12

A Framework for Policy-Based Data Integration in Palliative Health Care

Benjamin Eze
University of Ottawa, Canada

1. Introduction

Electronic Health (e-Health) processes are data-focused, event-driven, and dynamic. Palliative Health care presents a challenge of caring for patients from home by doctors, nurses and other patient providers that operate in domains and organizations that do not share common collaboration technologies. Integrating data across these providers is often difficult or impossible because of incompatible applications, as well as security and privacy concerns. In addition, these exchanges need to be systematically monitored for compliance with legislation, privacy, accreditation guidelines, and quality of care protocols.

To address these specific requirements for e-Health processes especially as it relates to Palliative healthcare, we extend traditional SOA infrastructure with policy-based processing of streaming event data based on a general publish/subscribe model in a business-to-business (B2B) healthcare domain. This entails the use of a policy-based message broker to execute subscription policies on streaming event messages. This approach provides great flexibility to the distribution and integration of data within a B2B network. Applying this framework to palliative healthcare provides a dynamic data sharing model for real-time monitoring of a patient health status, medical records, reports and other necessary information with healthcare providers operating in disparate domains.

2. Background

In palliative health care, patients are catered for from their homes and this brings with it added complexity in terms of who has the permissions to access patient medical records, under what conditions and what time span. Most importantly, there is also the need to have real-time monitoring of patient health status and the ability to provide timely and accurate interventions when emergencies occur. Using a common collaboration portal is often not appropriate. Portals provide a common port of call for information but fail to address the dynamic nature of the patient relationship as well as rules of association with their healthcare providers.

Recent initiatives advocate that healthcare should be delivered as integrated services that make data accessible within distributed processes across different contexts (Coiera and Hovenga, 2007). Community care, especially at-home care, usually requires the integration of care processes across several providers and organizations in a business-to-business (B2B) network.

We also recognize that effective and efficient collaboration for applications in a B2B depends on the ease of information flow between parties. Collaboration must cut across all the communications strata: software applications, databases, server processes, mobile devices and low level sensors (Foster et. al, 2002).

Service Oriented Architecture (SOA) (Huhns & Singh, 2005) has emerged as the standard framework for an extensible machine-to-machine interaction using the Internet as a veritable communication platform. SOA enables applications to exchange data across organization domains and firewall. As depicted in Figure 1, clients discover a service address from a registry and need to subsequently pool this service broker for data in a request/response style interaction. As a result many implementations of SOA entail unnecessary procedural interaction and data polling that could limit flexible data integration (Eze et al., 2010). This is particularly true with processes that run within a B2B network as opposed to a single enterprise (Doshi& Peyton, 2008).

Unlike SOA, event-driven systems (Niblet & Graham, 2005) are characterized by their ability to decouple service providers and consumers through messages (Etzion et. al, 2006) or data being exchanged. Event-driven Architectures (EDA) contextually decouple data producers and consumers only by their data exchanges and not through procedural calls (Eugster et. al, 2003).

The Publish/subscribe framework follows the EDA model but allows many data consumers to subscribe to data sources, and have event data sent to them as messages, as they become available. A combination of SOA with EDA publish/subscribe type interaction provides a more robust interaction. Service providers and consumers are totally decoupled by a central middleware, the message broker.

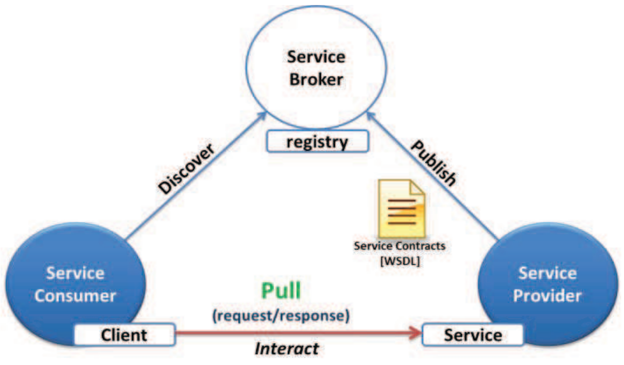


Fig. 1. Service Oriented Architecture Illustrated.

Data consumers indicate interest to data through subscriptions and data publishers only publish data to the broker as they become available. Clients do not poll data; they simply present an interface for a broker to push data to as they are made available from other publishers or data sources.

Figure 2 present a combined SOA and publish-subscribe style interaction. We can see that both SOA and event-driven architecture provide communication and data sharing flexibilities that are complimentary for efficient data sharing and enterprise collaboration. However, they still do not address fully the information management requirements of e-

Health monitoring processes. In particular, both frameworks lack support for a common data model across a B2B network to support data integration (Eze et al., 2010).

In an e-Health network that supports Palliative Healthcare, there may be many different types of data from many different organizational sources that must be integrated to provide a single consistent view of the information flow. Such complex data exchanges also need robust privacy (Peyton et. al, 2007) considerations and enforcement. As well, policy compliance is an issue with respect to privacy legislation, hospital guidelines, and procedures that dictate under what circumstances individuals can view data.

