PICNIC Technology[*]

Dimitrios G. KATEHAKIS[1], Morten BRUUN-RASMUSSEN[2],
Vesa PAKARINEN[3], David PIGGOT[4], Niilo SARANUMMI[3]

[1] *FORTH, Institute of Computer Science, (katehaki@ics.forth.gr), Greece*

[2] *MEDIQ (mbr@mediq.dk), Denmark*

[3] *VTT Information Technology (vesa.pakarinen@vtt.fi), Finland*

[4] *Integrity Consulting Partners/ GMS(P)B (davidmpiggott@hotmail.com), U.K./ Ireland*

[3] *VTT Information Technology (Niilo.Saranummi@vtt.fi), Finland*

Abstract. A key objective of the Professionals and Citizen Network for Integrated Care (PICNIC) project was to provide products for a European and potentially worldwide software market. The approach followed was through the delivery of a number of Open Source (OS) components, to be integrated into applications that deliver similar services across the participating regions, aiming at their exploitation by other regions and the industry. This chapter describes the technology developed during the lifecycle of the PICNIC project, focusing on the three core services of Clinical Messaging, Access to Patient Data, and Collaboration. For each service, the entire process of how to turn its functional specifications into reusable components and common data sets in order to support Information Technology (IT) services for the next generation of secure, user-friendly healthcare networks is presented by means of common documentation tools. Security and privacy issues are also addressed.

1 Introduction

Health telematics applications are presently revolutionising the developments not only in diagnosis, treatment, surveillance and rehabilitation of patients, but also on the side of the more collective aspects of healthcare and disease prevention such as clinical trials, epidemiology and health education. Moreover, for the first time in the history of healthcare the emerging *Health Information Infrastructure* (HII), i.e. telematics networks together with a set of technologies, health telematics applications and services, is making possible the rapid dissemination and sharing of health information and research results. This is leading to knowledge creation and to promotion of innovative approaches based on evidence collected in medical practice all over the world.

As it has already been presented in the “PICNIC Story” chapter, the project started with the aim of taking the concept of the 18 *Work In Synergy for Europe* (WISE) project [1] services required by a Regional Health Care Network (RHCN), in order to deliver a full range of IT services to any “member” healthcare organisation within the RHCN. The methodology followed in order to turn the functional specifications provided by IT into reusable components across services and regions, in order to support the next generation

[*] Dimitrios G. Katehakis, Morten Bruun-Rasmussen, Vesa Pakarinen, David Piggott, and Niilo Saranummi: “PICNIC Technology”, in “Regional Health Economies and ICT Services: The PICNIC Experience”, IOS Press, Volume 115 Studies in Health Technology and Informatics, pp. 61-91, hardcover, ISBN-13: 978-1586035389, August 2005. (http://www.ics.forth.gr/~katehaki/publications/picnic_technology2005.pdf)

RHCNs, is briefly described in Section 2 (From Functional Specifications to Common Components). More specifically, the process followed for the three services of top priority to the regions (i.e. Clinical Messaging, Access to Patient Data, and Collaboration) is presented in Section 3 (Core IT Services). Section 4 (Software Components) discusses performance and conformance issues related to the components identified and specified to be developed within PICNIC, while Section 5 (Security and Privacy) addresses issues related to security and confidentiality. Finally Section 6 (Conclusions) concludes on the PICNIC technology.

2 From Functional Specifications to Common Components

The kind of common tools and methods needed in PICNIC have been identified as being Unified Modelling Language (UML)-based [2]. During the course of the project certain elements from the Rational Unified Process (RUP) were also introduced in the organization and methodological process followed. The RUP activities create and maintain models, and rather than focusing on the production of large amount of paper documents, it emphasizes the development and maintenance of models (i.e. semantically rich representations of the software system under development) [3][4].

The RUP supports an iterative approach to development that addresses the highest risk items at every stage in the lifecycle, significantly reducing a project's risk profile. This iterative approach helps attack risk through frequent, demonstrable, executable releases that enable progress (i.e. frequent, executable releases that enable continuous end user involvement and feedback). Because iterations end with an executable release, the development team stays focused on producing results, and frequent status checks help ensure that the project stays on schedule. An iterative approach also makes it easier to accommodate tactical changes in requirements, features or schedule.

The RUP describes how to elicit, organize, and document required functionality and constraints; track and document tradeoffs and decisions; and easily capture and communicate business requirements. The notions of use cases and scenarios prescribed by the RUP process have proven to be an excellent way to capture functional requirements and to ensure that these drive the design, implementation and testing of software, making it more likely that the final system fulfils the end user needs [5].

In addition, the RUP supports *component-based software developmen*t. *Components are non-trivial modules, subsystems that fulfil a clear function.* The RUP provides a systematic approach to defining an architecture using new and existing components. These are assembled in a well-defined architecture, either ad hoc, or in a component infrastructure such as the Internet, .Net, or the Common Object Request Broker Architecture (CORBA), for which an industry of reusable components is emerging [6].

There are nine core process workflows in the RUP, which represent a partitioning of all workers and activities into logical groupings, as shown on Figure 1.

Although the names of the six core engineering workflows may evoke the sequential phases in a traditional waterfall process, *the phases of an iterative process are different* and *these workflows are revisited again and again throughout the lifecycle*. The actual complete workflow of a project interleaves the nine core workflows, and repeats them with varying emphasis and intensity.

At the initial phase of a project, such as PICNIC, most of the effort is directed towards Requirements, Analysis and Design. The goal is to describe what the system should do and allow the developers and the end users to agree on that description. To achieve this task PICNIC elicited, organized, and documented required functionality and constraints [8]; while at the same time it tracked and documented tradeoffs and decisions.

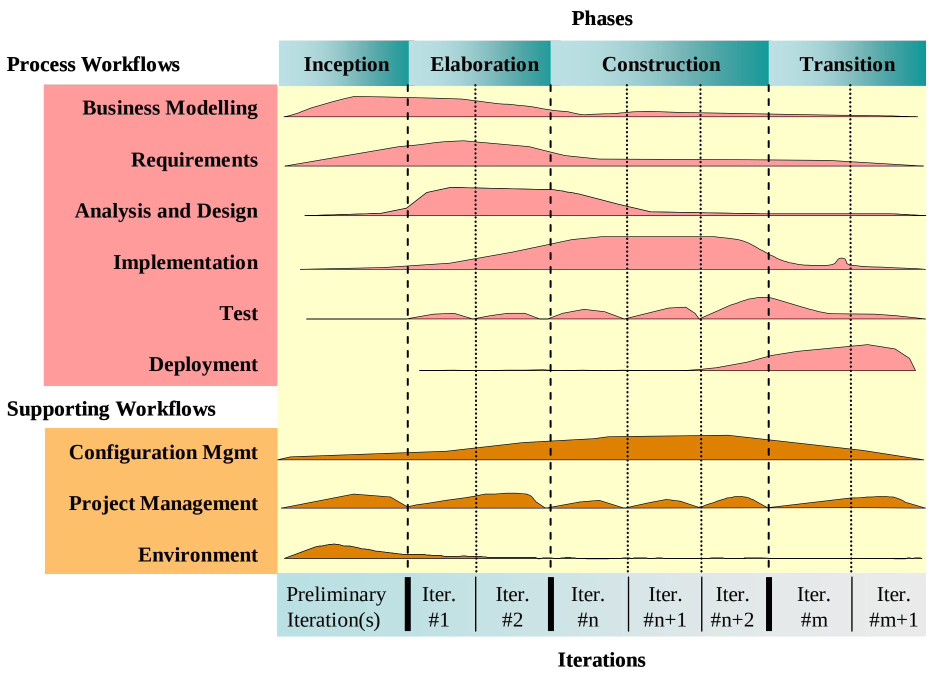


Figure 1 – The nine core process workflows (sequence of activities that produces a result of observable value) are divided into six core "engineering" workflows and three core "supporting" workflows (from [7]).

Actors were identified, representing the users, and any other entity that must interact with the system being developed. Use cases were also provided, representing patterns of behaviour each of the selected IT services exhibit. Because use cases were developed according to the actors' needs, the system was more likely to be relevant to each user. Therefore a harmonisation process had to be applied in order to bring all the related use cases in a state where commonalities, rather than peculiarities, were exposed. Each use case was described in detail.

The use-case descriptions showed how the IT service environment interacts step by step with the actors and what it does. The presented summary output results in a design model and optionally an analysis model. The design model serves as an abstraction of the source code; that is, the design model acts as a 'blueprint' of how the source code is structured and written. The design model consists of design classes structured into design packages and design subsystems with well-defined interfaces, representing what will become components in the implementation. It also contains descriptions of how objects of these design classes collaborate to perform use cases.

The final component specification documents produced [9][10][11] correspond to the elaboration phase, and as such they address the critical use cases identified in the inception phase, which typically expose the major technical risks of the project. While an evolutionary prototype of a production-quality component is always the goal, this does not exclude the development of one or more exploratory, throw-away prototypes to mitigate specific risks such as design/ requirements trade-offs, component feasibility study, or demonstrations to investors, customers, stakeholders and end-users.

3 Core IT Services

The application layer of any reference architecture consists of applications, which support user activities in the various areas of an organization. These applications are both information sources and/ or information access points. Clinical, diagnostic, and administrative information systems, diagnostic imaging repositories, medical libraries, and user-oriented services are all part of the application layer. All applications and services of the application layer make use of their own data model and user-interface. Clearly in PICNIC all regions shared the need for integration of systems into an interoperable infrastructure.

The main challenge today, in regard to the development of integrated RHCN offering advanced health telematics services, has to do with the development of the appropriate HII that will be in a position to support all the requirements of the healthcare domain. This requires good knowledge and understanding of the domain together with the definition of the corresponding and necessary reference architecture, which defines the framework for the integration of heterogeneous, autonomous, distributed information systems that are networked. In addition to that, the definition of public and stable interfaces and protocols is also required together with the provision of the appropriate middleware and user-oriented services.

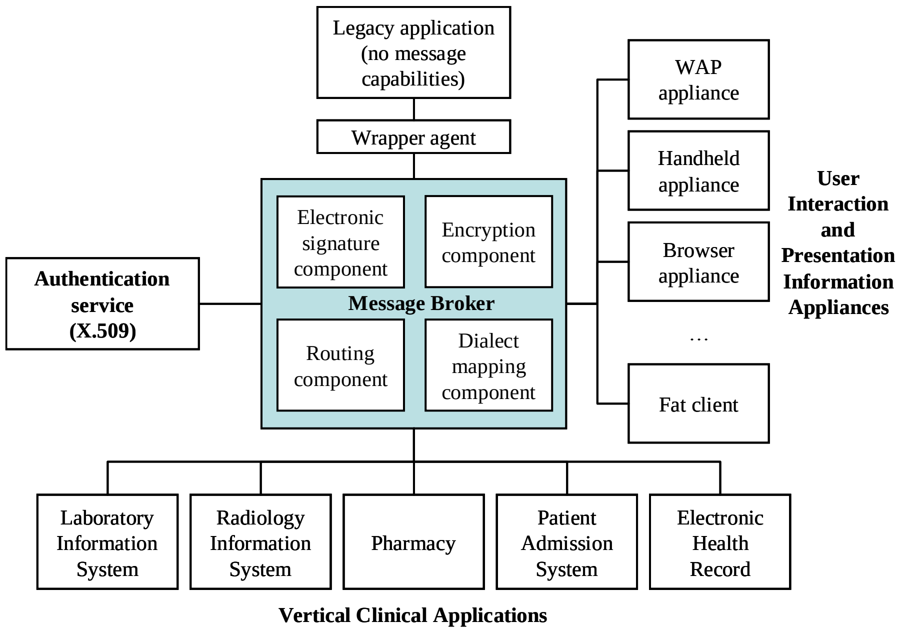
PICNIC developed some of the components that are required by the underlying HII, giving them a European perspective and aiming at piloting them across several European regions. The OS approach ensures that the mechanisms are in place for creating a community of developers sharing a common interest, and creating industries around it to provide the end-users added-value services of high quality. Their development has been concentrated in three groups, corresponding to the three greatest priority areas for PICNIC regions:

- *Messaging* concentrates on the exchange of clinical and administrative data between different applications and includes the use of a various number of already developed standards;
- *Access to Patient Data* focuses on the development of an integrated environment for professionals or citizens who need a uniform way to access parts of patient record data that are physically located in different clinical information systems, and should not be confused with autonomous Clinical Information Systems (CISs), message based communication of Electronic Health Record (EHR) data, centralized Clinical Data Repositories (CDR), or monolithic information systems that have embedded in their structure mechanisms for accessing directly host systems; and
- *Collaboration* is involved with the development of an environment for the provision of examination, monitoring, treatment and administration of patients through immediate access to expertise and patient information regardless of where the patient or relevant information is geographically located.

