

Designing the architecture for an integrated Information System for Universities: the challenge ahead.

Federico Gallerani¹

¹ CINECA, *Solution and services for university administration* Department, via Magnanelli 6/3, 40033 Casalecchio di Reno (BO) Italy, f.gallerani@cineca.it

Keywords

U-GOV, CINECA, Architecture, Information Systems, Legacy Systems, Business Workflow, SOA, ESB.

1. EXECUTIVE SUMMARY

In 2005 CINECA launched the U-GOV project that aims at developing a new integrated information system for universities in Italy. The challenge of the project was to put together the CINECA Payroll and Student legacy applications with new applications for Accounting and Research in an unique integrated environment.

1.1. Designing The Architecture

Achieving this ambitious goal required a mixed refactoring/re-projecting approach and a number of different actions to be taken in order to address all the major issues:

- Adopting a common technological platform and framework in order to standardize the development of all the new software modules.
- Designing a set of shared archives to let every software module access to people and organizational structure data.
- Re-engineering the previous generation of CINECA applications to assure integration based on shared archives and single sign-on.
- Adopting a completely new and configurable Business Cycle Workflow layer to facilitate the communication between the modules of the U-GOV system and support the cross-organization processes.
- Introducing a SOA approach to rationalize the inter-module communications and prepare for interaction with third party systems.

Thanks to these development lines, the U-GOV system today has achieved a good degree of integration between the previous generation of applications and the new modules. The back-end layer is the core of the system architecture and the integration is mainly based on shared data services. The Service Oriented Architecture (SOA) approach started from the human resource application: critical elaboration components have been encapsulated and can be reused now "as a service" by other modules.

Moreover during 2009 some Italian universities have started to test the new Accounting system and the configurable Business Cycle Workflow layer supporting process management.

1.2. Future Developments

Recently CINECA has begun including more sophisticated integration platforms to push forward the integration in the SOA direction. In particular the Enterprise Service Bus approach seems to be very promising, particularly for the ability to rapidly integrate proprietary and third party systems in a unique technological environment.

2. THE STARTING POINT

CINECA is an Italian Interuniversity Consortium founded in 1969 to support Higher Education Institutions (HEI) and the (Italian) Ministry of University and Research (MUR) in all key sectors of Information and Communication Technology:

- Supercomputing and research applications
- Administrative applications and IT services for Universities
- IT services for the Italian Ministry of University and Research.

Today CINECA is made up of 39 members: 36 Universities, 1 research institute, the CNR (National Research Council) and the Italian Ministry of University and Research.

In 2005 CINECA launched the U-GOV project aimed to develop a new integrated information system for universities. The initial driver of this project was the realization of a new accounting solution in order to substitute the existing one, technologically obsolete and expensive on the maintenance side.

Beside this, new functional areas had to be developed for the first time such as Research, Course Planning and Human Resource administration.

Finally, a strong attention was put on process integration, seeking to realize an integrated system where the existing Payroll and Student legacy applications could work together with new applications in an unique integrated environment.

2.1. Assessment of Existing Information Systems

At the very beginning of the project an accurate investigation was conducted in order to assess the status of the existing Information System for Higher Education and decide the better evolution strategy. The following table summarizes the situation at the beginning of 2005:

Functional Area	Application Type	Technologies	First release	Number of University users
Accounting (CIA)	Rich Client	SmallTalk, Oracle	2000	44
Careers & Payroll, HR Management (CSA)	Rich Client, Web	Delphi, PHP, Oracle	2001	67
Learning & Student Services (ESSE3)	Rich Client, Web	PowerBuilder, Java, Oracle	2003	68

Table 1. Before the U-GOV Project: overview of the previous generation of CINECA administrative applications for University.

