An Application Framework for Monitoring Care Processes

Aladdin Hussein Baarah

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree of

PhD in Computer Science

School of Electrical Engineering and Computer Science

University of Ottawa

Ottawa, Ontario, Canada

Abstract

Care process monitoring is important in healthcare domains to provide precise and detailed analytics on patients, providers, and resources participating in a care process and their status. These analytics are used to keep track of whether the quality of care goals set by healthcare organizations are satisfied and ensure that legislative and organizational guidelines are followed. The complexity of care process monitoring can vary depending on whether the care process takes place in a hospital or out in the community, and it can vary depending on the complexity of the information technology infrastructure that is in place to support the care process.

A Care Process Monitoring Application (CPMA) is a software application which collects and integrates data from various sources while a care process is being provided, in order to provide performance reporting of metrics that are used to measure how well the performance goals and guidelines for the care process are being met. In our research, we have studied how CPMAs are built in order to improve the quality of their engineering. The significant challenge in this context is how to engineer a CPMA so that the engineering process is repeatable, produces a CPMA of consistent high quality, and requires less time, less effort and less complexity.

This thesis proposes an application framework for care process monitoring that collects and integrates events from event sources, maintains the individual and aggregate states of the care process and populates a metrics data mart to support performance reporting. Our

contributions are the following: a state-based application meta-model of care process monitoring, a care process monitoring architectural pattern, and finally, a behavior driven development methodology for CPMAs based on our meta-model and architectural pattern.

Our results are validated through three different case studies in which we collaborated with two different health care organizations to build and deploy CPMAs for two different care processes (one hospital-based, the other community-based) in collaboration with healthcare clinicians and researchers.

Acknowledgments

First and foremost, I thank my Lord for granting me the power and the capacity to complete this thesis.

I owe my deepest gratitude to my supervisor, Prof. Liam Peyton, for his guidance and support, and for his encouragement, knowledge and his patience throughout my research. Without him this thesis would not have been completed or written.

It is hard to find appropriate words to express my sincere gratitude to my beloved wife Razan who has been supporting, understanding, caring and loving all the time throughout my research. This thesis would have not been possible without her help and understanding. Also, I owe so much thanks to Massah, my daughter, who was born during the writing of this thesis, for giving me unlimited happiness and fun.

My deepest gratitude is dedicated to my beloved father Prof. Hussein Baarah and my mother Fatima Baarah for their love and unconditional support and encouragement throughout my study and research.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
ACS	Acute Coronary Syndrome
BAM	Business Activity Monitoring
BPM	Business Process Management
BPMN	Business Process Model and Notation
CCL	Cardiac Cath Lab
CEP	Complex Event Processor
CIS	Clinical Information System
CPM	Care Process Monitoring
CPMA	Care Process Monitoring Application
CPME	Care Process Monitoring Engine
CW	Cardiology Ward
DAO	Data Access Object
DSR	Design Science Research
ECA	Event-Condition-Action
ED	Emergency Department
EDA	Event Driven Architecture
EHR	Electronic Health Record

EMA	Event Monitoring Agents
ETL	Extract, Transform, Load
EMR	Electronic Medical Record
GPS	Global Positioning System
HIS	Health Information System
IS	Information System
KPI	Key Performance Indicator
MDA	Model Driven Architecture
MPAF	Managed Process Application Framework
MQ	Message Queue
MVC	Model View Controller
OLAP	Online Analytical Processing
PPSMCS	Palliative Pain and Symptom Management Consultation Services
RFID	Radio-Frequency Identification
RTLS	Real Time Location Service
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SME	State Monitoring Engine
URN	User Requirements Notations
WSDL	Web Service Description Language

Chapter 1. Introduction

1.1. Problem Statement

Healthcare organizations have guidelines, business rules and policies to document care processes. Each healthcare provider who is involved and participating in the care process should know his duty and how to deliver his outcome to other parties within the organization. Often communication is ad hoc; by e-mails, phone calls, electronic documents and paper (Armellin, Betti, Casati, Chiasera, Martinez, & Stevovic, 2010). As a result, monitoring precisely what is taking place within a care process is difficult. Often the reports available to managers and stakeholders in healthcare are based on a very coarse level of granularity that gives only a rough idea of performance, weeks after the fact.

The complexity of managing and monitoring care processes varies depending on the type of care process. We distinguish between two types of care processes, community care processes and hospital care processes. In community care processes, patients are tracked on an ongoing-basis over a long period of time, usually for chronic disease management or as part of a process of regular health maintenance. Often the information technology infrastructure is either non-existent, rudimentary or fragmented. In hospital care processes, patients are tracked for a relatively short period of time, usually for acute care (a broken limb, car accident) or to receive treatment related to a chronic disease that requires a care service available only in a hospital. Usually, the information technology infrastructure is more pervasive and sophisticated, but often

many different systems are involved with poor interoperability between them (Mouttham, Peyton, & Kuziemsky, 2011).