3.1 Messaging

Clinical Messages was one of the first IT services developed in a RHCN and consists of highly structured patient-related information, concerning the treatment of an individual. Messaging makes the exchange of form-based information such as prescriptions, laboratory results, referrals, and discharge summaries almost automatically possible between different healthcare providers. When communicating clinical messages, a “store and forward” e-mail technique is often used providing the opportunity to communicate 24 hours/ day. Because such messaging is suitable for standardisation, national or regional standards make it possible to integrate clinical messages in IT applications, already in use by the professionals.

A very common way to set up the messaging infrastructure is through a "message broker", which is usually installed as a service in the network. The "message broker" is a middleware component, which has the capability of interpreting different messaging protocols and translating between various dialects. This technique can be of great value in preserving existing investment and facilitating the introduction of new technologies. The message broker can interact with an authentication service, which ensures validity of senders and receivers (see Figure 2).



Legacy application (no message capabilities)
Wrapper agent
Electronic signature component
Encryption component
Message Broker
Routing component
Dialect mapping component
Authentication service (X.509)
WAP appliance
Handheld appliance
Browser appliance
…
Fat client
User Interaction and Presentation Information Appliances
Laboratory Information System
Radiology Information System
Pharmacy
Patient Admission System
Electronic Health Record
Vertical Clinical Applications

Figure 2 – The message broker architecture for messaging.

The main actors of the Messaging IT service have been identified in PICNIC to be:

- the *End User* that can be either an *Administrator*, a *General Practitioner*, a *Nurse*, a *Physician*, a *Radiologist* or any *Healthcare Professional*;
- the *Maintainer* that represents a person who is responsible for the maintenance, expansion and enhancement of the system with new features; and
- the *Existing IT System* that represents current systems that holds the primary source of prescribing and validation information.

Features of the Messaging IT Service that can be translated into use cases are listed on Table 1.

Table 1 – Use Cases for the Messaging IT Service of PICNIC.

Use Case	Short description
Add New IT System	Adds a new system to the federation of IT systems. It is initiated by the *Maintainer*.
Access Control Rule	Defines, sets and updates the security roles and policies of the users when the try to access clinical information. It is initiated by the *Maintainer* or the *End User*.
Keep Information Up-to-date	Accesses the internal data structures so that they are kept up-to-date. It is initiated by the *Maintainer* or by an *Existing IT System*.
Access Rights and Authentication	Relates to authentication and the use of passwords and user groups to validate users and allow them to access clinical information.

Message Set-up	It is initiated by the *End User* and describes how the definition of the clinical message is set up.
Maintain Messages	It is initiated by the *End User* and describes how an existing definition of the clinical message is maintained (amendments/ deletions)
Access Clinical Information	It is initiated by the *End User* and describes how clinical information for the provision of the message content is accessed.
Message Communication	It is initiated by the *End User* and describes how the message is communicated to the intended recipients.
Provide Secure Information Communication	Provides security through all communication paths. It is used by the Access Clinical Information use case.
Keep Auditing Trails	Audits every interaction between the various entities of the system.
Notification	A notification is send to the *Maintainer* of the system if a change has been made to a Clinical Messages.
Terminology Service	A service or function that assists the user to translate codes between different coding systems.

The corresponding UML diagram is depicted in Figure 3.

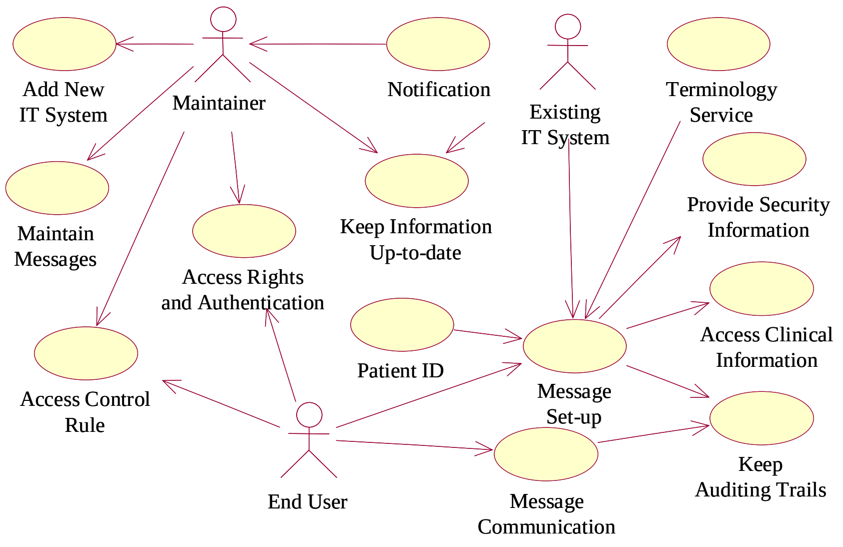


Figure 3 – Main use case diagram for the Messaging IT service.

The main software entities that can be represented as messaging components are the following:

- Message Set Up Service Component;
- Claims and Payments Service Component;
- Clinical Observation Access Component;
- Patient Identification Service Component;
- Shared Records Indexing Service Component;
- Shared Records Update Broker Component;
- Auditing Service Component;
- Authentication Service Component;
- Message Broker Component;
- Encryption Service Component;
- Resource Service Component;

- Terminology Service Component; and
- User Profile Service Component.

Clinical Document Architecture (CDA) Release 1

Today there are a number of internationally (partly) competing standards relating to messaging content. The earliest of these standards is the United Nations' Electronic Data Interchange (EDIFACT), which is today being used extensively also for healthcare messaging. Lately this committee has also decided to separate content from technology and does not specify EDIFACT as the only implementation technology. Since the mid-90s the Technical Committee 251 of the European Committee for Standardization (CEN-TC251) [12] gradually developed and refined an UML-based domain information model methodology that also heavily influenced the V3 methodology of Health Level Seven (HL7). Furthermore, today CEN-TC251 and HL7 work closely together in the messaging area. HL7 V2.x is another example of a widely used message standard that does not separate content from the envelope.

True separation is now the accepted norm and work is proceeding on those lines first to set up international standards for content and then to specify implementations with various technologies [13]. HL7's V3 [14] with its generic Reference Information Model is the current goal that is being pursued not only by the United States but also by several HL7 affiliates across the globe (more than 20 today) representing both local industries and users. Additionally, as mentioned above CEN-TC251 is heavily involved in this as well. Naturally, there are some that fear that this will lead to a standard set by the Americans. As in all volunteer standards work, the outcome is defined by those who participate. On the other hand, a standard is only used if there are benefits in using it.

Today CDA Release 1 (R1) is an American National Standards Institute (ANSI) approved standard. CDA is intended as a framework (architecture) for describing the structure of clinical documents. It focuses on the structuring of clinical documents. Future extensions (R2 and R3) will provide more guidance/ requirements for the structure of clinical documents. The standard is based on the notion that "all medical information" is stored and read as documents. CDA seeks to bring structure to these documents in order to make them interoperable.

Presently, there is no point (nor possibility) to require the data contained in a RHCN to comply with RIM. Instead, and for the time being, it makes sense to require that the contents that will be exchanged in the form of messages or shared in the context of a shared teleconsultation folder complies with the requirements of CDA R1. This is because:

- CDA R1 is only concerned with the structure, not the contents with the exception of the header for which there are mandatory requirements.
- Having a header complying with CDA "does not hurt anyone". Everyone has to provide info about the contents of any clinical document. Hence everyone will have to provide a header. Why not agree that this header information is structured in accordance with CDA R1?
- CDA allows documents to be exchanged with the Extensible Markup Language (XML). XML implementations are the goal for all concerned. Again, the same argument: why not agree to structure the header in a common way?
- CDA offers a migration path towards full Reference Information Model (RIM) compliance once V3 is available.

The *CDA Header* consists of:

- document information;
- encounter data;
- service actors (such as providers): responsible, authenticators, recipients, originators, organization, etc.;

- service targets (such as patients); and
- localization.

A CDA document is not a message. Messages can be used to exchange CDA documents in which case the CDA document is the "payload" and the message provides the wrapper. The message can be in HL7 v2.x or v3. The message contains the data on who the addressee is etc. The header info of a CDA document is meant for the identification of the document not to be used for addressing the document to a receiver.

The *CDA R1 Body* consists of:

- shared xml attributes (id, confidentialities, language);
- document body and sections (+ captions, non_xml body);
- document structures (paragraphs, lists, tables); and
- document entries (character data, content, links, coded, observation media, localization).

Harmonised CDA Header

The clinical document header for the harmonised PICNIC CDA R1 header message includes all of the elements in Table 2. Table 2 includes those elements within the overall CDA R1 standard that were considered relevant to the participating PICNIC regions in relation to their PICNIC prototypes [15].

Table 2 – CDA header elements.

Element	Contains	Mandatory
id	The document identifier within a specific environment and the identifier of that environment.	Yes
set_id	An identifier that remains constant across all document revisions/versions.	Yes
version_nbr	The version of the document used.	No
document_type_cd	Code Value, Display Name corresponding to the code value, and Code System Name.	Yes
origination_dttm	The date & time the service being documented took place.	Yes
copy_dttm	The date & time a document is released from a document management system that maintains revision control over the document.	No
confidentiality_cd	Code Value, Display Name corresponding to the code value, and Code System Name.	Yes
fulfills_order	Order type & order document id to which the defined document is a response.	No
patient_encounter	Encounter data include an optional, globally-unique encounter identifier; a required encounter time stamp; and an optional encounter location, which includes a globally unique location identifier and an address.	No
authenticator	Authenticator type and signature which can be used to hold a code which shows that an electronic signature is held on file, or is required to be provided for this document.	No
legal_authenticator	Legal authenticator type and signature which can be used to hold a code which shows that an electronic signature is held on file, or is required to be provided for this document.	No
originating_organization	A number of child elements, including originating organization	Yes

	type, name and id.	
provider	Provider type (i.e assistant performer, consultant, or performer) and function (e.g. admitting physician).	Yes
service actor	Service actor type.	No
patient	A number of child elements, including patient type and id.	Yes
local_header	Name and value of local attributes.	No

Every element has its own sub-structure composed of either child elements and/ or attributes.

3.2 Access to Patient Data

Access to Patient Data has been defined by PICNIC to be an IT service for professionals or citizens who need a uniform way to access parts of patient record data that are physically located in different clinical information systems. This end-user IT service provides fast, secure and authorized access to distributed patient record information from multiple, disparate sources. This service should not be confused with autonomous CISs, message based communication of EHR data, centralized CDRs, or monolithic information systems that have embedded in their structure mechanisms for accessing directly host systems. In that sense PICNIC provided an environment for integrated, round-the-clock access to clinically significant information which is kept at the place where it is produced and maintained by the most appropriate clinical information system in all cases. From the provided analysis, it has been deduced that the following elements are necessary in order to develop the service:

- information propagation from CISs to the middle layer of the HII;
- a component residing at the middle level of the "architecture" managing the required minimum data sets, as well as indexing; and finally
- a Graphical User Interface (GUI) to make available the Access to Patient Data to the end users (See Figure 4).