To support such flexibility, rules-based policies are used to govern system behaviour by providing reactive functionalities (Harrocks et. al, 2003). Executed through policy engines, rule-based policies are very adaptable to a wide range of applications, such as adapting composite services with constantly changing business processes. Rules languages in the form of event-condition-action tuples, when presented in XML (Bailey and Wood, 2003), become powerful tools for expressing both attribute and role-based access control policies. eXtensible Access Control Markup Language (XACML) (OASIS eXtensible Access Control Markup Language [OASIS], 2011)is a standardized common security-policy language that "allows the enterprise to manage the enforcement of all the elements of its security policy in all the components of its information systems".

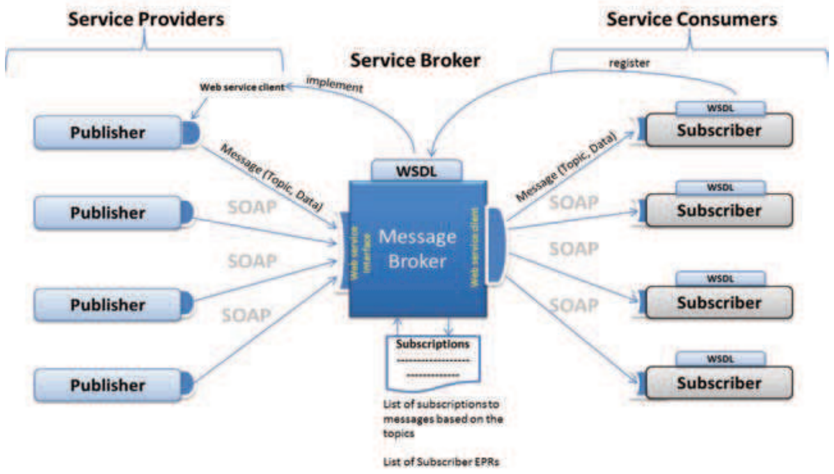


Fig. 2. SOA Publish/Subscribe Style Interaction.

It has been applied to compliance, including data sharing, as determined by the message context (Barth et. al, 2006).

In this chapter, our focus is to find the right fit of technologies and methodologies to support dynamic data sharing and collaboration as required with Palliative Healthcare. We present a framework that is designed to achieve this flexibly. This framework provides dynamic interaction and data sharing by extending the traditional SOA publish/Subscribe framework to support a common communication model through dynamic policies that can flexibly reflect dynamic changing rules of exchange as well as privacy and security considerations. Our framework supports policy-based processing of streaming event data using a general publish/subscribe model. This model provides good flexibility in the distribution and integration of data within a B2B network.

3. Palliative severe pain management scenario

Let us start with a Palliative Severe Pain Management Scenario. Palliative Severe Pain Management addresses one of the application areas of Palliative Care: Severe Pain Management (SPM). Severe pain is a pain score of 8 and above on a 10-point numeric rating scale (Kuziemsky et. al, 2008).

In discussing this scenario (Figure 3), we make the following assumptions:

- A patient is admitted into Palliative Care and starts receiving medical care from home. This patient is able to check and post periodic Pain Scores using the home PC, PDA or through automatic sensors.
- A patient assigned nurse automatically cumulates and analyses these pain scores to determine how the effective pain medications are on the patient.
- If the nurse's pain reports indicate severe pain above the acceptable threshold, the nurse sends a Pain alert to the patient physician for action. It should be noted that this nurse doesn't necessarily know the physician that will act on the alert.
- The patient physician then pulls historical reports to determine if the patient needs a new prescription. If required, the physician sends a new pain relief prescription back to the nurse who then administers the new prescription on the patient.
- The entire interaction is captured by the Palliative Care Portal (PCP) for reporting, non-repudiation and legal compliance.

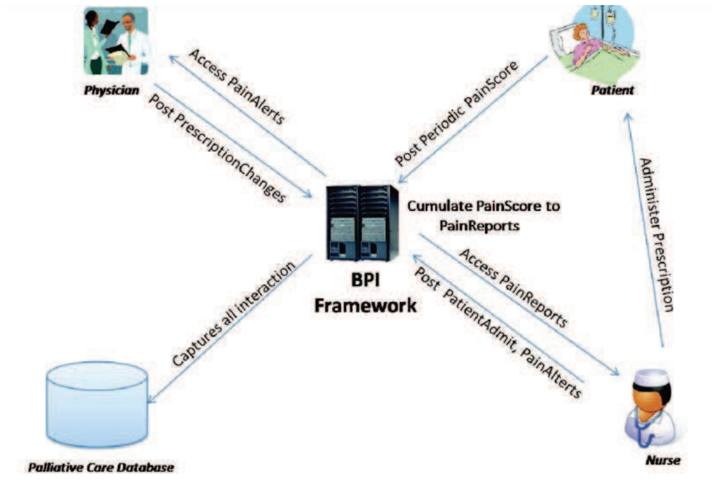


Fig. 3. Palliative SPM Scenario.

Palliative SPM Scenario described above requires an appropriate BPI framework for coordinating the data-sharing between the patient, nurse, doctor and the database infrastructure. Subsequent sections describe different approaches to solving the problems presented in this scenario and how our framework provides the most flexible architecture for achieving flexible data sharing and collaboration for all the players described in this scenario.

