



**United Nations
University**

UNU-IIST
**International Institute for
Software Technology**

Building a Dependable Messaging Infrastructure for Electronic Government

Elsa Estevez and Tomasz Janowski

April 2007

UNU-IIST and UNU-IIST Reports

UNU-IIST (United Nations University International Institute for Software Technology) is a Research and Training Centre of the United Nations University (UNU). It is based in Macao, and was founded in 1991. It started operations in July 1992. UNU-IIST is jointly funded by the government of Macao and the governments of the People's Republic of China and Portugal through a contribution to the UNU Endowment Fund. As well as providing two thirds of the endowment fund, the Macao authorities also supply UNU-IIST with its office premises and furniture and subsidise fellow accommodation.

The mission of UNU-IIST is to assist developing countries in the application and development of software technology.

UNU-IIST contributes through its programmatic activities:

1. Advanced development projects, in which software techniques supported by tools are applied,
2. Research projects, in which new techniques for software development are investigated,
3. Curriculum development projects, in which courses of software technology for universities in developing countries are developed,
4. University development projects, which complement the curriculum development projects by aiming to strengthen all aspects of computer science teaching in universities in developing countries,
5. Schools and Courses, which typically teach advanced software development techniques,
6. Events, in which conferences and workshops are organised or supported by UNU-IIST, and
7. Dissemination, in which UNU-IIST regularly distributes to developing countries information on international progress of software technology.

Fellows, who are young scientists and engineers from developing countries, are invited to actively participate in all these projects. By doing the projects they are trained.

At present, the technical focus of UNU-IIST is on formal methods for software development. UNU-IIST is an internationally recognised center in the area of formal methods. However, no software technique is universally applicable. We are prepared to choose complementary techniques for our projects, if necessary.

UNU-IIST produces a report series. Reports are either Research \mathcal{R} , Technical \mathcal{T} , Compendia \mathcal{C} or Administrative \mathcal{A} . They are records of UNU-IIST activities and research and development achievements. Many of the reports are also published in conference proceedings and journals.

Please write to UNU-IIST at P.O. Box 3058, Macao or visit UNU-IIST's home page: <http://www.iist.unu.edu>, if you would like to know more about UNU-IIST and its report series.

G. M. Reed, Director



**United Nations
University**

UNU-IIST
**International Institute for
Software Technology**

P. O. Box 3058
Macau

Building a Dependable Messaging Infrastructure for Electronic Government

Elsa Estevez and Tomasz Janowski

Abstract

The paper presents the development of a dependable messaging infrastructure for Electronic Government. Based on a few simple concepts like messages, members and channels, the infrastructure was developed to facilitate the exchange of messages by government agencies in a dependable and automated way. The dependability requirement was addressed on various levels: design, development and application. Considering design, the infrastructure comprises a small core offering plain messaging services, a repository of extensions to provide additional services, and a development framework to rigorously specify, implement and verify messaging extensions. Considering development, the infrastructure was built through rigorous use of modeling and analysis in various development stages. Considering applications, government agencies can use the infrastructure to exchange messages through carefully managed logical communication channels and the prudent use of necessary extensions, including extensions to implement required security measures. The paper presents the development and explains why the outcome satisfies the dependability requirement.

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.

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.

Contents

1	Introduction	1
2	Related Work	2
3	Government Collaborations	3
4	Dependability in Government	3
5	Messaging Infrastructure Development	4
5.1	Requirements Elicitation.....	4
5.2	Domain Modeling	7
5.3	Use Case Modeling	7
5.4	Architectural Design.....	8
5.5	Detailed Design.....	8
5.6	Implementation	9
5.7	Deployment	9
6	Infrastructure Supported Collaborations	10
7	Infrastructure Dependability Analysis	10
8	Conclusions	11
	References	11

1 Introduction

Electronic Government aims to provide a customer-focused, efficient and reliable access to public services. Among such services, seamless services provide the greatest benefits to customers – citizens or businesses, as they transcend the boundaries between functional areas and levels of the government. Offered through a one-stop portal and organized into clusters related to particular life events or business episodes, such services respond directly to typical customer needs.