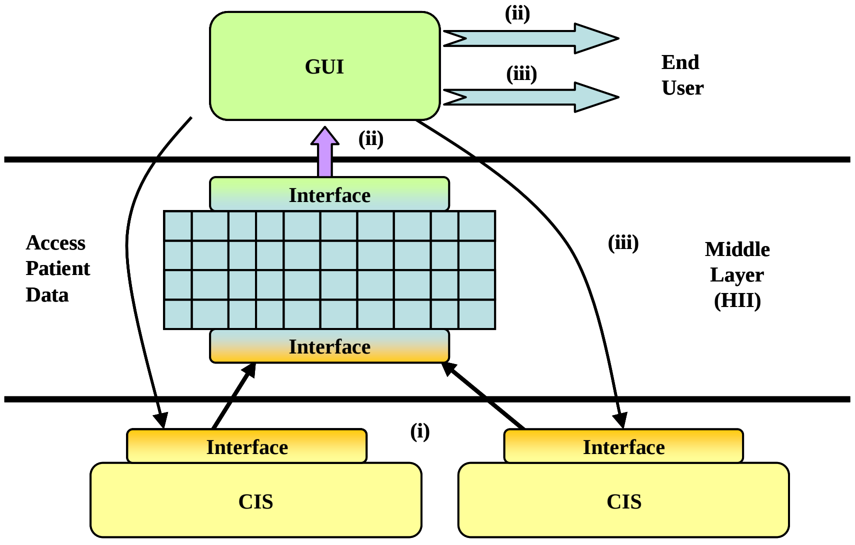


Figure 4 – In order for the end user to access patient data, (i) individual CISs propagate updated information to the middle layer, (ii) the end user retrieves indexes, selects information of interest and (iii) directly accesses clinically significant information.

Unique patient identification, patient consent on sharing the personal information, and organisational commitment in changing the traditional way of work, are all necessary requirements, as well as the existence of the appropriate legal framework that will allow the communication of personal record fragments among the involved actors.

The main actors of the Shared Records IT service have been identified to be:

- the *End User* that represents the ultimate user of the system and can be either a *Citizen* or a *Healthcare Professional* and expect the service to be able to offer role-based, secure access to reliable, patient information 24-hours a day;
- the *Maintainer* that represents a person who is responsible for the maintenance, expansion and enhancement of the system with new features; and
- the *Existing IT System* that represents a system that is a primary source of clinical information.

According to the description provided in the functional specifications harmonised document, the Access to Patient Data IT service features that can be translated into the use cases are listed in Table 3.

Table 3 – Use cases for the Access to Patient Data IT Service.

Use Case	**Short Description**
Add New IT System	Adds a new system to the federation of IT systems. The *Maintainer* initiates this use case.
Access Control Rule	Defines, sets and updates the security roles and policies of the users when the try to access clinical information. The *Maintainer* or the *End User* initiates this use case.
Keep Information Up-to-date	Accesses the internal data structures so that they are kept up-to-date. It is initiated by the *Maintainer* or by an *Existing IT System.* It uses three "sub-use cases": Provide Secure Information Communication, Keep Auditing Trails, and Semantic Mapping.
Access Rights Authentication	Relates to authentication and the use of passwords and user groups to validate users and allow them to access clinical information.
Access Clinical Information	It is initiated by the *End User* and described how he/she can access clinical information. It uses three "sub-use cases": Provide Secure Information Communication, Keep Auditing Trails, and Semantic Mapping.
Provide Secure Information Communication	Provides security through all communication paths. It is used by the Access Clinical Information use case and the Keep Information Up-to-date use case.
Keep Auditing Trails	Audits every interaction between the various entities of the system. It is used by the Access Clinical Information use case and the Keep Information Up-to-date use case.
Semantic Mapping	Performs the translation between languages and coding schemes. It is used by the Access Clinical Information use case and the Keep Information Up-to-date use case.
Add New IT System	Adds a new system to the federation of IT systems. The *Maintainer* initiates this use case.
Access Control Rule	Defines, sets and updates the security roles and policies of the users when the try to access clinical information. The Maintainer or the *End User* initiates this use case.
Keep Information Up-to-date	Accesses the internal data structures so that they are kept up-to-date. It is initiated by the *Maintainer* or by an *Existing IT System.* It uses three "sub-use cases": Provide Secure Information Communication, Keep Auditing Trails, and Semantic Mapping.

The corresponding high-level UML diagram is depicted in Figure 5.

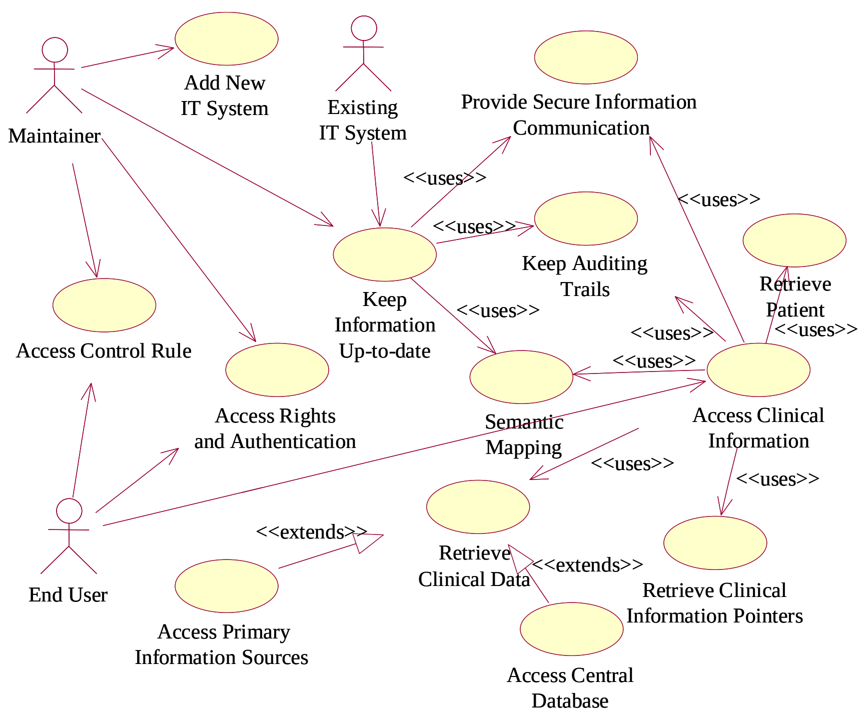


Figure 5 – Main use case diagram for accessing patient data.

The main software entities that can be represented as access to patient data components are the following:

- Clinical Observation Access Component (COAS);
- Patient Identification Service Component (PIDS);
- Shared Records Data Service Component;
- Shared Records Indexing Service Component (SRIS);
- Shared Records Update Broker Component (SRUB);
- Auditing Service Component (AUDS);
- Authentication Service Component (AUTS);
- Encryption Service Component (ENCS);
- Resource Service Component (RESS);
- Terminology Service Component (TERS); and
- User Profile Service Component (UPS).

A brief description of components developed in PICNIC follows, while their synergy is depicted in Figure 6.

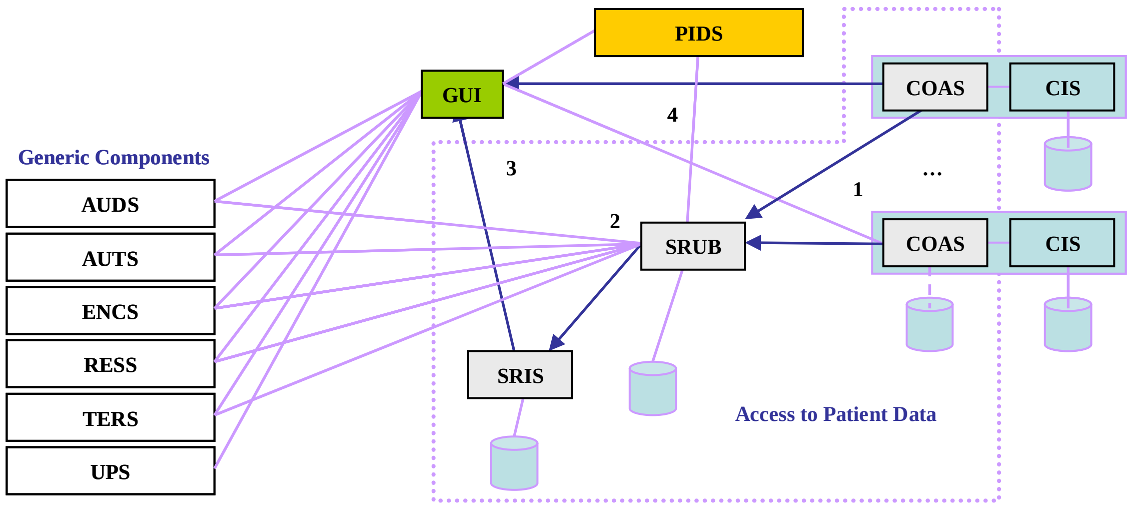


Figure 6 – The access to patient data architecture. (1) Indexing information can be either extracted by SRUB through COAS (pull model), or forwarded by locally installed COASs to SRUB (push model). (2) SRUB is responsible for transforming indexing information to a SRIS readable format. (3) The end user through SRIS accesses indexing information. (4) Once the location becomes known, the end user uses COAS to access clinical data. In most cases, the synergy of one or more of the auxiliary components already identified is required (e.g. TERS, RESS, etc.).

Patient Identification Service Component

The PIDS component allows for the unique association of distributed patient record segments to a master patient index. PIDS is required by all regions involved with accessing patient data for both identification (ID) and correlation purposes, as well as for the support of active propagation of dynamic information (i.e. to support instant notification, and in that way maintain consistency among the systems that manage patient demographics). PICNIC has adopted the Object Management Group (OMG) PIDS specification [16] that defines the interfaces of a CORBA PIDS that organizes person ID management functionality to meet healthcare needs.

The PIDS is designed:

- to support both the assignment of IDs within a particular ID domain and the correlation of IDs among multiple ID domains;
- to support searching and matching of people in both attended-interactive and message-driven-unattended modes, independent of matching algorithm;
- to support federation of PIDS services in a topology-independent fashion;
- to permit PIDS implementations to protect person confidentiality under the broadest variety of confidentiality policies and security mechanisms;
- to enable plug-and-play PIDS interoperability by means of a “core” set of profile elements, yet still support site-specific and implementation-specific extensions and customization of profile elements; and
- to define the appropriate meaningful compliance levels for several degrees of sophistication, ranging from small, query-only single ID domains to large federated correlating ID domains.

The adopted conceptual model of PIDS is depicted in Figure 7.

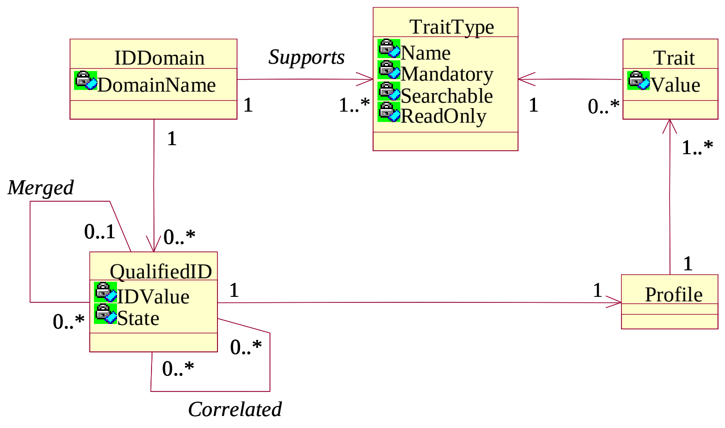


Figure 7 – The adopted OMG PIDS conceptual model: "ID Domain" defines a set of IDs among which there is to be one unique person ID value per person represented. A "Qualified ID" is represented by a sequence of characters that one or more systems in an "ID Domain" use to represent a person and bind related information. A "Trait", which is of a certain "Trait Type", is an attribute (i.e. information) that can be used to help identify a person. Traits are grouped to create a "Profile". Examples of traits include name, date of birth, gender, address, etc.

Shared Records Indexing Server Component

The SRIS component is required to provide the means for locating clinically significant information dispersed throughout the RHCN. SRIS manages data determined by the selected federated schema (i.e. indexed information can be related to existing encounters, allergies, personal information of relevance, etc.).

SRIS is designed:

- to support the ability to locate shared records information in a distributed environment;
- to leverage related specifications where appropriate (e.g. PIDS, COAS, TERS, RESS, security etc.);
- to leverage related specifications for local implementations;
- to support the ability to identify and retrieve properties of the information located (such as indication of the type, size and availability of located information, its format, count of matching instances, etc.);
- to provide for filtering, such as by patient, provider, information type, time frame, etc.;
- to support multiple, extensible location types (e.g., universal resource identifiers, COAS interoperable object references, paper-based, etc.); and
- to support the ability to be accessible from a plethora of devices e.g. PCs, mobile devices, and platforms, like the Hyper Text Transfer Protocol (HTTP) or CORBA.

This conceptual model of SRIS is depicted in Figure 8.