4. Web portal based approach

Web Portals are the most popular collaboration technology available on the Internet today. They are popular because they are easy to setup, central and easy to manage and control. It sells well when the participants belong to the same organization or union and agree to use a common collaboration portal. In addition, it must be operated by humans in real-time and all the data belongs to the portal custodian or host.

See Figure 4, the Web Portal connects the clinical management, patient care, and SPM applications as well as a number data sources into one application portal. Patients are admitted into Palliative Care when nurses create profiles for them. Patients are likely setup to use their home PC to periodically enter their *PainScore* to the portal through the Patient interface. The nurse periodically runs *PainReports* on each of the patient to analyze and determine the performance of their prescriptions. When anomalies are detected, a doctor is notified to advice on the next line of action.

In this approach, all parties: patients, nurses, and physicians are required to have access to the Web application portal in order to participate in Palliative Care. This doesn't reflect a true B2B interaction since it provides no mechanism for automatically integrating external but already existing clinical and hospital management systems.

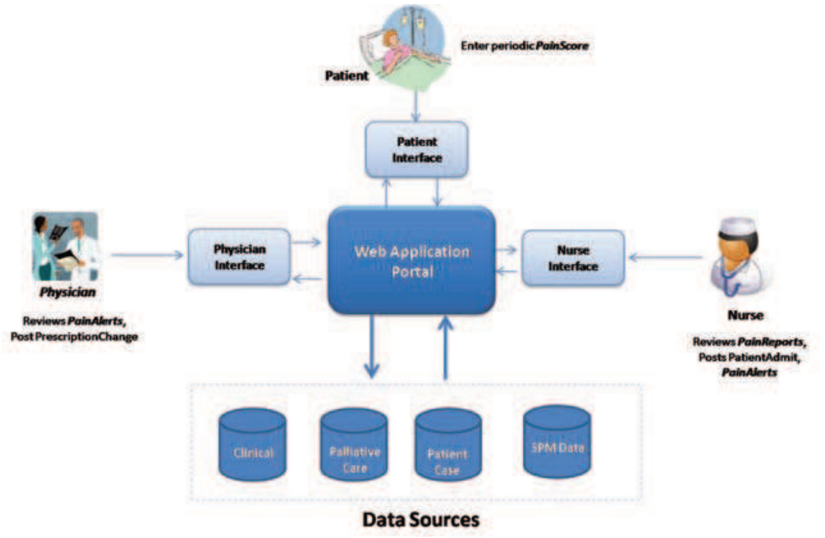


Fig. 4. Web Portal Based Approach to SPM.

The Web Portal based approach is not data-driven but interface-driven because interfaces define data and they are not easily modified to meet new requirements across the organizations using them.

Finally, the data sources are statically linked and represent a strong coupling. Rules of data exchanges are embedded in low-level business rules stuck in complex database views, stored procedures, applications methods and objects. They are usually not dynamic enough to capture new requirements. This is essentially a monolithic framework with static processes, interfaces, roles and permissions.

5. Traditional SOA Publish/Subscribe approach

SOA Publish/Subscribe approaches this problem by describing interactions as a network of events. Participating parties are seen as producers and consumers of event messages. This is very similar to the way emails are exchanged between various application domains. However in SOA publish/subscribe, these messages are application messages. Each event results in some data being communicated from an event source to an event destination.

In SOA publish/subscribe, the Palliative Sever Pain Management scenario described above is broken down into a set of *Topics* representing data exchanges. Figure 5 provides an architectural representation of this interaction. All communication amongst the partners uses HTTP/SOAP as the transport protocol.

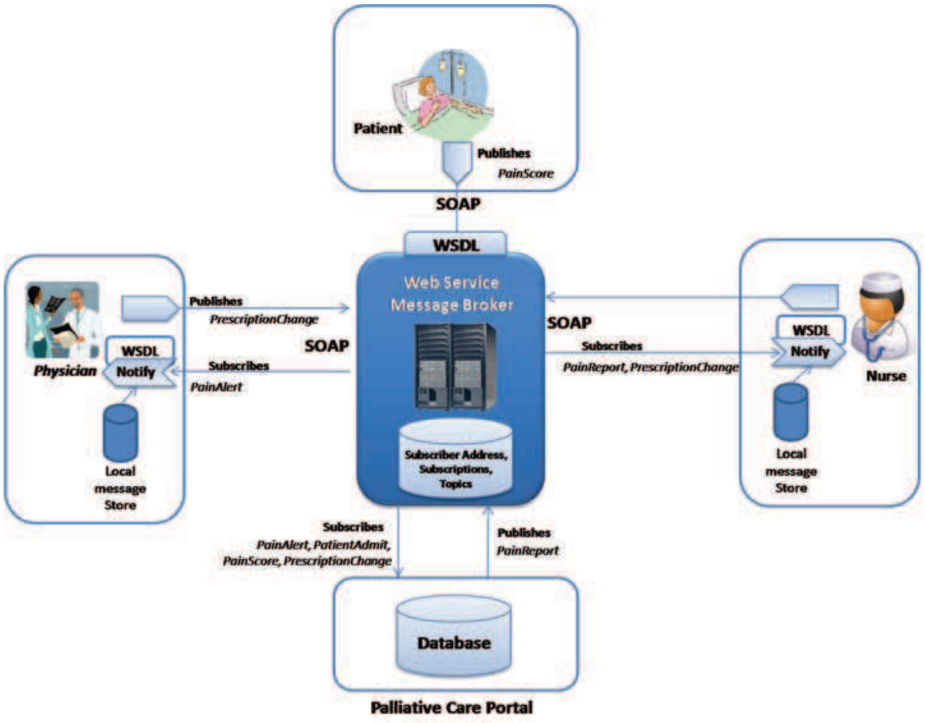


Fig. 5. SPM using SOA Publish/Subscribe Architecture.