However, the provision of seamless services is challenging for governments on many levels. On the organizational level, agencies must be able to integrate their processes to jointly deliver seamless services [1]. On technical level, coordination and communication between agencies must be ensured.

This paper focuses on technical challenges and provision of high-level communication infrastructure to facilitate the delivery of seamless public services. Although there are several well-established messaging frameworks and some solutions specifically designed for or applied to Electronic Government, none of them is natively supporting seamless e-Government services. Existing MOM frameworks generally provide a fixed messaging functionality and statically configured channels, and must resolve to external technologies to meet integration and interoperability requirements. As such, they are not able to directly address evolving needs of Seamless e-Government.

This paper presents the development of a messaging infrastructure to address such needs. The infrastructure comprises a core framework, a repository of horizontal (process-independent) and vertical (process-dependent) extensions, a mechanism to dynamically enable such extensions on top of the core framework, and a development method to rigorously specify, design and verify messaging extensions. The whole framework relies upon three simple concepts: members, channels and messages. A member is a registered user of the infrastructure, usually a government organizational unit. A channel is a medium for communication between authorized members. A message is a unit of communication, posted by members to particular channels. Members can subscribe to channels and send messages to other subscribers. Horizontal extensions include logging, validation and transformation, while vertical extensions include: process enforcement, policy monitoring and channel composition.

In addition to functional requirements that define a range of communication services offered through the infrastructure to software applications, dependability is another basic requirement. Governments are required to ensure the reliability, availability and security of seamless public services in order to conform to relevant legislations, fulfill public service quality pledges, avoid bad publicity and build trust with citizens.

The aim of the paper is to present the development of the messaging infrastructure and explain how it addresses the dependability requirements. The requirements were addressed at design, development and application levels as follows:

- 1) *Design* - The infrastructure was designed according to a rigorous process of specification, design and verification, supporting reliability and availability requirements. Various extensions can also be used to implement the required level of security, for instance through encoding of messages or authentication of members.
- 2) *Development* - The infrastructure was build following a rigorous process documented with suitable UML diagrams [3]. As a result, a reasonable level of reliability has been attained, especially in the core system. In addition to static verification of extensions, run-time checking of unverified extensions is considered.

- 3) *Applications* – The infrastructure makes it possible for agencies to exchange messages through logical channels and the prudent use of extensions. The reliability of message exchange is ensured by the underlying core framework and careful management of channels and message traffic by channel owners. Availability is ensured by the two-stage delivery of messages, from the sender to the owner and from the owner to subscribers. The required level of security is assured by the enforcement of policies and the use of security-oriented extensions.

The rest of this paper is organized as follows. Section 2 presents related work. Sections 3 and 4 describe example collaborations between agencies and analyze their dependability requirements respectively. Section 5 documents the development process of the infrastructure, from requirements, through domain and use cases modeling, to architecture and detailed design, to system implementation and deployment. Sections 6 and 7 explain how collaborations are supported by the infrastructure and analyze dependability requirements. Conclusions are presented in Section 8.

2 Related Work

Facilitating information and process integration, existing technical solutions for cross-agency coordination are essentially based on two approaches: Message-Oriented Middleware (MOM) [2] and Service-Oriented Architecture (SOA) [11].

In MOM, agencies asynchronously exchange data in the form messages. MOM enables applications to produce and consume messages using MOM-supplied APIs, and to transfer them through messages queues. MOM products include Java Message Service [4], Microsoft's MSMQ [5], Web Methods Enterprise [7] or IBM Web Sphere [6]. Several government projects also implement MOM, such as the Inter-Agency Messaging Service [9] for data related to birth and death notifications and the Hermes Messaging Gateway [10] for exchanging ebXML messages.

In SOA, the basic building block is a service - an abstract resource provided and consumed over a network, with ability to execute tasks [12]. SOA facilitates information integration, enabling the exchange of messages between applications running in different environments, and process integration through service orchestration. The EU-Publi.com project [14] and Microsoft's Connected Government Framework [13] implement SOA solutions.