These were the main issues:

- Application were almost completely not integrated; communication was based on data import/export via file transfer.
- Applications shared nothing on the technological layer but the same DBMS (Oracle).
- The "Accounting" system (CIA) was an old application (SmallTalk) and needed to be completely re-written in order to introduce new functionalities.
- The "Human Resource Management" system had limited functionalities and needed to be re-written as well.
- "Careers & Payroll" and "Learning & Student Services" systems had to be considered a strategic company asset and thus the requirement was to integrate them rather to re-write.
- Universities needed new applications for other "core" functional areas: Course Planning and Research Management.

Designing the architecture for an integrated Information System for Universities: the challenge ahead.

2.2. The Strategy

To achieve this ambitious goals CINECA adopted a mixed refactoring/re-projecting approach and planned a number of different actions to be taken in order to address all the major issues. The following table lists the main project focuses and the corresponding taken actions.

ISSUE	ACTION TAKEN
Rationalize the technological platform	Define an architecture that could support both new developments and integration of the existing systems and a framework for the development of new modules.
Share common information between processes	Design a set of shared archives to support integration at data level.
Integrate Legacy Systems	Adopt shared archives, define a service architecture, introduce single sign-on.
Configure Business Processes	Introduce a configurable Business Cycle Layer.

Table 2. U-GOV project main focuses and corresponding taken actions.

3. BUILDING THE ARCHITECTURE

The first problem to solve for the design of an integrated system was the definition of a global architecture that could host newly developed modules as well as consolidated legacy application and allow an acceptable degree of integration and interoperability.

Because of the big technological diversities found, in the U-GOV project it was necessary to divide the applications into two different groups of modules, with different interoperability requirements:

- **U-GOV Java Platform modules:** this set of applications groups all the newly designed web-based modules that can rely on web-enabled integration (same interface, web services, SSO, ..).
- **U-GOV Legacy modules:** they are the older applications that are loosely integrated, mainly on a database communication level.

Integration between U-GOV Java Platform modules must be as complete as possible: modules must interoperate natively when a process crosses modules' borders; common information must be organized in shared archives rather than replicated; user experience must be uniform across the modules.

Integration between U-GOV Legacy modules and Java Platform modules could be a little more loose without compromising the process vision: interoperability can be based on different standards, re-using existing mechanisms where present; some degree of data-replication is accepted, depending on the type of information; user interface differences are accepted in older non-web applications.

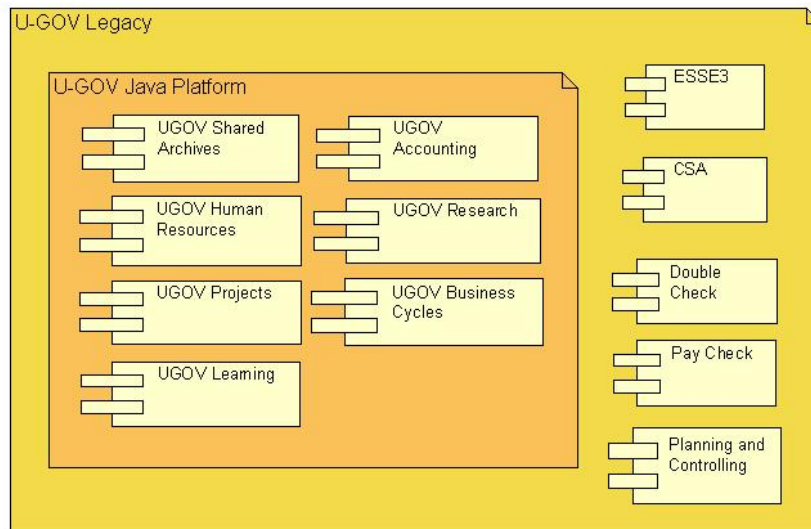


Figure 2. This image shows the U-GOV applications grouped in the two different module sets: U-GOV Legacy modules and U-GOV Java Platform modules.

3.1. The U-GOV Java Platform Architecture

From a technological point of view, the U-GOV Java Platform is a standard three-tier J2EE architecture composed of three main layers: DBMS, Application Server and Web Server. Each component in this architecture is based on the Java Enterprise pattern.

