NOTE TO USERS

This reproduction is the best copy available.

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Grant Middleton

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (E-Business Technologies)

GRADE / DEGREE

Department of E-Business Technologies

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Framework for Continuous Compliance Monitoring of B2B Processes

TITRE DE LA THÈSE / TITLE OF THESIS

Dr. Liam Peyton

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Dr. Morad Benyoucef

Dr. Craig Kuziemsky

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

A Framework for Continuous Compliance Monitoring of B2B Processes

Grant Middleton

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the M. Sc. degree in E-business Technologies

University of Ottawa

Ottawa, Ontario, Canada

August 2009

Your file Votre référence
ISBN: 978-0-494-61184-5
Our file Notre référence
ISBN: 978-0-494-61184-5

Abstract

Government regulations and legislation, organizational policies, and customer contracts all stipulate rules that an organization and its internal entities must adhere to. Compliance with said rules is often mandatory, and if compliance is not met, may lead to negative consequences. Continuous compliance monitoring involves collecting data on a continuous basis in order to track the relevant business processes to ensure compliance with government regulations and legislation, organizational polices and customer contacts. This thesis presents a framework for continuous compliance monitoring of business to business (B2B) processes in the context of a publish/subscribe architecture for event-driven business process integration. The framework integrates a streaming event data model with an agent-based surveillance portal to provide continuous support for dynamic exception alerts and performance management reporting. The government regulations and legislation, organizational policies and contracts are represented within the framework by event-condition-action (ECA) policies that can be monitored in terms of the event data collected by the surveillance portal. An eHealth scenario in which organizations collaborate to provide regulated at home care is used to illustrate our approach.

Acknowledgements

First and foremost I would like to thank my supervisor, Liam Peyton. Without his help and support this work would not have been possible. His guidance and encouragement throughout my research was invaluable.

I would also like to thank Ben Eze. Ben and I collaborated on some early work and his help and support greatly improved the quality of my work. I would also like to thank Clare McElcheran for supporting me and allowing me the use of her equipment. I also thank the entire Palliative Care research team for their helpful comments and contributions.

This work was supported by a research grant from the National Sciences and Engineering Research Council of Canada (NSERC) and by the University of Ottawa through an entrance scholarship.

Finally, I would like to thank my family for their support and encouragement throughout the writing of this thesis.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
B2B	Business-to-Business
BAM	Business Activity Monitoring
BPM	Business Process Management
CoT	Circle of Trust
ECA	Event Condition Action
EDA	Event-driven Architecture
EHR	Electronic Health Records
EPSI	Event-driven Publish/Subscribe Interface
ETL	Extract Translate Load
FIPA	Foundation for Intelligent Physical Agents
JADE	Java Agent Development Architecture
LHIN	Local Health Integration Network
PSC	Partner Service Component
QoS	Quality of Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
WSDL	Web Service Description Language

XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

Chapter 1. Introduction

1.1. Problem Statement

Government regulations and legislation, organizational policies, and customer contracts all stipulate rules that an organization and its internal entities must adhere to. Compliance with said rules is often mandatory, and if compliance is not met, may lead to negative consequences. Additionally, the number of rules in place is constantly growing as our society evolves and grows. With such a large number of rules in place, compliance management becomes a serious concern for any organization.

Compliance management involves a number of different tasks. Organizations need to discover which rules apply to them. They need to develop processes and data collection methods so that they can verify that the rules are being followed. They also need to develop processes and mechanisms for taking proactive action and notifying when rules are broken so corrective action can be taken. Finally, they need to provide support for external auditing and review. Each one of these tasks presents a particular challenge to an organization. In particular this thesis is focused on the processes and data collection methods being used to verify that the rules are being followed.

With the increased desire for business process automation, and with recent enhanced technology, businesses are bringing many of their business processes online. In addition, the number of business to business (B2B) interactions, in general, is growing as the world economy is increasingly globalized [Lee03]. These two growing trends lead to an increasing number of online-B2B processes. This increase can be seen in many

different areas. Supply chain management [Lancioni00], financial transactions [Lucking-Reilly01], and health care [Anyanwu03] are all examples of areas where the number of online B2B processes is growing. Within health care a specific example is community care, especially at home care, which frequently requires the cooperation and integration of care processes across several providers and organizations in a B2B network. Business Process Management (BPM) provides support for business processes and can help improve the effectiveness of the online-B2B processes.

Approaches to support online-B2B processes include emerging technologies like Service Oriented Architecture (SOA). However, SOA based infrastructures are not ideal. They only provide the capability of pulling or requesting data from the data source. Data cannot be pushed from the source to interested parties when it becomes available. SOA infrastructures have shown a lack of B2B compliance management abilities. A mechanism for defining a common data model across a B2B network, which could be tied to an approach to collecting data for compliance monitoring, and provide a policy decision point for ensuring compliance is needed.

Approaches to compliance monitoring include manual and web portal based techniques. Manual approaches involve manually examining physical documents in order to determine if compliance is being met. Web Portal based techniques build upon this method by capturing the data from the physical documents into a data warehouse and presenting the data to the user in a web browser. Compliance monitoring is still a manual task. Automating compliance monitoring would reduce the number of errors involved

with compliance monitoring and it would reduce the number of human work hours, thus saving time and money.

These approaches also lack the ability to continuously monitor for compliance. Currently compliance monitoring results are only made available to interested parties a significant amount of time after the occurrence of the activities involved. The data is often not made available immediately to interested parties and it is not possible for a human to analyze the data in a continuous fashion. If the data could be made available and analyzed continuously then compliance monitor results could be made available on a continuous basis. This would allow organizational members to respond to any negative outcomes by taking corrective action and thus improve the organization's compliance results. Emerging publish/subscribe technology offers data integration techniques which could support continuous compliance monitoring.

Informing organization members of a compliance failure is often done a significant amount of time after the failure occurred. In the time between the failure and the notification more compliance failures may have occurred. In order to improve compliance results, an organization's member should be alerted to compliance failures when they occur, or even alerted that a potential failure may occur. This would help reduce the number of compliance failures across the organization.

1.2. Thesis Motivation and Contributions

Current compliance monitoring practice is manual and inefficient. Tools such as web portals and data warehouses help support the manual effort, but they do not eliminate it. With increasingly large online B2B networks, we need a way to provide continuous

and automated compliance monitoring. We need standardized data publishing and collection across a B2B network, assembled into a database for performance management and audit review, but also streamed in real-time for a policy-based compliance monitor to raise alerts.

The contributions of this thesis are:

1. A set of key criteria that a solution for automated continuous compliance monitoring of B2B processes should support.
2. A framework for continuous compliance monitoring in which government regulations and legislation, organizational policies, and contracts are systematically expressed as declarative policies that can be monitored based on a streamed audit trail of published event messages within a B2B network
3. An architecture that supports the framework which includes a publish/subscribe message broker, a common data model and event registry, and a surveillance portal that provides both performance management reporting, and policy-driven compliance monitoring and alerting.
4. An agent-based architecture for policy-based compliance monitoring and alerting, including a survey and comparison of rule and agent based technologies for implementing the architecture.

1.3. Thesis Methodology and Organization

The methodology we employed during our work was problem formulation and gap analysis of existing approaches based on a literature review, followed by iterative development, gap analysis and evaluation of our proposed solution to the problem. In particular we took the following steps:

1. Identify and analyze problem (Compliance monitoring)
2. Review literature and industry approaches to problem
3. Identify the strengths and shortcomings of these approaches
4. Define objectives
5. Develop compliance monitor framework to address shortcomings and meet objectives
6. Perform case study (palliative care) to evaluate and compare the proposed framework to more traditional web portal based approaches.
7. Improve compliance monitor framework with agent-based approach based on review and re-evaluate
8. Draw conclusions and identify future work

The thesis is organized as follows:

In Chapter 2, we set the context for the problem we are trying to solve by giving background information on business process management, compliance monitoring, policies, events and B2B business processes. We then provide some background on event-driven and agent architectures. Finally, we discuss related healthcare systems.

In Chapter 3, we provide a big picture discussion to set the stage for our proposed framework. We identify a set of criteria that a continuous compliance monitoring framework should support. We then describe our approach: continuous compliance monitor framework, architecture to support the continuous compliance monitor framework, and a policy-based compliance monitor architecture.

In Chapter 4, we introduce our case study drawn from palliative care and focus in on a particular palliative care scenario. We describe the implementation and results of using both a "strawman" web portal architecture and our continuous compliance monitoring framework.

In Chapter 5, we evaluate our approach in comparison with the "strawman" web portal approach based on the set of criteria we identified in chapter 3. Additionally we survey and compare several candidate technologies that could have been employed when implementing the policy-based compliance monitor architecture.

Finally, in Chapter 6 we summarize our contributions and discuss possible future extensions that could build on this thesis.

Chapter 2. Background

2.1. Business Process Management

Most of the rules that an organization must comply with are stated in terms of the business processes that define the business. The rules stipulate: what steps need to be taken by the business, who can do what and when they need to do it, how quickly the business processes should be completed, and what results need to be produced by the business processes.

A business process can be defined as "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer" [Hammer94]. Business processes can range from simple processes contained within a single component of the business, to complex processes that cut across multiple organizations.

BPM is defined as "supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information" [Aalst03].

Increasingly as business processes are automated or take place on-line, data collected from those processes is used to monitor them. This is called Business Activity Monitoring (BAM). BAM is defined by the Gartner Group as "the real-time reporting, analysis and alerting of significant business events, accomplished by gathering data, key performance indicators and business events from multiple applications" [Dresner03]. BAM transforms and merges data from multiple sources, allows for the creation of

business calculations, and supports an intuitive user interface [Dresner02]. The data collected in BAM can be acted on immediately, routed to individuals for action, or it can be collected and stored. Typically, data is collected that characterizes or measures events associated with different stages or steps of the business process. One of the main goals of BAM is to reduce the response time to business events.

Business intelligence techniques can be used to analyze and interpret the data collected from BAM. Business Performance Management "can be described as a series of business processes and applications designed to optimize both the development and the execution of business strategy" [Frolick06]. Performance Management is used to tie evaluation of the success and compliance of business processes directly to the data collected about them, in terms of metrics. Metrics provide measurements of business processes which are relevant to the goals for those processes. These methods allow organizations to monitor the metrics that have been captured by information systems in order to measure performance and to diagnose the operational processes [Weske04]. The main goal of Business Performance Management is to improve the performance of the business operations.

Compliance is a form of business performance management. It is focused on ensuring business processes conform to the rules and constraints imposed on them. It is not just about achieving the business objectives of the organization.

2.2. Compliance

Organizations are constrained or obligated by government regulations and legislation, organizations guidelines, and contracts with customers. Management

initiatives such as the Balanced ScoreCard [Kaplan93] and Six Sigma [Breyfogle99] are examples of data driven performance management. They provide organizations with measurements and goals in order to improve business processes. Legislation, such as Sarbanes Oxley [Sarbox04], have imposed data driven performance management frameworks upon organizations in order to verify that compliance with financial regulations is being met. Contracts such as a Service Level Agreements (SLA) dictate the level of service between an organization (service provider) and its customer (service consumer) in terms of measurable quantities. A distinction should be made between the different levels of compliance. Compliance can be legally mandatory, such as Sarbanes Oxley, or it can be a guideline that does not legally require compliance, such as Six Sigma. These legislation, organizational guidelines and contracts dictate the expected behavior of the organization, as well as the entities within the organization in terms of measurable results. The possibility that an entity within the organization may deviate from its agreed upon behavior means that we need a mechanism for monitoring an entity's actual behavior [Milosevic02]. Data is collected to provide quantifiable measures of the behavior in order to verify the SLA.

Processes must be systematically monitored for compliance with legislation, organizational guidelines and contracts. A B2B interaction, such as community at home care, frequently requires the cooperation and integration of processes across several providers and organizations in a B2B network. In this context, the flow of information between parties in B2B networks is crucial for both effective and efficient collaboration as well as for verifying compliance. Continuous compliance monitoring involves

collecting data on a continuous basis in order to track and manage the relevant business processes to ensure compliance.

The compliance requirements of an organization will directly affect the execution and performance of the organization's business processes. For this reason compliance monitoring becomes a business process management activity. Compliance monitoring is concerned with how our business processes are performing with respect to the legislation, organization guidelines and contracts that govern an organization.

In addition to simply monitoring for compliance, action can be taken to help improve compliance performance. A policy notification mechanism allows a party to be warned of possible non-compliance, or to notify a party of an action that needs to be taken [Milosevic02]. We can use alert notifications to improve the compliance of our business processes with respect to legislation, organizational guidelines and contracts.

Frameworks for measuring compliance based on goal models and data collected are proposed in [Ghanavati07] and [Pourshahid08]. In model-based verification, compliance monitoring is defined and validated at design time. Models of the business processes and the relevant legislations, organizational policies, contracts, and metrics are created. The advantage of this approach is that strategies and mechanisms for compliance are well thought out and documented in advance. The disadvantage is that they do not articulate a systematic approach for verifying compliance once the processes are deployed. Typically, it is assumed that the organization has a data warehouse approach where relevant data is collected that is used for general performance management within the organization. The problem is to ensure that sufficient data is collected operationally

to validate that the compliance defined by the models is actually taking place as envisioned. In practice, leveraging existing data warehouses means that compliance is tracked days, weeks and even months after the fact. The compliance that can be validated is partial and inconsistent at best.

2.3. SLA Monitoring

One example of a type of contract that an organization must follow is the SLA. A SLA is a formal definition of the relationship that exists between a service provider and a consumer of that service [Verma99]. It is a contract between the service provider and the consumer that stipulates how the service will be provided and the Quality of Service (QoS). An SLA can be used for any type of service, in any industry, but including IP networks [Verma99] and specifically Web Services [Jin02]. Typically an SLA will contain the following information: the purpose of the SLA, the parties involved, the validity period, scope, restrictions, service-level objectives, penalties, optional services, exclusions, and administration [Jin02].

Compliance with the SLAs committed to by an organization is very important because of the ramifications associated with a failure to comply. Often non-compliance can lead to monetary losses by one or both of the participating parties. In order to determine compliance with an SLA, there needs to be a monitoring mechanism in place. The SLA itself will usually stipulate the mechanism for monitoring the service with regards to the SLA.

The main components of a SLA that need to be monitored are the QoS guarantees. These guarantees usually focus on such parameters as availability, throughput,

bandwidth, etc. Since a SLA is written in natural language there can be confusion about the meaning of such parameters. Approaches to solve this problem include SLA Templates [Rodosek01], and the WSLA framework [Keller03]. With a common vocabulary in place, monitoring for compliance becomes a task of monitoring network traffic to determine if the QoS guarantees are met.

2.4. Events

An event is defined as a state transition [Li05]. For example, the change of state of a light switch from the off position to the on position would constitute an event. In order to utilize event occurrences within software, an event must be represented in some format. Typically, an event is realized through the use of messages. A single message represents an event occurrence. This allows other entities to become aware of the event occurrence. Often these messages are distinguishable by an event type indicator and provide a set of attribute value pairs that represent the data associated with the event. To continue the example, when the light switch was changed from the off position to the on position, a sensor would capture this information in a message with event type 'Light Switch' and a set of attributes indicating the light switch involved, and the change of state that occurred. This message would then be made available to other interested parties in a structured format (e.g. eXtensible Markup Language (XML)). The source of an event, the light switch in this case, is known as an event producer, and the sinks that consume the events are known as the event consumers.

2.4.1 Composite Events

The occurrence of one event may mean little, but when coupled with other events, it may indicate something of importance. A composite event is a pattern of event occurrences [Li05]. A certain sequence of events, possibly within a certain timeframe, corresponds to a specific composite event. Similar to a single event, a composite event would typically be represented in a message format.

In order to observe composite events, a monitoring entity needs to be put in place. This monitoring entity would be given the definition of the composite events and would require access to all event messages. Upon detection of a composite event, the monitoring entity would produce a composite event message. The monitoring entity is an event producer and an event consumer.

2.5. Policies

A policy is a declarative specification of guidelines, rules of conduct, organization, and behaviour of entities in a given environment [Ross-Talbot04]. It reduces the complications involved in managing controls by defining strictly regulated access to resources.

Policies can be used by an organization to represent the government legislation, organization guidelines and contracts it must follow. A strictly defined policy language is needed in order to define the guidelines in a consistent and manageable fashion. In addition, the policy language should be machine readable in order to facilitate any policy automation an organization requires. For instance, continuous compliance monitoring

would require a machine readable policy language in order for the compliance monitor to understand the policies that it is monitoring.

2.5.1 Event-Condition-Action Policies

Most of the material found in this section can be found in [Eze09] [Middleton09], we summarize here for use in this thesis.

An Event Condition Action (ECA) policy pattern is used to represent industry regulations, internal business processes and policies by the majority of businesses [Boglaev05]. An obligation refers to the actions that an entity must or must not take when presented with the occurrence of a specific event [Shankar05]. Obligatory policies are often represented in the form of ECA rules.

ECA rules automatically perform a set of actions in response to events provided that certain stated conditions are met [Bailey02]. They take the form:

on [Event]

if [Condition]

then [Actions]

[Event] defines a state transition that causes a change in the active system being monitored. [Condition] defines comparisons of the event properties or attributes to a certain set constants. [Actions] defines those activities that must take place as a result of a successful or unsuccessful test of these conditions.

2.6. B2B Business Processes

There are a number of technologies and methods in place in order to support online B2B business processes. Here we discuss a few of them.

2.6.1 Service Oriented Architecture

SOA is a software architecture where software and support infrastructure are connected as a set of interoperable services that are accessible through standardized interfaces and messaging protocols [Papazoglou03]. This approach allows organizations to mix and match the services in order to support business processes with less effort than a traditional point-to-point approach. Each service can be considered as a business function that can be combined with other services in order to provide a larger business function.

