A Composite Domain Framework for Developing Electronic Public Services

Adegboyega Ojo, Tomasz Janowski and Elsa Estevez

April 2007

UNU-IIST Report No. 370

R

UNU-IIST and UNU-IIST Reports

A Composite Domain Framework for Developing Electronic Public Services

Adegboyega Ojo, Tomasz Janowski and Elsa Estevez

Abstract

The availability of domain frameworks to enable rapid development of Electronic Public Services (EPS) is essential to meet the increasing demand for mature EPS by various government stakeholders. This paper presents a composite domain framework comprising frameworks to build the Front-Office and Back-Office parts of an EPS. The framework supports a set of domain requirements obtained through a detailed analysis of over 30 concrete public services. After presenting these requirements, the framework is described in four stages - architecture, design, implementation and instantiation - all using UML to capture the artifacts built during development. We also illustrate the application of the framework through a case study in developing an Electronic Licensing Service by means of framework instantiation. We conclude with some comments on the complexity, flexibility and performance of the framework. This work was carried out as part of the e-Macao Project to build a foundation for e-Government in Macao, funded by the Government of Macao SAR.

Adegboyega Ojo is a Research Fellow at the Center for Electronic Governance - United Nations University - International Institute for Software Technology (UNU-IIST-EGOV) in Macao, on leave from the Department of Computer Sciences, University of Lagos, Nigeria where he is a Senior Lecturer. Before his affiliation with UNU-IIST, he led major software development projects involving large multinationals in Nigeria. He has been involved in e-government research and development activities since 2004 at UNU-IIST as a member of the project team working on the e-government development in Macao. His research interests include Software Infrastructure for Electronic Government, E-Government Measurement, Ontology, Semantic Interoperability, and Applications of Computational Intelligence. He holds a doctorate degree in Computer Science from the University of Lagos in Nigeria. He is also a member of the Computer Professionals of Nigeria.

Tomasz Janowski is the Head of the Center for Electronic Governance - United Nations University - International Institute for Software Technology in Macao (UNU-IIST-EGOV) and a Senior Research Fellow of UNU-IIST. Tomasz Janowski is the author or co-author of numerous publications in Computer Science, Software Engineering and Foundations. He has been a Program Committee member at many international conferences, supervised a number of international students and staff, and led several research, development and capacity-building projects. These include the e-Macao Program to build a foundation for Electronic Government in Macao and the UNeGov.net initiative to building a global community of practice for Electronic Governance. As part of the UNeGov.net initiative, he organized many workshops and schools on Electronic Governance in developing countries. His current research interests include foundations of Electronic Governance, tools and applications of formal methods, and rigorous development of enterprise software, particularly software for the public sector. Tomasz Janowski holds a PhD in Computer Science from the University of Warwick, England, and an MSc in Mathematics from the University of Gdansk, Poland.

Elsa Estevez is a Project Staff at the Center for Electronic Governance - United Nations University - International Institute for Software Technology in Macao (UNU-IIST-EGOV). She is also an Assistant Professor at the Universidad Nacional del Sur (UNS) in Bahia Blanca, Argentina where she teaches subjects related to Software Engineering. Prior to joining UNS, she has been working as a Project Leader, Project Manager, System Analyst and Programmer in Information Technology for major industries, including banking, in Argentina. While in UNU-IIST-EGOV, Elsa Estevez contributed to various tasks of the e-Macao and UNeGov.net projects, from research and development, through training, to networking. Her current research interest include: Electronic Government, developing software infrastructure for Electronic Government, and Component-Based Software Development. Elsa Estevez holds an MSc degree in Computer Science from UNS and a Licenciado Degree in Computer Science from Universidad de Buenos Aires, Argentina.

Contents

1 Introduction

A Framework is a partial design of an application in a given domain, leading towards its implementation [2]. Frameworks capture both commonalities and variabilities in a family of applications that belong to a particular domain, enabling effective re-use of components, design patterns and other development artifacts. As such, they increase the reliability of domain-specific applications and decrease the time required to build them. Specific applications are obtained by instantiating a framework - filling the hooks or hotpots in the framework, and supplying additional components as required [1].

Frameworks can be classified in different ways. One classification considers the nature of the components in the framework and the type of functionality they provide. Following this classification we identify Application Frameworks and Domain Frameworks [2]. Application Frameworks provide a whole range of functions required by any application, irrespective of its domain, such as a graphical user interface, database management or Internet connectivity. Examples of Application Framework are Microsoft and Java Foundation Classes (MFC and JFC) used to build Windows Applications, and also the Spring framework for building Java Enterprise Applications. In contrast, Domain Frameworks support applications that belong to specific vertical domain such as Banking, Health, Transportation, or Public Services.

This paper proposes a Domain Framework to enable rapid development of Electronic Public Services (EPS). In particular, the framework is targeted at building enterprise applications to support electronic delivery of authorization and certification types of public services. For instance, licenses and permits are issued through authorization services, while marriage or tax clearance certificates are issued through certification services. The framework is documented using a UML Framework Profile [5], through Design Class Diagrams, and its implementation is based on a family of Enterprise Application Frameworks such as Spring and Hibernate. We also show how the framework enables rapid development of a prototype Electronic Licensing Service through framework instantiation.