- **Database Layer.**

The U-GOV database identifies the set of components which host data objects and offer access services to the data. It is articulated in different logical schemes inside the same instance, each of which coincides with the database of a U-GOV module, or with the repository of common data shared through the whole system. The database layer uses Oracle 10i DBMS.

- **Application Layer.**

The Application layer hosts the business components that handle the applicative logic of the system. The various components, grouped together logically, make up the U-GOV modules, which in their turn define the functional Areas of the system. Application server layer is based on J2EE, EJB, JDO/JDBC standards.

- **Presentation Layer.**

The U-GOV presentation layer contains the components which handle management of the interaction between users and applicative pages of the system. As the business components of the application server, the presentation components are also organized according to the modules and Functional Areas of U-GOV.

The U-GOV system also provides its applicative functionalities through the use of web services. They therefore represent the public elements of integration with other information systems, such as university portals or other applications that have to inter-operate with U-GOV by exchanging data or orchestrating services. The presentation layer is based on HTML, JSF, XML and WS-* standards.

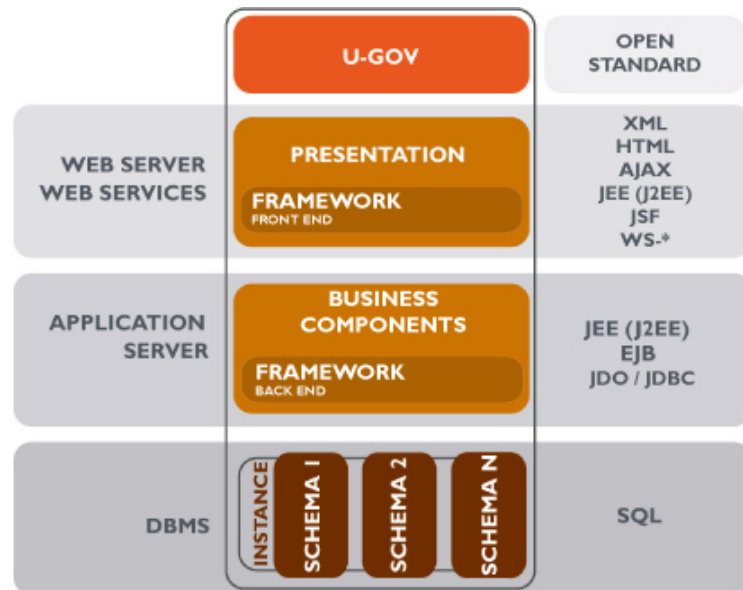


Figure 3. U-GOV Technological Architecture

3.2. The new U-GOV Technological Framework

In order to standardize the coding of all new software modules it was chosen to adopt a common development framework. Here the requirement was to build a robust development framework based on industry standards and to avoid critical vendor lock-ins, where possible.

The U-GOV framework contribute to standardize a set of common functionalities containing the best practices and patterns to be applied during development of the purely applicative components. The framework is divided into two different parts:

- the back end Framework provides the set of classes and services utilized by the system to access database objects. Specifically, the persistence engine contains the data management and memorization components which are performed on the database.
- the front end Framework provides the cross-functional interaction functionalities common to all presentation components as well as the mechanisms of system access authorization and control.

3.3. U-GOV Architecture: Logical View

In a logical-functional perspective, U-GOV is composed of a the shared data structure and the layer of "vertical" applicative modules. The shared data (covered in depth in the next chapter) is the logical module that host common information and services used by most business processes. The vertical modules (grouped in functional areas) are the components that deliver information and services to the end-user.