SOA defines the interaction between software as a set of messages between a service provider and a service consumer. Service providers publish a description of the service(s) (including the service interface) that they offer to a service registry. Service consumers search for services by searching the service registry. The service consumer, having found a service, will then bind with the service interface in order to interact with the service. The interaction is in a pull format where the service consumer will make requests to the service provider and receive responses from the service provider. This interaction can be seen in Figure 1.

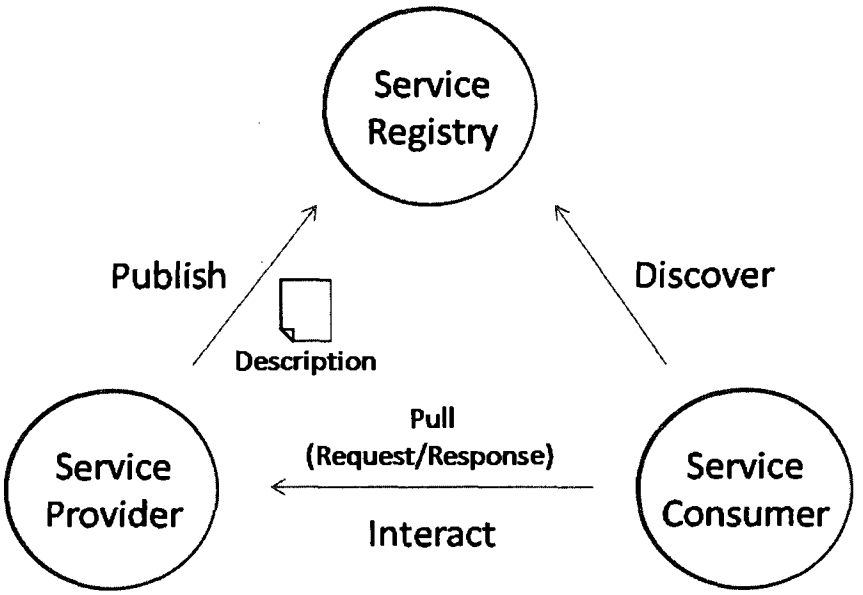

Figure 1 – Service Oriented Architecture

The Web Services architecture is one of the main approaches to realizing SOA. Web Services are based on three W3C standards: Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI). SOAP is the communication protocol, WSDL is the service description language and UDDI is the service registry.

2.6.2 Federated Identity Management

As the number of service consumers grows there becomes a need for a consumer profile management system. Identity management, a set of technologies and processes used to create, maintain and terminate user identities, has become a potential solution [Ahn05]. Such a solution is considered for a single organization. When services and business interactions start to cut across organizational boundaries the use of federated identity management comes into play.

The Liberty Alliance project is a consortium of IT vendors who joined together to create a standard and set of specifications for federated identity management. The key component of the project is the Liberty Alliance Circle of Trust (CoT) [Landau03]. The CoT is a B2B network where services are provided to users within a federated identity management framework. All identities and personal information within the CoT are protected by a designated Identity Provider. Participating organizations are still able to access and share data given that the individual's permission is obtained.

2.6.3 Audit Trail

In audit trail monitoring, each component or service within a B2B network is responsible for generating its own audit trail or log file that records what it is doing as an audit trail of events. As well, each process or application can log its events. A particular approach is where each service will log any events to an auditing service [Peyton07a]. The auditing service then stores all events in a database. The database is made available for querying and reporting, which could be used to document and monitor compliance. Audit trails are also useful for solving complaint requests. In the case of a complaint or issue the audit trail can be inspected to see what happened.

It is possible to collect the information from the log files into a data warehouse and analyze them for a traditional performance management view [Peyton08]. However, since continuous monitoring is not possible and the audit trail is often unstructured, extracting data can be challenging. Additionally, there is no standardized data model or correlation between the different audit trails. Distributed logging and auditing has not

been widely adopted because of privacy concerns [Shen04]. Revealing the system logs could lead to business loss or litigation [Shen04].

2.7. Event-driven Architecture

Event-Driven Architecture (EDA) represents a system architecture pattern whereby the behavior of the system is characterized by the production, notification and consumption of events. Service providers produce events and service consumers consume events. Distribution of event notifications in an EDA is in the form of messages. This means that service providers and service consumers are not coupled by API calls but rather by the data itself [Niblett05].

An essential component of the EDA is the event notification service [Carzaniga01]. This service supports an EDA by providing an infrastructure for disseminating event related data. The service accepts notifications from service providers and will forward the notifications to service consumers.

2.7.1 Publish/Subscribe

The publish/subscribe interaction paradigm "provides subscribers with the ability to express their interest in an event or a pattern of events, in order to be notified subsequently of any event, generated by a publisher, that matches their registered interest" [Eugster03]. Publishers are event producers and subscribers are event consumers. In between the publishers and subscribers is an event notification service, namely a Message Broker, that facilitates the publish/subscribe behavior by accepting

event notifications (messages) and notifying subscribers of events of interest. The publish/subscribe interaction can be seen in figure 1.

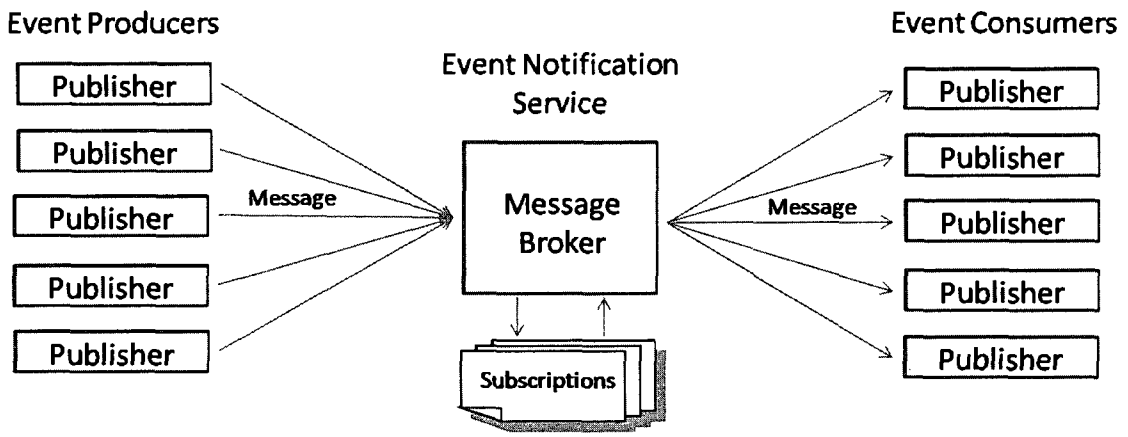

Figure 2 – Publish/Subscribe Paradigm

Event subscriptions can take a number of different forms: topic-based, content-based, type-based or a mixture of topic, content and type-based. The topic-based approach is where publishers publish messages with a specific topic, and subscribers subscribe to specific topics. Content-based is where subscribers subscribe to the content of the event message. Type-based is where messages are filtered into groups based on their type. Subscribers subscribe to specific types and not to the content or the topic of a message. Finally, there are approaches whereby both topics, and content, among other things, are considered when publishing and subscribing to events. For instance, the policy-based publish/subscribe framework proposed by [Eze09] uses policies to dictate event subscription and notification.

This paradigm offers a number of benefits: Space, Time and Synchronization decoupling. This means that publishers and subscribers do not need to know each other,

they do not need to be participating at the same time as each other, and publishers and subscribers are not blocked when performing an action.

2.8. Agent Architecture

The term agent can take many different meanings, here we will use the following definition: "a self-contained, concurrently executing software process, that encapsulates some state and is able to communicate with other agents via message passing" [Wooldridge95]. The architecture of an agent specifies "the construction of a set of component modules and how these modules should be made to interact" [Maes91]. The modules of the agent stipulate how the input data will determine the output of the agent, as well as the internal state of the agent [Maes91].

At the most basic level an agent will perform two types of duties: gather information about its environment and react to that information. The agent may retrieve data by way of sensors or by receiving messages from other agents. The reaction to information may come in the form of an internal state change, an outgoing message, or an action. Finally, an agent may communicate with other agents via a messaging interface. For a more comprehensive overview of Agents and Agent Architecture see [Wooldridge95].

2.9. Healthcare Systems

Healthcare presents a particularly unique problem because of the sensitivity of the data involved, the large number of independent organizations involved in providing healthcare, and the slow technology adoption rate. Here we discuss and present some of the different types of healthcare systems that exist today.

[Peyton07b] discusses the use of a CoT within a healthcare setting. A CoT within a B2B healthcare network is particularly useful because of the many privacy requirements that exist for healthcare. Such a system allows patients to control their own identity and to receive service from many service providers without having to worry about their identity being compromised.

The use of Health Insurance Portability and Accountability Act compliant auditing trails for privacy and security compliance monitoring of medical imaging systems is shown in [Chen05].

2.9.1 Event-driven Healthcare

Event-driven technologies have been studied within the context of health care. Here we mention and discuss some of this research.

[Hoot04] developed a tool that integrates publish/subscribe with existing Electronic Health Records (EHR). The tool allows medical providers to stay up to date, and thus clinically responsible, with the most recent reports and testing data associated with a patient's EHR. The medical provider is the subscriber, and the patient's EHR is the publisher.

[Yupho06] discusses routing protocols that could be useful for routing within a wireless sensor network within a healthcare setting. They distinguish between continuous and event-driven methods. The event-driven method will only send data when a particular event arises. For example, a temperature monitor on a patient will only send event data when the temperature of the patient has reached 103.7 degrees. This is done so that the

party that is interested in temperatures is not overloaded with temperature readings. Instead, the interested party only receives information when an exceptional event has occurred.

[Mansour06] presents an event-driven approach to implementation of clinical practice guidelines for managing clinical care. A clinical practice guideline directs patients and clinicians on how to handle clinical problems. Clinical events are considered to be the drivers of clinical practice guidelines, and thus are seen as the rationale for an event-driven approach to managing clinical care. The approach specifies ECA rules within an XML and uses active databases to implement an EHR.

2.9.2 Data Warehousing & Web Portal

A data warehouse is a "subject-oriented, integrated, time varying, non-volatile collection of data that is used primarily in organizational decision making." [Inmon92] Data warehouses are typically used in order to provide Business Performance Management functionality. Users are not necessarily concerned with individual records, but rather with varying degrees of summarized data.

The data warehouse is maintained separately from an organization's operational databases [Chaudhuri97]. A data warehouse contains consolidated data from a number of operational databases. Data from the operational databases is transferred to the data warehouse on a regular interval (nightly, weekly, etc.) through a process called Extract Transform Load (ETL). The ETL process involves extracting the data from the operational database, transforming the data to the data warehouses acceptable format, and then loading the data into the data warehouse. Since we are accumulating multiple

operational databases into one place, a data warehouses will be orders of magnitude larger than a typical operational database.

The typical user interface for access to the data warehouse's reporting, monitoring and analysis functionality is the web portal [Peyton08]. Web portals offer users access to the data warehouse via the web. Users navigate to the web portal with their web browser where they can then run reports and analyze or view data. Each user's interface may be customized to meet their individual needs.

A data warehouse is referenced in [Peyton08] that is used by one of Canada's largest teaching hospitals. [Kuziemsky08] proposes the use of a Web Portal based system, PAL-IS, for use in a palliative care severe pain management environment. Our work will be compared against the proposed PAL-IS system.

2.9.3 Health 2.0

Health 2.0 technologies refer to those technologies that leverage Web 2.0 in order to create online communities in which patients, medical professionals, and stakeholders can collaborate and share data [Hu09]. Two major providers of Health 2.0 services are Microsoft Healthvault [HealthVault09] and Google Health [Google09]. Both Microsoft Healthvault and Google Health provide very similar services. Microsoft Healthvault and Google Health allow users to store and manage all of their health information in one location. Both claim that the user's information is safe and secure and that it will never be sold or shared without the user's specific consent.

In addition to storing and managing your health information Google Health provides users with search features, allow users to share their health records with family members, doctors or caregivers, and provide reference links to related information. Since your data is all stored online, they can cross reference multiple data sources and offer services such as checking to determine if any of medications a person is being prescribed could potentially interact with other medications that the person is being prescribed.

Chapter 3. Framework for Continuous Compliance Monitoring of B2B Processes

This chapter describes our proposed framework for continuous compliance monitoring of B2B processes. In section 3.1 we present an overview of current approaches to compliance (manual and paper based techniques, Web Portal), highlighting some known issues and problems in order to introduce our proposed framework. In section 3.2 we identify the most critical requirements for continuous compliance monitoring that our framework needs to address. In section 3.4 we describe our framework in terms of its methodology for leveraging Internet technology to embed data collection, audit trail support and alerts directly within B2B processes in order to meet the requirements we have identified. In section 3.4 we present the architecture that supports the framework. Finally, in section 3.5 we present the policy-based compliance monitor component of our architecture and its agent-oriented approach for automating compliance monitoring.

3.1. Overview

The traditional approach to compliance monitoring is through the use of manual and paper based techniques. Network participants track their activities and results by recording them on paper. A policy officer, who is aware of all the regulations in place, must then verify if these regulations are being met. This is done by analyzing the paper recordings of activity.

A more current approach to compliance monitoring, which tries to improve on the manual paper based technique, is through the use of a Web Portal. Figure 3 shows a high level view of the interactions that take place within this type of architecture.

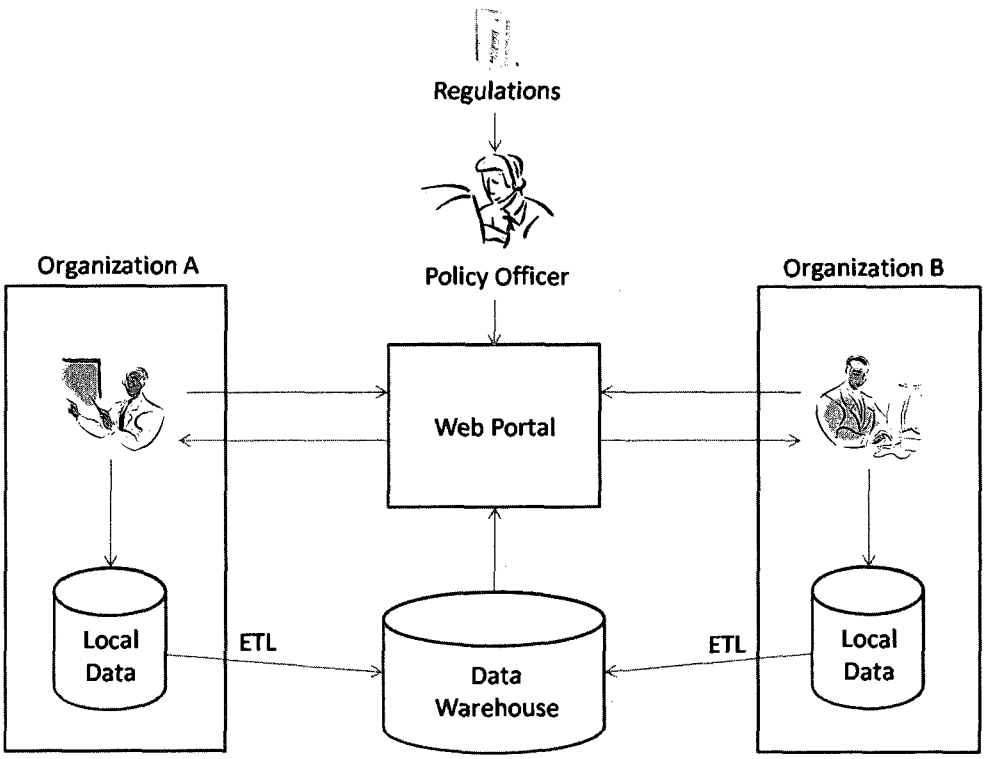

Figure 3 – High Level Web Portal Compliance Monitoring Architecture

There are multiple organizations interacting within a B2B network. Each organization maintains its own local databases in which it stores its local data. The data within these databases is made available to a network wide data warehouse through an ETL process or by members of the organization manually re-entering the data into the web portal to make it available to other organizations. In some cases, the information within the organization may only be available in paper form, and it is "re-entered" into the web portal in order to create an electronic record. The data warehouse is made available to the network participants by way of a Web Portal. All organizations are given

access to the web portal and can interact with it. In order to monitor for compliance, a policy officer is also given access to the Web Portal. The policy officer, who has knowledge of all regulations, will access the web portal in order to view data associated with the regulations. The policy officer will manually determine if compliance with the regulations in place is being met.

There are some faults with this type of system. Since data is not made available to the Web Portal as it is produced, and since the Policy Officer is manually checking for compliance at a convenient time, policy monitoring cannot be done at a real-time pace. We also see that policy monitoring is a manual process because a policy officer must check for compliance. Data is entered multiple times within the overall process thus creating a problem of double entry. Each organization has to decide what information needs to be available in the Web Portal for sharing with other organizations. This can be a timely task. Compliance monitoring is just one of the reasons that information is shared in the web portal, and as such it is not necessarily the primary focus of the system. All of the data is organized within one single data model for all purposes and organizations. Thus, the data model cannot be specialized to meet any one purpose or organizations need. Therefore, it could be difficult for the policy officer to find the information that they need for determining compliance.

Figure 4 shows a high-level view of an event-driven compliance monitoring architecture. This approach improves upon the Web Portal's faults. We discuss its advantages later in this section.