This thesis will examine the challenge of defining an application framework for designing, building and deploying *Care Process Monitoring Applications* (CPMA). A Care Process Monitoring Application is a software application that collects and integrates data from various sources while a care process is being provided, in order to provide metrics that are used to measure how well the performance goals and guidelines for the care process are being met. In our research, we have studied how CPMAs are built in order to improve the quality of engineering. The significant challenge in this context is how to engineer a CPMA so that the engineering process is repeatable, produces a CPMA of consistent high quality, and requires less time, less effort and less complexity.

Our initial hypothesis is that there are commonalities that all CPMAs will share regardless of the care process and whether or not it is community or hospital-based. Based on these commonalities, it should be possible to define an application framework for care process monitoring consisting of a general architectural pattern for care process monitoring, and a domain-specific application meta-model which can be leveraged in a systematic development methodology. Our main hypothesis is that an engineering process for developing CPMAs, based on such an application framework, will be repeatable, produce CPMAs of consistent high quality, and require significantly less time, less effort and less complexity than current approaches found in the literature or used in health care organizations today.

1.2. Motivation

One can think of a CPMA as a type of business intelligence (BI) application that provides performance reports for a particular care process. In order to build such a solution, however, one needs an information technology infrastructure that can integrate data and events across the care process to provide a good operational model of process details and related data sources linked to a good strategic model of the quality of care goals for the process and how they are measured. There are a number of related technologies and approaches that are relevant including: Business Process Management (BPM), Real-time Location Service (RTLS), Event-Driven Architecture (EDA), Service Oriented Architecture (SOA), Complex Event Processing (CEP), Data Warehouse (DW) and Business Intelligence (BI).

BPM can be used to re-engineer care processes using business process models (Weske, 2012) and provide events in near real-time while the care process is enacted online. BPM is typically set up within a single organization which can be effective for hospitals (Mouttham, Peyton, & Kuziemsky, 2011) but community care has to be coordinated across many separate organizations and individuals (clinics, hospitals, long term care facilities, doctors, residents, caregivers, etc.)

Wi-Fi enabled Real-time Location Service (RTLS) systems which generate location events as they are happening, which can be used to track the status and location of entities involved in the process in near real-time (Bellenger, Pawlowski, & Westhuis, 2010) (Guillemette, Fontaine, & Caron, 2008) .

EDA can be used to collect data from disparate sources across a network and deliver them to the event processing as they are happening in real time (Levina & Stantchev, 2009) (Niblett & Graham, 2005). Combining EDA with SOA (Michelson, 2006) integrates the event sources and event consumers with the event processing. CEP is a specialized type of rule-based event processing that can been leveraged to infer high-level business events and respond to them on the fly. It has been applied in healthcare to recognize and respond to critical medical events (Yao, Chu, Li, & Mullen, 2008).

Data collection and integration from distributed operational data sources can be performed in a DW in order to monitor the care processes and provide historical performance reports (Kimball & Ross, The Data Warehouse Lifecycle Toolkit, 2008). BI leverages a DW to support visualization and analysis using performance reports and dashboards in order to provide insights on the care process to end users (Azvine, Zheng, & Nauck, 2005).

Application frameworks are an important concept for systematizing and reducing the engineering efforts associated with application development. Models and agile methods that leverage user scenarios are relevant technologies. Modeling an application using the Unified Modeling Language (UML) is standard practice to systematize and reduce the complexity of communications among the development team and stakeholders (Booch, Rumbaugh, & Jacobson, 2005). More recently requirements languages such as the User Requirements Notation (URN) have been used to define performance measurement of business processes (Pourshahid, Amyot, Peyton, Chen, Weiss, & Forster, 2009, and model driven approaches to developing applications are increasingly common (Tegegne & Peyton, 2013). Test-driven development (TDD) is an agile method that adapts to changing business requirements and ensures that the

developers can understand the requirements before coding (Beck, Test Driven Development: By Example, 2003). More recently, Behavior-Driven Development (BDD) ensures that all participants (developers, tester and stakeholder) have the same understanding of the system based on usage scenarios (Sacks, 2012) (Soeken, Wille, & Drechsler, 2012).

Our approach has been to:

- understand how to characterize care process monitoring (CPM) in terms of a general architectural pattern that relates CPM to the relevant information technologies that can be applied;
- understand how to characterize a CPMA in terms of an application model that captures the key relationships between the sources of data in a care process and the metrics used to measure performance;
- understand how to develop a CPMA to monitor the critical behaviour in a care process using a systematic but agile development methodology.