This work is based on three premises: (1) citizens and businesses increasingly demand that public services are accessible through mature electronic channels, (2) public services can be grouped into a few categories, for instance Certification, Authorization, Control and Production [8], which capture inherent similarities between them, and (3) at present, no generic, open-source, open-standard-based Domain Framework exists for Electronic Public Services.

The rest of the paper is organized as follows. Section 2 provides an overview of the EPS domain. Section 3 identifies commonalities and variabilities among EPS. The framework is discussed in Section 4 - requirements, architecture and design, and implementation. The process for instantiating the framework is also described in Section 4 and elaborated through a case study in Section 5. Conclusions are provided in Section 6.

2 Electronic Public Services

Governments offer a large number of public services to their customers and stakeholders. According to the Governance Enterprise Architecture (GEA) Object Model [8], these services can be categorized into four classes: Certification, Control, Authorization and Production.

Through Certification Services, governments issue various kinds of declaration to applicants - citizens, business, visitors, etc. Examples of certification services include issuance of personal documents like passports, marriage certificates or criminal-record clearance. Control Services generally aim to ensure conformance to regulations, particularly through visits and inspections by government officials or authorized representatives. Through Authorization Services, government agencies grant permissions and approvals to the applicants. Authorization Services include granting licenses and permits as well as approving requests, say for welfare support, by the applicants. Production Services include infrastructural and utility services, such as electricity, water or telecommunication services.

The transformation of public services into Electronic Public Services is a common type of initiative carried out by governments as part of their e-Government and Public Sector Modernization programs. EPS can be delivered at different levels of maturity: from information on the procedure on how to apply for a service, through online support to downloading and uploading application forms and supporting documents, to fully transactional services. While the full transactional access is not required in all cases, citizens generally demand such services for certification and authorization types. However, large-scale development and deployment of mature EPS requires a significant investment in the underlying software infrastructure. Domain Frameworks, supporting rapid development of domain-specific aspects of applications, are essential elements in such an infrastructure.

3 Domain Analysis

We briefly analyze the domain of Electronic Public Services, considering its scope (Section 3.1), the amount of commonality and variability among EPS particularly belonging to the Authorization and Certification types (Section 3.2) and possible extensions (Section 3.3).

3.1 Domain Scoping

We assume that the space of public services can be roughly organized into four categories - Certification, Control, Authorization and Production, as specified in the GEA model [8]. While over 100 public services were surveyed and documented by the authors [9], only 31 of them were analyzed in depth as a basis for domain analysis and framework development. Specifically, 25 licensing services (Table 1) and 6 social welfare services (Table 2) delivered by the Government to business entities and citizens were studied. Both licensing and welfare services can be classified as GEA Authorization Services.

no	licensing services
1	temporary/permanent electricity license
2	construction and utilization license
3	aviation industry license
4	certificates of origin license
5	import and export license
6	trademark registration
7	temporary and permanent factory license
8	auditing firm license
9	media house registration
10	marine operations and works license
11	marine taxi license
12	food and animal origin license
13	food and beverage license
14	radio network license
15	radio station license
16	nursing home establishment license
17	pharmacy license
18	private educational institution license
19	adult education and tutorial centre license
20	tourist guide
21	travel agency license
22	bank license
23	money exchange agent license
24	remittance company license
25	financial intermediaries license

Table 1: Licensing services

no	welfare services
1	social houses
2	financial aids to students
3	retirement pensions
4	financial assistance to individuals and families
5	post-graduate scholarships
6	survivor pensions

Table 2: Welfare services

The analysis in [9] shows that authorization and certification services have similar underlying processes. Evidence also suggests that such services are frequently offered by governments and demanded by the public [10]. For example, 75% of the 20 basic public services by EU are certification and authorization services. We thus argue that our analysis based on the 31 services suffices for developing a representative EPS framework.

3.2 Commonalities and Variabilities

In this section we capture common features of EPS and identify the associated variabilities. The identification of variation points in the common feature is particularly important as they define the space of concrete services, or their variants, that can be developed.

We identify the common features of the domain through three basic steps: (1) abstracting the underlying processes for online delivery of the 25 licensing and 6 welfare services (see details in [11][12]), (2) integrating the processes for Certification and Authorization services as provided by the GEA model, and (3) aligning the processes obtained through the steps 1 and 2.

The generic process obtained in step 3 and described in Figure 1, consists of the following steps:

s1) Submission of the request/application by an applicant
s2) Submission of evidences by the applicant
s3) Checking for completeness and verifying correctness of the submitted information
s4) Assessing if the applicant is eligible for the service
s5) Evaluation of the application
s6) Requesting external opinions, if necessary
s7) Deciding on the application
s8) Communicating the outcome to the applicant