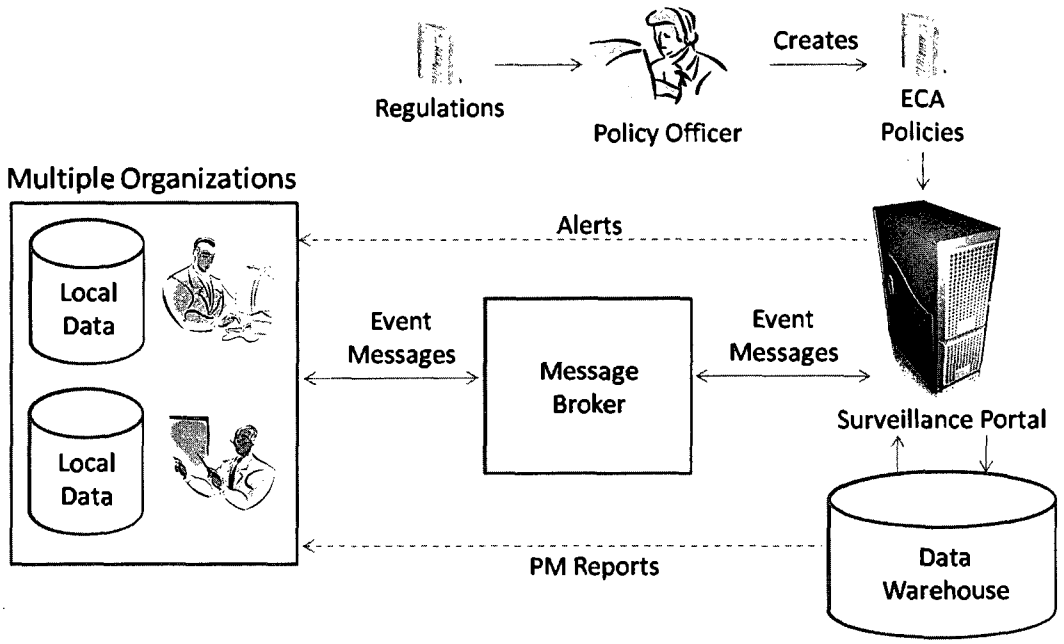

Figure 4 – High Level Event-driven Compliance Monitoring Architecture

The event-driven architecture is enabled by use of a publish/subscribe message broker that distributes event data between network participants. Within each organization, there are publishers and subscribers. Data is captured within the organization as event data and then made available to other interested parties (subscribers) by a publisher. To make the event data available to subscribers, the publisher publishes the event data as event messages to the message broker. The message broker then forwards those messages to subscribers. Subscribers subscribe to specific event messages with the message broker.

The Surveillance Portal is a special component that is interested in all event messages, and therefore has a subscription to all event messages, including event creation messages (creation of a new type of event message). This allows the Surveillance Portal to capture all messages, store those messages in a data warehouse for performance reporting, and to monitor the messages for compliance purposes. The Surveillance Portal

also provides alerts to compliance events. These alerts are seen as event messages and distributed to the interested parties.

The policy officer, who has knowledge of the regulations in place, will translate these regulations into ECA policies and then submit them to the Surveillance Portal for compliance monitoring. The Surveillance Portal monitors all messages to monitor for compliance with the ECA policies. Compliance monitoring can then occur at near real time.

In this approach, since data is made available as it is produced, and since policy monitoring is done by an automated system, compliance monitoring can be done at a real-time pace. Policy monitoring is not a manual process and data is only input once by its producer therefore eliminating any potential double entry. Organizations do not need to decide which of the information is published because all information is published. The message broker decides which information people are allowed to see. The data model can be defined locally by any of the subscribers and in particular, is done so by the Surveillance Portal. Since all data is shared electronically it is possible for the Surveillance Portal to automatically process much of the data related to compliance and provide it in a more convenient and actionable form for the policy officer.

3.2. Requirements

The following sections identify the requirements that a framework for continuous compliance monitoring of B2B processes should support. These requirements are a result of our literature review and out interactions with the Palliative Care team. Our focus is on the automation of compliance monitoring for B2B processes in which the necessary data

to measure compliance can be provided on-line as the processes are happening (in more or less real-time). The two main overall areas of requirements for continuous compliance monitoring are B2B Data Integration and Policy Monitoring. The data must be integrated across the B2B network in such a way so as to support a mechanism for continuous compliance monitoring. In order to provide compliance monitoring in general, we need to provide the system with a set of Policies based on the regulations in place, and we need to be able to monitor those policies for compliance. The two areas of requirements go hand in hand to support the overall goal of continuous compliance monitoring.

3.2.1 B2B Data Integration

B2B data integration means how data across a B2B network can be defined, published, shared and processed. That is to say that it incorporates a number of key areas with the goal of integrating data across a B2B network. Achieving this goal can improve efficiency and effectiveness of the business processes across the network. The main purpose of these requirements is to distinguish the key data integration elements that are needed to support a continuous compliance monitor.

Consistent Data Model

There must be a common data model across the entire network. This consists of a consistent, common set of attributes and data types as well as a consistent format. If the set of attributes and data types are not both consistent and common, then participants may start to use multiple attribute names or data types to represent the same thing. For example, using an integer value of '1' or '0' to represent a gender in one area, and using a string representation like 'Male' and 'Female' in another area.. In this case, the policy

monitor will not be able to properly monitor for compliance because the policies could be written using one particular set of attributes and data types and the event messages may be written using others. To prevent this, a look up list of enumerated values must be identified and formatted consistently. Users can search the list to determine the appropriate attributes and data types, thus eliminating any confusion from having inconsistent attributes and data types. Having a consistent format for the data model is needed to ensure that the system can store, and process any data without complications.

Extensible Data Model

The data model must be easily extended as new providers and new sources of information are added. A non-extensible data model requires that all data sources and data providers be identified prior to implementing the compliance monitor. This is not feasible because of the changing nature of B2B interactions. Therefore, we need to develop an extensible data model in order to provide continuous compliance monitoring.

Consistent Timestamps and IDs

Many policies and regulations consist of time restrictions. For instance, a patient must be visited every 24 hours by a nurse. In order to monitor such a policy, a timestamp must be provided when the information about the nurse visit is provided to the policy monitor. A consistent timestamp is one where the event in question actually occurred at the given time of the timestamp. Additionally, a consistent timestamp should be one that is in accordance with that of a consistent timer. This places restrictions on who can define the time. If the timestamps are not consistent, then the policy monitor will not be able to

accurately monitor for compliance. The compliance monitor results will become irrelevant because the timing information is inaccurate.

Many policies are dependent on individual entities, such as a patient or nurse. These entities must be represented by an identification (ID) code in order to allow entities to produce data, or to refer to other entities. The policy monitor in particular needs to be able to distinguish between entities in order to determine which entity is related to the particular policy being monitored. If the IDs used to represent entities are not consistent, the policy monitor results will not be accurate. If the policy monitor receives data from two different sources that refer to themselves with the same ID, then the policy monitor could come to conclusions that it would not have if it had known the two sets of data were from different sources.

Another aspect to consistent IDs is that of federated identity management. How can we ensure that two nurses from two different organizations are not given the same ID by their respective organization? How can we be certain that a single patient does not receive two different IDs from a hospital and a clinic? A federated identity management approach would remove these issues by managing all IDs across multiple organizations while still maintaining local identity within each organization.

Single sign on is another issue associated with consistent IDs. When a user signs on to one system, they do not want to continually enter username and password information when trying to access other connected applications. Users should be able to sign on once, and access all connected applications.

Data Entry

Data entry can be a tedious, time consuming task. When data is entered multiple times, errors can occur. Errors such as simple typing mistakes or entry into the wrong location (e.g. wrong database table) may occur. These types of errors can be reduced by limiting the number of times that data is entered. Thus, data should only be entered once, at the source online, and then published as events and made available to all relevant parties in near real-time. Although this does not ensure that typing mistakes do not occur, it will limit the frequency of their occurrence.

Data Distribution

Data distribution refers to the methods and processes by which data is distributed within the B2B network. In order to support continuous compliance monitoring the data distribution processes must be such that data is made available to the compliance monitor in a real-time fashion. This ensures that the compliance monitor will receive all data in real-time. This implies that data should not be solely made available in a pull format (entities must go and retrieve the data) but rather in a push format (data is pushed to interested entities). Making data available in both a push and pull format within the B2B network is ideal because it provides not only the compliance monitor and entities within the B2B network with real-time data but also allows these entities to query for data they may have previously not been interested in.

3.2.2 Policy Monitoring

Policy monitoring is the other major requirements group for continuous compliance monitoring. In order to provide continuous compliance monitoring, there

should be a mechanism for monitoring a set of policies that indicate the compliance requirements. There needs to be a way to match the policies that are articulated in the policy language, to the data collected from within the B2B network. We need to provide a support mechanism that allows users to be kept up to date on what is happening with regards to compliance. As well, we need to provide the ability to analyze what has happened with respect to compliance monitoring after the fact in order to ensure that everything has been correctly monitored.

The following requirements are important because they help provide and support policy monitoring capabilities.

Policy Language

In order to monitor a set of regulations for compliance, there is a need to translate those regulations into a common format. The common format should consist of executable policies that a machine can measure, monitor and act upon. This is to ensure that compliance monitoring can be automated and occur at a near-real time speed.

There are a number of characteristics that a machine processable language must have. It should be based on a standard. It should be able to articulate actions. This ensures that not only will we be able to determine if compliance is being met, but we can take action as a consequence to a specific policy. It should be able to integrate with the common data model. This allows the policies to be monitored based on the data that is available. If they are not integrated with the common data model, then there will be no connection between the data available and the contents of the policies.

Not only should the language be machine processable, but also human friendly. A language of 1s and 0s could be processed by a machine, but it would be difficult for someone to translate the regulations from natural language to a specific format of 1s and 0s. There should be a balance between the day to day language of humans and the specificity and standardization that a computer requires.

Since regulations can come in many different formats, and in many different contexts, we need a policy language that is expressive. It should be able to express any and all types of regulations.

Alerting

In order to provide entities within the B2B network with feedback related to the successfulness of their compliance with regulations, alerts should be raised. Alerts would flag violations with the regulations, but also alert entities of impending violations. These can act as triggers to entities in order to motivate them to take the appropriate actions that would avoid compliance failures. The alert needs to express the conditions under which the alert is raised. This could include, the purpose of the alert, the time of the alert, the circumstances that lead to the alert. Alerts should be delivered to all parties that require notification of the alert. The format of the alert should be such that the alert is easily recognized by anyone who reads it. Finally, the delivery of the alert should be done in a timely manner.

Performance Management

Performance management is necessary to support compliance monitoring. Full reporting must be supported in order to measure and evaluate business processes and

quantify the compliance with regulations. The performance management can help improve the organization's business processes by providing measurements of their effectiveness. The processes that are related to compliance can be improved by creating measurement goals associated with the processes. Performance management can offer us insight into the effectiveness of organizational guidelines by providing key measurements associated with the guidelines.

In order to provide performance management, a Business Intelligence tool should be linked to the common data model in place. Performance management across a B2B network would not be possible without this link because users would not be able to understand the data that they were viewing. The data must be represented with the same attributes and data types as any other data found within the B2B network.

Audit Trail

It must be possible to record a single, consistent audit trail of all events in the network in order to have a persistent record of all activity. This can ensure that if there is any confusion about what has happened in the past, that there is a record of activities available to clarify the order of events. For instance, if there are any complaints brought about within the B2B network, the audit trail can be investigated to determine if the complaint is valid. The audit trail can also be made available to any regulatory bodies in order to prove that the processes within the B2B network are sound and that compliance with regulations is actually being met.

3.3. Continuous Compliance Monitoring Framework

A framework consists of a technological architecture and a methodology for leveraging that architecture. The architecture, which is discussed in detail in sections 3.4 and 3.5, supports continuous data publishing, data integration and policy based automated compliance monitoring. Here we discuss the methodology for leveraging our proposed architecture.

A manual organization is one in which all business processes are manual and paper based. In such an organization, compliance monitoring consists of manually verifying whether the business processes match the requirements of the natural language regulations documents. The basic methodology of compliance monitoring in a manual organization is the following steps: determine what regulations are in place, determine what data is required to verify compliance with the regulations, create data gathering processes, gather all of the required data (in paper format), determine if compliance is being met, and create any responses required. Note that all data is contained and distributed in paper format.

A web portal based organization is one in which the main data distribution mechanism is through a web portal. Here, the computers, databases and the internet try to improve upon manual and paper based techniques. However, although data integration is not manual, policy monitoring is still a manual task. The methodology of such an organization is as follows: determine what regulations are in place, determine what data is required to verify compliance with the regulations, access web portal to gather data required for compliance monitoring (run reports), manually verify if compliance is being

met, and create any responses required. Note that all data is input into the web portal by organizational members and is available to the policy officer.

The proposed framework attempts to replace the manual effort of compliance monitoring within a manual or web portal based organization by using online automated techniques. Our methodology leverages Internet technology to embed data collection, audit trail support and alerts directly within B2B processes in order to meet the requirements we have identified.

There are three aspects to this methodology: defining the data, defining the policies, and data publishing.

Each member of the organization must follow these steps in order to define the data.

1. Determine an event that may not be registered yet (if any exist)
2. Check with the Message Broker to see if this event is already registered, if yes return to step 1, if no continue
3. Determine what name and attributes are associated with the event
4. Register the event with the Message Broker

All members of the organization are responsible for determining what data exists. Data is characterized in the form of events. When they discover any new events, they must register that event with the message broker. All events are registered with the

message broker in order to provide data publishing and distribution. In order to register an event with the message broker, they must first determine if that event already exists, and secondly they must determine the name of the event and the associated attributes. Attributes are the data content definition of the event. This process continues forever as new events can be created at any time within the organization.

The following steps are carried out by the Policy Officer in order to define the policies for the automated policy monitor.

1. Determine what regulations are in place (natural language documents)
2. Translate regulations into ECA policies
    a. For each regulation, determine which events are associated with it
        i. For each event, determine what conditions must be met for it to be triggered
        ii. For each event, determine the associated actions if the condition is met.
3. Submit the ECA policies to the policy monitor

Similar to the manual and web portal based organizations, the first step is to determine what regulations the organization must follow. This consists of gathering all of the natural language regulations documents together. Since an automated compliance monitor cannot read natural language, the documents must be transformed into ECA policies. For each regulation in place, an ECA policy must be created. This consists of determining what events are associated with the regulation, determining what conditions are associated with each event and determining what actions should be taken if the

condition is met. Finally, once the ECA policy has been written it should be submitted to the policy monitor. This process should continue forever as new regulations come into existence.

The following is carried out by all organization members in order to publish data and make it available to all organizational members in real time.

1. When an event occurs, immediately capture the data in an XML message
2. Immediately, publish the message to the message broker

In order to have real time continuous compliance monitoring, all data must be made available to the compliance monitor in real time. For this to occur, all data must be captured at its source in a suitable format. The format required by the message broker is an XML message format. Once the data is captured, it must be immediately published to the message broker. This ensures that all interested parties (including the compliance monitor) receive the data as quickly as possible. This process continues forever as new events will continue to occur.

3.4. Continuous Compliance Monitoring Architecture

The Compliance Monitor architecture can be seen in Figure 5. The data distribution component of this architecture is the Message Broker. Publishers publish event messages to it, and it forwards those event messages to any interested subscribers. A detailed description of the architecture of the Message Broker we propose for use can be found in [Eze09]. The Message Broker interacts directly with the Surveillance Portal. More specifically it interacts with the Event-driven Publish/Subscribe Interface (EPSI).

The interaction consists of sending and receiving event messages. Any messages that the EPSI receives are automatically forwarded to the Logger and to the Policy-based Compliance Monitor. The Logger takes the messages that it receives and inputs them into the MSG database. The MSG database is a store that contains all of the event data. A Business Intelligence Portal via a Data Warehouse is used to view and run reports against the MSG database. The Policy-based Compliance Monitor monitors all of the event messages in order to determine compliance with a set of policies. It can also produce compliance related alerts that are published in the form of event messages. The policies are stored in the Policy database. The policy database is loaded with a set of policies that are based on actual regulations that the organization must follow. Discussion of all of these components can be found in the following sections. Our work focuses on the Surveillance Portal, and we assume that the message broker architecture is already in place.

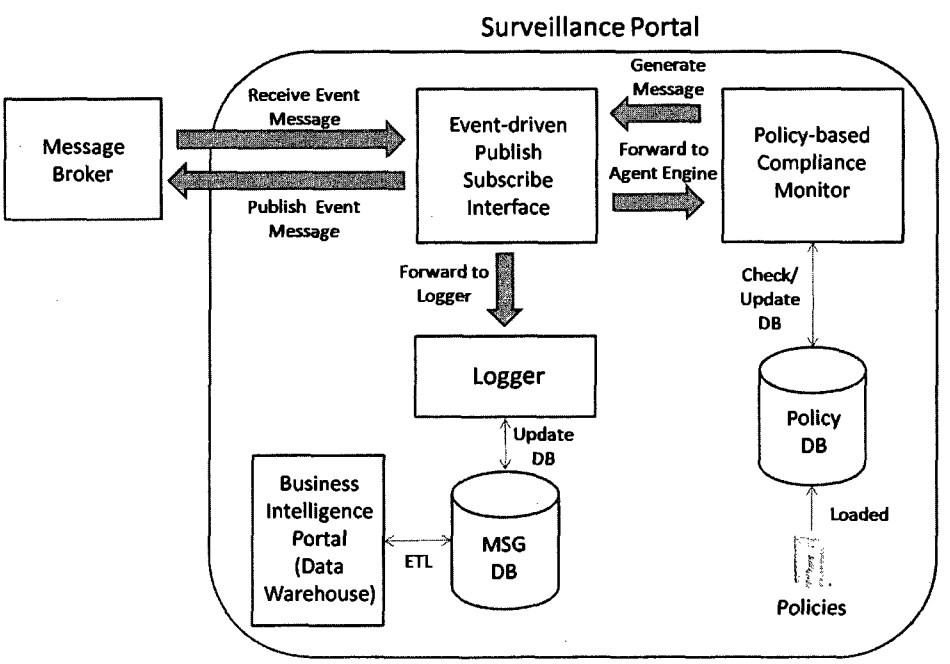

Figure 5 – Event-driven Compliance Monitoring Architecture

3.4.1 Message Broker

The Message Broker is the data distribution component to this architecture. Publishers publish event messages to the Message Broker, and it forwards those messages to any interested parties. Subscribers show their interest to specific events by submitting a subscription to the Message Broker. Events must be registered with the Message Broker. A registered event consists of a name and a set of attributes that describe the contents of the event. The list of registered events shows what events can be published and allows potential subscribers to know what events are available for subscription. The Surveillance Portal is subscribed to all event types that are registered with the Message Broker. This includes any new event creation messages. New event creation messages are published by the Message Broker when a new event has been registered with it. When an event message is published to the Message Broker, it is given a name (topic), and a set of attribute value pairs. The attributes consist of such things as the publisher's ID, the time of the event, as well as all of the attributes that were designated when the event was registered. Figure 6 shows the structure of the event message object.