1.3. Thesis Contributions

This thesis proposes an application framework for developing Care Process Monitoring Applications (CPMA) that collects and integrates events from the event sources, maintains the individual and aggregate state of the care process, and populates a metrics data mart to support performance reporting.

Our results are validated through three different case studies in which we collaborated with two different health care organizations to build and deploy CPMAs for two different care

processes (one hospital-based, the other community-based) in collaboration with healthcare clinicians and researchers.

The main contributions of the thesis are the following:

1. A state-based application meta-model of care process monitoring that can be instantiated for any care process to define its goals, performance measures, states, events and the relationships between them.
2. A Care Process Monitoring architectural pattern that defines a core set of system components and their interfaces that can be reused as the basis for any CPMA. This includes: a Care Process Monitoring Engine (CPME), Metrics Data Mart and Reporting Dashboard.
3. A behavior driven development methodology that articulates how to leverage the meta-model and architectural pattern in a repeatable, systematic manner to build CPMAs that are more effective and efficient than traditional approaches to building CPMAs.

The following publications have been published or in relation to the thesis:

1. O. Baderddin, **A. Baarah**, L. Peyton, "Analytics Driven Application Development for Healthcare Organizations", Accepted for publication in the Proceedings of HEALTHINF 2014, 3-6 March, 2014.
2. **A. Baarah**, O. Baderddin, L. Peyton, "An Application Meta-Model for Real Time Monitoring of Care Processes", submitted to Computational and Mathematical

Methods in Medicine, Special issue in Medical and Health Data Analysis in the Age of Big Data, July, 2013.

3. C. Kuziemsky, **A. Baarah**, A. Chamney, L. Peyton, "Systems Design and Interoperability Considerations for Health Information Exchange to Support Collaborative Care Delivery", submitted to Electronic Commerce Research and Applications (journal), July, 2013.
4. S. Baffoe, **A. Baarah,** and L. Peyton, "Inferring State for Real-Time Monitoring of Care Processes", 5th International Workshop on Software Engineering in Healthcare, at the International Conference on Software Engineering (ICSE 2013), San Francisco, May, 2013.
5. L. Peyton, A. Mouttham, K.A. Ali, **A. Baarah,** H.T. Mouftah, "Real-time Analytics and Quality of Care", Handbook of Systems and Complexity in Health., Springer, pp. 495-519, 2013.
6. **A. Baarah,** and L. Peyton, "Engineering a State Monitoring Service for Real-Time Patient Flow Management", 9th Middleware Doctoral Symposium of the 13th ACM/IFIP/USENIX International Middleware Conference, Montreal, Canada, 2012.
7. **A. Baarah,** A. Mouttham, L. Peyton, "Architecture of an Event Processing Application for Monitoring Cardiac Patient Wait Times", International Journal of Information Technology and Web Engineering, vol. 7, no. 1, pp. 1-16, 2012.
8. **A. Baarah,** A. Mouttham, L. Peyton, "Improving Cardiac Patient Flow Based On Complex Event Processing", IEEE Jordan Conference On Applied Electrical Engineering and Computing Technologies, Amman, Jordan, December 2011.

This thesis was completed in collaboration with other students as part of a general research program funded by NSERC and MITACS. Below is a list of theses in which two master students completed their thesis on June 2013. The remaining PhD and Master thesis are still in progress. For each thesis, I state the relationship to this PhD work:

- Alain Mouttham, "A Framework for Real-Time Analytics and Decision Support in Patient Flow Management", PhD thesis in Computer Science, University of Ottawa (in progress). This is a health informatics thesis that provides a comprehensive framework for both analytics and decision support in patient flow management. A care process monitoring application is one component of that overall framework to enable analytics. Mouttham's thesis sets the context in which to understand the problem I am addressing in this thesis and the software engineering approach I have taken to define an application framework for building care process monitoring applications. Mouttham's was also the main point of contact with the hospital for case study 1 and case study 2 in this thesis.
- Renaud Bougueng Tchemeube, "Location-Aware Business Process Management for Real-time Monitoring of Patient Care Processes", Master's thesis in Computer Science, University of Ottawa, June 2013. The main contribution of Tchemeube's thesis, related to my work, is to integrate BPM and RTLS technology into the information technology architecture supporting a care process in order to provide events to a Care Process Monitoring Application (CPMA). Tchemeube's was responsible for the BPM and RTLS components that were used in case study 1 and case study 2 of this thesis.