The main limitation of existing MOM solutions is a fixed, or at least statically configured messaging functionality, such as routing, validation or encryption, which is not sufficient to address all communication needs of e-Government systems. Also, most MOM solutions support static communication structures to carry out messaging and thus cannot address the evolving communication needs of e-Government systems, e.g. following changes in law and outsourcing arrangements. Limitations of SOA approaches include the lack of service-level agreements and lack of support for dynamic configuration of processes.

Regarding dependability, MOM and SOA both facilitate security implementation through services such as user authentication or messages encryption. In SOA, security standards focusing on integrity, confidentiality and authentication are still under development. Applicable to both approaches, SAML [15] is an XML framework to exchange assertions about authentication and authorization. While other dependability attributes are also formulated by vendors, given the general lack of formal models enabling behavioral reasoning about composition of messaging services, such arguments are generally not supported scientifically.

3 Government Collaborations

Agencies must collaborate with each other in delivering public services. These collaborations take place through the exchange of various types of messages by agency staff using various software systems. This section describes example scenarios related to the service to issue business licenses, and defines some collaboration patterns on this basis [16].

Local governments are responsible for issuing various types of business licenses: for importing and selling goods; for establishing food and beverage activities; for advertising in the public places, etc. In most cases, before issuing the license, the agency requests other agencies to provide opinions on the application. The following example opinions may be required by a licensing agency when considering the application for opening a food and beverage business: i) opinion about the applicant's infrastructure - public works; ii) opinion about compliance with fire prevention - fire brigade; iii) opinion about compliance with health regulations - health department; iv) information about the number of personnel employed - labor department. In turn, obtaining these opinions may demand the arrangement of interviews and on-site inspections. All require close collaboration between agencies and their staff to complete the licensing process in a timely and effective way.

Some communication patterns identified based on this service include: (1) request information from an agency and receive a response, e.g. the request to the fire brigade; (2) request a technical opinion from an agency and receive a response where both request and reply may include additional documents, so messages may have attachments, such as a building plan in the message to the public works agency; (3) receive a notification message, such as the health department receiving a notification about appointments; (4) track requests sent to the agencies, when for instance a request was issued and the agency did not reply; (5) acknowledging the receipt of a message by another agency, such as the fire brigade acknowledging the reception of a request and enquiring on-site inspection.

4 Dependability in Government

We consider seven dependability requirements [18] in government collaborations: availability, reliability, security, timeliness, survivability, recoverability and maintainability. The requirements are explained below:

- 1) *Availability* – When an agency requires a service from another agency, the mechanism for communicating the request must be available at all times. For instance, when the e-License application receives a request from a government portal, it will immediately send messages requesting information from other agencies, assuming the responsible agency applications are available at all times.
- 2) *Reliability* – Each time a message is sent as part of larger agency collaboration, the message must be delivered reliably. The completeness and integrity of data transferred must be also assured. It is unacceptable to have the data modified or to lose some attached documents.

- 3) *Security* – Confidentiality, authentication and authorization are critical requirements to provide government services. For instance, when the e-License application receives a technical opinion apparently from the health agency, it must be sure that the information was indeed sent by this agency. When sensitive information about applicants is exchanged, confidentiality must be guaranteed. When e-License presents the license application for a public officer to make a decision, this must be indeed an officer who has the authority to decide, not just process applications.
- 4) *Timeliness* – Responses to collaboration requests must be provided on time. Each step of the business process must be executed during the pre-defined time-window in order to fulfill the defined performance measures. This particularly applies to the collaborations requested from other agencies, as the responsibility is shared among units.
- 5) *Survivability* – Collaborations must be provided, albeit with degraded performance, in the face of accidental malfunctions or deliberate attacks. Since seamless services are accessed through a one-stop government portal, the user perceives the service as provided by the whole government. He or she is not aware about the agencies involved and what type of problems they could be facing.
- 6) *Recoverability* – Following the same argument, the infrastructure supporting agency collaborations must be able to recover from errors. The process driving such collaborations cannot depend on a user not making mistakes or hardware not failing.
- 7) *Maintainability* – Collaborations between agencies are dynamic and highly exposed to change due to innovation in government regulations, policies and procedures. The infrastructure software supporting such collaborations must assure that including new features or repairing existing features can be done with reasonable resources.