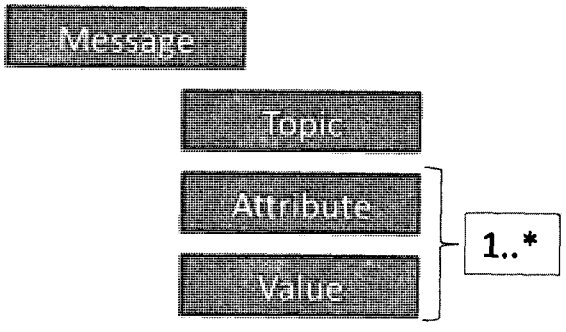

Figure 6 – Event Message Object

The Message Broker we propose to use is one in which subscriptions are represented as policies. A full detailed description of the architecture of the Message Broker can be found in [Eze09].

3.4.2 Event-driven Publish/Subscribe Interface

The EPSI is the interface between the Message Broker and the Surveillance Portal. The EPSI was proposed by [Eze09] and is referred to as a Partner Service Component (PSC). A PSC "is an application that assumes both the role of a publisher and a subscriber" [Eze09]. It is both a web service end point that can be called by the Message Broker, and a web service client that is implementing the WSDL of the Message Broker. This allows it to receive messages from the Message Broker and to publish messages to the Message Broker. The PSC also provides an interface for external applications or users to submit messages or receive messages. This interface consists of an Inbox folder and an Outbox folder. Applications will retrieve incoming messages from the Inbox and will put messages into the Outbox when they wish to publish a message.

The EPSI is a special version of a PSC. It has the added functionality of forwarding messages to the logger and the Policy-based Compliance Monitor (as opposed to simply putting the messages into an inbox). Any messages generated by the Policy-based Compliance Monitor are put into the Outbox for the EPSI to publish to the message broker. The EPSI will subscribe to all events, including the new event creation messages.

3.4.3 Logger

The Logger receives all messages that are delivered to the Surveillance Portal. The Logger will log all messages to the MSG database, thus making them available for extraction to a data warehouse (via ETL). In addition, the Logger is capable of creating new tables in the MSG database when new events are registered with the message broker. This is possible because the EPSI has subscribed to all new event creation messages from the broker. The new event creation message indicates that a new event type has been registered with the message broker. This means that publishers are now free to publish messages of this type. The new event creation message provides the event name as well as the attributes associated with the event. The Logger uses the event name as the table name and uses the attributes to create the columns within the table. When it receives an event message it can insert that message directly into the database by matching the event name with the table name and the associated attributes with the columns of the table.

3.4.4 MSG Database

The MSG Database is where all event data is stored. There is a unique table for all event types. The table is defined based on the name and the set of attributes associated with the event. The database is made available to the Business Intelligence Portal in order to provide performance management reporting on the data.

3.4.5 Business Intelligence Portal

The Business Intelligence Portal offers performance management reporting to users. A BI Tool, such as Cognos 8 Business Intelligence Suite from IBM [IBM09], is

connected to a data warehouse to provide the reporting functionality. Users must connect directly to the Business Intelligence Portal in order to generate and retrieve reports.

The data within the data warehouse is retrieved from the MSG database through an ETL process. Specifically, the data must be transformed to a structure (normalized or dimensional) suitable for reporting and analysis, and then loaded into the data warehouse.

3.4.6 Policy-based Compliance Monitor

The policy-based compliance monitor is where compliance monitoring occurs. It receives all messages from the EPSI. Once it receives a message, it monitors that message and compares it against a set of ECA policies. The name of the policy and attributes within the message correspond to the content of the policies. The name represents the event portion of the policy. The attributes represent the conditions portion of the policies. If the event name matches and the conditions hold, then the set of actions is carried out by the policy-based compliance monitor. One of the possible actions is an alert message generation. These alerts are in the same format as other event messages. The policy-based compliance monitor accesses the Policy database in order to view the policies in place, as well as to store key information. The inner workings of the policy-based compliance monitor are described in great detail in section 3.5.

3.4.7 Policy Database

The policy database is where the ECA policy definitions are stored. Additionally, it stores internal information that is useful for the policy-based compliance monitor. The ECA policies are defined based on the events that are registered with the Message

Broker. Each ECA policy consists of a set of events, conditions and actions. The events correspond directly to the set of events that are registered with the Message Broker. The conditions correspond to the set of attributes that are part of each event. Finally, the actions correspond to a set of internal tasks, or to producing event messages. The event messages that could be produced are referred to as alerts because they are alerting people within the organization of a compliance related event. Full details of the inner Policy database can be found in section 3.5.

3.4.8 Policies

Policies represent the various government regulations and legislation, organizational policies, and customer contracts that apply to the organization. These policies are natural language documents that stipulate the acceptable behavior of the organization. The data within the organization must be monitored in order to determine if these policies are being adhered to. The policies are translated into an ECA format and loaded into the policy-based compliance monitor so that they can be monitored.

3.5. Policy-based Compliance Monitor

The detailed inner workings of the Policy-based Compliance Monitor can be seen in Figure 7. The Monitor is made up of three components and three databases. The three components are the Agent Manager, Agent Timer and Agent Instances. The three databases are the Timer Events database, Policy database, and the Agent State database. The Timer Events and Agent State databases are internal database that support runtime instances only. Full descriptions of these components and databases are found in the following sections.

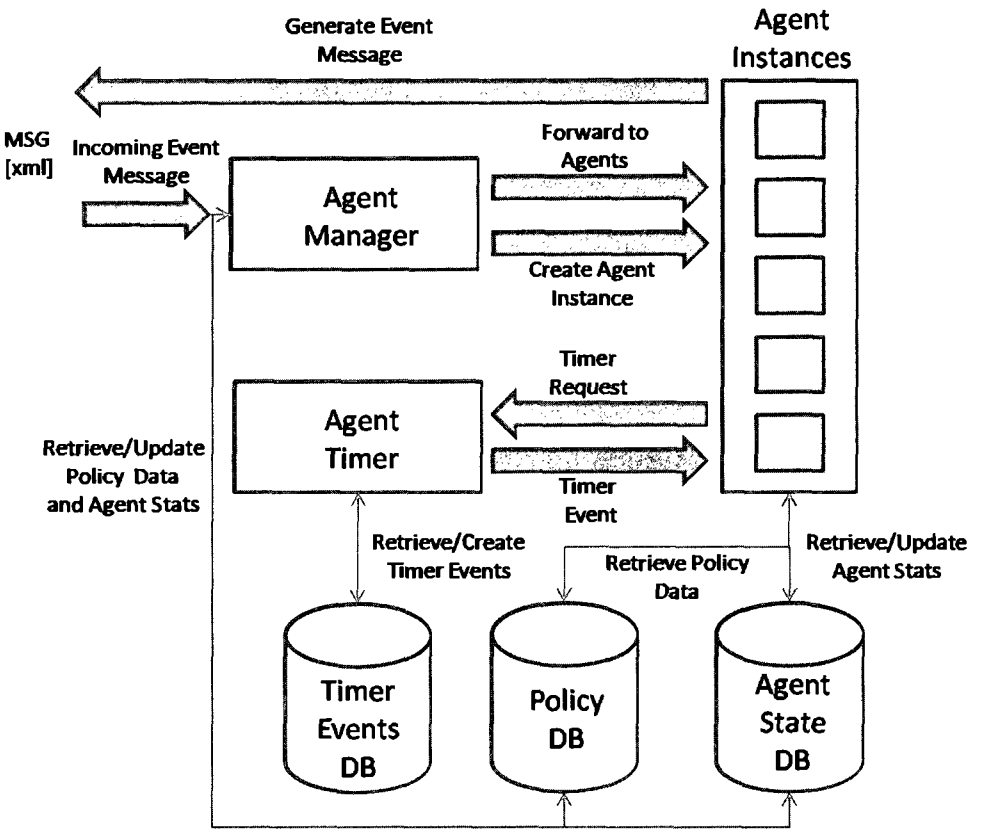

Figure 7 – Agent-based Design of Policy-based Compliance Monitor

3.5.1 Policy Definition

A policy is defined as an ECA policy. This means that it is represented by a set of Events, Conditions, and Actions. The events are the events that are registered with the Message Broker. The conditions are based on the attributes associated with the events. The actions are one of the following: generate alert message, create timer event, cancel timer event, change the internal state of the agent, or end the agent's life. Each event can be associated with many conditions and actions. Each policy may contain multiple events. Finally, the actions associated with the policy are only executed if the event occurs, and its associated conditions are true. Policies can be represented in XML format,

or they may be represented in database tables. These policies are stored in the Policy database.

A full example of an XML representation of a policy can be found in Appendix A. Here we show a small example in order to illustrate the policy definition.

```
<policy>
   <event>
       <name>PainAssessment</name>
       <isTrigger>no</isTrigger>
       <condition>
         <attribute>PatientID</attribute>
         <value>PatientID</value>
         <operator>=</operator>
         <TypeOfAttribute>message</TypeOfAttribute>
         <TypeOfVariable>variable</TypeOfVariable>
       </condition>
       <action>
         <type>CancelTimerEvent</type>
       </action>
       <action>
         <type>Exit</type>
       </action>
   </event>
</policy>
```

In this example, the ECA policy consists of a single event with a set of conditions and actions. The event in question is a 'PainAssessment' event. The conditions associated with this event are that the attribute 'PatientID' be equal to 'PatientID'. Finally, the actions associated with a match of the conditions are to cancel a timer event and to have the agent monitoring this policy terminated (as noted by the 'CancelTimerEvent' and 'Exit' type of action). The 'TypeOfValue' and 'TypeOfAttribute' are used to distinguish where the compliance monitor should look for the attribute value that is being compared (IE – in the database or in the message).

3.5.2 Agent Execution Model

Policy Definitions are stored in the Policy database. The schema for the Policy database can be seen in Figure 8. Policies are given a unique ID, a Name and a Description. Each policy has a number of Events, Conditions and Actions associated with them. Additionally, each Policy has a number of State_Variables associated with them. These State_Variables are primarily used to indicate to the Agent Manager if an attribute within an event message is of importance to that policy. This allows the Agent Manager to determine which Agents, based on the policy they are monitoring, to awaken and forward the message to. Agent instances are created by the Agent Manager by loading policy definitions and interpreting them. Each Agent Instance is responsible for monitoring a specific policy applied to a specific entity. For instance, a particular agent instance would be responsible for monitoring the policy found in Appendix A against a patient A, while a different agent instance would be responsible for monitoring the same policy against patient B.

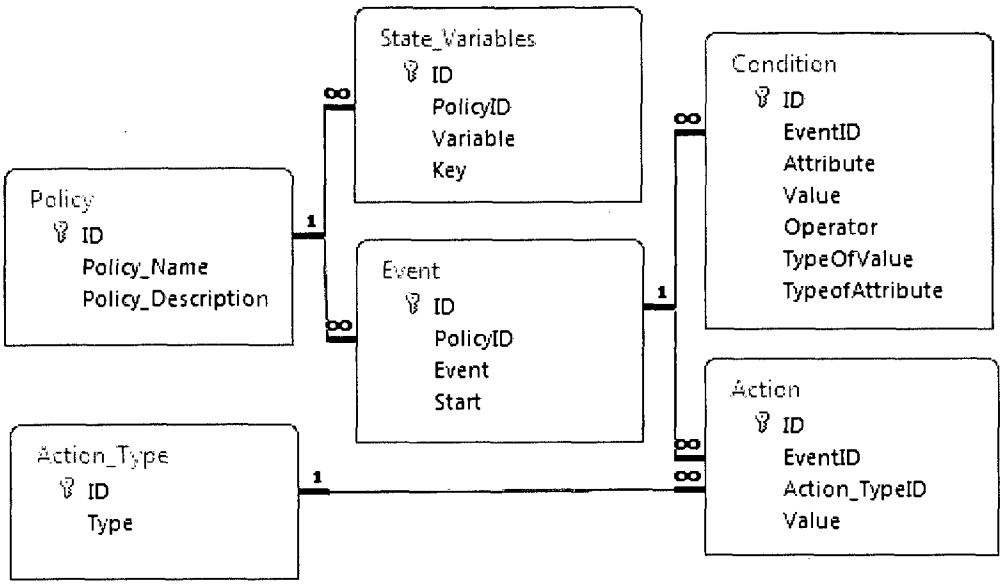

Figure 8 – Policy Object

In order to implement the Agent Instances, we store Agent Instance definitions within a database. This means that we store all of the data and variables associated with an Agent Instance within the Agent State database. There is no Agent Instance information stored in memory. Instead, there is a generic class that represents all agent instances. The generic class is called by the Agent Manager to activate any given Agent Instance. The Agent Manager provides the generic class with the Agent ID and the message that has just been received. The generic class will determine the state of the Agent Instance by accessing the Agent State database. The generic class will act as the Agent Instance, analyze the contents of the event message, update its state in the database, produce any messages or timer agent interactions required, and then close itself. Figure 9 shows the schema for the Agent State database. There are two tables in this database: Active_Agent_Variables, and Active_Agents.

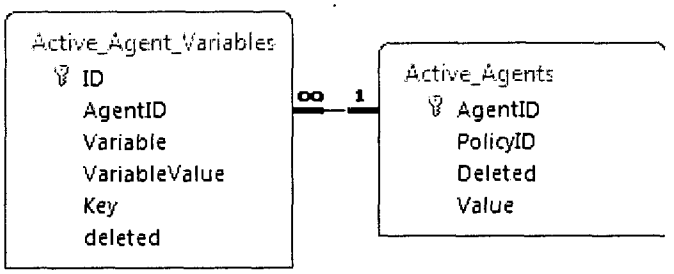

Figure 9 – Agent State

It would be useful to keep a complete change history of the Agent State database in order to have stats associated with the behavior of each agent. An action log table that contains a column that describes what action an agent took (IE changing or deleting a variable), along with a timestamp and the new value of the variable, would help determine what actions an agent has taken in the past. This would allow us to audit the database to determine or prove what actions an agent has taken.

3.5.3 Alerting

Alerting is used to inform network participants of a completed action, a compliance failure, or to warn them about an impending compliance failure. The Agent Instances will generate these alerts, in the form of event messages, which are published to the Message Broker. For each type of alert an alert event is registered with the Message Broker. Subscribers may subscribe to these alerts based on the alert event, or based on the attributes found within the event message. Alerts are defined within the policy definitions that they are associated with. Alerts are defined in the action section of the ECA policy definition.

Below we can see an example portion of a policy definition. An action of type 'GenerateMessage' is to be taken with a value of 'AdmitAssessmentFinished'. This

action would generate an alert in the format of an event message with name 'AdmitAssessmentFinished.' This message would be sent to the Message Broker who would then forward it to any interested parties.

```
<action>
  <type>GenerateMessage</type>
  <value>AdmitAssessmentFinished</value>
</action>
```

3.5.4 Agent Manager

The Agent Manager is responsible for two tasks: creating new agent instances, and forwarding event messages to the appropriate active Agent Instances. The Agent Manager receives all incoming event messages and reads the details of the message. If the message corresponds with a starting event and the agent responsible for monitoring this event has not previously been initiated, then the Agent Manager will create a new Agent Instance. The new Agent Instance will correspond with the appropriate policy and be initialized with the appropriate variables. The policy definition is loaded and interpreted in order for the Agent Instance to monitor it.

The Agent Manager will also forward the message to any active Agent Instances that may be interested in the event message. To achieve this, the Agent Manager will check the Agent State database and compare the 'variables' for each Agent Instance with the attributes of the message. If the 'variables' match with the attributes of the message, the message is forwarded to the Agent Instance.

3.5.5 Agent Timer

In order to incorporate the use of time within policies an Agent Timer is needed. The Agent Timer will accept timer requests from agent instances and will notify an agent instance when a specific timer event occurs. A timer event is an event message that is only seen internally to the policy-based compliance monitor (it is not published to the Message Broker). To request a timer event, the Agent Instance calls the Agent Timer, giving it the following values: Policy ID, Agent ID, Time Period, and Value. The Policy ID indicates which policy the timer event is associated with. The Agent ID indicates what Agent Instance the timer event is for. The Time Period indicates the unit of time. Finally, the value indicates the length of time. For example, the XML code below shows a possible action of type 'GenerateTimerEvent' that would cause the Agent Instance to make a request to the Agent Timer for a timer event of 1 (the 'value') minute (the 'timePeriod'). These two values represent the attributes of the event message. This type of event message does not require a name because it is internal to the system and automatically recognized. Figure 10 shows a timer event object.

```
<action>
  <type>GenerateTimerEvent</type>
  <timePeriod>m</timePeriod>
  <value>1</value>
</action>
```

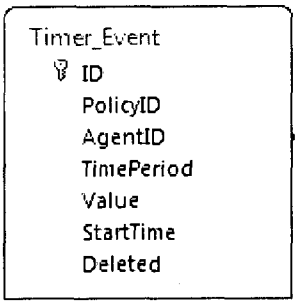

Figure 10 – Timer Event Object

In addition to these values provided by the Agent Instance, the Agent Timer will generate two other values: the ID and the Start Time. The Start Time indicates when the request was made and provides the Agent Timer with a starting reference time. The ID is used to distinguish between the different timer events, with each timer event having a unique ID. Finally, there is a Deleted value that is used to distinguish when a timer event is no longer active (the timer event has happened, or the timer event has been canceled by an Agent Instance). Since we do not actually delete the timer event from the table, we are able to track the history of timer events. All of these values are stored in the Timer Database table Timer_Event. The schema for the Timer Database can be seen in Figure 10.