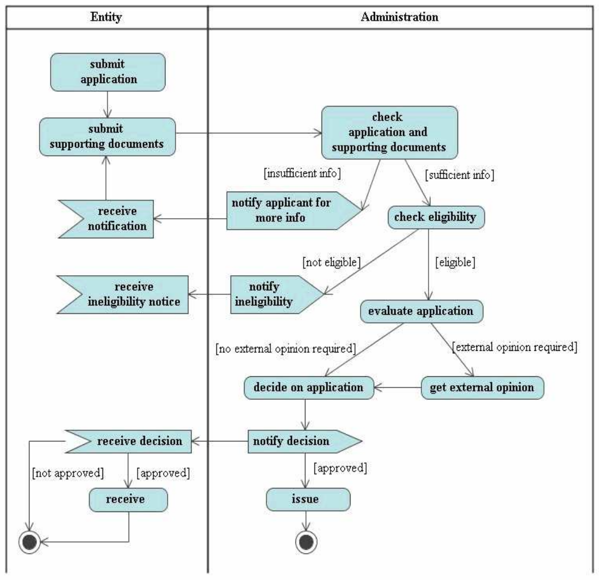


Figure 1: Generic EPS process

The features common to all authorization and certification services captured by the EPS process in Figure 1 include: (s1) application, (s2) evidence, (s3) completeness, (s4) eligibility check, (s5) evaluation, (s6) external opinion, (s7) decision and (s8) communication. Table 3 shows possible variations: s1 - different kinds of application forms; s2 - different lists and forms of evidences; s4, s5 and s7 - various criteria applied to define eligibility, carry out evaluation and make decisions; and c6 - different types of requests and protocols used to ask and receive external opinions. Variations s1, s2, s4 and s6 may be realized by parameterization of common features and dynamic binding of specific implementations for variants, while s5 and s7 may be supported through hooks and under-specification of the associated feature.

no	common feature	variations
s1	application	application forms
s2	evidence	lists and forms of evidences
s3	completeness	same
s4	eligibility check	criteria (structured)
s5	evaluation	criteria (unstructured)
s6	external opinion	types of requests, protocols
s7	decision	criteria (unstructured)
s8	communication	same

Table 3: Variations in common features

3.3 Extensional Features

In addition to customization through variations, the process in Figure 1 may also require extensions. For instance, as licensing services vary in terms of concrete business process steps, such steps must be accommodated in the Domain Framework and additional components must be developed to implement such steps. To this end, the architecture and design of the Domain Framework in Section 4 accommodates both hotspots and extensions.

4 Framework Development

In this section, we present requirements (Section 4.1), architecture and design (Section 4.2) and implementation (Section 4.3) details of the framework. We also briefly describe how the framework can be instantiated for designing and implementing concrete EPS (Section 4.4).

4.1 Domain Requirements

Domain requirements are largely derived from the analysis of the generic process in Figure 1. These requirements are roughly partitioned into three categories – Front-Office (FO), Back-Office (BO) and Mid-Office (MO). Several FO and BO components constitute a typical public service delivery environment. The MO represents the functions of a coordination office between several FO and BO components in a public organization.

FO Requirements: These requirements capture the first four process steps s1-s4 covering: submission of applications and evidences, checking completeness of submitted documents, establishing eligibility of applicants to receive the service, and tracking application status. FO requirements also cover basic user management, user notification, and dispatch of application requests to the MO. More information on FO Requirements are provided in [6]. Figure 2 presents FO Requirements as use cases.

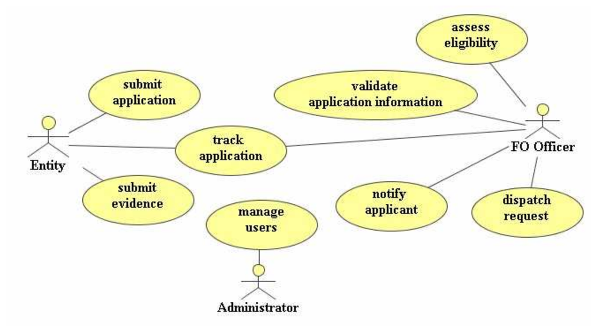


Figure 2: Front-Office use cases

BO Requirements: These requirements correspond to the steps s5-s8. They support processing of applications: receiving requests from the MO, certifying the accuracy of submitted information, evaluating applications, seeking external opinions on applications, deciding on the application cases, and communicating outcomes to the applicant. In addition to the core requirements, auxiliary requirements such as signaling end-of-task completion to the MO and user management are also included. Detailed information on BO requirements is provided in [7]. Figure 3 depicts the corresponding use cases.

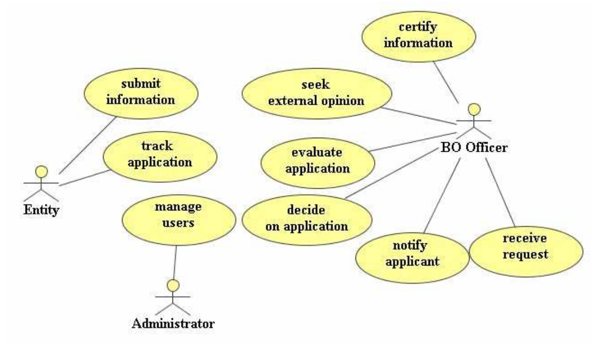


Figure 3: Back-Office use cases

MO Requirements: The Mid-Office coordinates tasks carried out by various Back-Offices through well-defined business processes. The use of business processes to coordinate BO workflows supports variability and flexibility in BO tasks (see Table 3). For instance, the task to evaluate a license application from a telecomm operator is very different from the task to evaluate an application for welfare assistance. MO requirements support: receipt of requests from the FO, definition of business processes, association of requests with business processes, and execution of business processes to coordinate processing of requests at the BO. The execution of a business process includes the creation of tasks for each step, assignment of roles to tasks, dispatch of tasks to the BO, and receiving progression signals. Detailed information on the MO is provided in [12]. Figure 4 depicts the use cases.