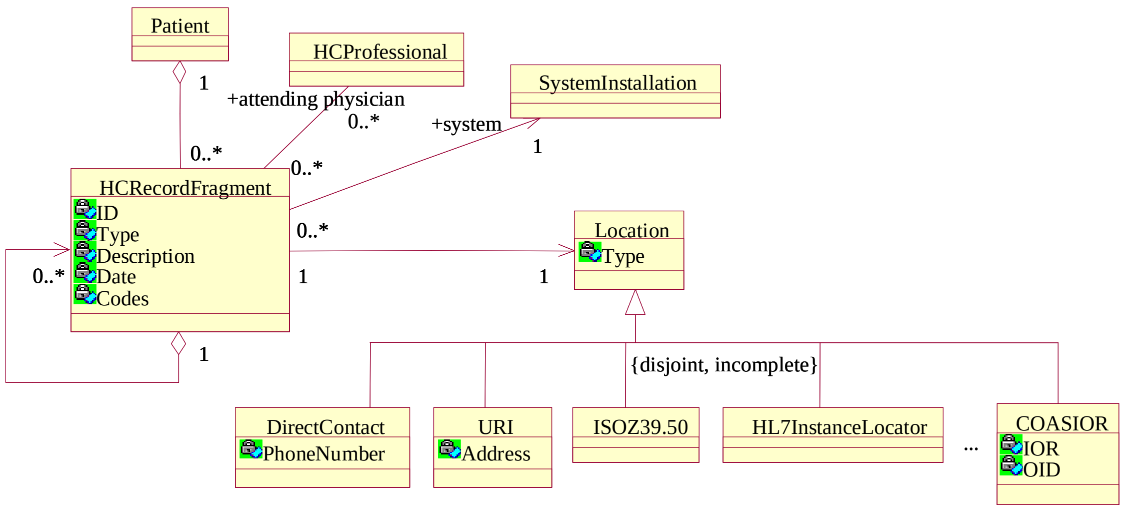


Figure 8 – The conceptual model of SRIS. A "Health Care Record Fragment" denotes the entity that contains clinical information, and the context in which this medical act took place. It contains only the indexing information and not the full information itself. "Location" represents a location where detailed information about a "Health Care Record Fragment" can be retrieved from.

Clinical Observations Access Service Component

COAS components are required for obtaining clinically significant information, captured at the point of care, directly from the corresponding clinical information systems. This service requires the implementation of standardized gateways for each clinical information system to enable for secure access to patient record data. PICNIC has adopted the OMG COAS specification [17], where "clinical observations" are defined to be "*any measurement, recording, or description of the anatomical, physiological, pathological, or psychological state or history of a human being or any sample from a human being, and any impressions, conclusions, or judgments made regarding that individual within the context of the current delivery of healthcare*."

All observations share a few common features:

- they are made on a specific subject of care (e.g., patient, organ, population);
- they represent a snap-shot of that subject in time, either at a particular time, or over some specified interval of time (time in this context includes the notion of both date and time);
- they are made, or recorded, by an instrument or a healthcare professional in some clinical context; and
- they are given (either by the patient, the healthcare institution, or society) some degree of confidentiality.

Observations can be quantitative, qualitative, and recordings. For example, vital signs and clinical laboratory results, trends in measured values, impressions from a clinical exam, correlation of several qualitative impressions, and images and manipulations of images such as digital subtraction angiography.

The adopted conceptual model of COAS is depicted in Figure 9.

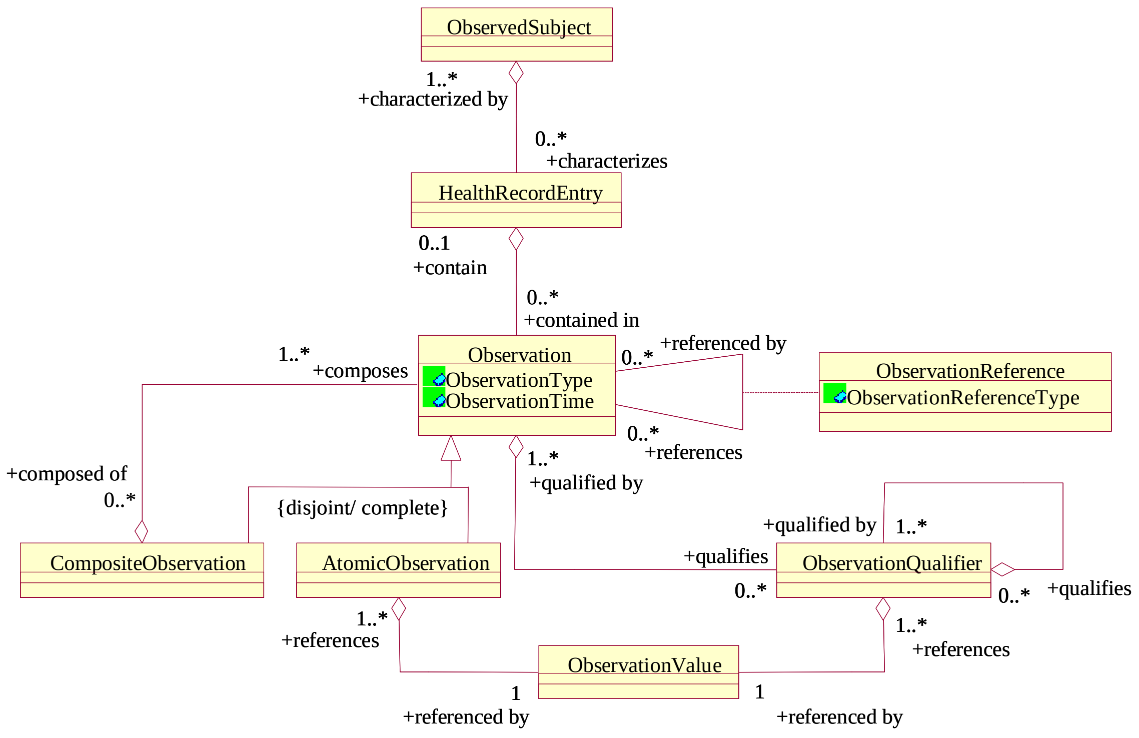


Figure 9 – The adopted OMG COAS conceptual model. An "Observation" may be Atomic or Composite. An "Atomic Observation" has an "Observation Value" associated with it, denoting the fact, note, or result of an observation. An "Observation Reference" defines an association between Observations.

Shared Records Update Broker Component

SRUB components are required for the provision of prompt and consistent propagation of indexing information to SRIS. In other words, the SRUB component is responsible for keeping SRIS up-to-date with new or updated information, and maintains a loose consistency between CISs and the SRIS. Each CIS is associated to a single SRUB that either periodically (on pre-determined time intervals), or on demand, submits properly formatted information to the IS.

Because COAS, in the context under consideration, is not designed for providing update/ notification information, it becomes necessary for the SRUB to maintain local indexing information for all the CISs of the federation that it is responsible for. This local indexing information, called *Update Cache*, is actually the partition of SRIS that corresponds to the specific CIS.

The conceptual model of SRUB is depicted in Figure 10.

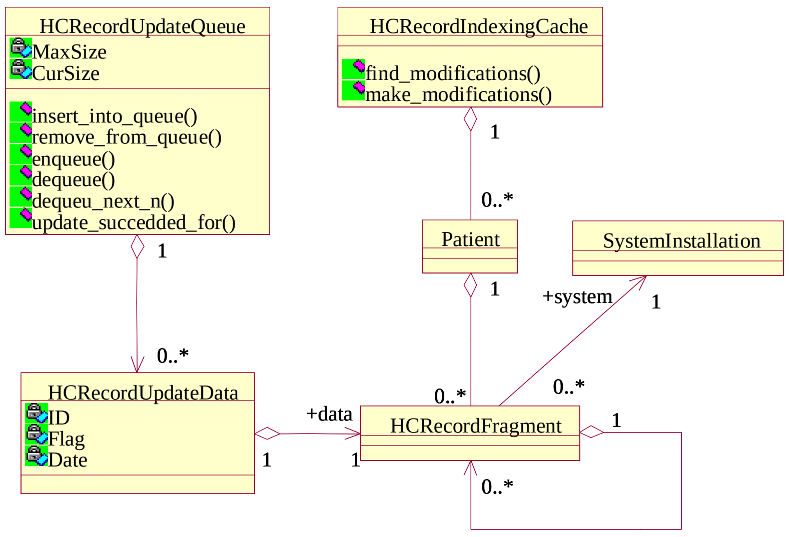


Figure 10 – The conceptual model of SRUB. Whenever a modification is submitted to SRIS and is committed, SRUB updates its own update cache in order to keep it synchronized. The "Health Care Record Update Queue" is a queue of the changes that need be submitted to SRIS. SRUB, after identifying the modifications required, it inserts them in the "Health Care Record Update Queue". When it is decided that SRIS must be updated, information is removed from the "Health Care Record Update Queue" and is transmitted to SRIS.

This design allows for the decoupling of SRIS update from the update of SRUB itself, making it possible to impose different policies in each of them. Two alternative policies that can be employed make use of the *pull model* and the *push model*. In the pull model the SRUB actively asks the CIS using the COAS as a gateway for collecting all available information. In the push model the SRUB is passively kept up-to-date, which means that the CIS sends the update information to the SRUB on demand. In this model an implemented event/ notification functionality of the COAS could be used to support these asynchronous updates of the SRUB.

Both of these models are useful in practice. The pull model is more general in that the SRUB can extract all the modifications performed inside the CIS. In the downside, the whole data kept in a CIS is a huge volume of data that must be transmitted to the SRUB where the differences must be determined based on the current and the previous datasets.

In the push model, only the modifications are transmitted to the SRUB, so the whole process is more efficient. The disadvantage though is that there is no general way to get modifications that have to do with "deletions" i.e. with information that does not exist any more. Following the event/ notification way of communication in COAS, a client can register its interest in receiving specific kind of information, e.g. for a specific patient or a specific type of clinical information, or information that satisfies some criteria. In all cases the notifications delivered to the client deal with new or updated information and not with information that is obsolete.

Since both of these models are needed, a combination of the two is a rational decision: the push model can be used for frequent updates and less frequently the pull model as a general check and repair method for maintaining consistency.

3.3 Collaboration

Telemedicine is an IT service used for the actual care of patients, making it possible to provide expert supervision to other professionals or directly to the patient. During the past ten years a large number of telemedicine projects have developed solutions for healthcare professionals at remote sites, to enable them to have access to the expert opinion of medical specialists via telemedicine in a variety of settings.

Today we know that telemedicine is not only for remote sites, where access to the healthcare system is difficult. More and more telemedicine services and applications have been implemented and are used by the healthcare professionals to optimize the treatment of patients. It is foreseen that during the next ten years, telemedicine services will be as important as the stethoscope, to practice medicine.

Some of the most important benefits consist in a rise in the quality of treatment and better utilisation of resources. At the same time the same technology helps remove the barriers that geography can put in the way of patient treatment. The healthcare experts are no longer any further away than the nearest Personal Computer (PC) with a network connection. The necessary specialist knowledge is available where it is needed – without the patient having to be moved to receive the best treatment.

The objective of PICNIC was to develop a framework where all kind of telemedicine application can be interconnected to perform collaboration between healthcare professionals. The aim was to develop a set of tools and guidelines, which can be used by all companies to connect their telemedicine applications to the RHCN.

The conceptual model for the Collaboration IT service is shown in Figure 11.