Chapter 4. Case Study

We evaluated our framework for compliance monitoring based on a case study in Palliative Care, in which organizations collaborate to provide regulated at home care. We also evaluated an existing Web Portal implementation (PAL-IS) [Kuziemsky08] to serve as a basis for comparison with our Continuous Compliance Monitoring framework. In section 4.1 we briefly define Palliative Care and highlight some of the issues with respect to compliance monitoring. Then, in section 4.2 we introduce a typical Palliative Care scenario as a focus for our case study. In section 4.3 we describe the PAL-IS Web Portal implementation of the scenario and its support for compliance monitoring. In section 4.4 we describe the implementation of our continuous compliance monitoring framework and its support for compliance monitoring. Finally, in section 4.5 we discuss in detail the implementation of the Policy-based Compliance Monitor and some of the alternatives considered in support of its agent-based architecture.

4.1. Palliative Care

The World Health Organization describes Palliative Care as "an approach that improves the quality of life of patients and their families facing the problem associated with life-threatening illness, through the prevention and relief of suffering by means of early identification and impeccable assessment and treatment of pain and other problems, physical, psychosocial and spiritual" [WHO09]. The goal of Palliative Care is not to cure, or delay the patient's disease, but rather to improve the quality of the patient's life by reducing the severity of the disease's symptoms.

Palliative Care faces a number of complex issues. One of which is that it cuts across the boundaries of multiple organizations. The patient, who is suffering from a life-threatening illness, is often living at his or her home. The patient has a team of healthcare professionals who help to maintain his or her quality of life. This team of healthcare professionals consists of doctors, nurses and other healthcare workers who are associated with various hospitals. The patient also has a case manager who is responsible for getting the patient the best care possible. The situation cuts across a lot of boundaries and involves many organizations and people. This truly is a complex B2B interaction.

Within healthcare, and particularly Palliative Care, we can find a large number of regulations and policies in place. These regulations and policies cover a wide range of requirements for how Palliative Care patients are to be treated. In particular, there are typically organizational guidelines and accreditation standards for quality of care that cut across the different organizations. We have selected a few of these guidelines and standards for use within our scenario.

Palliative Care is also an area of interest because it is an area where technology has not been widely adopted. PAL-IS utilizes technology by providing data access through web browsers. There is potential to greatly improve the efficiency and effectiveness of the services and collaborations within the B2B network of Palliative Care by bringing the business processes online and automating otherwise manual tasks. For instance, remote monitoring devices can provide real-time data and data analysis that would otherwise not be available to healthcare providers.

4.2. Palliative Care Scenario

As a focus for our case study, in order to evaluate our thesis we introduce a Palliative Care scenario (see Figure 11). This scenario revolves around a business process for severe pain management in the context of a patient receiving home care [Kuziemsky08]. There are 3 main participants in the scenario. Nancy is a cancer patient who is being monitored remotely through a system of community care that integrates healthcare professionals from several organizations. Nancy is responsible for entering pain scores into her home computer or PDA in order to help facilitate her home care. Nancy has a Case Manager, Wilma, from a provincial health organization who is responsible for ensuring the quality of care she and others in the region receive. Wilma is responsible for admitting and discharging patients from the Palliative Care program and posting data collected in that context. Nurse Betty is a nurse from a local healthcare organization assigned to visit Nancy and other patients in their homes as part of a team of mobile healthcare workers. Nurse Betty is responsible for posting Pain Assessments based on patient interactions and data.

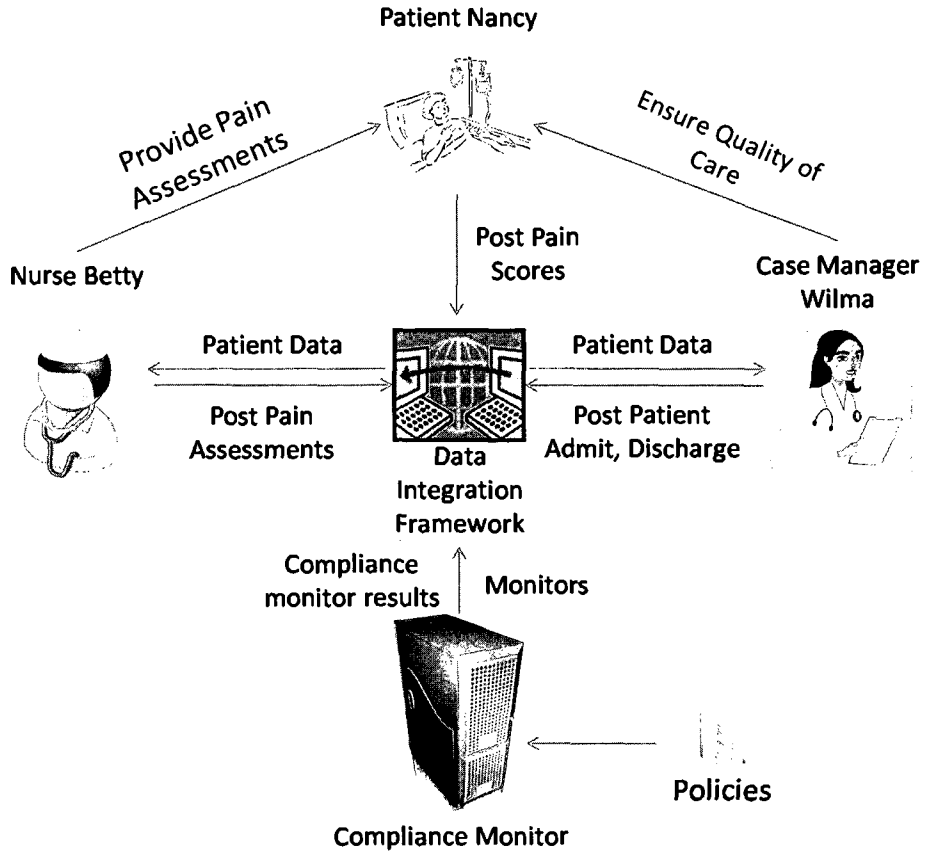

Figure 11 – Palliative Care Scenario

There are a number of provincially and federally mandated guidelines and regulations that are relevant to ensure that Nancy receives reasonable quality of care. Three examples that we will focus on, are Local Health Integration Network (LHIN) policies that must be followed to ensure accreditation and compliance with provincial/federal guidelines and regulations:

1. *Admit Assessment* - Once a patient is admitted into the system a Pain Assessment must be administered within 24 hours.
2. *Daily Assessment* - Each patient in the system must be given a Pain Assessment at least once every 24 hours.

3. *Response Assessment* - Once a pain score of greater than 7 has been entered by a patient, a Pain Assessment must be administered within 2 hours.

The participants, entities and interactions within the Palliative Care B2B network can be seen in Figure 11. The Palliative Care scenario requires a compliance monitor in order to provide a mechanism for tracking whether or not the policies in place are being met. The compliance monitor is responsible for collecting and analyzing data delivered by the data integration framework and providing Nurse Betty and Case Manager Wilma with feedback related to policy compliance by way of the data integration framework. The policies that are in place are supplied to the compliance monitor. Neither the compliance monitor, nor the data integration framework are necessarily computer based systems. For instance, currently the data integration framework is through person to person interaction and paper based techniques. As well, the current compliance monitor is a set of compliance officers (LHIN members, outside accreditors, etc.) who manually checks for compliance.

4.3. Web Portal

4.3.1 PAL-IS Web Portal Architecture

A web portal based information system (PAL-IS) for the Ontario LHIN was proposed by [Kuziemsky08] and is currently being implemented at the LHIN. In this system, many external data sources are connected together with a central application in order to provide a single point for accessing and inputting data. User interfaces are provided through the use of web pages. Users can input data, view data, execute queries

and read reports through the interfaces. Application developers define the interfaces and the coordination of the data sources.

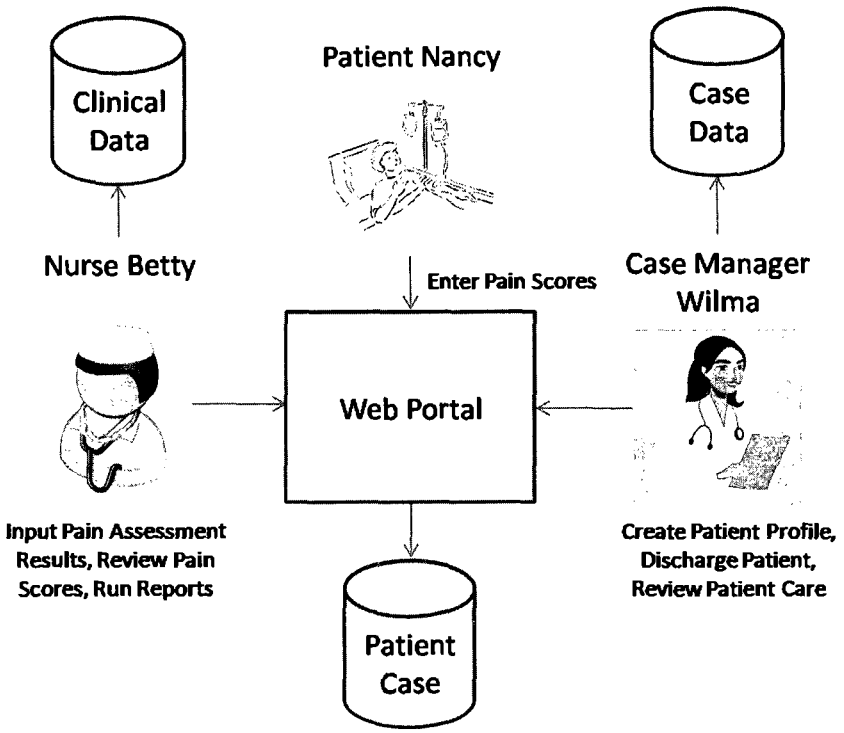

Figure 12 – PAL-IS Web Portal Framework

Figure 12 shows a high level view of the Palliative Care scenario implemented by the PAL-IS Web Portal framework. The details of the scenario can be found in section 4.2. The data integration framework within the scenario is the PAL-IS Web Portal. All participants access the portal in order to enter data and view data provided by other participants. Patient Nancy's web interface is customized for entering pain scores. Wilma's interface allows her to create user profiles, discharge patients from the system, and review the care that a patient is receiving (by running reports). Betty can input pain assessments, review patient's pain scores and run reports. Both Wilma and Betty maintain their own local database. This means that they are both entering the same data multiple times into different data stores. All data entered into the Web Portal is stored

within the Patient Case database. Compliance monitoring is achieved through manual review of the data found within the Web Portal's Patient Case database.

4.3.2 Compliance Monitoring

The Web Portal's Patient Case database attempts to ensure that there is a record of all activity within the B2B network. It is entirely dependent on the individuals within the network remembering to enter their data. Any authorized parties who are interested in reviewing the data can do so by logging into the Web Portal. In this scenario, Case Manager Wilma is interested in determining whether or not the three policies in place are being complied with. It is her job to continually ensure that the activities related to her patient are appropriate and adhering to the policies in place within the B2B network. In order to do this, she must maintain a collection of the policies in place. This can consist of memorizing the policies, maintaining a set of physical files, or maintaining a set of electronic documents. She must review the data that is available to her, by accessing the Web Portal, in order to determine if the policies are being met. Finally, she must create reports that indicate her conclusions with regard to compliance with the policies in place. These reports can then be made available to any other interested parties.

Outside accreditation and auditors, as well as policy officers within the LHIN responsible for compliance, will depend on reports that can be generated from the patient case database. Since the data within the patient case database may not be fully up to date, or may be missing key information, individuals will have to dedicate time to gathering missing data and handcrafting the reports.

4.3.3 Performance Management

Performance management reporting is offered through the Web Portal. End user reporting tools can attempt to create reports based on the data from within the patient case database. The data within the database may be incomplete or inconsistent depending on the effort that each participant gives to entering their data in a consistent and timely fashion.

4.3.4 Data Entry

Patient Nancy enters her pain scores directly into the Web Portal through a customized web interface. Nurse Betty and Case Manager Wilma both must keep their own data store in order to locally track the data that is important to them. This is the initial entry point for Nurse Betty and Case Manager Wilma's data. It offers a convenient and timely approach to capturing data. When Betty or Wilma has the time to, they will input similar data into the Web Portal via user specific web pages. Often this data will not be input until a much later date. It is possible that a data entry clerk could be hired to input the data. This would save time and effort for both Nurse Betty and Case Manager Wilma. This would also improve the timeliness of the Patient Case database, but not to the extent of the database containing real time data.

4.3.5 Alerting

Alerting is not supported or tracked within the PAL-IS Web Portal. Any alerting is done in a strictly manual ah-hoc way. Wilma would need to send an e-mail, make a phone call, or actually speak to Betty in order to alert her of failed compliance.

4.3.6 Analysis

The Web Portal approach provides some support for compliance monitoring within the context of B2B processes; however there are some short comings to this approach. These shortcomings are discussed below:

- Compliance monitoring is a manual process, and thus is not continuous or real time. Anyone interested in compliance must make his or her own conclusions and manually gather the important data from the patient case database. The data cannot be continuously monitored and therefore will only be analyzed for compliance at convenient times. There is no single person who takes care of compliance monitoring and there is no systematic approach between all of the interested parties. This approach could lead to errors due to its manual nature.
- Data entry is done multiple times. Each participant maintains a local data store and must, at some point in the future, transfer this data to the Web Portal.
- Data is not available to the Web Portal in real time. Since the Web Portal is not the primary data capture point, data is not input in real time. Instead it is captured in some other format, and then transferred to the Web Portal. This does not support continuous monitoring.
- The set of regulations is outside of the system. Regulations are written in natural language and stored in written documents. There is no strict record of what is being monitored, or of the results of the monitoring.

- Although alerting is possible, it is not practical. Alerts cannot be pushed out to intended recipients. They require users to connect to the Web Portal through a web page in order to receive alerts. If a user does not connect in a timely manner, alerts can become old and no longer useful. As well, Alerts will only be based on data, not time.
- Data model is defined by developer at design time. Thus, it also requires a developer to alter. This is due to the fact that data is viewed through strictly defined user interfaces (web pages).

4.4. Continuous Compliance Monitor

We took an iterative approach to developing the Continuous Compliance Monitor framework. Our initial attempt at developing the framework involved addressing the requirement that data be made available in a real-time continuous fashion. In doing so we developed the event based data model, along with the publish/subscribe infrastructure and Surveillance Portal data capturing ability. The Surveillance Portal collected all of the data being distributed within the B2B network and made it available for audit trail and performance management purposes. The details of the audit trail can be found in section 4.4.2.

On our second pass through, we improved the framework by adding automated compliance monitoring. This consisted of representing regulations within the system's database as ECA policies that could be interpreted by a set of agents in order to monitor them for compliance. The details of this can be found in section 4.4.3. We also improved

the abilities of the audit trail by including all of the Policy-based Compliance Monitor interactions within a database.

4.4.1 Continuous Compliance Monitor Architecture

Figure 13 shows the Palliative Care scenario being implemented within the context of the Continuous Compliance Monitor Framework. The architecture is able to support a publish/subscribe infrastructure for monitoring compliance by using a Message Broker that collects and distributes all care-related events that occur in the network, and a Surveillance Portal that processes the raw data stream of events, and generates its own reports and alerts. Full details of the Continuous Compliance Monitor methodology and architecture can be found in sections 3.3 and 3.4.

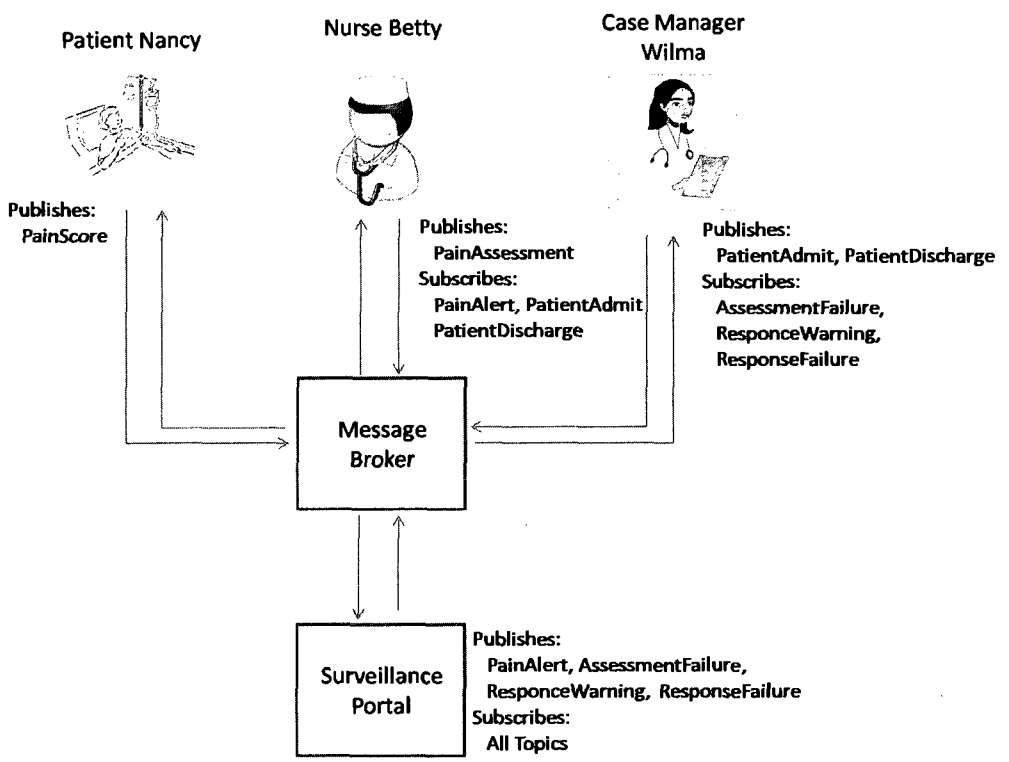

Figure 13 – Continuous Compliance Monitor Architecture

The publish/subscribe Message Broker is the data integration component to this architecture. We used a policy-based message broker [Eze09] because it provides a flexible framework for event-driven business process integration using publish/subscribe, SOA and declarative policies for controlling data sharing. Publishers will publish messages to the broker and the broker will forward the messages to any subscribers who are interested in the message. A subscriber's interest in a message is specified by a subscription with the message broker.