This is realized by implementing a Partner Service Component for each collaborating party. This tool helps a participating party to push and receive data from the B2B network. Then there is a message broker that creates the virtual message pathway for message distribution. A Partner Service component implements a generic web service *notify()* interface that the message broker pushes messages to as well as a web service client that is used internally for publishing messages meant for other parties in the B2B network, through the same message broker.

The message broker plays the role of a message router. It receives all communication and routes them to the destination subscribers. In addition, the message broker maintains a list of all participating partners through a registration process for both publishers and subscribers. It also keeps a list of subscribers' topic subscriptions. Table 1 summarizes the roles played by the Partners in this scenario.

In this approach data sources need to publish data to the message broker while data consumers are required to have registered and active subscriptions to the topic of interest with the message broker.

For example the nurse plays the role of a publisher of *PatientAdmit and PainAlert* data, as well as a subscriber to *PainReport and PrescriptionChange.* Since the topics are pre-defined, the Palliative Care Portal subscribes to all the topics. This way, it is able to receive all the messages sent on these topics. We assume these messages are simple data blobs, with no support for content filtering.

Node/Service	**Role**	**Topic of messages published**	**Message Topic Subscriptions**
Broker	Message Broker	None	None
Nurse	Publisher/Subscriber	PatientAdmit PainAlert	PainReport PrescriptionChange
Patient	Publisher	PainScore	None
Physician	Publisher/Subscriber	PrescriptionChange	PainAlert
Palliative Care Portal	Publisher/Subscriber	PainReport	PatientAdmit PainAlert PainScore PrescriptionChange

Table 1. Palliative SPM Management in a Publish/Subscribe Topology.

The SOA publish/subscribe implementation described in this section supports our Palliative SPM scenario by decomposing complex BPI problems into multiple atomic events with multiple interactions using publish/subscribe. Unlike the Web Application portal approach, the definition is not statically tied to a B2B. Rather they are dynamically defined as run-time attributes to the framework.

However, despite the flexibilities introduced by this approach, it still utilizes publish/subscribe broadcast type interaction. Topic advertisement from arbitrary publishers is not a secure way of decomposing processes. It fails to define a mechanism for defining more flexible subscriptions that filters message contents for specific patterns, support message transformation and describe how and when the notification should be delivered.

6. Policy-based SOA Publish/Subscribe data sharing model

This architecture is functionally similar to the SOA publish/subscribe described in the previous section but with additional components for dynamic BPI using a policy model.

Consider the data sharing model in Figure 6, where data is being published and subscribed to by applications, devices and in some cases business processes. This approach tries to marry topic-based Publish/subscribe with content-based filtering. Self-describing topics identified by a name, and set of attribute value pairs, describe the published messages. The topic registrar is a service that defines as well as describes topics to the message broker.

In addition, the registrar could optionally define policies that restrict message publishers to defined topics. Subscriptions on the other hand describe interest to messages. Like topics and messages, policies are used to describe this interest flexibly, as well define attributes for message transformation, aggregation, and delivery Quality of Service.

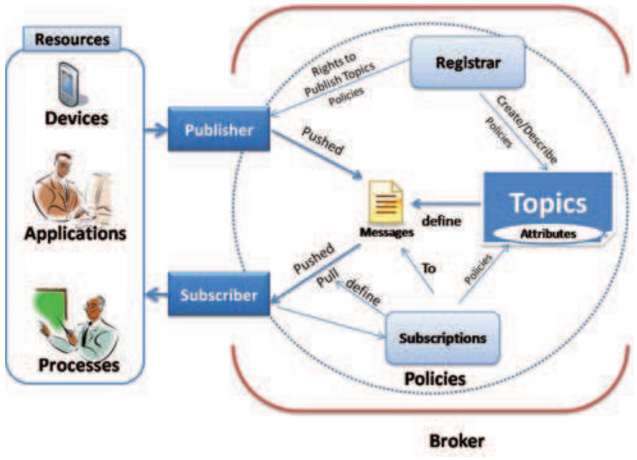


Fig. 6. Policy-based Pub/sub data sharing model.

6.1 Policy types

The policy framework described above is to extend an SOA publish/subscribe with added flexibilities and functionalities. As described in Figure 7, our framework supports four types of policies:

1. **Topic Policy**: Topics describe a message to the message broker. A *Topic Policy* provides a means of controlling "who has access to publish what". Topic creation can be controlled by delegating this responsibility to select publishers called *Topic Registrars*.