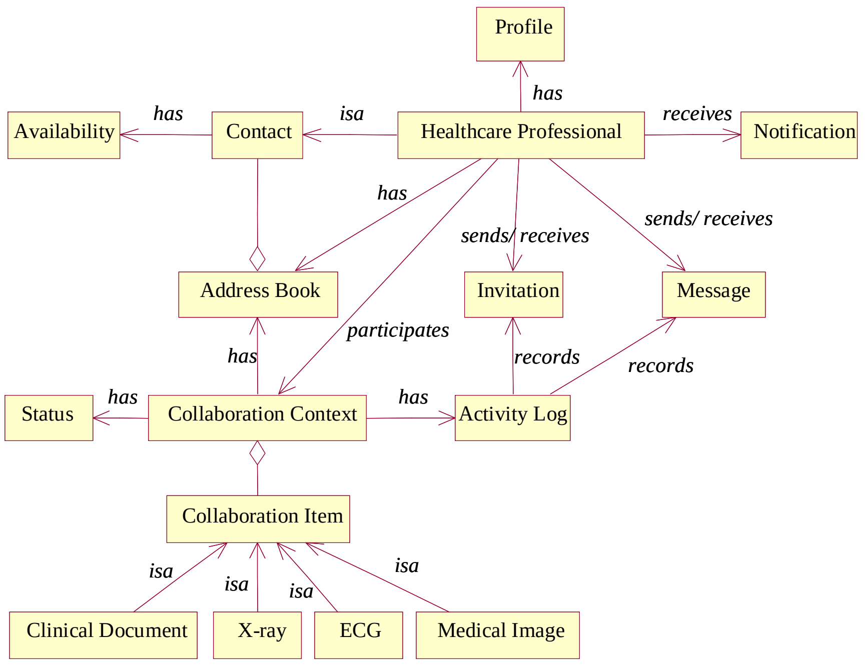


Figure 11 – The conceptual model of the Collaboration IT service. “Collaboration Context” maintains all information related to a collaborative activity. All participants are stored in “Address Book”. “Collaboration Items” represent all clinical data that are shared in the context of the collaboration. Those may be “Clinical Documents”, “X-rays”, “ECGs”, “Medical Images”, etc. The “Activity Log” keeps a detailed record of all activity that involves the collaboration context, including any posted “Message” and/ or “Invitation” to participate in the collaborative activity. The “Collaboration Context” may be linked to a workflow, in which case the state of the workflow is stored in “Status”. A “Healthcare Professional” is the typical user of the Collaboration IT service and all personal preferences are stored in a “Profile”.

A healthcare professional may be interested in particular events concerning a collaboration context and may wish to be notified in a special way and/ or on a specific device. A healthcare professional using the service may play the role of the consultant, participant, or referrer. In the role of the participant, a healthcare professional may participate in a chat conference posting messages, or inviting more specialized healthcare professionals. In the role of the referrer, the healthcare professional uses a CIS that stores patient data and provides access to the Collaboration IT service that facilitates access to clinical information and automatic creation of clinical documents (compliant to the HL7/ CDA R1 architecture), based on templates customized for different clinical problems. These documents could be linked to the problem at hand and the EHR.

The main actors of the Collaboration IT service have been identified in PICNIC to be:

- the *Healthcare Professional* who is a generalization of different healthcare individuals (*medical expert*, *medical doctor*, *nurse*, *general practitioner*, and *social worker*) that belongs to a health organization and uses the Collaboration IT service in the treatment of a patient. The Collaboration IT-service can be used to establish collaboration with other *Healthcare Professionals* and share data on-line as well as off-line.
- The *Instant Messaging System* is a technology platform that provides presence (on-line and off-line) for the *Healthcare Professional*, who participates in the Collaboration IT service. The *Instant Messaging System* can be considered as part of the collaboration Context Manager, where the Collaboration Context Manager is the front-end (client) and the Instant Messaging System is the back-end (server).
- The *Notification Agent* actor has the responsibility to send notifications to *Healthcare Professionals* involved in a specific collaboration. The *Healthcare Professionals* may in advance request and specify what event and conditions he should be notified about.
- The *Clinical Information System* actor is any end user application used by the *Healthcare Professionals*. The *Clinical Information System* has implemented the needed interface to the use of the Collaboration IT-service.
- The *Collaboration Context Manager* actor takes care of storage and retrieval of all clinical data or pointers to clinical data for a collaboration session. The *Collaboration Context Manager* provides also the capability to *Healthcare Professionals* to share the clinical data via a CIS. The *Collaboration Context Manager* communicates with the Instant Messaging system.
- The *Patient* actor is an individual that is admitted to the hospital or consults another Healthcare facility for treatment of a specific health problem.
- The *Resource Directory System* is a directory service, which gives access to information about healthcare professionals, healthcare organizations and collaboration contexts.

The main use cases for the Collaboration IT Service are shown in Figure 12.

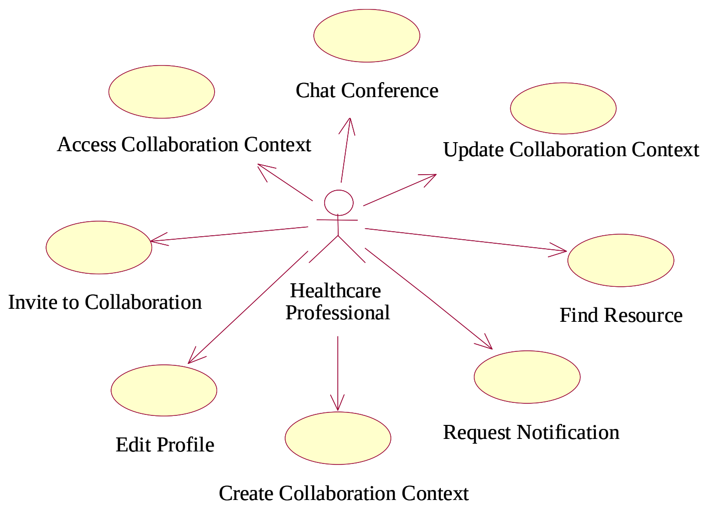


Figure 12 – Main use case diagram for the Collaboration IT service.

According to the description provided throughout the services description and the subsequent harmonised functional specifications, the Collaboration IT service features can be translated into the use cases listed on Table 4:

Table 4 – Collaboration IT service use cases.

Use Case	Short description
Create Collaboration Context	When the *Healthcare Professional* encounters a new episode, a new *Collaboration Context* can be defined, in which all information about the patient can be shared with other *Healthcare Professionals*. Via the *Collaboration Context* other *Healthcare Professional* can be invited to take part in the treatment of the patient.
Access Collaboration Context	Collaboration between *Healthcare Professionals* for a specific patient, will always take place via a *Collaboration Context*, where all kind of clinical data are shared. The *Collaboration Context* includes functions for inviting new *Healthcare Professionals*, updating the clinical data, which are shared and engaging in a chat conference.
Find Resource	Provides information about all kind of resources available. The resources can be healthcare organizations, healthcare professionals involved in a *Collaboration Context.* A resource can also be a *Collaboration Context* itself.
Request Notification	The *Healthcare Professionals* may request to be notified about specific events relevant to a *Collaboration Context* and receive notifications as requested.
Invite to Collaboration	The *Healthcare Professional* may invite other *Healthcare Professionals* to participate in a *Collaboration Context*.
Chat Conference	One of the *Healthcare Professionals* can launch a conference in order to discuss with other *Healthcare Professionals*.
Update Collaboration Context	The *Healthcare Professionals*, which have been invited to participate in a specific *Collaboration Context* and put documents into the context. The documents in the Context are shared information.
Edit Profile	Each *Healthcare Professional* can edit her/ his personal data and notification preferences that are stored in a *Profile*, in order to make it possible to fulfil specialised needs.

The main software entities that can be represented as collaboration components are the following:

- Clinical Observation Access Component;
- Collaboration Service Component;
- Patient Identification Service Component;

- Shared Records Indexing Service Component;
- Auditing Service Component;
- Authentication Service Component;
- Encryption Service Component;
- Resource Service Component;
- Terminology Service Component; and
- User Profile Service Component.

Their synergy is depicted in Figure 13, a brief description of components developed in PICNIC follows.

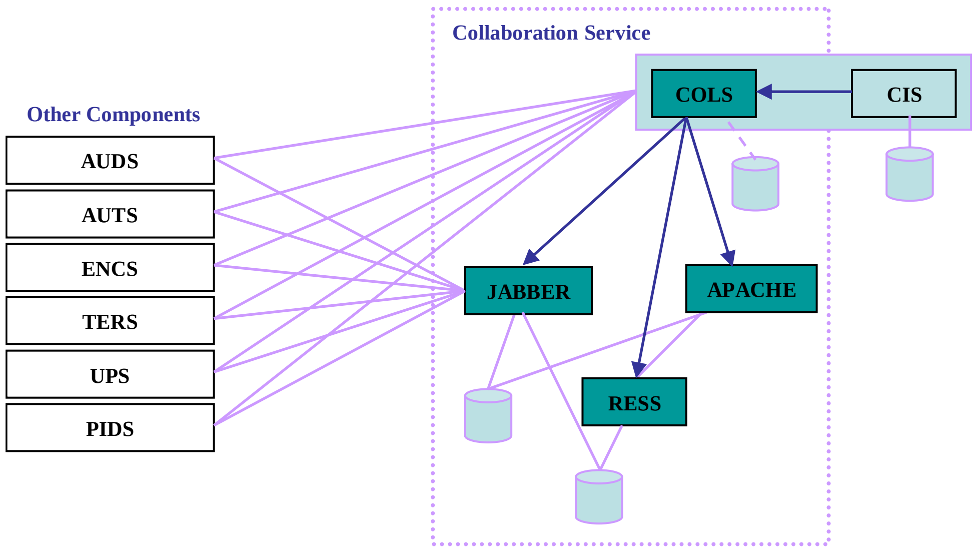


Figure 13 – The collaboration architecture. The collaboration service comprises of COLS and RESS, while interaction with other components is indirect through the CIS. The Jabber Instant Messaging System [18] can be interfaced in order to improve accountability and keep a detailed record of all the activity in a collaboration context. The Apache HTTP server [19] is used for the exchange of data between the participants.

Collaboration Service Component

COLS components are required to establish a collaboration context that enables not only the active sharing of clinically significant information, but also receiving feedback, feed through and awareness information from all participating actors.

COLS is designed:

- to support multiple, extensible resource types (e.g. health care professionals, organizations, collaboration contexts, services, devices, pricelists, etc.) with respect to their type and way of expression of internal information;
- to support the ability to locate resource information in a distributed environment;
- to provide for filtering, such as by resource type and attribute, type, time frame, etc.;
- to support for site-specific and implementation-specific extensions and customization of profile elements;
- to support the ability to be accessible from any devices e.g. PCs, mobile devices, and platforms, like HTTP or SOAP;
- to support on-line presence of connected users;

- to support exchange of various type of information free text, structured information by use of standards (eq. EDIFACT, HL7);
- to support conference facilities (chat, video and sound; and
- to support subscription and notification of events in a collaboration.

The computational boundary classes of COLS are depicted in Figure 14.

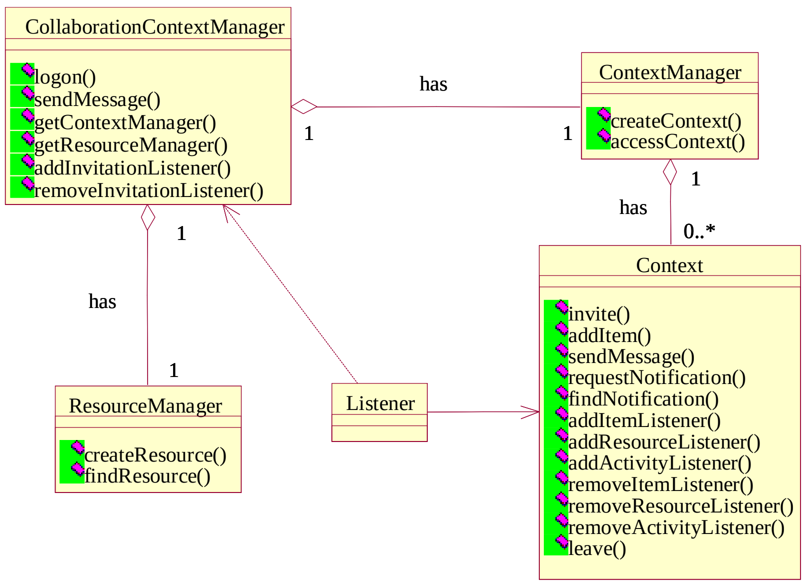


Figure 14 - COLS Class diagram.

Resource Service Component (RESS)

The RESS component constitutes an integral part of the PICNIC architecture to enable for the discovery of existing components and services of the healthcare infrastructure. It is intended to maintain and provide information about all related healthcare agents. According to CEN-TC251 these include any healthcare person, healthcare organization, healthcare device or healthcare software component that performs a role in a healthcare activity [20] within the health system under consideration.

This component ought to be the central repository of location information and the place where potential clients (users, components, applications) are directed in order to find and identify the offered components/ services in the distributed environment of a RHCN.