- Shirley Baffoe, "A Generic BI Application for Real-Time Monitoring of Care Processes", Master of Science in Electronic Business Technologies, University of Ottawa (June 2013). Baffoe's experimented with two alternative implementations and designs of the care process monitoring engine (CPME) that conformed to our proposed architectural pattern in case study 1 and case study 2 of this thesis. In particular, she leveraged the application model for cardiac care (that we built using our application meta-model) to implement the CPME in case study 2.
- Austin Chamney, "A BI Application Framework for integrating mobile forms data and Care Process Monitoring", Master of Science in Computer Science, University of Ottawa (in progress). Chamney's experimented with an alternative implementation and designs of the CPME that conformed to our proposed architectural pattern in case study 3 of this thesis. In particular, he leveraged the application model for palliative care (that we built using our application meta-model) to implement the CPME in case study 3. Chamney's thesis, which he is just starting, will focus on the integration of AJAX mobile apps for form-based data collection with the metrics data mart in our architecture.

1.4. Research Methodology

In order to conduct our research, we should follow a research methodology that is proven and leveraged by other researchers. The aim of our proposed approach is to improve the engineering of CPMAs. Since this is categorized as the development and design IT systems, we use the Design Science Research (DSR) methodology. According to (Vaishnavi & Kuechler,

2012), this type of research methodology has been proven and been exploited by numerous research communities in the Information Science (IS) discipline.

DSR involves constructing and evaluating an innovative artifact "intended to solve identified organizational problems" (Hevner, March, Park, & Ram, 2004). In order to determine whether the proposed artifact is novel or not, it must be considered as a more effective solution for the problem or as a solution for a problem that has never been solved (Hevner, March, Park, & Ram, 2004).

In particular, we used the process model proposed by Peffers et al. (2006) for conducting DSR. We followed exactly the six main steps iteratively (two and a half times) to construct a complete artifact (an application framework) that is an effective solution for the problem of engineering CPMAs. The six steps we followed are: problem identification and motivation, objectives for a solution, design and development, demonstration, evaluation and communication.

In the first iteration, we followed these steps:

1. **Problem identification and motivation**: Identify the problem of monitoring care processes in near real-time motivated by the possibility of leveraging emerging technologies such as CEP, BPM and RTLS.
2. **Objective of the solution**: Build an architecture to collect, process and integrate events from the underlying care processes as they are taking place in order to provide performance reports in near real-time.

3. **Design and development**: Propose an architecture and design a CPMA system and build a prototype based on that architecture using traditional software engineering methodology.
4. **Demonstration**: Create a demonstration environment that would allow domain experts and clinicians to walk through a real cardiac care scenario using the prototype CPMA.
5. **Evaluation**: Evaluate, validate and review results of the first case study with a domain expert and the clinical staff at Osler Hospital who walked through the prototype in the demonstration environment.
6. **Communication**: Publish a journal paper (Baarah, Mouttham, & Peyton, 2012), a conference paper (Baarah, Mouttham, & Peyton, 2011) and a poster to share results.

In the second iteration, we have observed the solution during the demonstration and identified gaps we wished to address. Based on that, we have refined the problem identification, objective of the solution and the design in order to improve our proposed solution. We followed these steps:

7. **Problem identification and motivation**: Identify the problem of how best to engineer any CPMA motivated by the possibility of defining an application meta-model and general architecture pattern that would capture commonalities across all CPMAs for any care process so that implementation of a CPMA could follow a specific simplified test driven agile software methodology.

8. **Objective of the solution**: Develop an application framework for care process monitoring that would enable CPMAs to be engineered in a systematic, repeatable fashion with less effort, less time and less complexity.
9. **Design and development**: Develop an application meta-model and an architectural pattern, and proposed a software methodology based on a literature review and experiences from the first iteration to define an application framework that could be used to build a CPMA for any care process.
10. **Demonstration:** Re-implement the CPMA from the first iteration using the new application framework. Reuse the same demonstration environment with the same scenario as the first case study in iteration 1 with domain experts of Osler Hospital, as well as hospital staff and technical experts from two new hospitals (Ottawa Hospital, Queensway Carleton).
11. **Evaluation**: Evaluate and validate the application framework based on the experience of re-implementing the CPMA.
12. **Communication:** Submit a journal paper, publish two conference papers (Baarah & Peyton, 2012) and (Baffoe, Baarah, & Peyton, 2013) a poster. A Master's thesis related to this research (Baffoe, 2013) was also published.