The messages that are being passed around must belong to an event type that is registered with the message broker. Events are defined by a publisher and are registered with the Message Broker. A registered event includes a list of attribute value pairs that define the messages published under this event. An event must be registered before a message for that event can be published.

The Surveillance Portal provides the compliance monitoring functionality for this architecture. Policies are provided to the Surveillance Portal in an ECA format. It logs all messages and analyzes them to determine compliance with the policies in place. It produces alerts and warnings that are associated with the policies. Additionally, the Surveillance Portal offers audit trail and reporting functionality.

There are 8 different types of events that occur in this scenario. Each event is represented by a message that gives the details pertaining to the event (in attribute value format). All messages have an attribute that represents the publisher of the message and an attribute that represents the time the event occurred. Below we list all of the events, and describe them. The description includes the attributes that will be found in the event

message, as well as a list of publishers and subscribers for each event. All events are subscribed to by the Surveillance Portal.

- *PatientAdmit* – This is the initial event that indicates a patient is being admitted into the system.
    - The attributes for this event are: Patient ID
    - The publishers of this event are: Case Manager Wilma
    - The subscribers to this event are: Nurse Betty
- *PatientDischarge* – This event indicates that a patient is no longer the responsibility of the system.
    - The attributes for this event are: Patient ID
    - The publishers of this event are: Case Manager Wilma
    - The subscribers to this event are: Nurse Betty
- *PainScore* – A pain score is a patient's own interpretation of his or her pain level. It is a value ranging from 1 to 10. This event indicates the submission of a pain score by a patient.
    - The attributes for this event are: Patient ID, PainScore
    - The publishers of this event are: Patient Nancy
    - The subscribers to this event are: Nurse Betty
- *PainAssessment* – A pain assessment is a nurse's evaluation of a patient's pain. This event indicates that a nurse has given a pain assessment to a patient.
    - The attributes for this event are: Patient ID, PainScore
    - The publishers of this event are: Nurse Betty.

- The subscribers to this event are: Case Manager Wilma

- *PainAlert* – This event indicates that a patient has expressed a pain score greater than 7.
  - The attributes for this event are: Patient ID, PainScore
  - The publishers of this event are: Surveillance Portal
  - The subscribers to this event are: Nurse Betty, Case Manager Wilma
- *ResponseWarning* – This event indicates a warning that a pain assessment needs to be administered to a patient soon. This is with respect to the *Response Assessment* policy
  - The attributes for this event are: Patient ID, Time Remaining
  - The publishers of this event are: Surveillance Portal
  - The subscribers to this event are: Nurse Betty, Case Manager Wilma
- *ResponseFailure* – This event indicates that a pain assessment has not occurred with respect to the *Response Assessment* policy.
  - The attributes for this event are: Patient ID,
  - The publishers of this event are: Surveillance Portal
  - The subscribers to this event are: Nurse Betty, Case Manager Wilma
- *AssessmentFailure* – This event indicates that a nurse has failed to give a pain assessment to a patient. This is associated with the *Admit Assessment* and *Daily Assessment* policies.
  - The attributes for this event are: Patient ID
  - The publishers of this event are: Surveillance Portal
  - The subscribers to this event are: Nurse Betty, Case Manager Wilma

4.4.2 Audit Trail

An audit trail is offered within this system. All events messages exchanged within the B2B network are captured by the Surveillance Portal (including event messages sent from the Surveillance Portal). Each event message is stored within a database. These event messages include key information such as the time of the event, the publisher of the event, and all event message attributes. Interested parties can query this database in order to determine the sequence of events that has occurred within the B2B network. Queries can be tailored to select data that is related to specific policies. This allows interested parties to verify the results of the compliance monitoring done by the Surveillance Portal.

An added benefit to capturing all event messages is that the data is now available for performance management reporting. The data can be transferred through an ETL process to a data warehouse. Business Intelligence tools can then be connected to the data warehouse to provide performance management reporting functionality. This reporting can be used to verify the results of the Surveillance Portal's compliance monitoring in addition to helping provide key information for business process improvement.

All interactions internal to the Policy-based Compliance Monitor are captured and maintained within the databases of the Surveillance Portal. Each table has a specific column to indicate when a row is deleted, this ensures that all internal interactions are maintained within the database and made available for future viewing (IE – they are never removed from the database). This data can be viewed or queried by an interested party in order to validate or determine the sequence of events that has occurred within the Policy-based Compliance Monitor. This constant capturing of the internal interactions is

useful for maintaining the Policy-based Compliance Monitor because it helps support the debugging process.

4.4.3 Agents

Each Agent is responsible for monitoring a specific policy. For this reason, we consider the ECA policy definitions to also represent Agent definitions. So an Agent definition is a set of ECA rules that create an ECA policy. These rules are created from elements of the common data model, as well as from a select set of predefined actions that an agent can perform. The first step to creating the ECA rules is to identify the appropriate events from the common data model. These events represent the Event portion of the ECA rules. Based on the selected Event, we must determine the conditions that need to be met. These conditions are selected from the attributes that are associated with the Event. Finally, we must select the appropriate actions that occur if the event and conditions hold true. An Agent can perform 5 different types of actions: *GenerateEvent*, *GenerateTimerEvent*, *CancelTimerEvent*, *SET* a variable local to the agent, and *EXIT*. Figure 14 below shows an example definition of an Agent. This example represents the Agent that monitors the *Response Assessment* policy.

```
ON event = PainScore
IF PainScore > 7 & PatientID = p & Status = unitialized
DO Status = initialized &
GenerateTimerEvent (1Hour, PatientID p)

ON event = Timer
IF Status = initialized & PatientID = p
Do Status = noresponse
GenerateEvent (ResponseWarning)
GenerateTimerEvent (1Hour, PatientID p)

ON event = Timer
IF Status = noresponse & patient = p
Do GenerateEvent (ResponseFailure)
GenerateTimerEvent (1Hour, PatientID p)

ON event = PainAssessment
IF patient = p
DO CancelTimerEvent (PatientID p)
Exit

ON event = PatientDischarge
IF PatientID = p
DO CancelTimerEvent (patient p)
Exit
```

Figure 14 – Example Agent Definition

When an Agent instance is first created it is in an unitialized state. When a PainScore event occurs with a Painscore > 7 it changes to an initialized status, and is woken up every hour by a timer event until either a PainAssesment or PatientDischarge event occurs. If neither event has happened after one hour, then a ResponseWarning event is published. After that a ResponseFailure Event is published every hour. Note that a failure message is sent every hour, once the 2 hours has expired, until either the pain is assessed or the patient is discharged.

Here is an example (shown by listing the messages involved) of a situation where a patient has submitted a pain score greater than 7, and is waiting for a pain assessment.

In this case, it is more than 3 hours before a pain assessment is done. Before that happens, 3 alerts (1 warning, 2 failures) are triggered. Following the Pain Assessment, the agent instance that was monitoring the policy is terminated.

```
PainScore (PublisherID: Nancy, Time Stamp: 2009/01/05
12:00, Pain Score: 8)
PainAlert (PublisherID: SurvP, Time Stamp: 2009/01/05
12:00, PatientID: Nancy, Pain Score: 8)
ResponseWarning (PublisherID: SurvP, Time Stamp:
2009/01/05 13:00, PatientID: Nancy, Time Remaining: 1
hour)
ReponseFailure (PublisherID: SurvP, Time Stamp:
2009/01/05 14:00, PatientID: Nancy)
ReponseFailure (PublisherID: SurvP, Time Stamp:
2009/01/05 15:00, PatientID: Nancy)
PainAssessment (PublisherID: Betty, Time Stamp:
2009/01/05 15:22, PatientID: Nancy, PainScore: 5)
```

Note that any subsequent PainScore events with pain greater than 7 from Patient Nancy during this interaction would be ignored by the Agent. Messages generated by the Agent are published under the ID of the Surveillance Portal.

4.4.4 New Event Creation

A new event is created, Temperature, in order to publish the patient's temperature (possibly by a monitoring device). This new information is useful for Nurse Betty to determine Nancy's health status. The Message Broker will now make this event available to subscribers (and other publishers as well). The Message Broker will send the Surveillance Portal a new event creation message indicating the new event name and attributes. Based on the attributes associated with the Temperature event, the Compliance Monitor's logger will automatically update its message database with a table for the temperature event. All subsequent Temperature messages received by the Surveillance

Portal will be logged to the message database. Any regulations that are in place and that are based on a patient's Temperature can now be translated into ECA policies that can be provided to and monitored by the Surveillance Portal. In order to provide any new alert messages with these new ECA policies, the Surveillance Portal must register the Alert events with the Message Broker.

4.4.5 Analysis

- Event-driven Publish/Subscribe Message Broker allows for real time data integration. This supports real time compliance monitoring, and continuous compliance monitoring. Data is received by the Surveillance Portal almost immediately after it is produced by an event source. This data is immediately provided to the Policy-based Compliance Monitor in order to determine compliance with the many ECA policies.
- The use of Agents supports the notion of scalability. The work of monitoring policies is split amongst multiple agents. This implies that the system can handle a large number of distinct ECA policies. Many policies can be monitored at the same time by multiple Agents.
- The data model is dynamic. When new event sources or event types are needed, they can immediately be registered with the Message Broker. Consequently, the events are known to the Surveillance Portal and can then be captured and processed.

4.5. Surveillance Portal Implementation

When implementing the Surveillance Portal's Policy-based Compliance Monitor we took into consideration a few possible different technologies that could have been utilized. We considered developing the agents within the Java Agent Development Architecture (JADE) environment. JADE is a java based agent development environment. Using JADE would have been ideal, but it required much more time and effort and did not provide the debugging features that we required. For the policy language, we considered using JRules and eXtensible Access Control Markup Language (XACML). JRules is a rules based language that would allow us to represent policies as rules. JRules does not support timing abilities nor does it provide the abilities to handle state. JRules would have been an acceptable choice. Similarly, XACML does not provide all of the abilities that we required. A full discussion of the technologies we considered for the Policy-based Compliance Monitor can be found in section 5.4.

The Surveillance Portal was implemented in the java programming language as a database application. The implementation consisted of three databases and a set of java classes. Figure 16 shows the class diagram for the Surveillance Portal. The Logger class represents the Surveillance Portal's Logger. The Message Handler class represents the Surveillance Portal's EPSI. The AgentManager, AgentTimer and AgentInstance classes represent the Surveillance Portal's Policy-based Compliance Monitor.

In order to test the implementation, messages were dropped into an inbox for the Message Handler class to retrieve. Any messages that the Message Handler output were retrieved from an outbox folder. All messages were stored to a message store database,

and all internal interactions were captured in an internal state database. This allowed us to determine if the system was functioning properly and to debug the system if it was not. Figure 15 shows the architecture of the implementation of the Surveillance Portal.

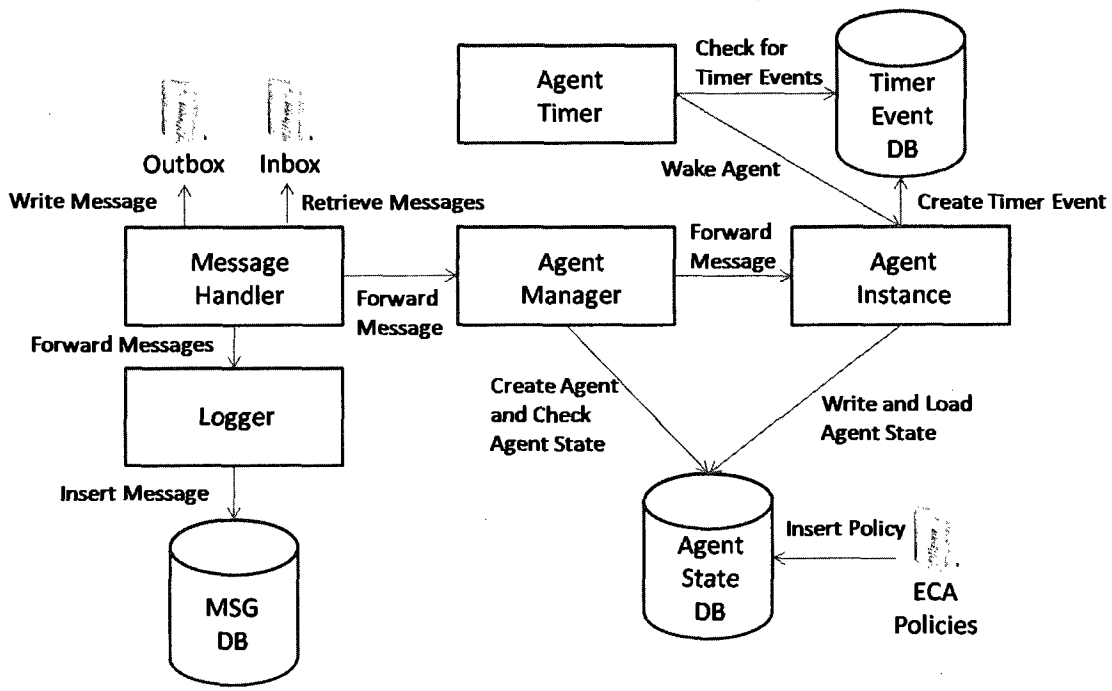

Figure 15 – Surveillance Portal Implementation Architecture

All agents in this system are stored in the Agent State database. When a new agent is needed, it is created within the database tables by an AgentManager object. The functionality of agents was achieved by having a single generic AgentInstance class that mimics any agent. When an agent needs to be awoken because of an incoming message, an AgentInstance object is created, with the agent's specific attributes (obtained from the agent state database) loaded into the object. The AgentInstance is passed the event message; it analyzes the message, and then updates its state in the database. The AgentInstance object then exits.

In order to provide timing capabilities, an AgentTimer class was developed. An AgentTimer object is created, and every few seconds it checks for timer events. The timer events are stored in a timer event database and indicate what agent should be woken and when that agent should be woken. The AgentTimer, upon observing a timer event, will create an AgentInstance object, load it with the appropriate attributes, and provide a timer event message for the AgentInstance.

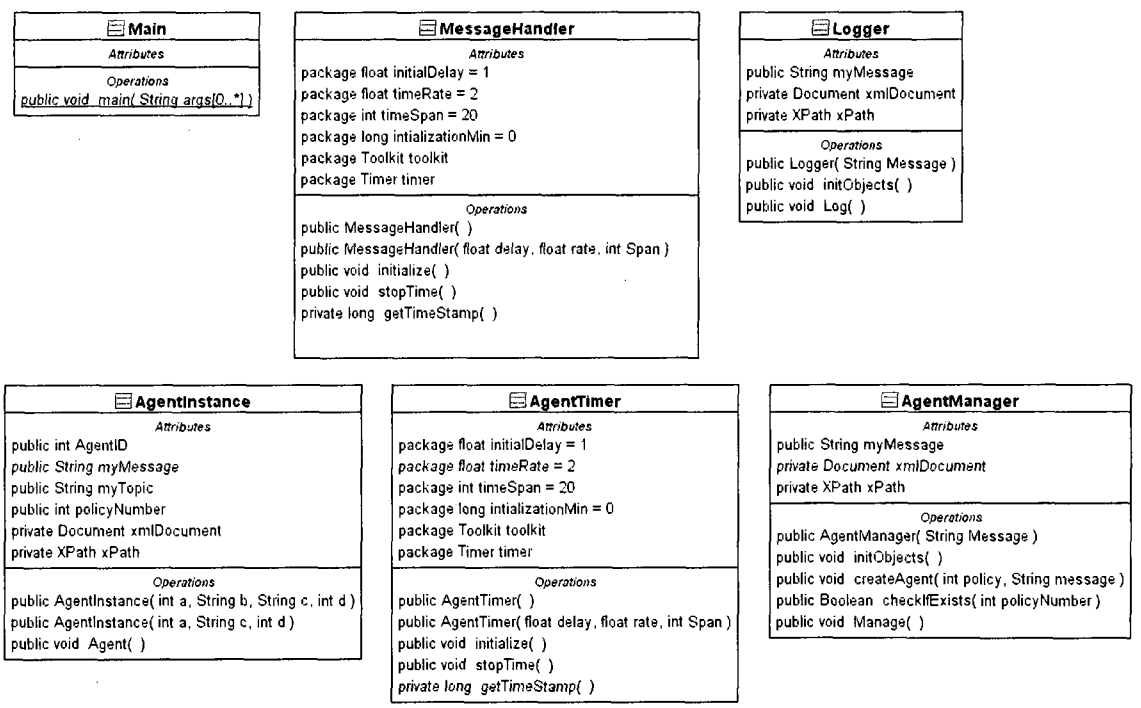

Figure 16 – Surveillance Portal Class Diagram

All event messages are represented in XML using an XML schema that corresponds to the common data model. All data elements required by the AgentInstance are extracted from the event message by the MessageHandler using XPATH queries. All Alerts are represented by event messages. When an Alert is needed, the AgentInstance object will create the appropriate message object and forward it to the MessageHandler.

The MessageHandler will convert the message object into an XML document that is dropped into an outbox.

In summary, we were able to achieve a reasonable implementation using messages for alerts, as well as for data. Policies were implemented in our own custom-built agent framework that was optimized to execute the policy language we defined in our database schema.

Chapter 5. Evaluation

The results of our case study were used to evaluate our thesis contributions along a number of different dimensions. In sections 5.1 and 5.2 we compare the conceptual framework of the Surveillance Portal against the conceptual framework of the PAL-IS Web Portal. The comparison is done in line with the requirements set out in section 3.2. In section 5.3 we compare the engineering effort required for each approach. Finally, in section 5.4 we compare the technology used to implement our Policy-based Compliance Monitor against other possible approaches, which include JADE, XACML, and JRules.