5 Messaging Infrastructure Development

We present seven stages in the development of the messaging infrastructure: requirements elicitation, domain modeling, use case modeling, architecture design, detailed design, implementation, deployment.

5.1 Requirements Elicitation

Functional requirements are classified into five categories for managing: members, channels, messages, activity sessions and extensions.

TABLE 1 lists the requirements for managing members. A member is a registered user of the system, usually a government agency, with name and password. Members are divided into: a visitor member who is only allowed to send messages to the administrator (MEM01); a regular member allowed to send and receive messages through all channels it subscribes to (MEM02); and the administrator, a distinguished member who can register other members (MEM03). Three membership operations are provided: registration (MEM04), un-registration (MEM05), and password change (MEM06).

MEM01	Managing Visitor Member Type
MEM02	Managing User Member Type
MEM03	Managing Administrator Member Type
MEM04	Registering a New Member
MEM05	Un-Registering a Member
MEM06	Changing a Member Password

Table 1: Member-related requirements

The requirements for managing channels are shown in TABLE 2. A channel is a medium for communication between members. A channel accepts messages from authorized members and disseminates them to other members. Every channel has an owner. Channels may be simple - able to transfer messages in a single atomic operation, or complex - composed from existing channels. In addition, there are pre-defined channels – always available for administrative operations, such as creating a channel; and user-defined channels - opened, subscribed and used by members. Pre-defined channels include the administration channel (CHA01) to connect each member with the Administrator, and the visitor (CHA02) channel connecting Visitor with Administrator.

In order to create a channel (CHA03), a member sends a request to the Administrator. If successful, the Administrator assigns an identifier to the channel and the member becomes the Channel Owner. Later, the owner can also destroy the channel (CHA04), by sending the corresponding request to the Administrator.

The owner is authorized to subscribe (CHA05) and unsubscribe (CHA06) members to/from its channel. A subscriber has the right to read, write or use (both read and write) the channel. Initially, it can only read. Later, it can request the owner to upgrade (CHA08) or degrade (CHA09) its status, and enquire about its status (CHA07).

CHA01	Managing the Administration Channel
CHA02	Managing the Visitor Channel
CHA03	Creating a Channel
CHA04	Destroying a Channel
CHA05	Subscribing to a Channel
CHA06	Unsubscribing from a Channel
CHA07	Consulting Channel Subscription Status
CHA08	Upgrading Channel Subscription
CHA09	Degrading Channel Subscription

Table 2: Channel-related requirements

TABLE 3 lists the requirements for managing messages. A message is the unit of communication, consisting of a header, a body and several attachments (MES01). The header (MES02) identifies what channel the message is sent to, who is the recipient, etc. The body (MES03) contains the main content. Attachments (MES04) are the documents send with the message.

Any subscriber with the right to write or use the channel can send a message to it. If sent by the owner, the message is received by all subscribers (MES05). If sent by a subscriber, it is first forwarded to the owner (MES06) who then decides if to send it to all subscribers. Any channel subscriber with the right to read or use the channel can receive a message from it (MES07). In addition, a member must be able to send messages to other members using pre-defined channels (MES08).

MES01	Composing a Message
MES02	Obtaining Message Header
MES03	Obtaining Message Body
MES04	Obtaining Message Attachments
MES05	Sending a Message by Owner
MES06	Sending a Message by Subscriber
MES07	Receiving a Message
MES08	Forwarding Messages between Members

Table 3: Message-related requirements

TABLE 4 shows requirements for managing activity sessions. After registration, a member may use the system during activity sessions. Each session begins when the member logs-in (SES01) and ends when the member logs-out (SES02). The system can recover member information (SES03) between sessions.