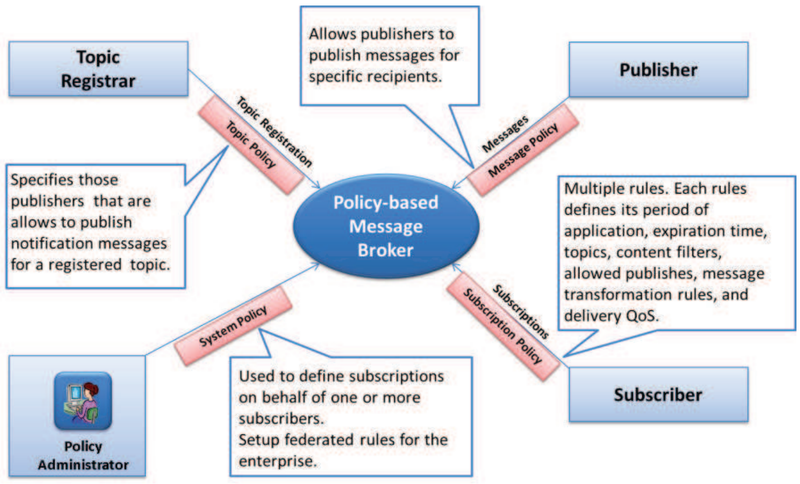

Topic Registrar
Allows publishers to publish messages for specific recipients.
Publisher
Topic Registration
Topic Policy
Messages
Message Policy
Specifies those publishers that are allows to publish notification messages for a registered topic.
Policy-based Message Broker
Multiple rules. Each rules defines its period of application, expiration time, topics, content filters, allowed publishes, message transformation rules, and delivery QoS.
System Policy
Subscriptions
Subscription Policy
Used to define subscriptions on behalf of one or more subscribers. Setup federated rules for the enterprise.
Subscriber
Policy Administrator

Fig. 7. Policy Types.

2. **Message Policy:** In publish/subscribe, messages are sent off from publishers to be routed to subscribers based on their subscriptions. In this approach, publishers of messages have no mechanism for controlling the recipients. Messages are simply broadcast to subscribers based on their subscriptions to a topic. In B2B, it is often required that certain communication be restricted to select parties. Message policies describe a means of achieving this flexibility in publish/subscribe. They define intended recipients to a message, thus allowing us to support various variants of broadcast type messaging as well as one-to-one messaging using publish/subscribe.
3. **Subscription Policy:** Simply stated, subscriptions are used by subscribers to indicate interests to a type of message usually by the topic or other meta-data as seen in content-based subscription. Our framework extends subscription through declarative policies to support scheduling, a more efficient combination of topic and content filtering, context-based message transformation and delivery QoS.
4. **System Policy:** It is often not appropriate security wise or in terms of management of data-driven rules of association to rely on applications/users to describe their interest to data appropriately. System policies are subscriptions done on behalf of subscribers by a policy administrator. System policies help define federated rules for data sharing with an enterprise view to its application. They can be automated and tied to an organization security policy and operations.

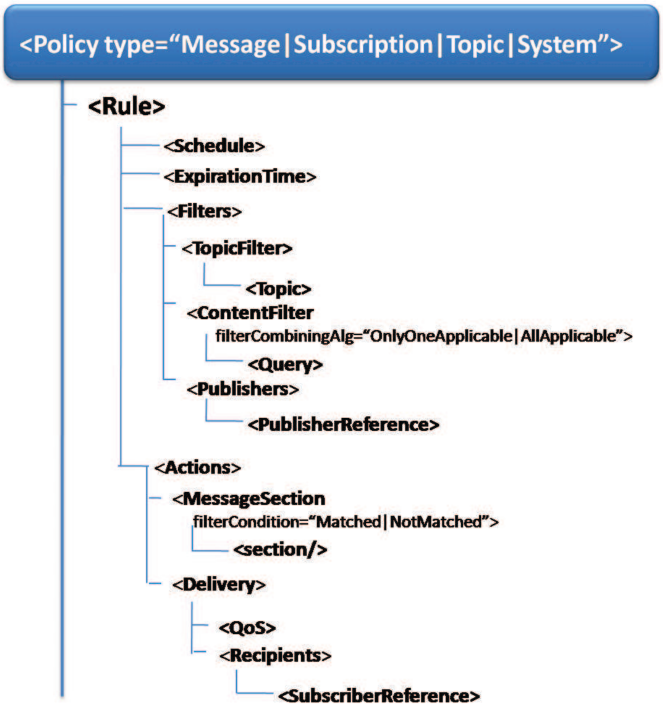
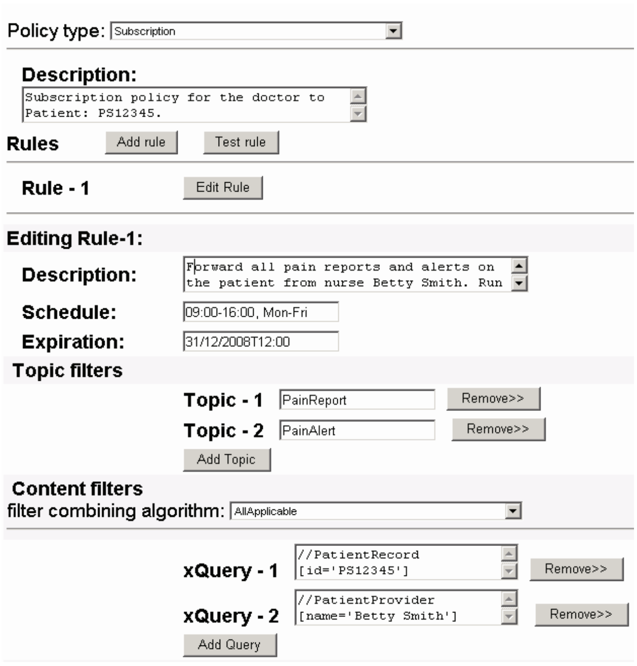


Fig. 8. Policy Representation.