RESS components are useful for identifying available resources and the means for accessing them. Examples of resources include pharmacies on-duty, hospitals and clinics, clinical information systems available at a regional level, methods and technologies available for accessing primary information, and protocols for exchanging information with them. A resource service component may also provide information about equipment/ devices available at certain facilities, as well as used by certain categories of people, together with the corresponding equipment/ device characteristics.

In this context, it is apparent that any definition of resource ought to include, without being limited to, the following:

- software entities, e.g. CORBA servers;
- healthcare facilities, e.g. hospitals, clinics;
- equipment/ physical devices, e.g. microscope, or computed tomography;
- healthcare professionals/ hospital personnel; and
- users.

It is difficult to anticipate every potential kind of resource that exist or could exist in the future and would be of interest, so the implementation of the RESS should be as much generic as possible, without imposing any constraints that could limit its use when future demands for new resources appear. Additionally, the information managed by the RESS for each resource should be expressed in a generic manner. An important consideration has to do with the fact that the attributes for each type of resource varies significantly. For example, for a web server it is important to know the host name, address and port number, while for a physician the name, address and specialty of the subject is important.

The resource service will keep information about:

- type of health care specialist available;
- pricelist for different type of resources;
- profile for organization and individual health care specialists;
- type of information, which can be exchanged (images, video, sound, structured text, unstructured text, booking of resources, type a standard used); and
- reimbursement.

RESS is designed:

- to support multiple, extensible resource types (e.g. people, services, devices, etc.) with respect to their type, relation and way of expression of internal information;
- to support the ability to locate resource information in a distributed environment;
- to provide for filtering, such as by resource type and attribute, information type, time frame, etc.;
- to support for site-specific and implementation-specific extensions and customization of profile elements;
- to compliance with the overall PICNIC architecture; and
- to support the ability to be accessible from a plethora of devices e.g. personal computers, mobile devices, and platforms, like HTTP or CORBA.

The conceptual model of RESS is depicted in Figure 15.

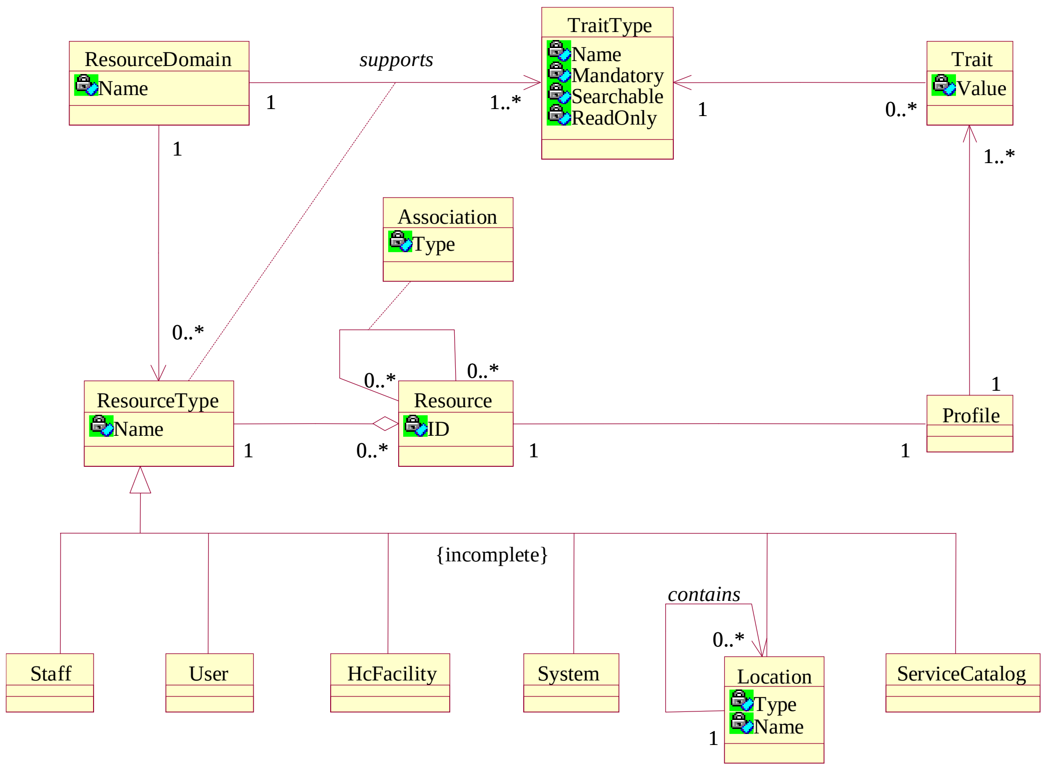


Figure 15 – The conceptual model of RESS. Each "Resource" is uniquely identifiable within the context of a "Resource Domain". Each "Resource Domain" supports several "Resource Types". Two or more Resources can be interrelated through an "Association". In a given "Resource Domain", a number of "Trait Types" are supported for a given "Resource Type". "Trait" is an instance of a "Trait Type" and consists of its type and value. Finally each "Resource" has a "Profile", which bundles its entire set of "Traits".

4 Software Components

A formal way to identify the exact nature and role of each of the components required by any health information network was used throughout PICNIC. The unified software development process [21] is the adopted methodology, describing how to elicit, organise, and document required functionality and constraints; track and document tradeoffs and decisions; and easily capture and communicate the healthcare domain requirements. The notions of use cases and scenarios prescribed by the RUP have proven to be an excellent way to capture functional requirements and to ensure that these drive the design, implementation and testing of software, making it more likely that the final system fulfils the end user needs [22].

UML was used in order to specify, visualise, and document models of software systems, including their structure and design, in a way that meets all available requirements. UML is the tool used to transform system capabilities into requirements, and use cases including both basic and alternative flows of events in the form of interaction diagrams. In specifying components, adequate care was taken so that their internal semantics are neutral, in the sense that each component ought to be able to be configured differently under the different execution environments it is expected to operate in.

Work conducted in PICNIC identified the health services [23] within a RHCN and an IT response [24]. The analysis of these identified commonalties in models and tools and established a common base for the development of generic tools and common software components [25]. It also led to a description of components identified and an initial high level view of the each IT service's functionalities, involved actors, as well as the components, forming subsystems of each IT service. These were taken through different steps starting with the initial decisions on which software components to include and how best to develop them were documented, ending with the production of the actual specifications for those components selected to be deployed in more than one region and/ or application domains [26].

Apart from the middle layer components described above, a number of platform services that provide basic functionality are in all technology platforms (e.g. Microsoft, Java, CORBA, World Wide Web Consortium – W3C etc.) are needed so that the 'global system' exhibits the required behaviour in terms of security (authentication, encryption, auditing, etc.), concurrency control, event handling, transaction management, workload balancing, etc.

4.1 Performance Issues

Any middleware platform should not sacrifice efficiency in favour of usability, flexibility, and feature richness. The performance factor is critical especially in domains such as the healthcare where, in some cases, "*the right answer delivered too late becomes the wrong answer*". In such application domains, middleware components' developers should pay close attention to performance dimensions such as the following:

- *Throughput:* The component should be able to handle a large number of requests per unit time, e.g. per second or per "busy hour";
- *Latency:* The component should minimise the request/ response processing delay when a client calls an operation;
- *Jitter:* The component should minimise the standard deviation of the latency in order to increase predictability and determinism;
- *Scalability:* The component should be able to sustain its performance when some external condition changes, such as when the load increases (load scalability) or when the number of hardware units, such as central processing units, increases (system scalability).

Various techniques can be deployed to achieve higher efficiency:

- *Concurrency:* multithreading or multiprocessing [27] can increase the concurrency of the system, i.e. its inherent parallelism. Study of the more appropriate collaboration patterns between component's modules should be done in order to minimise the synchronisation overhead and the threads' contention for shared resources. Another way to increase concurrency is to utilise asynchronous mechanisms like asynchronous input/ output [28] or asynchronous method calls [29].
- *Caching:* Complex and/ or expensive calculations, or tasks in general, can be minimised by keeping a cache of recent results [30]. Much attention ought to be given to the analysis of what are the system's resources that should be kept in a cache, in addition to the memory considerations and the cache replacement algorithms.
- *Load Balancing:* Distributing workload in a cluster of computer hosts or some other computation nodes such as threads or process usually increase the real parallelism of the system [31]. Additionally, it minimises the danger of a single-point failure and increases the availability of the system.

Achieving these goals is a difficult task most of the times because it is a common phenomenon that these different mechanisms do not collaborate in a synergistic manner. A lot of experience and study is usually needed in order to overcome the inherent complexities in developing and maintaining software components that offer the advantages described above. The use of design and architectural patterns leverages the development of robust, efficient, and reusable middleware components [32].

4.2 Conformance Levels

Four levels of conformance can be proposed for further consideration. Each one of them is progressive, in that it builds upon prior levels and is necessary for subsequent levels:

- Architecture conformance
- Interface conformance
- Semantic conformance
- Qualified code conformance

Architecture conformance refers to the required interoperability technology. Examples include Microsoft’s Component Object Model/ Distributed Internet Architecture [33][34], Sun Microsystems’s Enterprise JavaBeans/ Java 2 Enterprise Edition [35][36], as well as the Object Management Group’s Component Object Request Broker Architecture [37]. Since 2001, interest has been growing in using Web services, a set of technologies based on XML, as a component model for Internet communications [38].

Interface conformance refers to conformance to one or more interfaces described at the specification part of the component definition. Examples include the classes defined for PIDS, COAS, SRIS, and SRUB. An implementation claiming conformance to any of these classes must conform to all of the interfaces specified for that class. An implementation may claim conformance to multiple conformance classes as long as it is conformant to each one it claims.

Semantic conformance refers to conformance related to the semantics used for communicating structures containing values of data. Examples include the various domain models that can be used with PIDS, or for accessing COAS.

Qualified code conformance refers to conformance to a naming convention for the use of terms from other standards. Qualified codes are globally unique concept codes formed by combining the coding scheme id and the local concept code within that coding scheme.

5 Security and Privacy

Security refers to two main issues:

- security (resilience) of the hardware, operating system and the application software; and
- control of the use of information.

The main focus in PICNIC was the latter especially as it dealt with private and confidential patient information. The former, while being equally important, is an issue that must be solved in the selection of the software platform and the tools used to create the execution environment and the applications.

Security management is a particular instance of the general information system management functions. Information system security management services are concerned with the installation, maintenance, and enforcement of information domain and information system security policy rules in the information system intended to provide these security services. In addition to these core services, security management requires event handling, auditing, and recovery. Standardisation of security management functions, data structures, and protocols will enable interoperation of security management application processes across many platforms in support of distributed security management.

The concept of an *information domain* provides the basis for *security* protection. An information domain is defined as a set of users, their information objects, and a security policy. An information domain *security policy* is the statement of the criteria for membership in the information domain and the required protection of the information objects. Security policy implements regional and/ or local requirements and needs. It must be based on national legislation on the protection of personal data when collected, processed and/or transferred and on the laws regulating the practice of medicine and delivery of healthcare services by healthcare professionals. Security within each information domain must be established in accordance with the respective security policy. For communication between the information domains, a trusted end-to-end communication policy must be established (see Figure 16).

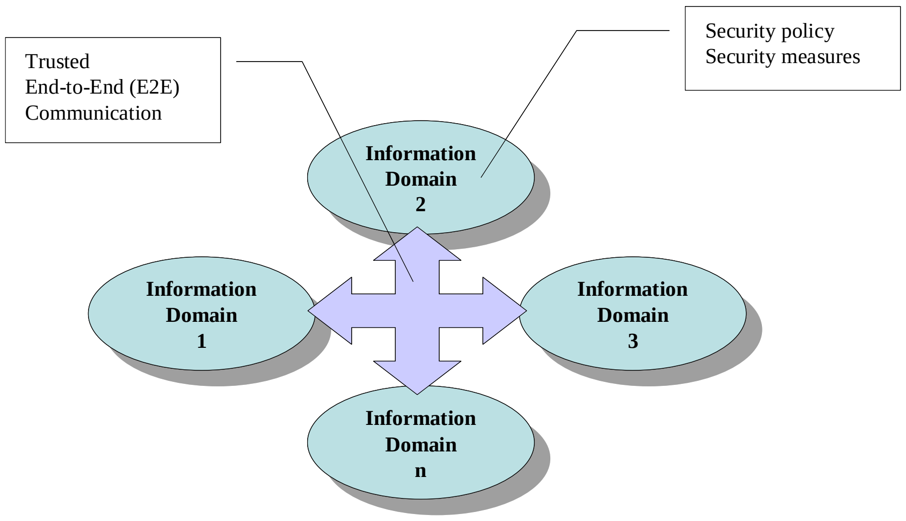


Figure 16 – Bridging information domains securely.