SES01	Logging-in to a Session
SES02	Logging-out from a Session
SES03	Restarting a Member

Table 4: Session-related requirements

Besides enabling a simple exchange of messages, messaging applications require additional services which are provided through various extensions. TABLE 5 depicts the related requirements. The framework provides a repository of extensions (EXT01) and a mechanism to enable such extensions dynamically. Extensions can be: horizontal – applied to channels regardless of the process that drives message exchange or vertical – part of a concrete business process. Extensions can be enabled (EXT02), disabled (EXT04) and configured (EXT03). For instance, the validation extension requires an XML Schema to validate messages. Three horizontal extensions are provided as illustration: (1) *Auditing* (EXT05) - creating a historical record of messages transferred through a channel; (2) *Validation* (EXT06) - validating messages posted to a channel according to a given message format; and (3) *Transformation* (EXT07) - processing messages in transit from one format to another.

EXT01	Managing a Repository of Extensions
EXT02	Enabling Extensions
EXT03	Configuring Extensions
EXT04	Disabling Extensions
EXT05	Providing Channel Audit Extension
EXT06	Providing Validation Extension
EXT07	Providing Transformation Extension

Table 5: Extension-related requirements

Non-functional requirements referring to availability, reliability, security and scalability are listed in TABLE 6. The Administrator must always be available (AVA01). The framework must assure basic delivery properties such as: messages must be delivered at least once (REL01) and must be delivered in a sequence (REL02). Two security requirements are: authentication (SEC01) and authorization (SEC02). Authentication ensures that the access to data is restricted to those who can provide a proof of identity. Authorization decides whether an entity can access a particular resource. Both are assured by the earlier functional requirements. Scalability allows increasing total throughput under an increased load when new resources are added. For instance, a system is scalable if registering new members does not increase the time for sending messages (SCA01).

AVA01	Availability of the Administrator Member
REL01	Assuring At-Least-Once Delivery
REL02	Assuring Non-Permutation for Delivery
SEC01	Authentication of Members
SEC02	Authorization for using Channels
SCA01	Assuring Throughput Capacity

Table 6: Dependability requirements

5.2 Domain Modeling

The model in FIGURE 1 presents the key concepts of the messaging infrastructure. A message has a header, a body and zero or more attachments. A channel is owned by a member and is subscribed by members who wish to use it for sending and receiving messages. Channels may be user-defined or pre-defined. Three types of members are: User, Visitor and Administrator. Visitor and Administrator must exist at deploy time.

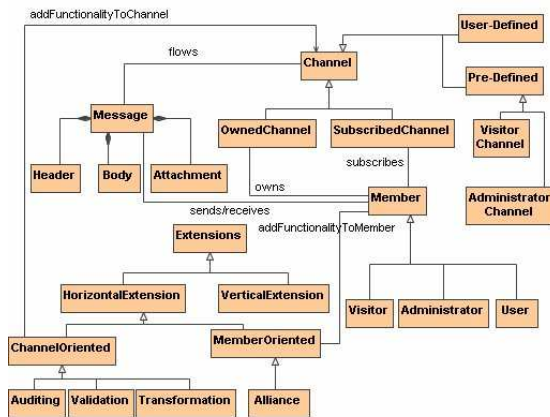


Figure 1: Conceptual Model

The infrastructure defines two kinds of extensions - horizontal and vertical. The former are classified into: channel-oriented – applied to messages flowing through a channel and member-oriented – applied to members. The latter combine various elements of the infrastructure to fulfill the communication and control requirements of particular business or administrative processes. As illustration, we defined four horizontal extension applied to channels - auditing, validation and transformation, and one to members - alliances.