Figure 8 shows a tree representation depicting a policy as a set of rules that describe a message. Notice that:

1. The <*TopicFilter*> element contains one or more <*Topic*>elements that define the topics for messages that it applies to.
2. <*ContentFilter*> element contains one or more XPath queries defined in <*Query*> elements and can be combined on the basis of the filter combining algorithm.
3. <*Publisher*> elements contain a list of allowed publishers described by the <*PublisherReference*> elements
4. <*Actions*> describe message transformation and rules for message delivery. Message transformation depends on the result of the result of filter execution.
5. <*Delivery*> element defines message delivery QoS. The <QoS> element values can be set as "BestEfforts", "Reliable" or batched/queued "Pullpoints".


Policy type: Subscription

Description: Subscription policy for the doctor to Patient: PS12345.

Rules

Rule - 1

Editing Rule-1:

Description: Forward all pain reports and alerts on the patient from nurse Betty Smith. Run

Schedule: 09:00-16:00, Mon-Fri

Expiration: 31/12/2008T12:00

Topic filters

Topic - 1: PainReport

Topic - 2: PainAlert

Content filters

filter combining algorithm: AllApplicable

xQuery - 1: //PatientRecord [id='PS12345']

xQuery - 2: //PatientProvider [name='Betty Smith']

Fig. 9. Describing Policies.

Figure 9 shows a sample subscription policy that will send Patient (PS12345) Pain Reports and Pain Alerts from the nurse Betty Smith to a doctor. This will only be sent from 09:00 – 16:00, Monday to Friday and expires on the 31st of December, 2008.

6.2 Policy-based partner service component

People, devices and sensors, surveillance infrastructure share real-time data in adhoc B2B networks using an SOA publish/subscribe model. Because of differences in technology, capacities and running platforms, a special component (Partner Service Component) interacts with these applications through an application message *inbox/outbox* type interface. This component is both a publishers and a subscriber to a policy-based SOA message broker. It also handles those publish/subscribe administrative details on behalf of these B2B resources using policy-based declarative subscriptions.

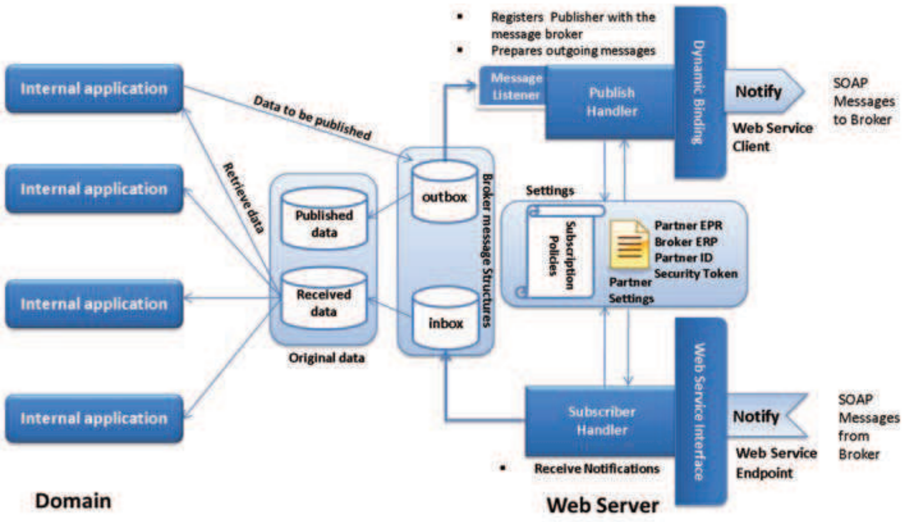


Fig. 10. Extended Partner Service Component.

In this framework, we extend the traditional SOA Message broker to support declarative policies. We also extend the PS component to support policy-based subscriptions and publishing (Figure 10). We use a partner service (PS) component as an adapter interface between the policy-based message broker and the application or process that publishes or subscribes to data. For example, a patient "John Smith" using a monitoring device attached to his wrist sends a *painscore* periodically. This device communicates locally to drop this *painscore* data structures into the *message outbox* of the partner service component. The Patient's PS component could be hosted on the patient home computer or a specially issued PDA from the hospital. It can also be in the form of a smart-phone application.