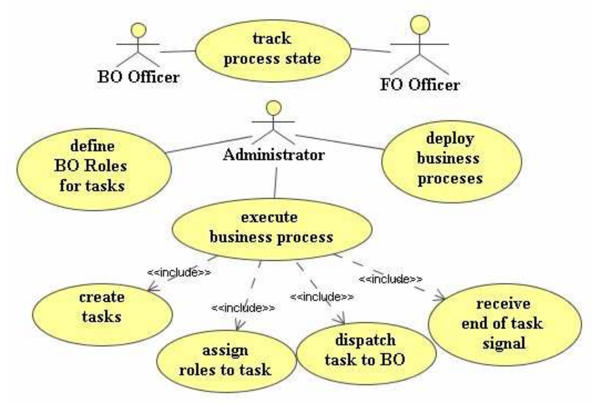


Figure 4: Mid-Office use cases

4.2 Architecture and Design

This section presents a high-level architecture and design of the Framework, partly using a UML Profile for describing frameworks [5]. In particular, we specify the kernel of the framework and show how it accommodates variabilities associated with the generic EPS process.

Architecture: The EPS Framework consists of three complementary frameworks to support the development of enterprise applications for FO, MO and BO functions of a typical public agency. Figure 5 depicts these frameworks as part of the Composite EPS Framework package.

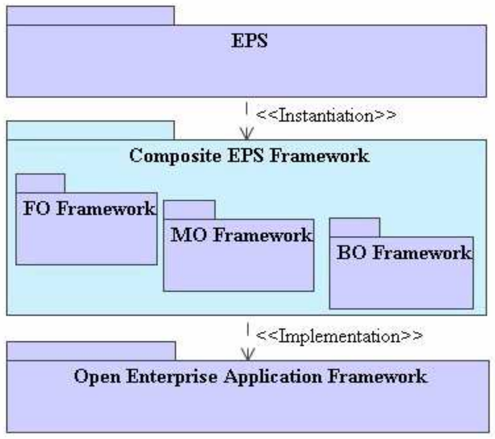


Figure 5: Composite EPS Framework

A complete EPS is obtained through the instantiation of the three frameworks. For instance the FO, BO and MO parts of the EPS are obtained by instantiating the corresponding frameworks. The EPS Framework itself relies on the open enterprise application framework to obtain basic middleware support for database and internet connectivity, dependency management, messaging, etc.

Detailed Design: The design of the EPS Framework is presented using a design class diagram with stereotypes, constraints and comments for specifying variation points. Three types of stereotypes are used to document the framework - Abstract, Template and Hook. The base class for Abstract is Class, while Template and Hook describe variation points related to methods. Abstract classes, as expected, must be sub-classed in the instantiation phase. Template methods define abstractly the interactions between abstract classes, while Hook methods supply concrete implementations [5]. The Covariant constraint shows classes that cooperate with each other. These classes are mutually affected by changes to either of them. For instance, specific request and response types are Covariants as shown in Figure 6. Tagged values indicated as notes show values for specific variation point attributes, e.g. a binding time. For instance, Figure 6 shows that requestbody and attachment attributes of the RequestTypeImpl class can be specified at run-time through dynamic binding. The details of the frameworks are described below.

FO Framework Design: Figures 6 and 7 show design classes for the FO Framework. Figure 6 depicts the operations exposed by the FO Framework through the FrontOfficeService component. Each operation supports specific requests from client applications such as submission of applications - submitApplication() and status tracking - trackApplication(). These operations are template methods since they invoke hook methods that requires specializations or implementations in the handler components. These operations also rely on pairs of request (RequestTypeImpl) and response types (ResponseTypeImpl) as parameters.

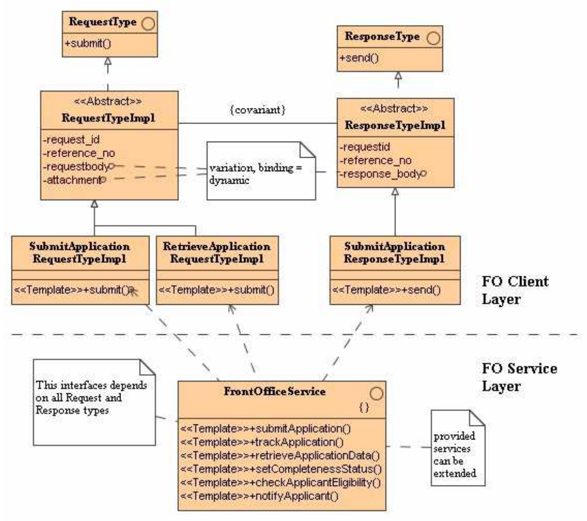

RequestType
+submit()
ResponseType
+send()
<<Abstract>>
RequestTypeImpl
-request_id
-reference_no
-requestbody
-attachment
{covariant}
<<Abstract>>
ResponseTypeImpl
-requestid
-reference_no
-response_body
variation, binding = dynamic
SubmitApplication RequestTypeImpl
<<Template>>+submit()
RetrieveApplication RequestTypeImpl
<<Template>>+submit()
SubmitApplication ResponseTypeImpl
<<Template>>+send()
FO Client Layer
FO Service Layer
This interfaces depends on all Request and Response types
FrontOfficeService
{}
<<Template>>+submitApplication()
<<Template>>+trackApplication()
<<Template>>+retrieveApplicationData()
<<Template>>+setCompletenessStatus()
<<Template>>+checkApplicantEligibility()
<<Template>>+notifyApplicant()
provided services can be extended