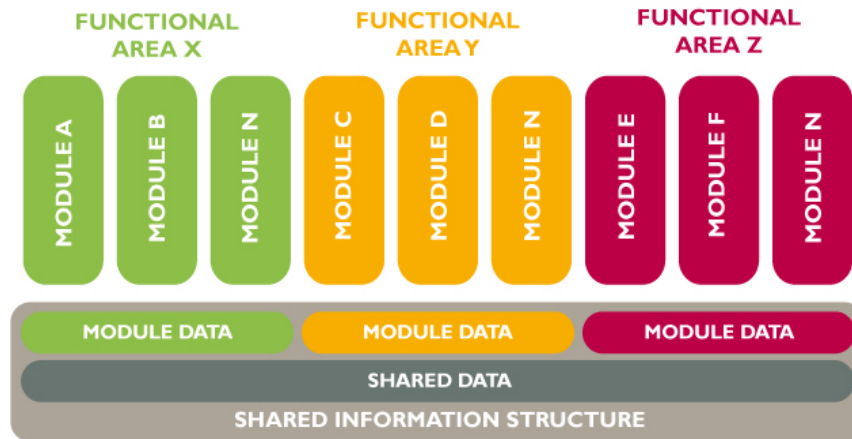


Figure 4. U-GOV Logical Architecture

4. SHARED DATA STRUCTURE

U-GOV overcomes the fragmentation of university information beginning at the data level. A unified database gives a solid foundation to govern and to map the university processes through cross-functional areas. This avoids the dispersion of information in different systems and archives.

Personal data (people) and Organizational Structure are the most critical information shared among all functional areas. This information is collected in a unique common archive that can be accessed from every vertical module. Personal data are organized in a three-level database structure that can be extended adding process specific information (extensions).

Level	Purpose
1 - Address Book	Stores basic identification data (Name, Date of birth, Fiscal Code, ecc.) and assign a common identifier for all the application.
2 - Persons	Stores common information related to the person, regarding addresses, contacts, preferences, ecc.
3 - Process Specific	Stores information related to the person but specific to one particular role that the person may cover: human resources, student, professor, ecc.

Table 3 Three-level database structure for Personal data.

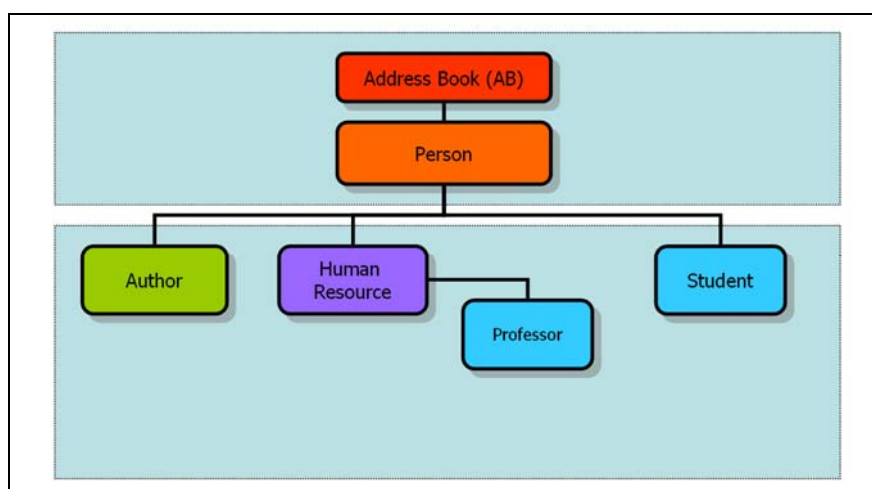


Figure 4. Shared Data Structure for personal data

The use of shared data registers throughout the applicative layer enables data sharing through the different processes, avoiding information duplication. Hence modifications to data are visible in real time in all applications. This approach allows unified management of the information in the whole university avoiding duplication of data and enabling cross-functional data analysis that correlate different information regarding the same person or structure. For example it becomes very easy to find every information (publications, courses, project, activities) regarding a professor in order to compose his Curriculum Vitae.

The shared data structure thus represents a fundamental premise for the development of university governance and to ensure and facilitate integration and evolution of the services offered by the University.

5. INTEGRATION OF LEGACY APPLICATIONS

A very critical question concerns the re-engineering of existing applications in order to assure a minimum degree of integration with the new breed of web-based services. Since legacy applications usually are the result of years of (not always linear) development and maintenance, every modification made to these procedure can be rather critical and expensive. Moreover, it's very likely that in existing applications some kind of discontinued or obsolete technology is used, making re-engineering an even more difficult and time-consuming task.