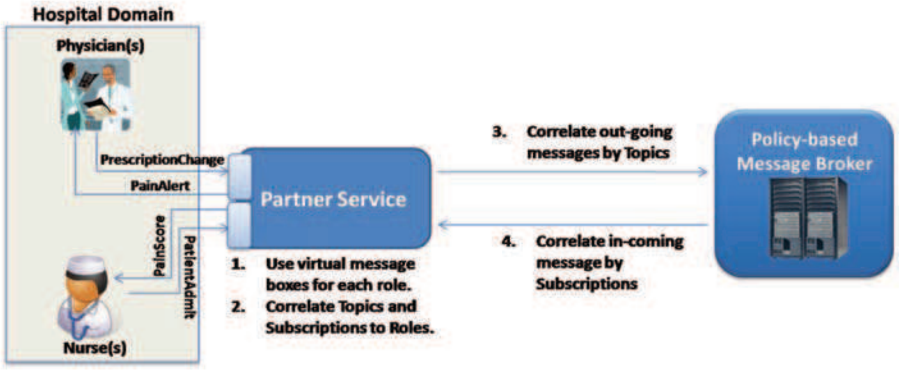

Hospital Domain
Physician(s)
PrescriptionChange
PainAlert
PainScore
PatientAdmit
Nurse(s)
Partner Service
1. Use virtual message boxes for each role.
2. Correlate Topics and Subscriptions to Roles.
3. Correlate out-going messages by Topics
4. Correlate in-coming message by Subscriptions
Policy-based Message Broker

Fig. 11. Supporting multiple groups and roles in a Partner Service Component.

The PS component is then pre-configured to dynamically bind and forward this data to the policy-based message broker. The component correlates messages by their topics and subscriptions. For example, since "NurseBetty" subscribes to *PainReport* and

PrescriptionChange, it stores these subscriptions and correlates them to outbound and in-bound messages. This way, it is able to manage data for multiple applications. It could even sit as a message proxy in a domain environment, ensuring that various parties and processes receive the appropriate messages.

In Figure 11, we illustrate how the same PS component correlates nurses and physician subscriptions in a hospital domain environment.

6.3 SPM scenario using the policy-based SOA publish/subscribe approach

Putting it all together, the policy-based message broker (Figure 12) is our core middleware. It manages interactions with all the participating parties Partner Server components on behalf of collaborating applications through a Policy engine and the policy database. Community-care appointed Policy administrators use a special tool to deploy subscription policies to the broker on behalf of collaborating partners. In addition it supports both synchronous and asynchronous data exchanges, self-describing topics since topic registration includes schemas that describe data published on the topic.

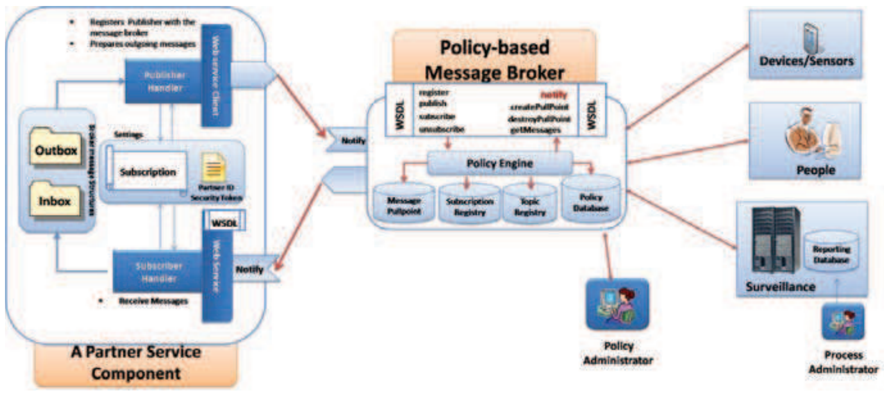


Fig. 12. Policy-based Message Broker.

Figure 13 shows the architectural represented of the interaction. Notice a special surveillance infrastructure hosted by the Palliative Care Portal. It subscribes to the policy message broker internal topic creation event and is able to receive the schema definition for each new topic. It subsequently uses this schema to create automatic table representation for capturing the event data. It then subscribes to messages attached to this topic. Unlike the SOA publish/subscribe, all these operations are performed automatically.

The Palliative Care Portal performs the surveillance role in our framework. Since it captures all messages published by data publishers, this infrastructure is able to provide various reports on the data exchanges. Various parties are able to subscribe to these reports automatically. In addition it is able to provide complete non repudiation for the entire B2B and perform automatic report generation based on threshold incidences and pattern triggers on data. These potentials are investigated in greater details in (Middleton et. al, 2009).

The entire infrastructure uses HTTP/SOAP as the communication protocol and therefore supports uninhibited communication on Internet. The policy-based message broker and all the interacting PS components are Web services endpoints.

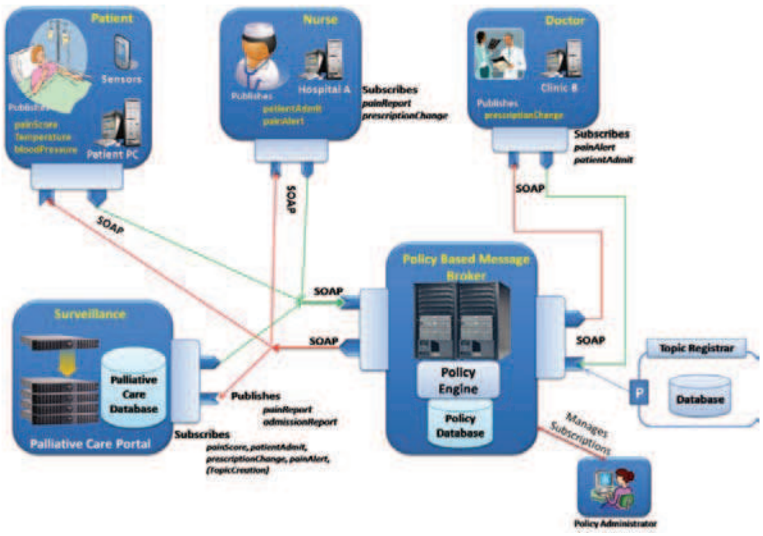


Fig. 13. Policy-based SOA Publish/subscribe Architecture.