In the third iteration, we simply repeated the last three steps in a different healthcare domain (community care) to validate that our application framework can be applied in different healthcare processes. Those steps are:

13. **Demonstration**: Implement a CPMA for palliative care. Deployed a prototype CPMA for use by domain experts and clinicians at Bruyère Hospital to monitor and report on a real palliative care process.
14. **Evaluation**: Evaluate and validate the experience of the palliative care CPMA and compare the results of applying our framework to a community care process CPMA with the previous results obtained for a hospital care process.
15. **Communication**: A journal paper and conference paper have been submitted together with this thesis to publish our results.

1.5. Thesis Organization

The thesis is organized as follows:

In chapter 2, we give an overview of care process monitoring and provide background material that is related to it such as clinical pathways, business analytics and data warehouses. We also give a very brief overview of different aspects of application development that are relevant to the engineering of care process monitoring applications. Finally, we discuss related work and highlight differences from our approach.

In chapter 3, we define the problem statement with respect to monitoring care processes and we provide a description of the problems we aim to address in the three case studies we use to evaluate our application framework. Also, we perform a gap analysis and define a set of criteria that are leveraged for evaluating our application framework prototype for each of the three case studies. In addition, we compare our application framework against related work described in chapter 2.4.

In chapter 4, we present our application framework for hospital and community care process monitoring. The elements of the application framework which are the application meta-model, the architectural pattern and the methodology for creating Care Process Monitoring Application (CPMA) are illustrated in detail in this chapter.

In chapter 5, we introduce three different real-life case studies drawn from a large community hospital located in Ontario. We describe the role of the thesis researcher in all case studies. However, the initial case study describes a prototype of the Care Process Monitoring Application (CPMA) that was built with the collaboration of IBM. In the second case study, we refined and re-engineered the first case study using our complete application framework. In the third case study, we describe a CPMA related to palliative care in a community settings based on our complete application framework and analyze the differences between community and hospital care processes as it relates to our framework.

In chapter 6, we evaluate our application framework against the three case studies. The evaluation criteria defined in chapter 3 is leveraged to compare the three different versions of the application framework in the case studies against ETL-Data warehouse. Furthermore, we use the evaluation criteria to compare our application framework to the related work in chapter 2. Finally, we discuss the limitations and the assumptions of our application framework.

Finally, in chapter 7, we summarize our conclusions and discuss future work.

Chapter 2. Background and Related Works

In this chapter, we first explain care process monitoring and describe the concepts relevant to it. Then, background on application development is given, including software engineering development methodologies, architectural patterns, models and application frameworks. After that, we provide background on technologies relevant to care process monitoring. Finally, in the related work section, we identify other approaches to care process monitoring in the literature.

2.1. Care Process Monitoring

The aim of care process monitoring is to provide analytics and performance management reports to measure to what extent quality of care goals are achieved for a care process.

2.1.1 Care Processes and Clinical Pathways

A care process involves a sequence of steps which are carried out by different medical staff (nurse, physician, administrative) in order to provide treatment to patients for a specific type of health condition. Usually, there are a number of quality of care goals defined for the care process which must be met in order to comply with regulations of the healthcare organization providing the care as well as of various accreditation organizations and governmental regulators (Middleton, Peyton, Kuziemsky, & Eze, 2009). The care process is often executed across different units of a hospital, each unit with its own information system. Or, the care process may be executed by a collaboration of different healthcare organizations in the community, each with its own information system as well (Reichert & Lenz, 2007) (Armellin, Betti, Casati, Chiasera,

Martinez, & Stevovic, 2010). Often these separate information systems or "data silos" in different hospital units or different healthcare organizations cause a lack of insight into overall performance of the care process as well as an incomplete picture of resource status and availability (NIH, 2006).

Palliative care is an example of a collaborative care process that takes place in the community which we will use in this thesis. In palliative care, different healthcare organizations in the community (nursing, physiotherapy, psychotherapy, clergy, etc.) collaborate to manage the pain and symptoms of patients with a terminal illness at home (Haugen, Nauck, & Caraceni, 2011) (Ferris, et al., 2002). In Canada, the palliative care process should adhere to the care process guidelines and quality of care indicators set by Accreditation Canada (2013). Data sharing across healthcare organizations is difficult, although there has been recent work to leverage Electronic Health Records (EHR) for palliative care in the community (Liu, 2010).

Clinical guidelines have been developed by the medical community for many care processes, especially those usually delivered by hospitals. According to Field & Lohr (1990), clinical guidelines are defined as "systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances". In fact, clinical guidelines are recommendations for medical practitioners to follow for the purpose of helping them to make decisions concerning the best medical care based for particular patient conditions. Therefore, clinical guidelines help to improve patient care and reduce risks (Ghattas, Peleg, Soffer, & Denekamp, 2010). For example, Intermountain Healthcare (Care Process Models, 2013) catalogs all its care processes. For each clinical process, references to the

appropriate guidelines is made, along with target quality of care metrics for the process, and in some cases a listing of the major steps or tasks to follow in providing treatment.