Figure 6: FO Framework services

Specific instances of EPS may extend the operations provided by the FO Framework. Extending these operations implies that additional request and response types must be provided for new operations. Request and Response types are also required by client applications to invoke the FO Framework Services. Figure 7 shows how the services provided by the FrontOfficeService interface are implemented by handler components in the FO Handler Layer, providing hook methods through its interface. The handler components in turn rely on Object-Relational Application Framework for persistence. The dependency management for the entire Framework is also provided through the application framework used. Detailed design of the FO Framework is described in [6].

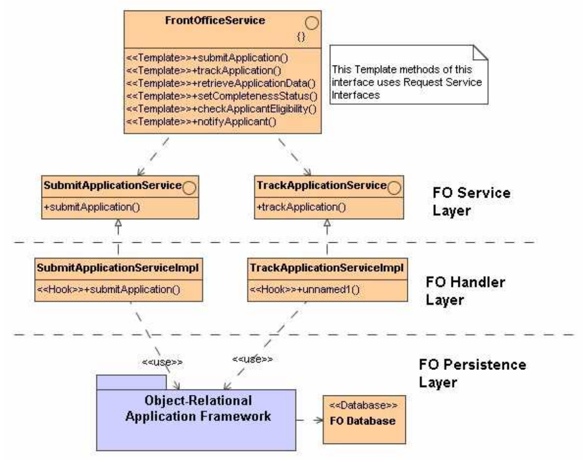

FrontOfficeService
{}
<<Template>>+submitApplication()
<<Template>>+trackApplication()
<<Template>>+retrieveApplicationData()
<<Template>>+setCompletenessStatus()
<<Template>>+checkApplicantEligibility()
<<Template>>+notifyApplicant()
This Template methods of this interface uses Request Service Interfaces
SubmitApplicationService
+submitApplication()
TrackApplicationService
+trackApplication()
FO Service Layer
SubmitApplicationServiceImpl
<<Hook>>+submitApplication()
TrackApplicationServiceImpl
<<Hook>>+unnamed1()
FO Handler Layer
<<use>>
<<use>>
FO Persistence Layer
Object-Relational Application Framework
<<Database>>
FO Database

Figure 7: FO Services implementation

BO Framework Design: The BO Framework is designed similarly to the FO. Like FO, the BO Framework exposes a service interface to client applications as in Figure 8. The operations in this interface include: evaluating applications - evaluateApplication(), making decisions - makeDecision() and requesting collaborations - requestCollaboration(), all related to BO functions. The difference between FO and BO components lies in the degree of customization required for BO components, as BO functions tend to vary significantly from one service to another. For instance, the steps for licensing a telecommunication operator would be much more complex than the steps for welfare benefits.

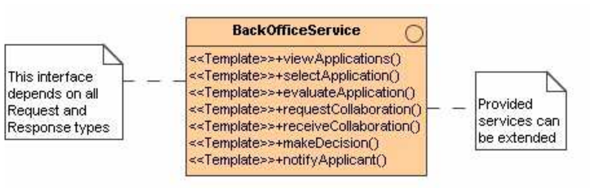

This interface depends on all Request and Response types
BackOfficeService
<<Template>>+viewApplications()
<<Template>>+selectApplication()
<<Template>>+evaluateApplication()
<<Template>>+requestCollaboration()
<<Template>>+receiveCollaboration()
<<Template>>+makeDecision()
<<Template>>+notifyApplicant()
Provided services can be extended

Figure 8: BO Framework services

To support this degree in variation, handler components supporting evaluation (s5), collaboration (s6) and decision (s7) steps are defined abstractly to accommodate a variety of implementation components which can be orchestrated through the MO workflow; see the two components implementing evaluation in Figure 9. Detailed design for the BO Framework is provided in [7].

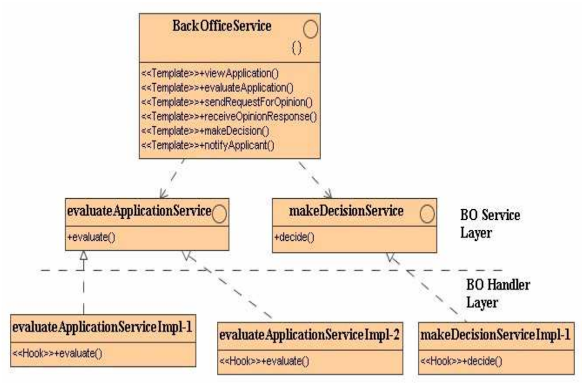

BackOfficeService
{}
<<Template>>+viewApplication()
<<Template>>+evaluateApplication()
<<Template>>+sendRequestForOpinion()
<<Template>>+receiveOpinionResponse()
<<Template>>+makeDecision()
<<Template>>+notifyApplicant()
evaluateApplicationService
+evaluate()
makeDecisionService
+decide()
BO Service Layer
BO Handler Layer
evaluateApplicationServiceImpl-1
<<Hook>>+evaluate()
evaluateApplicationServiceImpl-2
<<Hook>>+evaluate()
makeDecisionServiceImpl-1
<<Hook>>+decide()

Figure 9: Flexibility through multiple implementations

MO Framework Design: FO and BO are both defined as white-box Frameworks - the hotspots in the framework require concrete implementation of the abstract classes. In contrast, MO is a black-box framework where simple parameterization and dynamic binding are all that is required for instantiation. The framework consists of three kinds of components - Definition, Execution and Logging as in Figure 10. The first provides components for parameterization, the second executes business process definitions loaded through Definition, while the last logs all activities of the instantiated Workflow engine.