For all these reason it is very important to understand deeply the requirements and focus on the key aspect of integration.

The key integration requirements for the U-GOV project were:

- Share common data among different processes.
- Make different systems interoperate when they participate into a unique process.
- Improve usability and uniformity of user experience.

To answers to these requirements three types of solution have to be adopted:

- (1) Modify the legacy application so that they can access the shared archives described earlier.
- (2) Define services to enable interoperation between legacy systems and new modules in the key processes.
- (3) Adopt single sign-on mechanisms to reduce the need to log into different application to perform a single task.

5.1. Accessing shared data from legacy applications

Since the most important legacy systems to be integrated - "Career & Payroll" (CSA) and "Learning & Student Services" (ESSE3) - were applications based on Oracle DBMS, the most direct approach was to adapt them to access the new shared data structure to retrieve information about people and organizational structure.

Luckily it was sufficient to modify the data access components of these legacy application in order to read information contained in U-GOV shared data structure. This relatively low impact modification immediately allowed legacy applications to access the shared data without the need of data replication procedures.

Additionally, to give a simplified data representation that legacy systems could use and to enable database-level logic to operate without knowing the internals of U-GOV relational database structure, a set of *public views* has been added in U-GOV.

5.2. The Service Approach

With *service approach* we usually refer to a layer of services exposed by an application that gives access to data and functions offered by the system. This layer of services contributes to hide the complexity of the system internals and defines a contract between the service *supplier* and the service *consumer*.

In U-GOV services are exposed in two different ways:

- *EJB service classes* permit components that reside in the Java Platform to communicate in the most performing way;
- *Web Services* enable communication between heterogeneous technologies as in the case of Legacy systems and Java Platform components.

As an example, some services are exposed to update the shared data; these services are used by legacy systems to perform modifications to personal data in the context of the processes they implements. On the other side, payroll calculation services are exposed by the legacy CSA system; this allow to re-use the CSA business logic from newer U-GOV components.

5.3. Single Sign-on and federated authentication

Single sign-on functionalities are always a crucial element to consider when integrating applications. The ability to switch from an application/service to another with the same users/password and even without the need to re-enter user and password greatly improves the user experience and the perception of an integrated work environment.

The U-GOV project has adopted Shibboleth 2.x technology to achieve single sign-on and federated authentication. Shibboleth is the reference implementation for the SAML 2.0 specifications and offers the following elements:

- Authentication
- Single Sign On between different services/web resources
- Federation
- Release of user attributes (assertions).

Shibboleth thus is not an Identity Management System; this means that user creation and management must be carried out independently, for example in a LDAP system. Adopting Shibboleth as the standard authentication method for all the web applications in U-GOV enabled single sign-on between new and legacy web modules.

