaedke (Eds.). LNCS 3579, ©Springer Verlag, 465-471, July 2005, http://www.pst.informatik.uni-muenchen.de/personen/kochn/melia-icwe2005-transf.pdf

[MEL 05] Santiago Meliá, Jaime Gómez and Nora Koch. Improving Web Design Methods with Architecture Modeling. In 6th International Conference on Electronic Commerce and Web Technologies (EC-Web 2005), Copenhagen, Denmark, Kurt Bauknecht, Birgit Pröll, Hannes Werthner (Eds.). LNCS 3590, ©Springer Verlag, 53-64, August 2005. http://www.pst.informatik.uni-muenchen.de/personen/kochn/melia-ecweb05-websaProfile.pdf

[MIC] Software Measurements, Mark Micallef, Slides, http://www.cs.um.edu.mt/~ecac1/files/csa2821_software_measurement.pdf

[MOW 97] Thomas J. Mowbray, Raphael C. Malveau, Corba Design Patterns, , John Wiley & Sons Ed., 1997

[NAY 05] Warren P. Naylor, COTS and Safety – Are They Mutually Exclusive?, Federal Aviation Administration (FAA) web site, 2005, http://www.faa.gov/aio/common/documents/Safety/COTSaf.pdf

[OAS 06] OASIS (Organization for the Advancement of Structured Information Standards), Reference Model for Service Oriented Architecture 1.0, Public Review Draft 1.0, 10 February 2006, http://www.oasis-open.org/committees/download.php/16628/wd-soa-rm-pr1.pdf

[PAR 04] Product Review: Parasoft WebKing, WebSphere Journal, oct. 2004, http://websphere.sys-con.com/read/46844.htm

[PAR] Parasoft WebKing®, Automated Web application testing – performs Web site risk analysis, functional testing, load and performance testing, and security analysis. http://www.parasoft.com/jsp/products/home.jsp?product=WebKing

[PER 01] Tim Perdue, Best Practices: PHP Coding Style, in PHPBuilder.com, 2001, http://www.phpbuilder.com/columns/tim20010101.php3?page=1

[REI 00] Donald J. Reifer, Web Development: Estimating Quick-to-Market Software, 15th International Forum on COCOMO and Software Estimation, 2000.

[REI 02-a] Web Objects Counting Conventions, Donald J. Reifer, Web Objects White Paper, March 2002, http://www.reifer.com/documents/WOCounting.doc

[REI 02-b] Donald J. Reifer, Reifer Consultants, Inc., Estimating Web Development Costs: There Are Differences, June 2002, http://www.stsc.hill.af.mil/crosstalk/2002/06/reifer.html

[RIS 03] Réseau d'Ingénierie de la Sûreté de fonctionnement, Compte Rendu de la première réunion du groupe de travail « Intergiciel et sûreté de fonctionnement », LAAS-CNRS, 3 Juin 2003, http://www2.laas.fr/RIS/GT/IM/

[RUH 03] Melanie Ruhe, Ross Jeffery, Isabella Wieczorek, Using Web Objects for Estimating Software Development Effort for Web Applications, Ninth International Software Metrics Symposium (METRICS'03)

[SCH 01] Object Interconnections: CORBA and XML — Part 3: SOAP and Web Services, Douglas C. Schmidt and Steve Vinoski, C/C++ Users Journal, 2001, http://www.cuj.com/documents/s=7990/cujcexp1910vinoski/vinoski.htm

[SCH 96] Schwabe D., Rossi G. et Barbosa S. D. J. (1996). Systematic hypermedia design with OOHDM. In proceedings of Hypertext'96 Conference, Washington DC, USA., March 16-20, 1996

[SEM 05] PHP Test Coverage, Semantic Designs , http://www.semdesigns.com/Products/TestCoverage/PHPTestCoverage.html

[SER 03] ServerWath, Application Servers – WebSphere, 2003, http://www.serverwatch.com/stypes/servers/article.php/15989_3101851

[SOF 06] JavaScript Coverage Validator , Software Verification Ltd http://www.softwareverify.com/javaScriptCoverageValidator/index.html

[SOF] Software QA Testing and Test Tool Resources, web site, http://www.aptest.com/resources.html

[STR 06] Apache Struts Project, web site, last update 2006, http://struts.apache.org/

[TAK 97] Takahashi K. and Ling E. (1997). Analysis and Design of Web-based Information Systems, Proc. of Sixth International World Wide Web Conf., April. pp.377-389.

[TCP] Summary Comparison of TCP/IP Transport Layer Protocols (UDP and TCP), web site, http://www.tcpipguide.com/free/t_SummaryComparisonofTCPIPTransportLayerProtocolsUDP.htm#Table_160

[THE 05] TheServerSide, Application Server Matrix, web server, Last update: 03/27/05, http://www.theserverside.com/reviews/matrix.tss

[THO 98] Thomson J., Greer J., and Cooke J. (1998). Algorithmically detectable design patterns for hypermedia collections, Workshop on Hypermedia development Process, Methods and Models. Hypermedia,.

[VIT 99] Vitali F., Versioning hypermedia, ACM Computing Surveys, Dec. 1999, http://www.cs.brown.edu/memex/ACMCSHT/50/50.html

[W3C 03] CSS & XSL, W3C, last update 2003, http://www.w3.org/Style/CSS-vs-XSL

[WHI 99] Whitehead Jim, The Future of Distributed Software Development on the Internet From CVS to WebDAV to Delta-V, 1999, http://www.webtechniques.com/archives/1999/10/whitehead/

[WHI a] Whitehead JR E. JAMES, WIGGINS MEREDITH, WEBDAV: IETF Standard for Collaborative Authoring on the Web, http://ftp.ics.uci.edu/pub/ietf/webdav/intro/webdav_intro.pdf

[WHI b] Whitehead Jim, DeltaV: Adding Versioning to the Web, WWW10 Tutorial Notes, http://www.webdav.org/deltav/WWW10/deltav-intro.htm

[WIK 06] Wikipedia, Usage share of web browsers, web site, last update, 1/4/2006, http://en.wikipedia.org/wiki/Usage_share_of_web_browsers

[WIK 06b] Wikipedia, Comparison of web servers, web site, last update April 2006, http://en.wikipedia.org/wiki/Comparison_of_web_servers

[WIK 06c] Wikipedia, Ajax (programming), web site, last update 7/4/2006, http://en.wikipedia.org/wiki/AJAX

[WIK 06d] Wikipedia, Matrix of Application Servers, web site, last update April 2006, http://en.wikipedia.org/wiki/Matrix_of_Application_Servers

[WST 06] Software QA and Testing Resource Center, Web Site Test Tools and Site Management Tools, http://www.softwareqatest.com/qatweb1.html

[ZIM 04] Olaf Zimmermann, Pal Krogdahl, Clive Gee, Elements of Service-Oriented Analysis and Design, IBM web site, June 2004, http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/

Note: web-sites listed in this section have been accessed in March-April 2006.