Clinical pathways are multidisciplinary care plans that specify the sequence and timing of actions necessary for achieving expected patient outcomes (El Baz, Middel, van Dijk, Oosterhof, Boonstra, & Reijneveld, 2007). These are typically defined at hospitals for the most common and critical care processes. Clinical pathways have been leveraged as a method to design and monitor care processes through collecting data as the clinical pathways are executing (Gattnar, Ekinci, & Detschew, 2011) (Vanhaecht, Panella, Zelm, & Sermeus, 2010). In addition, clinical pathways are used to standardize care processes, monitor variances, and measure and evaluate the patients (Li, Liu, Yang, & Yu, 2013).

Acute Coronary Syndrome (ACS) (Dyck, et al., 2012) (Erdem, Geisler, & Flather, 2010) is an example of clinical pathway that describes the essential steps and guidelines that should be followed for cardiac patients from the time they enter the emergency room until they are admitted and operated on and then discharged. In case studies 1 and 2 of this thesis, we use the clinical pathway that William Osler Hospital has defined for ACS.

2.1.2 Business Analytics

In order to monitor care processes, data must be collected and reported to measure how well they are meeting quality of care goals dictated by organizational, governmental, and accreditation regulations. The aim of *business analytics* is to collect and monitor data generated from executing a business process and perform some sort of analysis according to the values of performance measures. The analysis indicates how well the organization is performing its

business process. As a consequence, the analysis results are displayed to the users in a simple format such as a dashboard or displayed to them as reports that help them in decision making (Azvine, Nauck, & Ho, 2003). The most well-known framework for measuring performance in a systematic manner that links business processes to performance management is the balanced scorecard (Kaplan & Norton, 1996). In the balanced scorecard, metrics are defined to measure performance for four key strategic areas of an organization which are: learning and growth, internal business process, customers and finance, using data collected from operational processes.

In near real-time business analytics, in order to support decision making, metrics, charts and alerts should be available in near real-time. The objective of near real-time analytics is to minimize the time between the occurrence of critical situation and the time when an action is initiated (Chieu & Zeng, 2008). This is critical for care process monitoring, if one wants to effectively managed care processes in a systematic way across a large healthcare organization or large community. Near real-time analytics should provide information in a timely fashion so that care providers can respond and take corrective action if care processes are not performing as expected.

2.1.3 Performance Management and Key Performance Indicators

Performance Management (PM) focuses on monitoring business performance by leveraging set of *Key Performance Indicators* (KPIs), systems, processes, as well as methodologies. PM can be used to bridge the gap between the operational level, which focuses on monitoring operational process, and the strategic level, which focuses on defining and

deducing KPIs according to the business goals that should be satisfied towards the processes (Kemper, Rausch, & Baars, 2013).

The purpose of KPIs is to measure whether the goals of the organization are satisfied or partially satisfied. Often there is a disconnect between high-level and strategic goals measured by KPIs and the low-level details of individual steps in a business process (Leggat, Bartram, & Stanton, 2011) .

Ideally, KPIs is used to monitor the performance of the business processes of the organization by calculating the KPIs from the events which are captured when the business processes are executed. However, each strategic goal also needs to be linked to these KPIs in order to measure to which extent the performance of the organization is far from or near to its goals (Kuziemsky, Liu, & Peyton, 2010). One of the characteristics of KPIs is the SMART (Specific, Measurable, Attainable, Realistic and Time- Sensitive) criteria (Shahin & Mahbod, 2007).

The current state of the art to provide performance management for care processes is that the data is collected and processed manually from existing clinical information systems that are located in different units of the hospitals or in different healthcare organizations. Increasingly in larger hospitals, there is a systematic batch-oriented process to Extract, Transform and Load (ETL) data from source systems into a data warehouse (Kimball & Ross, Health Care, 2002) and then apply business intelligence tools to process that data. This approach can still take days or weeks to provide reports on care processes, which is not practical for day-to-day operations (Mawilmada, 2011).

2.1.4 Electronic Health Record and Data Warehouses

According to (Gunter, Terry, & Powell, 2005), an *Electronic Health Record* (EHR) is defined as a "longitudinal collection of electronic health information about individual patients and populations". An EHR has information regarding each patient such as the medical history of the patient, lab test results, radiology images, etc. An EHR can be shared across a group of cooperating healthcare organizations (Yina, 2010). An EHR contains a database repository that allows physicians to exchange date from different hospitals (Kierkegaard, 2011) (Neal, 2008). The primary goal of an EHR is to improve the quality of care of patients by integrating healthcare data related to the patients gathered from paper records and multiple electronic sources (Gunter, Terry, & Powell, 2005).