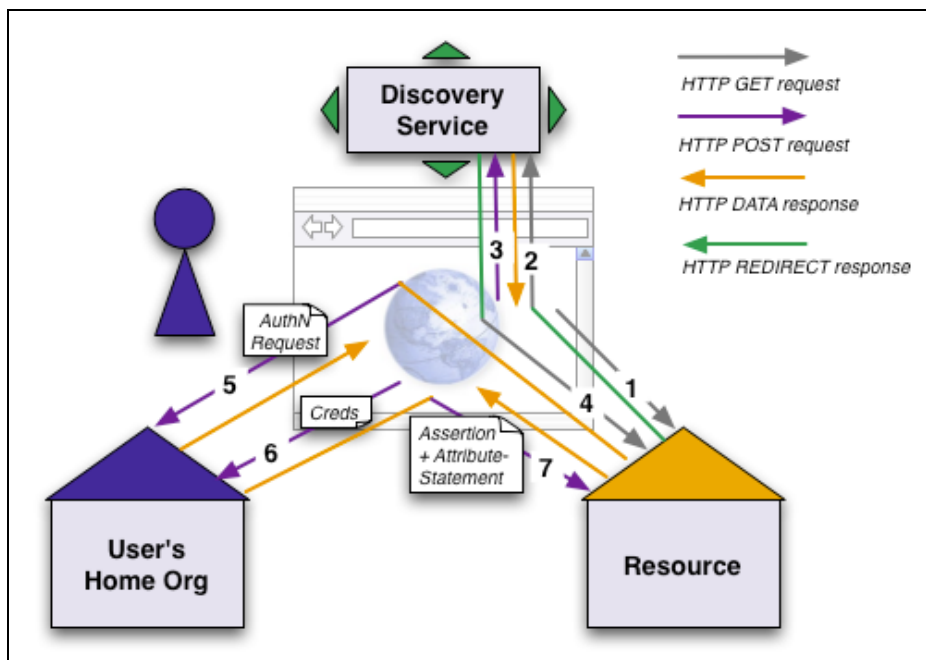


Figure 5. Federated Authentication Schema

6. BUSINESS CYCLE WORKFLOW LAYER

The context in which universities are operating nowadays is complex and subject to change. It is therefore fundamental to adopt information systems that can accommodate the continuous evolutions of their organization.

This flexibility is obtainable only through a strong separation between the “horizontal” business process logic - that describes the way each organization works and is subject to variations and depends on inner and outer university norms - from the “vertical” transactional components (here called *vertical engines*) which carry out transactions, computations and registration specific to a particular functional area (i.e. accounting, warehouse, human resources, etc.).

The reason for this separation is to gain the ability to re-configure the business process logic rather than hard-wire it into the system. Thanks to the separation between changing process rules (re-configurable) and vertical transactional services, U-GOV may be easily adapted to changes in management rules without direct modifications to the code. This feature permits to change the process configuration to put into practice decisions concerning university governance.

U-GOV is thus characterized by a software layer common to all modules, specifically dedicated to the management of university processes which defines a model of interaction and integration between the applicative components.

6.1. Ability to adapt to different requirements

The *Business Cycle Workflow Layer* (BCWL) has been created to allow the separation between business process logic and transactional components. Thanks to this approach, U-GOV does not impose a fixed organizational scheme, but it can be shaped on the structure and on the processes of a single university. This is fundamental in order to reduce the costs, either in the starting phase, where the new U-GOV system can emulate the previous application, or throughout time, adapting to the evolutionary processes.

The BCWL is based on the *process configuration* concept. A process “configuration” is the applicative representation of a process internal to the University. It therefore describes a workflow in terms of rules, documents, phases, constraints and decisions mapped out within the system. The configuration allow the different services exposed by the vertical engines in the modules to be coordinated and invoked in the correct points of the process.

6.2. The Business Cycle Workflow Architecture

The Business Cycle Workflow Architecture is intended to satisfy the following requirements:

- Separate *business processes* from “vertical” *engines* making the system easier to reconfigure when the business rules change.
- Promptly record *business events* separately from their effect in different applicative domains: the delivery of a ordered good is an event by itself that must be recorded and traced before and autonomously from its effects on accounting and warehouse.
- Allow governance of all the processes always permitting to know what happened where and when.

The architecture is composed of two layers (as shown in the next figure):