7. Conclusions

Palliative Healthcare presents a unique challenge of caring for patients from home. Doctors, nurses, and other patient providers usually operate from domains and organizations that do not share common collaboration technologies. In this chapter, we have presented a scenario that describes a dynamic data sharing relationship between patients, nurses, and doctors from various domains with Palliative Care Portal acting a central database for the streaming events in the B2B network.

We have reviewed how the Policy-based SOA publish/subscribe was used to extend the basic Palliative Severe Pain Management scenario to support the requirements for data-driven interaction described in this chapter.

1. Data is dynamically defined at run-time using a combination of topic name and data attributes, thus providing an automatic connection between the management and logging of streaming data to reporting and historical information.
2. Data topics are not advertised by any arbitrary publisher as it is with classic SOA publish/subscribe. Rather, the Topic Registrar defines event topics alongside policies that control the rights to publish on the topics. For example, by publishing the *PatientAdmit* topic creation event, we automatically have the surveillance infrastructure create logging table for the topic and subscribe to data from the topic publishers.
3. All communication and data exchanges on this framework are received by the policy-based message broker (or a network of brokers) using a similar generic interface. Messages representing management operations are handled similarly like notification messages. Various handlers tie to the message context handler to process various types of message. This makes it very convenient to handle authentication, synchronous and asynchronous communication for all interactions.
4. The Partner Service components provide a great tool for easily integrating applications participating in SOA publish/subscribe. We have demonstrated its use as a local

middleware for publishing, subscribing, receiving and correlating messages for applications and users.

Most importantly, this framework demonstrates through the scenarios described in this chapter, the flexibility of defining XACML-based subscription policies on a streaming data model. Policy-based subscriptions provide the means of defining interest to data through topic and content filtering, message transformation and delivery QoS. Data queries provide access to stateful as well as historical data and reports to customers. Message policies provide a means of performing directed point-to-point notification using a publish/subscribe infrastructure.

8. References

BaileyJ., PoulovassilisA., and WoodP. (2003). An event-condition-action language for XML, In Proceedings of the 11th International Conference on World Wide Web, Honolulu, Hawaii, USA, pp. 486-495.

BarthA., DattaA., MitchellJ., and Nissenbaum H. (2006). Privacy and contextual integrity: Framework and applications, in IEEE Symposium on Security and Privacy,pp. 184-198.

CoieraE., and HovengaE. J. S. (2007), Building a sustainable health system, IMIA Yearbook of Medical Informatics 2007, vol. 2, no. 1, pp. 11-8.

DoshiC., PeytonL. (2008) Trusted information process in B2B networks, in Proceedings of the 10th International Conference on Enterprise Information Systems, Barcelona, Spain.

EtzionO., ChandyM., AmmonR.V., and Schulte R. (2006).Event-driven architectures and complex event processing, IEEE International Conference on Services Computing, Chicago, 2006, p. 30.

EugsterP. T., FelberP. A., GuerraouiR., and KermarrecA. (2003). The many faces of publish/subscribe. ACM Computing Surveys, vol. 35, no. 2, pp. 114-131.

Eze, B., KuziemskyC., PeytonL., MiddletonG., MoutthamA. (April 2010).,A Framework for Continuous Compliance Monitoring of eHealth Processes in Journal of theoretical and applied electronic commerce research, vol5/issue 1 p 56-70.

FosterI., KesselmanC., NickJ., and TueckeS. (2002). Grid services for distributed system integration, Computer, vol. 35, no. 6, pp. 37-46.

HarrocksI., AngeleJ., DeckerS., KiferM., GrosofB., and WagnerG. (2003), What are the rules? IEEE Intelligent Systems, vol. 18, no. 5, pp. 76-83.

HuhnsM. N. and SinghM. P. (2005), Service-oriented computing: Key concepts and principles. Internet Computing, IEEE.Vol. 0/issue 1, p 75-81.

KuziemskyC. E, Weber-JahnkeJ., Lau F., Downing G.M. (2008). "An Interdisciplinary Computer-based Information Tool for Palliative Severe Pain Management." Journal of the American Medical Informatics Association15(3): 375-382.

NiblettP. and GrahamS. (2005).Events and service-oriented architecture: The OASIS web services notification specifications, IBM Systems Journal, vol. 44, no. 4.

PeytonL., HuJ., DoshiC., and SeguinP. (2007). Addressing privacy in a federated identity management network for e-health, in 8th World Congress on the Management of eBusiness, Toronto, 2007. Internet Computing, vol. 9, no. 1, pp. 75-81.

MiddletonG., PeytonL., KuziemskyC., EzeB. (2009). “A Framework for Continuous Compliance Monitoring of eHealth Processes”, World Congress on Privacy, Security, Trust and Management of eBusiness.