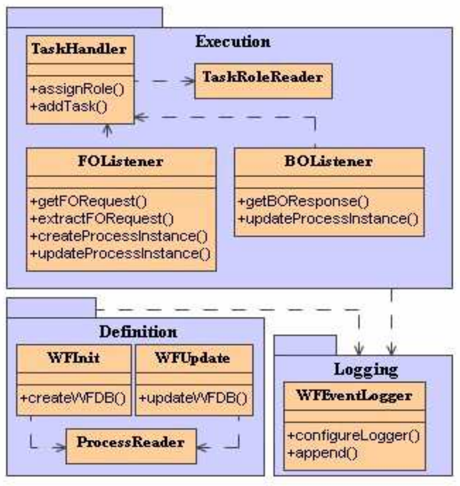


Figure 10: Mid-Office components

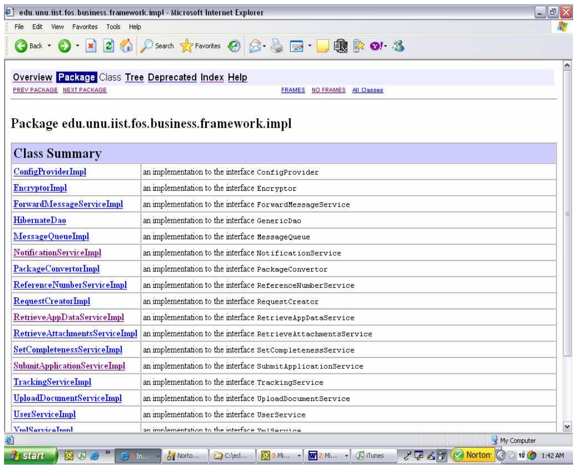
The Execution subsystem provides components for retrieving requests from FO Applications through the Message Oriented Middleware (MOM), dispatching created tasks to specific Back-Office roles for processing, and receiving end-of-task processing signals from BO applications also through a message queue.

The MO framework plays a major role in achieving variability at the BO. By using business processes to drive invocation of BO services, additional components can be added to the BO framework to implement variations, and orchestrated through a business process.

4.3 Implementation

The three frameworks comprising the EPS framework were all implemented using open-source technologies and based on open-standards, particularly Enterprise Java. Figure 11 depicts part of FO framework documentation.

FO and BO Frameworks: The FO and BO Frameworks were implemented using J2EE technologies, including Web Services, and the Spring Application Framework. The J2EE Technology (java.sun.com/j2ee) provides support for web and business components as part of enterprise applications. Spring (springframework.org) is a lightweight Application Framework for developing enterprise applications. Spring provides several features including: transaction management, data integration services - Object-Relational Mapping and Data Access Objects, and support for Model-View-Control framework and for Aspect-Oriented Programming. Dynamic binding and persistence of the FO and BO frameworks were both implemented through Spring. Specifically, Hibernate (hibernate.org) was adopted for Object-Relational Mapping within the Spring Application Framework. In addition, databases were implemented using MySQL Relational Database Management System (mysql.com).



Package edu.unu.iist.fos.business.framework.impl

Class Summary	
ConfigProviderImpl	an implementation to the interface `ConfigProvider`
EncryptorImpl	an implementation to the interface `Encryptor`
ForwardMessageServiceImpl	an implementation to the interface `ForwardMessageService`
HibernateDao	an implementation to the interface `GenericDao`
MessageQueueImpl	an implementation to the interface `MessageQueue`
NotificationServiceImpl	an implementation to the interface `NotificationService`
PackageConvertorImpl	an implementation to the interface `PackageConvertor`
ReferenceNumberServiceImpl	an implementation to the interface `ReferenceNumberService`
RequestCreatorImpl	an implementation to the interface `RequestCreator`
RetrieveAppDataServiceImpl	an implementation to the interface `RetrieveAppDataService`
RetrieveAttachmentsServiceImpl	an implementation to the interface `RetrieveAttachmentsService`
SetCompletenessServiceImpl	an implementation to the interface `SetCompletenessService`
SubmitApplicationServiceImpl	an implementation to the interface `SubmitApplicationService`
TrackingServiceImpl	an implementation to the interface `TrackingService`
UploadDocumentServiceImpl	an implementation to the interface `UploadDocumentService`
UserServiceImpl	an implementation to the interface `UserService`
XmlServiceImpl	an implementation to the interface `XmlService`

edu.unu.iist.fos.business.framework.impl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Overview Package Class Tree Deprecated Index Help
PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES All Classes
My Computer
start In... Norto... C:\ecl... 3 Mi... 2 Mi... iTunes Norton 1:42 AM

Figure 11: Part of FO Framework documentation

MO Framework: The Mid-Office Application, also called the Workflow Engine, was implemented using JBoss jBPM - a J2EE-based open source platform for managing business processes and workflows (jboss.com). The platform supports two process description languages - Java Process Definition Language (jPDL) and Business Process Execution Language (BPEL). The MO Framework implementation currently supports the former.