- Business Cycles and Business Documents Layer
- “Vertical” Engines Layer

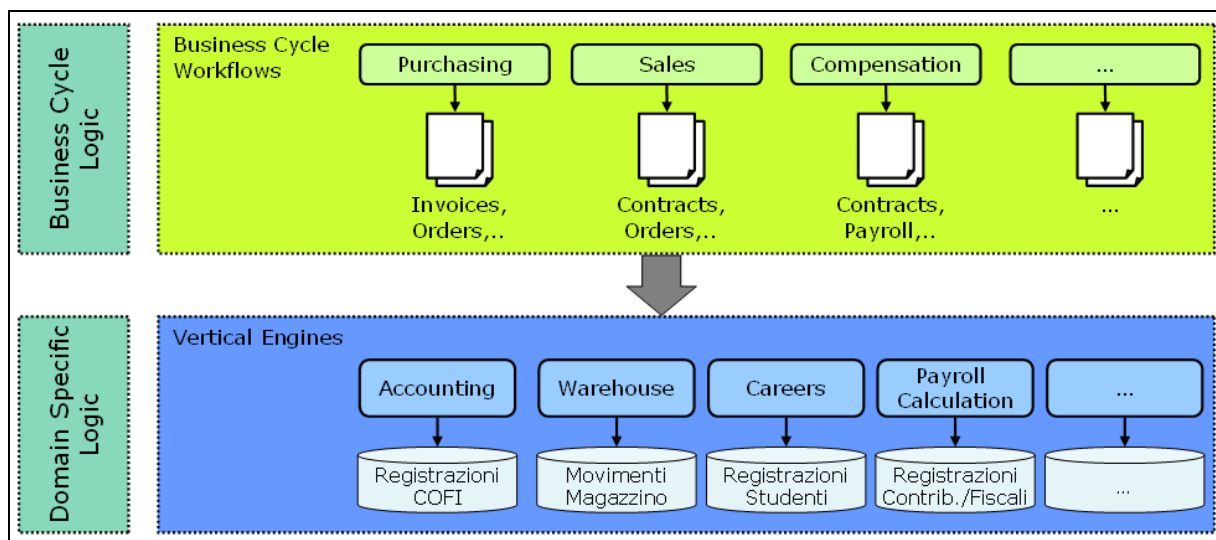


Figure 6. U-GOV Business Cycle Architecture

In this model the Business Cycle Layer is where the process logic can be configured creating new *documents* and defining their workflow and rules. Here the system administrator is free to define new document types, specifying the information they hold and designing the workflow steps where and the invocation of “vertical” services supplied by the engines. System users instantiate new documents following the process rules defined in the configurations. Documents instances are linked one to the other to trace the process execution steps.

On the other side, the Engines Layer are fixed transactional components that expose the services used by the Business Cycle Layer. Engines tasks usually are to record transactions (for example accounting records) and make computation (for example calculate payroll for a person).

7. THE CHALLENGE AHEAD

Recently CINECA has begun exploring more sophisticated integration platforms to push forward the integration in the SOA direction. In particular the Enterprise Service Bus approach seems to be very promising, particularly for the ability to rapidly integrate proprietary and third party systems in a unique technological environment.

7.1. An Enterprise Service Bus Architecture for the University?

An Enterprise Service Bus (ESB) is a distributed infrastructure used for enterprise integration. It consists of a set of service containers which integrate various types of IT assets. The containers are interconnected with a reliable messaging bus. Service containers adapt IT assets to a standard services model, based on XML message exchange using standardized message exchange patterns. The ESB provides services for transforming, orchestrating, and routing messages as well as the ability to centrally administer the distributed system.

Some work has been done to determine whether this type of architecture can be used to address integration issues between U-GOV modules. In particular some test have been conducted on Sun’s open source implementation of JBI service bus specifications, Open ESB.