The European Union (EU) data protection directive is currently being implemented into the national legislation of the EU Member States. This means that every EU Member State must have the same level of legal protection of personal data. The same does not apply to healthcare and medicine. There each country still has its own mandate. *Consequently, security policies and their implementations in the RHCN's will be different.* However, although they are different, there are several commonalties between the regions and in the technologies deployed. Each will have to address the following issues:

- Data protection (including audit trails)
- Data confidentiality (including encryption)
- Authentication of users
- Authorisation (including assigned roles regulating access rights to e.g. patient data)

Additionally, security policies must deal with the *informed consent of patients* (customers), which is required for legal access to patient data. Consent may also have qualifiers e.g. restricting access to only part of the patient data or restricting the period of time that the consent is given. Finally, the choice of what security features to implement must be based on *risk assessment* in the context of the intended service.

For these reasons the approach adopted in PICNIC for security was that each pilot region will have to implement it in accordance with their national legislation and guidelines and based on their software platform security functions. Due to the particular nature of healthcare and the private and confidential information that is produces, stores and processes a number of projects have been carried out over the years on these issues. These include EU-funded projects such as CHARM (Comprehensive Health Assistance and Resource Management) [39], HARP (Harmonization for the Security of the web technologies and Applications) [40], and RESHEN (Regional Secure Healthcare Networks) [41].

5.1 Security within Information Domains

The general stages for implementing security are shown in Figure 17. In the following the main elements of authentication and authorisation will be briefly described.

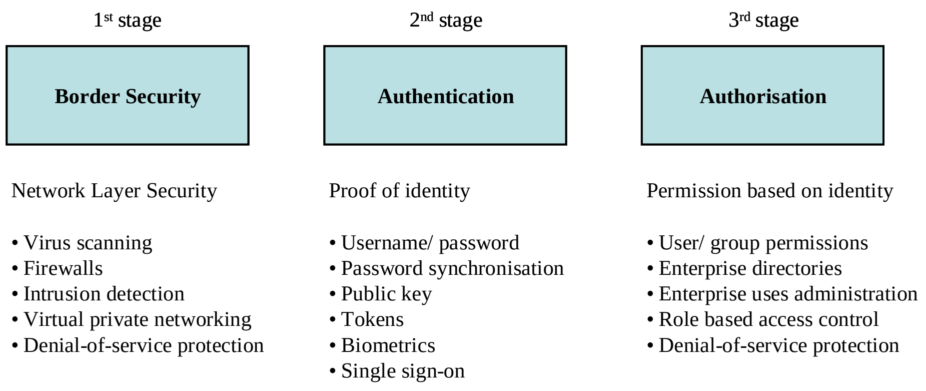


Figure 17 – A Layered Defence (from [42]).

Access & User Identification

In order to access system resources and patient data users must be identified. I.e. access to resources and data must be controlled so that unauthorised access is prevented. Access rights can be managed on two levels:

- authentication (the person is who (s)he claims to be); and
- authorisation (permitting access to resources and data based on a qualified role, role-based access).

A concept often associated with access is *Single-Sign-On* (SSO). This means that when a person logs into a system, is authenticated and authorised in a role (s)he gets access to all resources and data that that role entitles. In a healthcare context users often have to use several information systems. In such cases SSO is useful as it allows access to all information systems that the user is authorised to use in that role with only one log-in. SSO can be implemented e.g. using the Clinical Context Object Workgroup (CCOW) standards produced by HL7.

Consent (by patient)

Information may not be made available or disclosed to unauthorised individuals, entities or processes without the consent of the patient. This is a fundamental right of individuals that they shall have the power to keep information about themselves from being disclosed to anyone [43]. Therefore, the individual (patient) must agree/ consent to disclosing her/ his private information. In healthcare, consent refers to a communication process between the caregiver and the patient, and may refer to consent for treatment, special procedures, release of information, and advanced directives. There a several kinds of consent:

- *Expressed consent:* Oral or written agreement. Because it is difficult to prove that oral consent was given, most expressed consent is expected to be recorded with a signature.
- *Implied consent:* An action other than an expressed consent on the part of a patient that demonstrates consent. For example, the presentation of a person to a caregiver implies, to a certain extent, consent to at least basic consent.
- *Informed consent:* In some countries/ healthcare organisations, for a consent to be valid, the patient must be informed. The patient should be given the opportunity to ask questions, to indicate comprehension of the information provided, and to grant permission freely and without any coercion for performance of a procedure or course of treatment, as well as the opportunity to withhold or revoke such permission at any time without prejudice.

Audit trail

Audit trails refer to data collected and potentially used to facilitate a security audit. It comprises of a chronological set of records that provides evidence of system activity. These records can be used to reconstruct, review, and examine transactions from inception to output of final results. The records can also be used to track system usage and detect and identify intruders.

Audit trails are needed to ensure a*ccountability* of actions of individual persons or entities, such as obtaining informed consent or breaching confidentiality.

5.2 Trusted E2E Communication

The International Organization for Standardization (ISO) Technical Committee 215 has been working on a Technical Report on “trusted end-to-end information flows” [44]. Figure 18 illustrates the principle. The idea in this approach is that it is the responsibility of the information domains to negotiate under what terms they are able and willing to exchange information.

Each information domain has its own security policy stating e.g. what conditions must be met to access or store/ modify patient data. The requesting information domain has to provide information about its security measures to the replying information domain. This may include information about how the user that has originated the request has been authenticated and authorised and that an explicit consent has been given by the patient for this request to access her/ his data. It is the responsibility of the replying side to examine these certificates and ascertain their authenticity and based on these decide whether to provide the requested service or deny it. Both sides will additionally have to create audit trails of the transaction independent of whether the service was delivered or not.

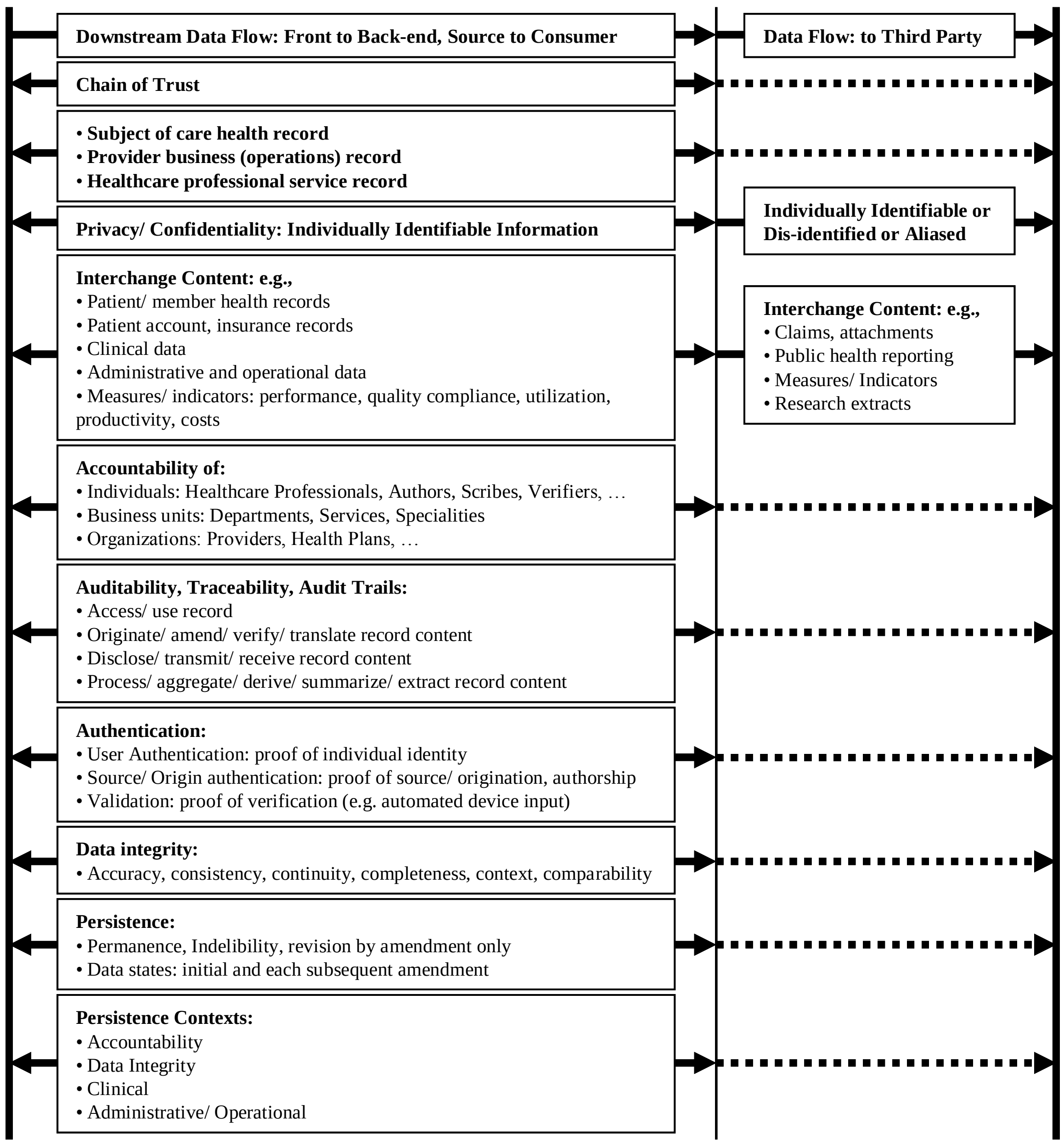

Downstream Data Flow: Front to Back-end, Source to Consumer
Data Flow: to Third Party
Chain of Trust
• Subject of care health record
• Provider business (operations) record
• Healthcare professional service record
Privacy/ Confidentiality: Individually Identifiable Information
Individually Identifiable or Dis-identified or Aliased
Interchange Content: e.g.,
• Patient/ member health records
• Patient account, insurance records
• Clinical data
• Administrative and operational data
• Measures/ indicators: performance, quality compliance, utilization, productivity, costs
Interchange Content: e.g.,
• Claims, attachments
• Public health reporting
• Measures/ Indicators
• Research extracts
Accountability of:
• Individuals: Healthcare Professionals, Authors, Scribes, Verifiers, …
• Business units: Departments, Services, Specialities
• Organizations: Providers, Health Plans, …
Auditability, Traceability, Audit Trails:
• Access/ use record
• Originate/ amend/ verify/ translate record content
• Disclose/ transmit/ receive record content
• Process/ aggregate/ derive/ summarize/ extract record content
Authentication:
• User Authentication: proof of individual identity
• Source/ Origin authentication: proof of source/ origination, authorship
• Validation: proof of verification (e.g. automated device input)
Data integrity:
• Accuracy, consistency, continuity, completeness, context, comparability
Persistence:
• Permanence, Indelibility, revision by amendment only
• Data states: initial and each subsequent amendment
Persistence Contexts:
• Accountability
• Data Integrity
• Clinical
• Administrative/ Operational

Figure 18 – Trusted end-to-end information flows (from [44]).

5.3 Security Technologies

Public Key Infrastructure and Certification Authorities

Public Key Infrastructure (PKI) is used to describe the processes, policies, and standards that govern the issuance, maintenance, and revocation of the certificates, public, and private keys that the encryption and signing operations require. PKI is used in order to enable two entities that do not know each other to exchange information using an insecure network such as the Internet. The infrastructure is based upon asymmetric cryptography and each entity (user, information system, etc.) is provided with a pair of keys (a private and public key).

Certification Authority (CA) is a trusted party that can vouch for the binding between names or identities and public keys. In some systems, certification authorities generate public keys. The public key certificate binds a user's name to a public key signed by a trusted issuer.

Smart cards are mostly associated with PKI and CAs as an access card containing encoded information and sometimes a microprocessor and a user interface. The information on the code, or the information generated by the processor, is used to gain access to a facility or a computer system.