4.4 Framework Instantiation

The complementary nature of the three frameworks was utilized to provide a holistic support for developing EPS applications. The full architecture of a framework-enabled EPS is depicted in Figure 12. In order to build such applications using the framework, the framework instantiation process consists of two basics steps.

First Step: This step entails the instantiation of each individual framework to build three separate applications: FO, MO and BO. Instantiating the FO Framework involves providing specific implementations for all hook methods in the implementation components in the handler layer, such as implementing the submitApplication method of the SubmitApplicationServiceImpl component. Additional services could be provided to the FO application by extending FrontOfficeService interface and providing necessary request and response type implementations for the new services. Since MO is a black box framework, instantiation is done by simply providing business process definitions for different request types together with a role-task assignment map, and dynamically loading these definitions through configuration files. The process for instantiating the BO Framework is similar to FO. However, due to variations expected from different instances, a strategy is to provide multiple implementations of basic BO handler component interfaces (see evaluation components in Figure 9). Using the MO application, a complex evaluation can be realized by sequential composition of simpler evaluation sub-steps.

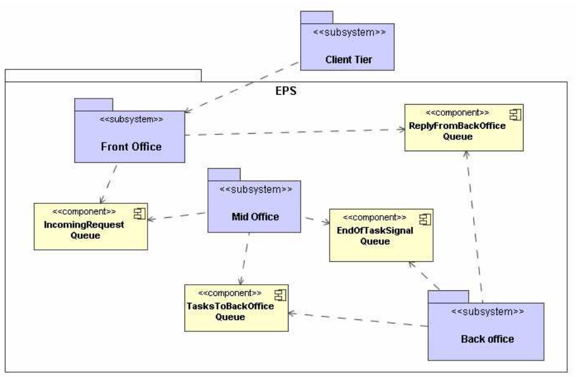
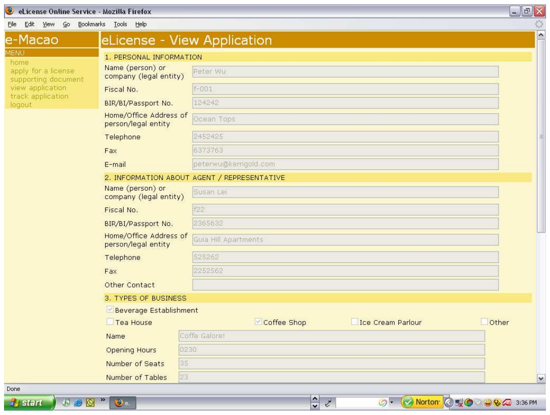


Figure 12: Full EPS architecture

Second Step: The second step entails the composition of these three applications into a single EPS by setting up connectors (specifically message queues) between them. Figure 12 shows the full architecture of an EPS. Four message queues are used as connectors between FO, BO and MO applications. The IncomingRequestQueue queue connects FO and MO applications in one direction **-** for sending requests from FO to MO applications. The TasksToBackOfficeQueue queue connects the MO and the BO and so does EndOfTaskSignalQueue. Tasks are inserted in the TasksToBackOfficeQueue by MO applications for processing by BO applications, and BO applications insert task completion signals into EndOfTasksSignalQueue. The fourth queue - ReplyFromBOQueue connects the FO and BO applications directly to implement synchronous requests, for instance for tracking application status.

5 Case Study – Electronic Licensing Service

We briefly describe the development of a prototype Electronic Licensing system based on our framework. The system implements the basic domain use cases described in Section 4.1, realizes the architecture described in Section 4.2 and is based on the implementation presented in Section 4.3. FO and BO parts of the Electronic Licensing system were developed by instantiating the EPS framework as described in Section 4.4, developing both client and web tiers for the FO part, and developing a Swing client tier for the BO part. Figure 13 shows an example screenshot produced by the FO application.



eLicense - View Application

1. PERSONAL INFORMATION

Name (person) or company (legal entity)	Peter Wu
Fiscal No.	f-001
BIR/BI/Passport No.	124242
Home/Office Address of person/legal entity	Ocean Tops
Telephone	2452425
Fax	6373763
E-mail	peterwu@kerrigold.com

2. INFORMATION ABOUT AGENT / REPRESENTATIVE

Name (person) or company (legal entity)	Susan Lei
Fiscal No.	f22
BIR/BI/Passport No.	2365632
Home/Office Address of person/legal entity	Guia Hill Apartments
Telephone	525262
Fax	2252562
Other Contact	

3. TYPES OF BUSINESS

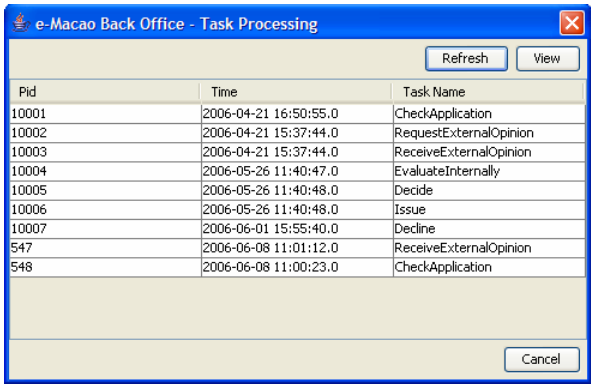
[✓ Beverage Establishment]