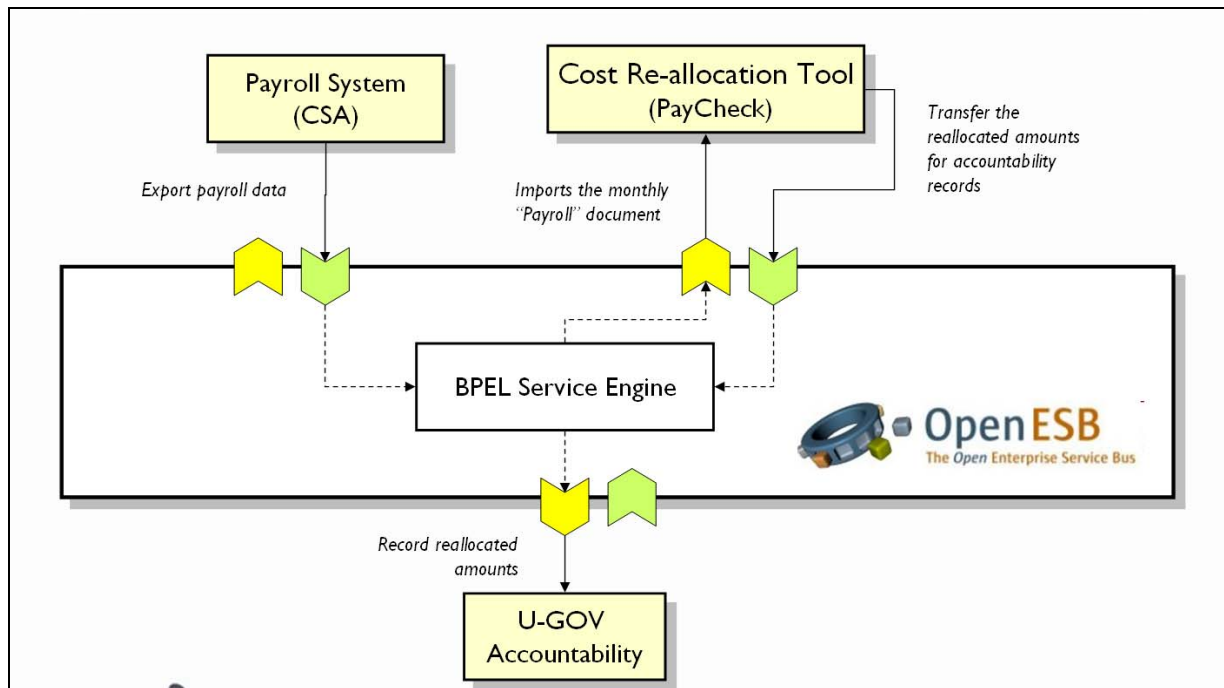


Figure 7. ESB architecture

Here are some considerations about the adoption of an ESB architecture in the context of a complex information system like U-GOV.

Key benefits

- Faster and cheaper integration. Ready to use service engines and binding components.
- Increased flexibility. Easier to change as requirements change.
- Standards-based technology. Avoid vendor lock-in.
- Scales from point solutions to enterprise-wide deployment (distributed bus).
- More configuration rather than coding.

Key disadvantages

- Integrated systems must have some sort of service or "port" (DB, WS, FTP, mail, CORBA, ...).
- Ongoing management of service and message versions is required to ensure the intended benefit of loose coupling.
- New skills needed to configure, manage, and operate an ESB.
- Extra overhead and increased latency caused by messages traversing the extra ESB layer, especially as compared to point to point communications.
- It normally requires more hardware resources than a simple point to point approach.

8. RESULTS AND CONCLUSIONS

Thanks to the guidelines described earlier in this document, the U-GOV system has achieved today a good degree of integration between the previous generation of applications and the new modules.

The back-end layer is the core of the system architecture and the integration is mainly based on fine-grained data access services and sharing of logical views at database level.

The service approach started from the human resource application: critical elaboration components have been encapsulated and can be reused now "as a service" by other modules; other services were exposed to access information about human resources.

Moreover during the current year some universities has started to test the new accounting system and the configurable Business Cycle Workflow supporting process configuration and management.

U-GOV today is activated in more than 30 Italian Universities and can be considered a successful project under many points of view, but still there are things that would benefit from some sort of improvement.

For sure more emphasis must be posed in the near future on the realization of coarse-grained business services as opposed to the fine-grained data services that are present today. This type of services is required for a successful introduction of more structured integration architectures (i.e. Enterprise Service Bus). The bigger impediment that must be overcome is on the organization side more than on the technical one, because it requires a change in the way people analyze problems and find solutions.

Another future evolution will take into account the ESB infrastructure. The introduction of an ESB to accommodate the many integration requirements is still a solution that have to be discussed. It would certainly improve and rationalize the management of integration logic but, as stated before, it needs to be supported by a robust and documented service layer and by new competences in the organization.