The security infrastructure comprises the following services:

- CAs that control and manage the PKI, publish public key certificates, and impose policies in their domain of authority.
- Registration Authorities that collect the required eligibility certificates from interested parties, verify their authenticity, and subsequently inform CAs.
- Certificates Management Systems (CMS) for management of certificates during their entire duration of validity. The CA controls and uses the CMS.
- X.500 directories that store public key certificates as well as public information for the holders of certificates and are used for the verification of digital certificates.
- The certificate of the users, which is published by the CA, is stored together with the user's private key, in a microprocessor card.

Digital Signature

Digital signature is a means to guarantee the authenticity of a set of input data the same way a written signature verifies the authenticity of a paper document. Digital signatures are required in many cases during the provision health services to a citizen. It comprises a cryptographic transformation of data that allows a recipient of the data to prove the source and integrity of the data and protect against forgery. To sign a document, the document and private key are input to a cryptographic process, which outputs a bit string (the signature). To verify a signature, the signature, document, and user's public key are input to a cryptographic process, which returns an indication of success for failure. Any modification to the document after it is signed will cause the signature verification to fail (integrity). If the signature was computed using a private key other than the one corresponding to the public key used for verification, the verification will fail (authentication).

The basic process for the exchange of digitally signed messages is outlined below:

- The sender of the message produces the digest of the message (i.e. the representation of the message in the form of a single string of digits using a one-way hash function) which it/ she/ he subsequently encrypts with its/ her/ his private key. This way the digital signature is formed.
- The digital signature and the sender's certificate are dispatched with the message. The certificate contains the identity of sender and its public key.
- The recipient confirms the sender's certificate, and then using the public key of the sender calculates the initial digest of the message.
- The recipient also produces the digest of message directly from the received message.
- If the two digests are same then the signature is authentic.

Digital signatures can be attached to any electronically transmitted message, including ones transferring EHR data. The digital signing of XML based clinical documents is a special instance where the nature of the clinical workflow may require that each participant only signs that portion of the document for which they are responsible. Older standards for digital signatures do not provide the syntax for capturing this sort of high-granularity signature or mechanisms for expressing which portion a party wishes to sign.

The joint standards effort between W3C and the Internet Engineering Task Force (IETF) includes a standard for XML Signature [45] [46]. It is being created with the ability to sign only specific portions of the XML tree rather than the complete document. This will be relevant when a single XML document may have a long history in which the different components are authored at different times by different parties, each signing only those elements relevant to them.

6 Conclusions

PICNIC by following a top down approach has identified the need for certain key components. These components, when implemented within the regional pilots, are expected to realize their potential as more of them become available. This is because component-based software development shifts the focus from new software development to the integration of existing components to perform new tasks; while at the same time it addresses the issues of large-scale system development in the areas of coupling, distribution, and multiple platforms.

The Internet is imposing new integration challenges as it extends into every corner of every organisation. Today, the issue of dealing with new platforms and applications that must interoperate with legacy systems becomes more important. As computers and networks become faster and cheaper and interconnection standards evolve new technologies constantly appear for new application niches. In developing the next generation RHCN, an architecture is required, being a strategic resource with the potential of enabling and supporting the RHCN in meeting its goals (see also the Architecture chapter). The software engineering view of this architecture acts as the unifying framework that provides structure and semantics and realises the components potential as more of them become available.

Eventually it is expected that the creation of application models that will be in a position to be translated automatically into software (capable of running in the target deployment environment) will eliminate much of the effort currently required for software development. This approach will offer an architecture always ready to deal with yesterday's, today's and tomorrow's execution environment. It will also make application and IT services integration across middleware boundaries easier; while at the same time it is expected to provide much wider cross-platform interoperability among different platform services. Today the new Model Driven Architecture (MDA) that OMG adopted in 2001 [47] focuses exactly on this.

With component-based software, the significance of deciding whether it is better to buy an integrated suite from a single vendor or choose individual best-of-breed applications from several vendors is reduced because standardised component interfaces facilitate the combinations of components from different vendors. The web services architecture, which today is still being developed, extends component-based software by allowing some of the components to be operated as IT services that can be discovered through publicly available directories, and provides the means for accessing components spanning the entire Internet.

Acknowledgements

The authors would like to thank everyone involved in the implementation of the “IT Response” and “Common Components Development” work packages of the PICNIC project; namely: Paddy BURKE (Ireland), Ita BROWN (United Kingdom), Richard CAMPIN (United Kingdom), Ana CARRIAZO Perez de Guzman (Spain), Jaime Nieto CERVERA (Spain), Catherine CHRONAKI (Greece), Jose Antonio COBENA (Spain), Paraic COLREAVY (Ireland), Mary CORRIGAN (Ireland), Declan McCORRY (United Kingdom), Baldur JOHNSON (Iceland), Leslie FINLAY (United Kingdom), Caroline GALLAGHER (Ireland), Hulda GUDMUNDSDOTTIR (Iceland), Anna HAFBERG

(Iceland), Jose Maria de la HIGUERA (Spain), Stefan IGELMANN (Germany), Timo ITALA (Finland), Tove KAAE (Denmark), George KAVLENTAKIS (Greece), Ole LYKKE (Denmark), Alpo KOMMINAHO (Finland), Stavros KOSTOMANOLAKIS (Greece), Evi MAVROGIANNAKI (Greece), Tuire MIKOLA (Finland), Marino G. NJALSSON (Iceland), Pentti PALOMAKI (Finland), Thorgeir PALSSON (Iceland), Susann Duedal PEDERSEN (Denmark), Alkis POULIS (Greece), Jakob POULSEN (Denmark), Michael PSARROS (Greece), Manuel Lopez SERRATO (Spain), Stelios SFAKIANAKIS (Greece), Theo STIDSEN (Denmark), Manolis TSIKNAKIS (Greece), Jari VIITANEN (Finland), and Manuel VILLACORTA (Spain).

References

[1] Building Regional Healthcare Networks in Europe, J. Oates and H. Bjerregaard Jensen (editors), IOS Press, 2000.
[2] UML Home Page http://www.uml.org
[3] Grady Booch, Object Solutions, Addison-Wesley, 1995.
[4] Ivar Jacobson, M. Griss, and P. Jonsson, Software Reuse—Architecture, Process and Organization for Business Success, Harlow, England, AWL, 1997.
[5] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard, Object-Oriented Software Engineering—A Use Case Driven Approach, Wokingham, England, Addison-Wesley, 1992, 582p.
[6] Alan W. Brown (ed.), Component-Based Software Engineering, IEEE Computer Society, Los Alamitos, CA, 1996, pp.140.
[7] Maria Ericsson, Developing Large-scale Systems with the Rational Unified Process, Rational Software White Paper, 2000 http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/sis.pdf
[8] PICNIC Deliverable 4.2, Functional Specification and Minimum Data Set for New Services, March 14, 2001.
[9] PICNIC Deliverable 6.2, Messaging Components Harmonized Specification, May 10, 2002.
[10] PICNIC Deliverable 6.3, Access to Patient Data Component Specifications, March 11, 2002.
[11] PICNIC Deliverable 6.4, Collaboration IT Service Component Specifications, June 4, 2002.
[12] Technical Committee 251 of the European Committee for Standardization http://www.centc251.org/
[13] R. Dolin et al. "An Update on HL7's XML-based Document Representation Standards." Proc. Of AMIA 2000.
[14] HL7, 'Version 3 Standard Clinical Document Architecture Framework Release 1.0', 2000.
[15] David Piggott, Catherine Chronaki, Morten Bruun-Rasmussen, Dimitrios G. Katehakis, Vesa Pakarinen, Niilo Saranummi, Jari Viitanen, Timo Itälä, Reporting Experiences from Using the HL7 Clinical Document Architecture in the PICNIC, HL7 International CDA Conference, Berlin, Germany, October 7-10, 2002.
[16] Object Management Group, Healthcare Domain Task Force, Person Identification Service Specification http://www.omg.org/technology/documents/formal/person_identification_service.htm
[17] Object Management Group, Healthcare Domain Task Force, Clinical Observation Access Service Specification http://www.omg.org/technology/documents/formal/clinical_observation_access_service.htm
[18] Jabber Software Foundation http://www.Jabber.org
[19] The Apache Software Foundation http://www.apache.org/
[20] European Committee for Standardization (CEN/TC 251), WG I, Information Models: "ENV 13606 - Electronic Healthcare Record Communication", 1999.
[21] Jacobson I, Booch G, Rumbaugh J, "The Unified Software Development Process", Addison-Wesley, 1999.
[22] Jacobson I, Christerson M, Jonsson P, Övergaard G: "Object-Oriented Software Engineering—A Use Case Driven Approach", Wokingham, England, Addison-Wesley, 1992, 582p.
[23] PICNIC Deliverable 3.2 "New Model for Providing Services", July 11, 2000 (www.medcom.dk/picnic).
[24] PICNIC Deliverable 4.1 "New Services", March 6, 2001 (www.medcom.dk/picnic).
[25] PICNIC Deliverable 4.2 "Functional Specification and Minimum Data Set for New Services", March 14, 2001 (www.medcom.dk/picnic).
[26] PICNIC Deliverable 6.1 "Identification of Generic Tools and Common Component, Part I", July 10, 2001. PICNIC Deliverable 6.2 "Specifications of Messaging Common Components", May 10, 2002. PICNIC Deliverable 6.3 "Specifications of Access to Patient Data Common Components", March 11,

2002. PICNIC Deliverable 6.4 “Specifications of Collaboration Common Components”, June 4, 2002 (www.medcom.dk/picnic).
[27] D. A. Solomon: Inside Windows NT, 2nd edition, Microsoft Press, 1998. B. Lewis: Threads Primer - A Guide To Multithreaded Programming, Prentice Hall, 1995.
[28] Information Technology – Portable Operating System Interface (POSIX) – Part 1: System Application: Program Interface (API) [C Language], 1995.
[29] Object Management Group, CORBA Messaging Specification, OMG Document orbos/98-05-05 ed., May 1998.
[30] P. Cao and S. Irani: Cost-aware WWW proxy caching algorithms, Proceedings of the USENIX Symposium on Internet technologies and Systems (USITS), Monterey, CA, Dec. 1997.
[31] O. Othman, C. O’Ryan, D.C. Schmidt The Design and Performance of an Adaptive CORBA Load Balancing Service, http://www.cs.wustl.edu/~schmidt/PDF/load_balancing.pdf.
[32] D.C. Schmidt, M. Stal, H. Rohnert, F. Buschmann, Pattern-Oriented Software Architecture, volume 2, Wiley, 2000.
[33] Microsoft Component Object Model http://www.microsoft.com/com/default.asp
[34] Christopher Et Al Blexrud, Jonathan Crossland, Dino Esposito, Jason Hales, Whitney Hankison, Vishwanath Honnaya, Tim Huckaby, Slava Kristich, Edward Lee, Rockford Lhotka, Brian Loesgen, Stephen Mohr, Simon Robinson, Ash Rofail, Brad Sherrell, Scott Short, Dan Wahlin, Professional Windows DNA: Building Distributed Web Applications with VB, COM+, MSMQ, SOAP, and ASP, Wrox Press Inc; 1st edition, September 29, 2000.
[35] Enterprise JavaBeans Technology http://java.sun.com/products/ejb/
[36] Java 2 Platform, Enterprise Edition (J2EE) http://java.sun.com/j2ee/
[37] CORBA Web Site http://www.corba.org/
[38] Web Services Activity, W3C Architecture Domain http://www.w3.org/2002/ws/
[39] CHARM (Comprehensive Health Assistance and Resource Management) IST-2000-25389 project http://www.charm.cup2000.it
[40] HARP (Harmonization for the Security of the web technologies and Applications) IST-1999-10923 project http://www.telecom.ntua.gr/~HARP/HARP/HARP1.htm
[41] RESHEN (Regional Secure Healthcare Networks) IST-2000-25354 project http://www.biomed.ntua.gr/reshen
[42] McConnell, Hamilton: Information assurance in the 21st century, supplement to IEEE Computer, Security and Privacy – 2002.
[43] DIRECTIVE 2002/58/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications) http://europa.eu.int/eur-lex/pri/en/oj/dat/2002/l_201/l_20120020731en00370047.pdf
[44] ISO/ CD TC 215: TR 21089, Health Informatics -- Trusted end-to-end information flows.
[45] The XML Signature initiative: Joint W3C/ IETF XML-DSig Working Group http://www.w3.org/Signature/
[46] XML Signature W3C Recommendation. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
[47] OMG Model Driven Architecture. http://www.omg.org/mda/