5.1. B2B Data Integration

The requirements for B2B data integration were outlined in section 3.2.1. Table 1 summarizes our evaluation in comparing the PAL-IS Web Portal with our Continuous Compliance Monitor as it relates to B2B data integration.

5.1.1 Consistent Data Model

Both the PAL-IS Web Portal and the Continuous Compliance Monitor have a well defined centralized data model. The difference between the two arises when a change to the data model is required. A change in the Continuous Compliance Monitor's data model would require creating a new event and then altering any policy definition's associated with the new event. A change to the PAL-IS data model would require a new version of the system. The new version would feature the new data model and new interfaces. This is because a change to the data model would render the user interfaces inaccurate. It is the PAL-IS Web Portal's strictly defined user interfaces that limit its

ability to change on the fly. The Continuous Compliance Monitor's user interfaces are independent of the data model. Ultimately the difference between the two approaches is the added effort required to create a new version of the system. The Continuous Compliance Monitor does not require a lot of effort to accommodate the changes, whereas the PAL-IS requires extra effort to make the version changes.

Table 1 – Comparison based on B2B data integration

Feature	PAL-IS Web Portal	Continuous Compliance Monitor
Consistent Data Model(Common vocabulary)	Yes, but data from source providers converted into predefined, single use tables.	Yes, well defined central data model with standard access mechanism.
Extensible Data Model	No, centrally defined tables require version changes for modification	Yes, can change freely with new tables (topics) added at any time by publishers
Message Timestamps	Consistent, but irrelevant because data is not real time	Broker and publisher timestamps gives consistency
Consistent IDs	Possible, part of the ETL process, and organization ID policy	Possible, includes Message Broker registration, and management of publishing practice
Data Entry	ETL and double entry(multiple versions of data)	Single stream of source events
Data Distribution	ETL + Manual (more rigid, predefined, batch style)	Event Driven Publish/Subscribe (more fine grained, extensible and real-time)

5.1.2 Extensible Data Model

The Continuous Compliance Monitor offers an extensible data model because of its event-based nature. When an addition needs to be made to the data model, a message can be generated by an event producer that would facilitate the change. This is done via the new event creation message. The Message Broker will receive this message, verify that it is an allowed change, and then add the new event, and its attributes, to the list of

available events. These additions can be made at any time. The Surveillance Portal would receive a new event creation message from the Message Broker which would allow it to create the appropriate table within the MSG database. This allows the message broker to capture all data, no matter when it is defined.

In contrast, the PAL-IS Web Portal does not offer runtime extensibility of its data model. The data model is defined at design time; therefore any additions to the data model would require a new version of the system before the addition could be made. Again, this is because the PAL-IS Web Portal's user interfaces are hard coded and need to be edited to account for new data sources.

5.1.3 Message Timestamps

Both the PAL-IS Web Portal and the Continuous Compliance Monitor offer message timestamps. The PAL-IS timestamps are however irrelevant because the data being input into the system is not done so on a real-time basis. This means that the timestamps will not be accurate to the time that the data was generated and thus will not be as useful for performance management or audit trail purposes. The timestamp will simply indicate when the data was input into the system.

The Continuous Compliance Monitor's timestamps are much more useful. These timestamps will indicate when the actual event occurred and thus will be useful for performance management and audit trail purposes. The timestamps are supplied by both the Message Broker and the publishers in order to provide consistency. The data is delivered to the Surveillance Portal continuously at real-time without any

transformations. Therefore, the timestamps captured by the Surveillance Portal will be a direct indication of the time that the data was produced.

5.1.4 Consistent IDs

Consistent IDs are possible within both the Continuous Compliance Monitor and the PAL-IS Web Portal. When data was entered into the PAL-IS Web Portal it would contain some IDs associated with the data, as well as the ID of the publisher. The IDs used would be determined by the data entry person. Which IDs used would be determined through an ID policy within the organization. This policy would indicate what IDs should be used for what entities. Additionally, IDs would be managed within the ETL process. During the translation phase, IDs would need to be checked and modified to fully support a consistent set of IDs.

Consistent IDs within the Continuous Compliance Monitor could be achieved through data publishing practices and use of the Message Broker. Publishers of data are registered with the Message Broker and are registered with a unique ID. All data published is labeled with the ID of the publisher. Additionally, a data publishing policy within the organization would indicate how publishers would use IDs within the data being published.

5.1.5 Data Entry

The Continuous Compliance Monitor offers data entry by way of publishing a stream of event messages to the Message Broker. Data is entered a single time and is captured by the Surveillance Portal. There is only one version of the data. The PAL-IS

Web Portal captures its data through an ETL process from multiple operational databases. The operational databases are maintained locally by different organizations and by different individuals. The data input process for each database is by way of human input. Each organization may be capturing similar data within its own operation databases. Since the databases may contain duplicate data, there is a problem of double entry. Double entry is when the same data is entered more than one time. This requires the time and effort of multiple individuals. Time and effort can be saved by reducing, or eliminating double entry.

5.1.6 Data Distribution

By data distribution we mean the methods and processes by which data is distributed amongst the B2B network. Data within a Web Portal environment is manually input into multiple local databases within each organization, and is then transferred to the data warehouse via an ETL process. Data is taken from the data warehouse through the Web Portal interface. This ETL process is a more rigid, predefined, batch style type of process. Data distribution, from a user's perspective, is in a pull style, where all data that is needed is pulled from the Web Portal.

The Continuous Compliance Monitor distributes data through an event-driven publish/subscribe interaction. This process is finer grained than the ETL, it is extensible and it occurs in real time. Additionally, users can access data through the Business Intelligence Portal. This portal is similar to the Web Portal's data warehouse, except that data is collected through the publish/subscribe mechanism. Data distribution, from the

user's perspective, is both push and pull. This is to say that data is pushed to the user, and the user is able to go and pull data from the Business Intelligence Portal.

5.1.7 Summary of Results

In summary, the compliance monitor framework is more dynamic and flexible because of its well defined, extensible data model. It provides a useful timestamp capability and only requires participants to enter data a single time. The PAL-IS portal is rigid and hard to adapt because of its predefined data model and because a new version must be created in order to alter or extend the data model. It requires users to enter data multiple times and does not have a useful timestamp feature. Both frameworks provide consistent IDs through organizational practices.

5.2. Policy Monitoring

The requirements with respect to policy monitoring were outlined in section 3.2.2. Table 2 summarizes our evaluation of the PAL-IS Web Portal with the Continuous Compliance Monitor in regards to policy monitoring.

Table 2 – Comparison based on policy monitoring

Feature	PAL-IS Web Portal	Continuous Compliance Monitor
Policy Language	Natural Language Documents	Natural Language documents translated into machine processable ECA policies
Alerting	Manual, awkward, not real time	Yes, real-time
Performance Management	Yes, but data may not be up to date	Yes
Audit Trail	Manual, time delays, not policy-based	Policy-by-Policy integrated and automatic event log

5.2.1 Policy Language

In order to monitor policies, there needs to be a policy language in place to represent policies. Table 3 summarizes the comparison between the PAL-IS and the Continuous Compliance Monitor with respect to the policy language.

Table 3 – Comparison based on Policy Language

Feature	PAL-IS Web Portal	Continuous Compliance Monitor
Explicitly Represented	Yes, natural language documents	Yes, natural language documents, ECA Policies within the Portal
Machine Processable	No	Yes
Ease of Expression	NA	Easy, XML or database tables (ECA)

Policies are not explicitly written within the PAL-IS Web Portal, but there are sets of natural language documents that indicate the policies that are to be monitored. In contrast, the Continuous Compliance Monitor has a policy language which allows a policy officer to explicitly represent policies. The explicit representation of policies within the Continuous Compliance Monitor is a step further beyond the natural language documents that exist. The Continuous Compliance Monitor's policy language is machine processable, whereas the natural language documents used for the PAL-IS Web Portal are not machine processable. Since policies can be expressed in either XML or as data in database tables and because events and their attributes are strictly defined, it is relatively easy for a human to express the policies. The ease of expression is also enhanced by the

fact that the policies are written in the form of ECA rules. ECA rules are easy to understand and easy to create.

5.2.2 Alerting

In order to inform B2B network participants of impending compliance issues, or of a failure to meet compliance, an alerting mechanism is needed. The Continuous Compliance Monitor offers the ability to generate alerts. Alerts can be generated in response to events where certain conditions are met. Alerts may be generated at any time to inform network participants of compliance related issues. Alerts are generated in real time and are made available to subscribers in real time. Alerts are defined within the ECA policy for each specific regulation.

Alerting mechanisms within the PAL-IS system are not immediately obvious. Since there is no specific policy language (natural language documents), and no specific monitoring approach, alerting is only possible in an ad-hoc fashion. Alerts would need to be distributed by manual approaches. Sending an e-mail, making a phone call or actually speaking to other people are the approaches to alerting. There is potential to build alerting mechanisms into the Web Portal, but ultimately alerting is an awkward process within the PAL-IS.

5.2.3 Performance Management

Performance Management is possible within both systems. PAL-IS suffers from data input delay and thus the Performance Management results will not be fully up to date with the current state of data. Both systems require human interaction to develop the data

model for the Performance Management process. This may be slightly easier for the PAL-IS because the data is captured during an ETL process where the data can be manipulated and transformed into a usable format. Since the data model is not created during the design phase, work is required to manipulate the data within the Continuous Compliance Monitor into an acceptable format for Performance Management purposes.

5.2.4 Audit Trail

An audit trail is possible for both systems. Within the Continuous Compliance Monitor an audit trail is provided in a real time policy by policy event log format. It is possible to view the exact series of events, including alerts, which occur for a specific policy, in real time. An audit trail for the PAL-IS system would require manual effort, would be delayed, and would not be based on policies. In addition the audit trail would not capture the true series of events because the data is not input in real time.

5.2.5 Summary of results

In summary, the Continuous Compliance Monitor is better because PAL-IS has no mechanisms for monitoring policies, no true alerting functionality, and only represents policies in their original natural language documents. It can only duplicate the support for performance management reporting and an audit trail if sufficient data is entered or ETLed into the central data warehouse. Data is only made available when the users choose to make it available. The Continuous Compliance Monitor offers real-time alerting, full audit trail support, and has a machine processable ECA policy language.

5.3. Evaluation of Engineering Effort

Table 4 provides a comparison between the PAL-IS and the Surveillance Portal with respect to the Engineering Effort required for each system.

Table 4 – Evaluation of Engineering Effort

Feature	PAL-IS Web Portal	Continuous Compliance Monitor
Development efforts	Programming is simple, but requires determining ETL process, policies and data model definition prior to deployment	More complicated programming, effort needed for automated publishing of data, policy language definition needed
Deployment Efforts	Connect BI tool to data source, train users on web access	Connect BI tool to data source, train users on policy language and data input, provide local publishing capabilities for all users
Data Publishing and Distribution	Multiple data entry (Double entry)	Single data input
Maintenance Efforts	Requires change to ETL, web interface, and data structure when there is a data model change. Data cleaning takes effort	Adding agents, events, etc. is straight forward. Changing policy language will require effort to update agent definitions.
Competencies of Developers	Moderate, ETL data cleaners required	Moderate, policy experts required

The PAL-IS Web Portal requires engineering to create a web-accessible data warehouse that collects data, via an ETL process, from multiple independent databases. The effort is not only in the development of the databases, the data warehouse and the web portal, but also in determining up front all of the policies, data, ETL process, and manual processes required to gather the data. Effort needs to be put in to transform the data during the ETL process, which includes data cleaning. In addition there is a large

overhead for maintenance because any change made to the data, the interfaces, or the processes requires a recompilation of the system. There is also extra effort required because of the double entry of data that will occur.

The Continuous Compliance Monitor on the other hand has a higher overhead for developing the publish/subscribe infrastructure and automated policy monitoring. There is much more time and effort spent on developing the actual system. However, once the system has been developed, there is little ongoing maintenance required. Policies and data do not need to be identified prior to implementation. Effort is required to deploy the publish/subscribe interfaces to users computers. The level of skill of the developer is not much greater than the level of skill needed for the PAL-IS Web Portal.

The main difference between the two is the extra development time for the Continuous Compliance Monitor vs. the extra effort and time for maintenance of the PAL-IS Web Portal. Additionally, the PAL-IS Web Portal requires that all policies and data be discovered up front or the system will need to be recompiled. The Continuous Compliance Monitor has the advantage here because all data and policies can be added at any time after the system is started. Additionally, the Continuous Compliance Monitor does require a bit more effort to deploy.

5.4. Policy-based Compliance Monitor

In this section we compare our Policy-based Compliance Monitor (description found in section 3.5) against three other possible implementation approaches. Table 5 summarizes the comparison between the different possible approaches for implementing the Policy-based Compliance Monitor.

Table 5 – Comparison between Possible Policy-based Compliance Monitor Approaches

Feature	XACML	JADE	JRules	Policy-based Compliance Monitor
Policy Language	Could be used as the Policy Language, has issues with state and timing, requires small changes	Could read and write database tables in the same way	JRules rule language could be used to represent policies	XML or Database ECA Rules
Data Model Integration	Data model would be defined by XACML Attributes	Possible, requires coding database access or XML parsing	Some effort, requires to map to JRULES data definitions	Fully Integrated as is
Timer Functionality	XACML does not handle timers	Yes	No, but could use wrappers	Yes
Tool Support	Yes, policy editors, policy verification engines, evaluation engines	Yes, Java (IDEs), not for policy language	Yes, both Java and JRules (Integrated tool support)	Yes, limited to Java and Database (Integrated), not for the policy language

5.4.1 Policy-based Compliance Monitor

Here we discuss the notable features of the Policy-based Compliance Monitor. For full design and implementation details, please see section 3.4.

The Policy-based Compliance Monitor is fully integrated with the event-based publish/subscribe framework and is able to receive events at a real time pace and respond to events with alerts, in the form of event messages, directly to the publish/subscribe interface. The Policy-based Compliance Monitor is fully integrated with the B2B data

model. The additions of new elements within the B2B data model are atomically integrated within the Policy-based Compliance Monitor.

The Policy-based Compliance Monitor will only ever be recompiled if a change is made to the policy definition language. In such a case the programming of the Policy-based Compliance Monitor itself may need to be altered to facilitate the use of the new policy language features. Currently, creating a new alert message will require a recompilation of the Policy-based Compliance Monitor, but this could be resolved with some effort. Additions to the B2B data model do not require any changes to the Policy-based Compliance Monitor. Adding new policies can be done on the fly and does not require any changes to the Policy-based Compliance Monitor, but rather simply additional database entries.

The Policy-based Compliance Monitor supports a policy language in the form of ECA rules. Events are selected from the list of available events, conditions are created based on the attributes of the chosen events and specific operators, and finally the actions are selected from a predefined set of actions which includes generating predefined alerts, requesting or canceling timer events, or simply ending the agent's life.

Tool support is limited to generic tools for programming and database design. For example, tools such as Eclipse are available to support, among other things, editing and debugging Java code.

5.4.2 XACML

XACML (eXtensible Access Control Markup Language) is an OASIS standard for describing access control policies. It is based on the XML language. A XACML policy describes a set of rules and policies that are combined to create a policy set that applies to a particular decision request [OASIS03]. XACML could be considered as an option to replace the policy language that we used.

Since XACML is an access control language and we are not performing access control, we would not be using XACML for its intended purpose. Since XACML uses attributes to represent the objects within the environment of concern, all event related data (the data model) would need to be represented as attributes. For instance, a ResourceAttribute would represent any resources, and an ActionAttribute would represent any actions. This requires some effort and it is not immediately obvious how all events could be represented.

The ECA policies could be translated directly to XACML policies. Events would be represented as XACML Targets. A Target expresses a simple predicate over a set of attributes. The attributes are the elements of the data model. A condition can be represented as a XACML Condition. Finally, the Actions can be represented as XACML Obligations. Obligations consist of instructions with a set of parameters.

However, there are some areas where XACML would need small or even insignificant moderations. For instance, the conclusion of a XACML policy is a Permit or Deny statement. This is not appropriate terminology for an ECA rule, but it could be ignored by the developer.

An advantage for XACML policies is that they are written in file format. Files are faster to read than a database, are easy to alter and are easy to create. XACML has existed for a number of years and thus has had time to develop and become mature. For this reason tool support for XACML is available. The policy language we use does not have any tool support.

Ultimately the effort required to use XACML as a non access control policy language is likely greater than the effort to create a new policy language. We would need to define the entire data model within XACML, and it is not immediately obvious how straight forward this is. XACML is a complicated language and thus a XACML expert would be required to create the data model and the XACML policies. Additionally, XACML policies do not have timing capabilities. Although it is possible to use XACML, it is not an ideal option.

5.4.3 JADE

JADE (Java Agent Development Environment) is a platform built on top of the Java programming language for developing multi-agent systems, where multiple agents can interact and be distributed across many computer systems [TIG09]. JADE is compliant with the Foundation for Intelligent Physical Agents (FIPA) specifications [FIPA09], which are a collection of standards that "promote the interoperation of heterogeneous agents and the services that they can represent" [FIPA09]. In addition, Jade provides a Graphical User Interface for managing a set of agents, and provides multiple debugging features. These debugging features include a sniffer for monitoring

messages passed between agents, as well as a dummy agent that can be used for message testing.

A number of points were considered when considering the use of JADE for the implementation of the Policy-based Compliance Monitor. Firstly, in order to use JADE, not only would an understanding of Java be required, but also an understanding of the JADE environment. Secondly, each Agent in the JADE environment must be specifically coded. This is in contrast to the Policy-based Compliance Monitor where we have only created one generic Agent that can represent all Agents. The effort and time required to create multiple agents would be much larger than the time and effort required for one generic agent. Finally, although JADE provides debugging features, these features are not ideal for the low level debugging required for such a system. The built in debugging of the Policy-based Compliance Monitor proved to be extremely useful.

Overall we need to consider that we were creating a Policy-based Compliance Monitor with a single generic agent, and not multiple Agents. In this case, the method chosen is ideal for debugging, stat collection, time requirements, and the effort level required. Jade is a much more advanced, productized option for creating agents. Jade would offer better scaling abilities. In an ideal situation, JADE would be the optimal choice for the implementation of the Agents.