Tea House [✓ Coffee Shop] Ice Cream Parlour Other

Name	Coffe Galore!
Opening Hours	0230
Number of Seats	35
Number of Tables	23

Figure 13: Screenshot of FO Application

The MO Framework was parameterized by providing process definition in jPDL as well as XML files to describe the assignment of roles to tasks. Figure 14 shows a list of tasks generated by the MO application for different process instances in response to different licensing requests, all dispatched to BO for processing.



Pid	Time	Task Name
10001	2006-04-21 16:50:55.0	CheckApplication
10002	2006-04-21 15:37:44.0	RequestExternalOpinion
10003	2006-04-21 15:37:44.0	ReceiveExternalOpinion
10004	2006-05-26 11:40:47.0	EvaluateInternally
10005	2006-05-26 11:40:48.0	Decide
10006	2006-05-26 11:40:48.0	Issue
10007	2006-06-01 15:55:40.0	Decline
547	2006-06-08 11:01:12.0	ReceiveExternalOpinion
548	2006-06-08 11:00:23.0	CheckApplication

Figure 14: Screenshot of BO Application

By building application-specific aspects of e-Licensing systems, for instance Client and Web tiers for FO and BO applications, and implementing the hook methods in the FO and BO handlers, we obtain a concrete EPS.

6 Conclusions

We presented a Domain Framework for rapidly building EPS. Through an extensive survey of government operations and a study of 31 concrete EPS, we derived domain requirements and provided an architecture, design and implementation for a composite framework to support these requirements. The framework consists of three frameworks - Front-Office, Mid-Office and Back-Office.

While frameworks can radically reduce the amount of development effort [4], they may fail to deliver expected productivity gains due to their design and effect of other frameworks in the instantiation environment. In [2], a set of guidelines is provided for achieving flexibility, reusability and usability of frameworks. A number of them were implemented here: (1) avoiding implementation dependencies by relying on interface dependencies, (2) avoiding tight coupling of components and (3) relying on open standards. Domain specific frameworks can also suffer from composition problems during instantiation [2][3], arising from feature interaction between composed frameworks. We avoided this by: (1) building a composite framework with complementary frameworks to cover a complete EPS development and (2) accommodating auxiliary application frameworks at lower architecture layers to avoid direct interference. In order to avoid performance degradation through excessive use of dynamic binding, we kept a balance between white-box (extension, sub-classing and interface implementation) and black box (dynamic binding and configuration files) aspects of our Domain Framework.

The development of the Framework is ongoing, with additional instantiations planned to guide its modification and evolution to better support the EPS domain.

References

[1] Predonzani, P., Succi, G. and Vrenazza T., Strategic Software Production with Domain-Oriented Reuse, Artec House Inc., Norwood MA 02062, 2000.

[2] J. van Gurp and J. Bosch, Design, Implementation and Evolution of Object Oriented Frameworks: Concepts and Guidelines, Journal of Software-Practice and Experience, volume 31, no 3, pp. 277-300, 2001.

[3] M. Mattson, J. Bosch and M. Fayad, Framework Integration – Problems, Causes and Solutions, CACM, volume 42, no 10, pp. 81-87, 1999.

[4] S. Moser and O. Nierstrasz, Measuring the Effects of Object-Oriented Frameworks on the Software Process, IEEE Computer, September 1996, pp. 45-51.

[5] Y. Sanada and R. Adams, Representing Design Patterns and Frameworks in UML – Towards a Comprehensive Approach, Journal of Object technology, ETH Zurich, vol. 1., no. 2. July-August 2002.

[6] A. Ojo, C. Tang Ieong, G. Oteniya, T. Chi Pio and T. Janowski, Front-Office Framework for e-Government – Development Report, e-Macao Report 8, August 2006.

[7] E. Estevez, W. Chon, W. Chan Tang, A. Ojo, G. Oteniya and T. Janowski, Back Office Framework for e-Government – Development Report, e-Macao Report 9, August 2006.

[8] V. Peristeras and K. Tarabanis, Governance Enterprise Architecture (GEA): Domain Models for E-Governance, (Ed.) Marijn Janssen, ICEC'04, 6th International Conference on E-Commerce, ACM, 2004.

[9] T. Janowski, A. Ojo, E. Estevez, State of Electronic Government in Macao, Volume 2, e-Macao Report 2, April 2005.

[10] Capgemini, Online Availability of Public Services: How is Europe Progressing?, Web based Survey on Electronic Public Services Report of the 6th Measurement, June 2006.

[11] G. Oteniya, E. Estevez, D. Zhen Zhong, A. Ojo, T. Janowski, Electronic Delivery of Social Welfare Services – Development Document, e-Macao Report 4, October 2005.

[12] A. Ojo, G. Oteniya, C. Keng Fong, E. Estevez, T. Janowski, Electronic Delivery of Licensing Services – Development Document, e-Macao Report 5, October 2005.