Even though the terms EHR and *Electronic Medical Record* (EMR) are often used interchangeably, there is a variance between those two terms and one should distinguish between them (Habib, 2010). EMR is an electronic record of a patient created and maintained by a single physician and staff from a single organization and it is considered as the primary data source for EHR. The key difference between EMR and EHR is that the patients' information in EMR is not shared across hospitals (Kierkegaard, 2011) (Habib, 2010). Currently, most community care is not supported by an EHR but rather it is supported by disparate EMRs maintained by each care provider.

The core advantage of EHR is the integration of healthcare data from disparate systems located at different departments within the hospital or across multiple organizations (Yina, Application of EHR in Health Care , 2010). The EHR architecture supports patient data storage and data exchange, but there is usually a disconnection with care processes (Kuziemsky,

Williams, & Weber-Jahnke, Towards electronic health record support for collaborative processes, 2011). Patient data is often entered into the EHR hours or days after the fact, so it is difficult or impossible to determine actual service times and wait times in near real-time. In addition, patient data focuses on administrative resources rather than clinical steps and thus it can be hard to extract or visualize the essential steps of the clinical process. Finally, EHR databases are not typically optimized for reporting.

A *data warehouse* is a mechanism for collecting and organizing data so that it is optimized for reporting. The data is often collected and reorganized in an attempt to provide a view of the performance of clinical processes. A data warehouse is a central entity that contains consolidated data from distributed operational data sources within the same organization or from disparate organizations. The purpose of building a data warehouse is for decision making and support within an organization and for producing performance reports (Kimball & Ross, 2008).

A data warehouse is constructed by extracting data from a variety of operational databases, then transforming these data into a certain format that is acceptable by the data warehouse, and finally loading it to the data warehouse. This process is called Extract Transform Load (ETL). The warehouse should be updated frequently on a regular basis (e.g., weekly) to reflect the changes that may occur in the source data (Chaudhuri & Dayal, 1997).

2.1.5 Business Process Monitoring

A *Business Process* is defined as "a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization but it may interact with business

processes performed by other organizations" (Weske, 2012). While business processes are executed, many events are generated and captured in order to provide a detailed visibility of the organizational processes. These events should be monitored and analyzed in order to detect critical situations and to trigger actions quickly. In addition, monitoring business process events is beneficial for decision making.

2.1.6 Managed Process Applications

A *managed process application* is a software application that monitors a process that contains a set of coordinated activities to achieve specific organizational goals. Moreover, a managed process application monitors and optimizes performance by collecting data from the business process and measures metrics (Tegegne & Peyton, 2011).

A managed process application must collect data to facilitate business process optimization in a flexible manner (Forrester, 2008). It enables an organization to monitor and measure the performance of their business processes (Ko, Lee, & Lee, 2009). These measurements are often used to identify improvements that enhance the overall organization. Metrics are used to measure how well a particular process is being performed (Kronz, 2006).

For a healthcare organization that provides care to patients, one important objective is to provide care to patients in a timely manner. Care processes need to be monitored to ensure compliance with medical guidelines, and the status and location of patients, physicians, medical equipment, beds and other entities needs to be monitored.

In addition to healthcare, logistics companies such as delivery services benefit from managed process applications since their business processes including the tracking of the status

and location of entities (e.g., shipments) need to be monitored. Manufacturing organizations also use managed process applications to monitor their processes and key resources involved in the process.

2.2. Application Development

In this section we give a very brief overview of different aspects of application development that are relevant to the engineering of care process monitoring applications.

2.2.1 Software Engineering Development Methodology

The most well-known but simplistic approach to software methodology is the *waterfall methodology* (Nikiforova & Nikulsins, 2008) in which a rigidly defined sequence of steps which is defined and well documented must be followed. To be successful with this approach, there should be a well-defined understanding of the requirements before development starts. Otherwise, any change in the requirements during development will lead to critical problems (Leau, Loo, Tham, & Tan, 2012) (Singh & Bakshi, 2013). Typically, the following set of software engineering activities are performed (Sommerville, 2011) (Scacchi, 2002):

1. Requirements gathering and analysis. The output is a software specification, which defines the detailed functional and non-functional requirements of the software.
2. System modeling. In this stage, the software is depicted and represented as models using special kinds of graphical notations. These models can be leveraged in either the requirements stage or in architectural design.
3. Architectural design. The purpose is to design the software architecture by assembling the components that construct the overall system and which reflects the requirements

specification. The architecture design is depicted as a block diagram showing all components and the interfaces among them.