5.4.4 JRules

JRules [ILOG09] and other Rule Execution engines provide a mechanism for executing a set of business rules based on a set of input values. The JRules platform allows users to write a set of rules in the ILOG rule language. The JRules engine would

be presented with a set of input data; it would analyze this data with respect to the set of ILOG rules, and then output a conclusion. These rules mainly consist of If-Then statements, and are based on a data model defined in a set of Java classes. A user-interface is provided by JRules for creating and editing rules.

When considering JRules for the implementation of the Policy-based Compliance Monitor the following points were considered. JRules does not provide a mechanism for timers. In addition, the JRules engine does not provide a mechanism for capturing state information. Since the data model is defined by Java classes, a change in the data model would require a Java developer to edit the set of Java classes. In addition, a change to the data model would require a recompilation. JRules has a number of tools to support its use. Finally, JRules comes with its own expression engine. This expression engine is far more advanced than the expression engine we created. It would save time and effort to use this engine.

The timer functionality, as well as the capturing of state information, could be implemented by creating wrappers around the JRules engine. Any input data presented to the JRules engine would initially be interpreted by the wrappers in order to determine timer and state information. Additionally, the conclusions of the JRules engine would be fed through the wrappers. The wrappers would not only be responsible for monitoring incoming and outgoing data, but for providing the engine with the timer and state data when timing events occurred. If this method were chosen, the definition of the rules would need to take into consideration the timer data from the wrappers.

The use of JRules for the Policy Monitoring Engine is a possibility. Separating timing and state from the rules themselves would be somewhat more complicated than including state in the policy definition. In addition, data model changes require a developer whereas data model changes to the Policy-based Compliance Monitor don't necessitate the need for a developer. Ultimately, the overall level of effort for both options is similar, but JRules would have been a good choice because of its prebuilt expression engine.

Chapter 6. Conclusions

6.1. Summary of Contributions

In section 1.2 we outlined the key contributions of this research. Here we discuss the impact of this work.

Contribution 1: A set of requirements for continuous compliance monitoring of B2B processes.

In section 3.2 we outlined the requirements that a continuous compliance monitor needed in order to monitor the processes within a B2B network for compliance. The two major areas of requirements were B2B data integration and policy monitoring. By identifying these requirements we were able to better understand and conceptualize a possible framework for continuous compliance monitoring. It also provided a systematic framework for evaluation and comparison of our approach with other approaches.

Contribution 2: A framework for monitoring compliance in which government regulations and legislation, organizational policies, and contracts are expressed as ECA policies and monitored based on a data stream audit trail of messages within a B2B network.

A framework for continuous compliance monitoring was proposed in section 3.3. Within the framework, government regulations and legislation, organizational policies and contracts are translated from natural language documents into machine processable ECA policies. All data within the B2B network is input as a stream of event data that can

be captured by a monitoring device. The monitoring device examines the incoming event data and analyses the ECA policies to determine if compliance is being met. This requires someone who has an understanding of the regulations in place and of the ECA policy language. This also requires that all participating members be aware of the data publishing standards within the B2B network.

Contribution 3: An architecture that supports the framework which includes a publish/subscribe message broker, a common data model and event registry, and a surveillance portal that provides both performance management reporting, and policy-driven compliance monitoring and alerting.

An architecture to support the framework proposed in contribution 2 was proposed in section 3.4. This architecture consists of a Message Broker, a common data model and event registry and a Surveillance Portal. This architecture was successfully implemented. The Message Broker provides a data integration mechanism for all members of the B2B network. Publishers may publish data, and this data will be distributed to all those members that have subscribed to it. The Message Broker also provide event and entity registration. The Surveillance Portal, which is subscribed to all messages, receives all event registration information. This adds a dynamic nature to the Surveillance Portal because it will always be fully aware of any new event types. The Surveillance Portal can then capture all messages. The Surveillance Portal provides compliance monitoring abilities by way of the policy-based compliance monitor. ECA policies are provided to the Surveillance Portal and are monitored based on incoming event data and time information. Additionally, the Surveillance Portal can publish alerts,

provide performance management reporting, and allow a policy officer to add additional ECA policies whenever they are required without having to reboot the system.

Contribution 4: A policy-based architecture for compliance monitoring and alerting, including a survey and comparison of rule and agent based technologies for implementing the architecture.

In section 3.5 we describe our policy-based compliance monitor. In section 5.4 we provide a comparison between different rule and agent based technologies that could have been used to implement the policy-based compliance monitor. It consists of an Agent Manager, an Agent Timer, Agent Instances and a set of databases for storing timer event data, policies, and agent state. The monitor was implemented. We compared XACML, JADE, and JRules against our monitor. Our comparison concluded that there were some better and some worst options available. The better options however would have required more time than was available, so the approach we took was deemed acceptable and successful.

6.2. Thesis Limitations

The framework we presented is suitable for domains that have a certain set of characteristics. Within the domain, we must be able to represent the data as a set of events in the form of attribute value pairs. We must be able to represent government regulations, organizational policies and customer contracts as actionable policies that relate directly to the events within the domain.

The requirements and framework we derived were based on one case study. If additional cases were considered, there is potential for additional requirements and framework components to evolve.

6.3. Future Work

6.3.1 Federated Identity Management

In order to manage IDs within the B2B network a Federated Identity Management system should be implemented. The system would allow users across all organizations to maintain local and global identities while still being able to access services across the entire network. An identity provider would manage all local IDs and provide global IDs across the network. Such a system would provide security and privacy by hiding user's identities through the use of pseudonyms and requiring user authorization when any data is requested. In the Palliative Care scenario, this would allow each hospital to provide their patients with their own IDs without needing to worry about other hospital IDs. This is because all of the local IDs will be registered with the identity provider. The identity provider would then allocate a global ID, and the system would be able to transform local IDs into global IDs when interacting across the B2B network.

6.3.2 Interface for Policies

The ECA policies that are used in this framework are either written in XML, or inserted directly into a database. Both of these approaches are not ideal. XML is tedious for a human to work with and directly entering data into database tables is inefficient and complex. A specifically designed interface for creating ECA policies would be a useful addition to this work. The interface could be connected directly with the Policy database,

but provide a more efficient approach to creating ECA policies. A specially designed interface would provide a simplified approach in comparison to entering the policies directly into a set of database tables. In particular the interface could be connected directly with the event data model. This would allow users to select the specific events, conditions and actions from lists located within the interface. Once a user has selected a specific event, the interface would limit the conditions to those attributes that are associated with the event. This would help limit the number of mistakes made while creating ECA policies.

References

[Aalst03] Aalst, W. M. P., Hofstede, A., Weske, M.: Business Process Management: A Survey, In: Proceedings of the International Conference on BPM. Eindhoven, Netherlands. (2003)

[Anyanwu03] Anyanwu, K., Sheth, A., Cardosa, J., Miller, J., Kochut, K.: Healthcare enterprise process development and integration. In: J. Res. Practice Inf. Technol. Sp. Iss. Health Knowl. Manag, 35(2). pp. 83--98. (2003)

[Bailey02] Bailey, J., Poulovassilis, A., Wood, P.: An Event-Condition-Action Language for XML. In: Proceedings of the 11th international conference on the World Wide Web, pp. 486--495. Honolulu, USA. (2002)

[Ahn05] Ahn, G-J., Lam, J.: Managing Privacy Preferences for Federated Identity Management. In: Proceedings of the 2005 workshop on Digital Identity Management. pp. 28--36. (2005)

[Boglaev05] Boglaev, Y.: Interchange of ECA Rules with Xpath expressions. Paper presented at the W3C Workshop on Rule Languages for Interoperability. (2005)

[Breyfogle99] Breyfogle, F.W.: Implementing Six Sigma: Smarter Solutions Using Statistical Methods. Wiley, NY. (1999)

[Carzaniga01] Carzaniga, A., Rosenblum, D., Wolf, A.: Design and Evaluation of a Wide-area Event Notification Service. In: ACM Trans. Comput. Syst., 19(3), pp. 332--383. (2001)

[Chaudhuri97] Chaudhuri, S., Umershwar, D.: An Overview of Data Warehousing and OLAP Technology. ACM SIGMOD Record, 26(1), pp. 65--74 (1997)

[Chen05] Chen, X., Zhang, J., Wu, D., Han, R.: HIPPA's compliant Auditing System for Medical Imaging System. In: Proceedings of the 27th International Conference of the Engineering in Medicine and Biology Society, pp. 562--563. Shanghai (2005)

[Dresner02] Dresner, H.: Business Activity Monitoring: 'New Age' BI?. Gartner Research LE-15-8377 (2002)

[Dresner03] Dresner, H.: Business Activity Monitoring. Gartner Symposium, Cannes, France. (2003)

[Eugster03] Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.-M.: The Many Faces of Publish/Subscribe.In: ACM Comput. Surv., 35(2), pp. 114--131. (2003)

[Eze09] Eze, B.: An Integrated Trusted Processes Framework for Consumer-facing B2B networks. M.Sc. thesis, University of Ottawa, Canada. (2009)

[FIPA09] Foundation for Intelligent Physical Agents.: http://www.fipa.org/, last retrieved: Apr 28, 2009, (2009)

[Frolick06] Frolick, M., Ariyachandra, T.: Business Performance Management: One Truth.In: Journal of Information Systems Management, 23(1), pp. 41—48. (2006)

[Ghanavati07] Ghanavati, S.: A Compliance Framework for Business Processes Based on URN. M.Sc. thesis, University of Ottawa, Canada. (2007)

[Google09] Google Health. www.goggle.com/health, last retrieved July 31, 2009 (2009)

[Hammer94] Hammer, M., Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. HarperBusiness. (1994)

[HealthVault09] Microsoft Healthvault. www.healthvault.com, last retrieved July 31, 2009 (2009)

[Hoot04] Hoot, N., Weiss, J., Giuse, D., et al.: Integrating Communication Tools into and Electronic Health Record. Medinfo. (2004)

[Hu09] Hu, J., Peyton, L.: A Framework for Privacy Assurance and Ubiquitous Knowledge Discovery in Health 2.0 Data Mashups. In: Ubiquitous Health and Medical Informatics: The Ubiquity 2.0 Trend and Beyond. S. Mohammed, J. Fiaidhi, (Eds.), IGI Global, Hershey, PA, USA (2009)

[IBM09] www-01.ibm.com/software/data/cognos/, last retrieved: July 31, 2009 (2009)

[ILOG09] www.ilog.com, last retrieved: July 31, 2009 (2009)

[Inmon92] Inmon, W.H.: Building the Data Warehouse. John Wiley. (1992)

[Jin02] Jin, L., Machiraju, V., Sahai, A.: Analysis on Service Level Agreement of Web Services. HP Labs, Tech. Rep. HPL-2002-180, (2002)

[Kaplan93] Kaplan, R., Norton, D.: Putting the Balanced ScoreCard to Work. In: The Harvard Business Review. pp. 134--147. (1993)

[Keller03] Keller, A., Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web services. In: J. of Net. and Syst. Mng. 11(1), pp. 57--81. (2003)

[Kuziemsky08] Kuziemsky, C., Weber-Jahnke, J., Lau, F., & Downing, M.: An Interdisciplinary Computer-based Information Tool for Palliative Severe Pain Management. In: Journal of the American Medical Informatics Association, 15(3), pp. 375--382. (2008)

[Lancioni00] Lancioni, R.A., Smith, M.F., Oliva, T.A.: The role of the Internet in supply chain management. Industrial Marketing Research. 29(1). pp. 54--65. (2000)

[Landau03] Landau, S.: Liberty ID-WSF and Privacy Overview, version 1.0, Liberty Alliance Project. http://projectlibrary.org/liberty/resource_center/papers, last retrieved July 31, 2009. (2003)

[Lee03] Lee, S.C., Pak, B.Y., Lee, H.G.: Business value of b2b electronic commerce: the critical role of inter-firm collaboration. In: Electronic Commerce Research and Applications. 2(4). pp. 350--361. (2003)

[Li05] Li, G., Jacobsen, H.-A.: Composite Subscriptions in Content-Based Publish/Subscribe Systems. In: Proceedings of ACM/IFIP/USENIX 6th International Middleware Conference. pp. 249--269. Grenoble, France. (2005)

[Lucking-Reilly01] Lucking-Reilly, D., Spulber, D.: Business-to-Business Electronic Commerce. In: Journal of Electronic Perspectives. 15(1). pp. 55--68. (2001)

[Maes91] Maes, P.: The agent network architecture (ANA). SIGART Bulletin, 2(4), pp. 115--120. (1991)

[Middleton09] Middleton, G., Peyton, L., Kuziemsky, C., Eze, B.: A Framework for Continuous Compliance Monitoring of eHealth Processes. In: World Congress on Privacy, Security, Trust and Management of eBusiness, August, 2009. (2009)

[Milosevic02] Milosevic, Z.; Josang, A., Dimitrakos, T., Patton, M.A.: Discretionary Enforcement of Electronic Contracts. In: Proceedings of the 6th International Conference on Enterprise Distributed Object Computing. pp. 39--50. (2002)

[Niblett05] Niblett, P., Graham, S.: Events and service-oriented architecture: The OASIS Web Services Notification Specifications. In: IBM Systems Journal. 44(4). (2005)

[OASIS03] OASIS.: eXtensible Access Control Markup Language (XACML) OASIS Standard. http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf, last retrieved: July 31, 2009 (2003)

[Papazoglou03] Papazoglou, M.P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: Proceedings of the 4th International Conference on Web Information Systems Engineering. pp. 3--12 (2003)

[Peyton07a] Peyton, L., Hu, J.: A Service-oriented Architecture for Managing Privacy Compliance in Collaborative Environments. In: International Journal of Business Process Integration and Management. 2(4), pp. 292—301 (2007)

[Peyton07b] Peyton, L., Hu, J., Doshi, C., Seguin, P.: Addressing Privacy in a Federated Identity Management Network for EHealth. In: Eight World Congress on the Management of eBusiness. pp. 12--18 (2007)

[Peyton08] Peyton, L., Zhan, B., StEPSIen, B.: A Case Study in Integrated Quality Assurance for Performance Management Systems. In: The 6th International Workshop on Modeling, Simulation, Verification and Validation of Enterprise Information Systems. Barcelona, Spain (2008)

[Pourshahid08] Pourshahid, A., Chen, P., Amyot, D., Forster, A.J., Ghanavati, S., Peyton, L., and Weiss, M.: Toward an integrated User Requirements Notation framework and tool for Business Process Management. In: The 3rd International MCeTech Conference on eTechnologies, pp. 3--15. Montréal (2008)

[Rodosek01] Rodosek, G., Lewis, L.: Dynamic Service Provisioning: A User-centric Approach. In: Proceedings of the 12th Annual IFIP/IEEE International Workshop on Distributed Systems: Operations & Management. Nancy, France, pp. 37--48. (2001)

[Ross-Talbot04] Ross-Talbot, S., Tabet, S., Chakravarthy, S., Brown, G.: A Generalized RuleML-Based Declarative Policy Specification Language for Web Services. Paper presented at the W3C Workshop on Constraints and Capabilities for Web Services. (2004)

[Sarbox02] Sarbanes-Oxley Act of 2002. Retrieved from www.soxlaw.com (2002)

[Shankar05] Shankar, C., Ranganathan, A., Campbell, R.: An ECA-P Policy-Based Framework for Managing Ubiquitous Computing Environments. In: Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services. pp. 33--42. (2005)

[Shen04] Shen, Y., Lam, T. C., Liu1, J.-C., Zhao, W.: On the Confidential Auditing of Distributed Computing Systems. In: Proceedings of the 24th International Conference on Distributed Computing Systems, pp. 600--607. (2004).

[TIG09] Telecom Italia Group.: http://jade.tilab.com, last retrieved: July 31, 2009

[Verma99] Verma, D..: Service Level Agreements on IP Networks. Macmillan Technical Publishing. (1999)

[Weske04] Weske, M., Aalst, W. M. P., Verbeek, H. M. W.: Advances in business process management. In: Journal of Data & Knowledge Engineering, 50(1), pp. 1--8. (2004)

[WHO09] World Health Organization.: http://www.who.int/cancer/palliative/definition/en/, last retrieved: July 31, 2009

[Wooldridge95] Wooldridge, M., Jennings, N.: Intelligent Agents Theory and Practice. In: Knowledge Engineering Review, 10(2). (1995)

[Yupho06] Yupho, D., Kabara, J.: Continuous vs. Event Driven Routing Protocols for WSNs in Healthcare Environments. In: Pervasive Health Conference and Workshops. (2006)

Appendix A

Example XML Policy Definition for the *Admit Assessment* policy

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
   <id>1</id>
   <name>Admit_Assessment</name>
   <description>Once a patient is admitted into the system a Pain Assessment must be
administered within 24 hours.</description>
   <event>
      <id>1</id>
      <name>PatientAdmit</name>
      <isTrigger>yes</isTrigger>
      <action>
         <type>GenerateTimerEvent</type>
         <value>1</value>
      </action>
   </event>
   <event>
      <id>2</id>
      <name>PainAssessment</name>
      <isTrigger>no</isTrigger>
      <condition>
         <attribute>PatientiID</attribute>
         <value>PatientiID</value>
         <operator>=</operator>
         <TypeOfAttribute>message</TypeOfAttribute>
         <TypeOfVariable>variable</TypeOfVariable>
      </condition>
      <action>
         <type>CancelTimerEvent</type>
      </action>
      <action>
         <type>Exit</type>
      </action>
   </event>
   <event>
      <id>3</id>
      <name>Timer</name>
      <isTrigger>no</isTrigger>
      <action>
         <type>GenerateMessage</type>
         <value>AdmitAssessmentFailure</value>
      </action>
      <action>
         <type>GenerateMessage</type>
         <value>AdmitAssessmentFinished</value>
      </action>
      <action>
         <type>Exit</type>
      </action>
   </event>
</policy>
```