5.3 Use Case Modeling

The top-level use case diagram is shown in FIGURE 2, depicting four main groups of services provided:

- 1) *Core Administrative Services* – required to configure the system. Fulfills MEM04, MEM05, MEM06, CHA03 to CHA09, and session-related requirements.
- 2) *Operational Services* – required by Administrator and Users for sending and receiving messages. Fulfills all messages-related requirements.
- 3) *Extended Administrative Services* – required by users in order to request extended functionality. Fulfills all requirements related to extensions.
- 4) *Configuration Services* – provided for configuring the system and creating communication structures. Fulfills MEM01, MEM02, MEM03, CHA01 and CHA02.

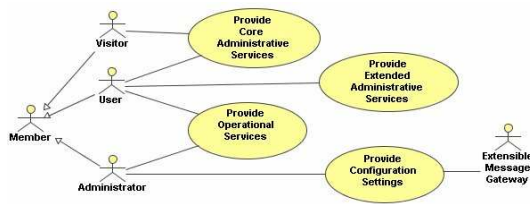


Figure 2: Top-Level Use Case Diagram

5.4 Architectural Design

The structural view is shown in Figure 3, showing a completely distributed architecture. Applications requesting services use a Messaging Component with two-layered architecture. The first includes components implementing the business logic: Core – operational and core administrative services; Extensions – extended administrative services; and XMLSchemas – the structure of messages. The second contains Database – managing data persistence with local databases. In the figure, ExternalApplication is any application using the infrastructure. Administrator has a component implementing the Administrator member, who is like any member and so uses the Messaging Component.

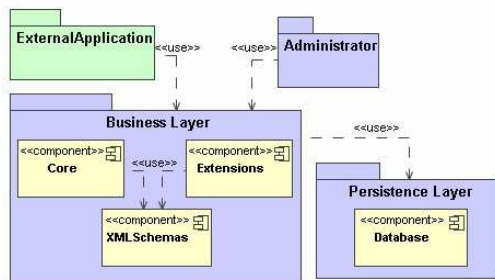


Figure 3: Architecture – Static View

5.5 Detailed Design

A design diagram is presented in Figure 4. The Core component offers the interfaces I-Member and I-Visitor which are used by external applications, and ApplicationListener which is implemented by external applications. I-Visitor offers services for registering and retrieving members. I-Member offers other member services. ApplicationListener allows applications to receive messages.

Core also contains classes for the channels owned and subscribed by members – ChannelOwned and ChannelSubscribed. Both are sub-classes of Channel and are related to the class Member that keeps the relationships with its channels. Message class enables composing and decomposing messages. ServerSocket and ClientSocket enable to send/receive messages.

Extensions component contains classes for managing extensions. The ChannelOriented class provides functions to enable, configure and disable channel-oriented extensions. It also defines abstract operations processMessage and configureExtension for extensions to implement. Auditing, Validation and Transformation extensions were implemented.

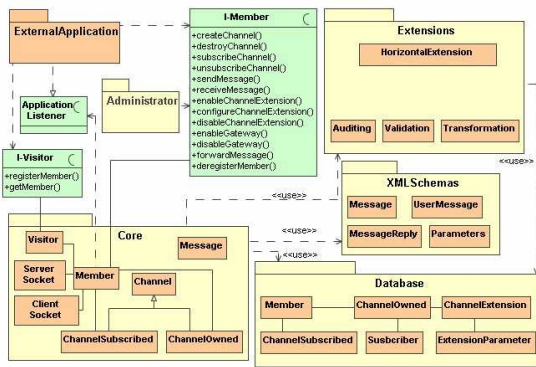


Figure 4: Detailed Design Class Diagram

5.6 Implementation

The infrastructure is implemented in Java using open-source technologies. The database is MySQL and Hibernate maintains the object-relational mapping. Messages are written using XML with XMLBeans. Validation relies on XML Schema and transformation on XSLT. The system consists of one package implementing: Core, Extensions, XMLSchemas and Database components. One copy of this package must be installed in every computer hosting an application that requires communication services.

5.7 Deployment

Figure 5 shows a possible deployment of the system. One node hosts the administrator - AdminNode and various agency servers host government applications communicating through the infrastructure. On each server a Messaging component is deployed. For instance, AgencyA server hosts the e-Welfare service, AgencyB hosts the e-Tax application and AgencyD hosts the government portal. In AgencyD, three e-services rely on the system: e-License, e-Health and e-Driving.

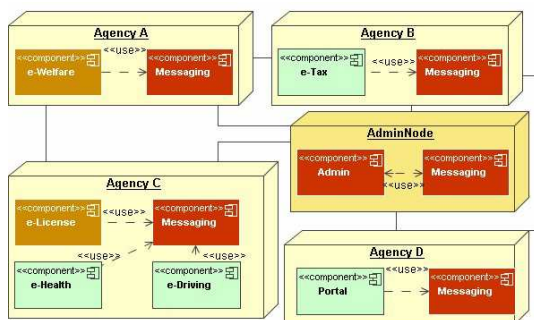


Figure 5: Deployment Diagram

6 Infrastructure Supported Collaborations

Consider how the infrastructure supports the cross-agency collaboration presented in Section 3.

Suppose a company applies for a business license through e-License application. The process involves: company I maintaining the portal; municipality M issuing licenses and hosting e-License; public works agency W checking the technical infrastructure; fire brigade F checking compliance with fire regulations; health agency H checking compliance with health regulations; and labor agency L for labor control. The business process is: the applicant submits the request through the portal; I forwards the request to M; M sends requests for opinions to W, F, H, and L; M collects the opinions and decides whether to issue the license.

The implementation connects the portal with e-License through the channel c1, and e-License with applications at W, F, H, and L through the channels c2, c3, c4, and c5. The extension *Process-Enforcement* is enabled for controlling the sequence of messages, and is configured with the pattern $(I.M^5.(W|F|H|L))^*$ - I sends the application request; M forwards four requests for opinions; the four contacted agencies respond in any order; and the whole sequence is repeated for each request. *Channel-Composition* is also applied to split the messages received from c1 to other five channels.

Regarding the example communication patterns in Section 3, we conclude that the infrastructure supports all of them: (1) requesting information - a message sent through a channel; (2) requesting technical opinions - sending a messages with supporting documents as attachments; (3) notifications - a message is received from a channel; (4) tracking requests - enable a *Tracking* extension to keep historical records of the messages sent through the channel; (5) sending acknowledgements - sending a reply message.

7 Infrastructure Dependability Analysis

This section performs informal analysis of the seven dependability requirements described in Section 4:

- 1) *Availability* - This is achieved by assuring the availability of members to external applications. Members at the sender side are always available, since an application can always restart them. Each time a member is restarted (SES03), it contacts the Administrator who is always available (AVA01). As members at the recipient side may not be available, the extension *OneDelivery* can be used to control messages sent to subscribers (REL01). While trying to send a message to an unavailable subscriber, the extension is aware about this and stores the message for a later delivery.
- 2) *Reliability* - messages cannot be lost (REL01) and are delivered in sequence (REL02). Message completeness can be assured through *Validation* extension by controlling mandatory elements based on XML Schema definition. Message alteration can be detected by attaching signatures through the *Signing* extension. In addition, a rigorous approach applied to build the infrastructure; a design based on a simple core and extensions; the ability to specify, implement and verify new extensions, all contribute to fulfilling the reliability requirement.
- 3) *Security* - User authentication is assured since members must provide names and passwords to initiate sessions (MEM02). Authorization for requesting messaging services is implemented through different subscription levels (CHA07, CHA08, CHA09). Extensions are enabled, configured and disabled by their owners only (EXT02, EXT03 EXT04). Confidentiality can be introduced attaching *Encryption* and *Decryption* extensions to channels.

- 4) *Timeliness* – The sending of a message is instantaneous, the timing of the receipt depends on the availability of the recipient in the system. The delay can be monitored and controlled through *Tracking* and *Scheduling* extensions. The former enables tracking the progress of messages through complex channels. The latter enables assigning priorities to messages to determine the order in which messages are dispatched by owners.
- 5) *Survivability* and *Recoverability* – The key to satisfying both is the guarantees that: the Administrator is always present in the system (AVA01), all members can always be restarted (Ses03), and messages are never lost (REL01). In addition, the design of the infrastructure into a minimum but fully functional core and a set of pluggable extensions directly support both requirements. These properties ensure that a minimum level of functionality is always available.
- 6) *Maintainability* – This is assured on two levels: application - the system supports maintainability since extended functionality can be enabled and disabled at run-time without the need to modify existing applications; infrastructure - due to the rigorous development process, modular design and complete documentation of the development with models, the system is relatively easily to maintain.

8 Conclusions

We presented a messaging infrastructure for e-Government to enable dependable exchange of messages between agencies and support collaborative delivery of seamless services. Our motivation was to provide a technical solution for seamless e-Government, with special focus on dependability due to its importance for e-Government - the delivery of public services must be reliable, secure, etc.

Following related work, government collaborations to support the delivery of seamless services were illustrated, and it was explained how dependability issues are relevant to them. The development of the infrastructure was presented in detail, including some development artifacts. The paper also argued in what ways the implementation supports seamless services and how it addresses dependability requirements.

The main ideas of the messaging infrastructure were first introduced in [17] and the infrastructure was developed as part of the e-Macao Project [16]. The main contribution of this paper is the presentation how dependability requirements are relevant for messaging support for seamless e-government, how the messaging infrastructure was developed, and how the development and its outcome address such requirements.

Future work includes building a formal model for the infrastructure and building and validating a theory for reasoning about behavioral properties of extensions.

References

- [1] T.Field, E. Muller and E. Lau, “The e-Government Imperative”, Organisation for Economic Cooperation and Development <http://www1.oecd.org/publications/e-book/4203071E.PDF>.
- [2] Software Engineering Institute (SEI), “Message-Oriented Middleware”, Carnegie Mellon University, www.sei.cmu.edu/str/descriptions/momt_body.html.

- [3] J. Arlow and I. Neustadt, "UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design" 2nd ed. Addison-Wesley, 2005.
- [4] Sun, "Java Message Service" java.sun.com/products/jms.
- [5] A. Dickman, "Designing Applications with MSMQ", Addison Wesley, 1998.
- [6] IBM, "WebSphere", www.ibm.com/software/websphere.
- [7] WebMethods Enterprise, "WebMethods", www.webmethods.com/meta/default/folder/0000005452.
- [8] J. Sebek, "Delivering E-Government Services to Citizens and Businesses", 2nd Int. Conference EGOV 2003, Springer, pp. 125-128, September 2003.
- [9] P. Collins, "Implementing a Messaging Infrastructure for Intra-Governmental Cooperation", 3rd European Conference on E-Government, pp. 69-80, July 2003.
- [10] Center for E-Commerce Infrastructure Development, The University of Hong Kong, "Hermes Messaging Gateway", <http://www.cecid.hku.hk/hermes.php>.
- [11] World Wide Web Consortium, "Web Services Architecture", www.w3.org/TR/ws-arch/#whatis, 2004.
- [12] World Wide Web Consortium, "Web Services Glossary" <http://www.w3.org/TR/ws-gloss/>, 2004.
- [13] Microsoft, "Connected Government Framework", www.microsoft.com/interop/govt/cgf/default.aspx.
- [14] E. Tambouris and K. Tarabanis, "e-Government and Interoperability", 5th European Conference on E-Government, pp. 399-407, June 2005.
- [15] OASIS, "OASIS Security Services (SAML)" http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.
- [16] E. Estevez and T. Janowski, "Extensible Message Gateway for e-Government - Development Document", e-Macao Project Report 11, 2006, <http://www.emacao.gov.mo>.
- [17] E. Estevez and T. Janowski, "Government-Enterprise Ecosystem Gateway for Seamless e-Government", Proceedings of the 41st Hawaii International Conference on System Sciences, January 2007.
- [18] I. Sommerville, et.al., "Dependability and Trust in Organisational and Domestic Computer Systems", in Trust in Technology: A Socio-technical Perspective, Springer London, 2006, pp.169-193.