4. Implementation. In this phase, the system is implemented using either the conventional programming languages such as Java and C or reusing existing similar systems and refactoring them to meet the requirements.
5. Testing. This phase ensures that the developed software meets both the software specifications as well as the users' expectations. The testing procedure starts from testing program entities such as classes, then component testing and system testing.

Rather than insisting that each of these activities be performed sequentially and completed in their entirety, modern practice embraces an iterative and incremental development methodology (Larman & Basili, 2003) to handle change and uncertainty by delivering the software as increments or versions each with a set of features or functionalities. When the increment is tested and is not satisfactory to the customer then, only this increment will be changed. The two most common approaches currently are the Rational Unified Process from IBM on the one hand, and agile methodologies, like XP and Scrum on the other. One of the key innovations from agile methods has been the concept of test-driven development and, in particular, behavior driven development. A short overview of these methodologies and techniques is given below.

Rational Unified Process

The *Rational Unified Process* (RUP) (Kruchten, 2004) is an iterative software development methodology for developing high quality software within timelines and budget by leveraging the *Unified Modeling Language* (UML). One of the key insights from RUP is the

separation between the dynamic view, which presents the phases of the project over time (Inception, Elaboration, Construction, Transition), and the static view, which presents the activities conducted during the development process (requirements analysis, design, coding, testing, project management). By means of this separation, each phase of the development process is no longer linked to a single activity, but rather all activities are relevant in varying degrees to each phase (Aked, 2003). Another key features of RUP, is that the methodology is configurable depending on the project to accommodate different techniques, tools and even activities for different sizes of development teams and types of applications (Rational, 2003).

Agile Software Development

Agile software development (Martin, 2003) (Biju, 2008) aims to address constantly changing requirements and deliver high quality software quickly.

According to (Cohen, Lindvall, & Costa, 2004), *Extreme Programming* (XP) and *Scrum* are considered as the best-known and widely used agile methods. In XP (Beck & Andress, 2004), the goal is to deliver high quality software and respond to changing user requirements over time by delivering small increments of the system frequently. The requirements are based on user stories which are broken down into several tasks and prioritized by the customer for implementation. The key innovations that have been introduced in XP are:

1. Test- first development. Writing tests for each intended task (system functionality) before implementing those tasks.
2. Pair programming. The developers work in pairs at the same workstation to develop software with one observing and suggesting while the other codes. This facilitates

refactoring the code continuously which enhances the quality of the software and the speed at which that quality is achieved.

3. Customer involvement with the development team. A representative of the customer is part of the development team and participates in the requirements phase by providing the user stories, as well as in the test development and validation.

Scrum (Cohen M., 2009) is similar to XP in that it mandates small increments and customer participation in the development team. In Scrum, though, the small increments are defined as fixed increment "sprints" of between 2 to 4 weeks. The starting point for each sprint is to assess the remaining work in the product backlog and determine which product backlog item (features) should be selected for the sprint.

The key difference between developing software following XP or Scrum is that Scrum does not require features to be defined by user stores, nor does it require test first development or pair programming (Biju, 2008).

Test Driven Development (TDD)

Test driven development (TDD) (Beck, 2003) is an agile method that focuses on writing automated test cases for new functionality of the system prior to coding the functionality.

TDD promotes red-green-refactor cycles (Hammond & Umphress, 2012), where red stands for writing a test case for the small increment of the system functionality that initially fails because the code has not been implemented yet, and green stands for writing the minimal code needed to pass the test, and finally, refactor stands for improving code while ensuring the test cases still pass.

One of the main advantages of TDD is that in order to write correct test cases, the developers need to understand the requirements well before starting to code. TDD has been adopted for small and medium sized projects and has been leveraged by a wide range of developers and researchers (Janzen & Saiedian, 2005) (Jeffries & Melnik, 2007).

Behavior Driven Development (BDD)

Behavior Driven Development (BDD) (North, 2006) is a refinement of test-driven development in which detailed specifications of the desired behavior of the system are determined by each stakeholder. The aim is to foster communication and collaboration between the members of the project (developers, testers, and customer), and ensure that all participants have the same understanding of the system (Sacks, 2012) (Soeken, Wille, & Drechsler, 2012).

In effect, BDD is based on TDD, but different in terms of the way the tests are written and specified. In BDD, tests are comprehensible and are designed in such a way that they assist the development team (developers, business analysts, testers) as well as the stakeholders to specify and write their tests (Solis & Wang, 2011).

One of the main characteristics of BDD is user stories for features, which represent what is performed by the user, and scenarios, which describe the expected behavior of the system when the features are implemented. Both follow a common pre-defined template written in a specific ubiquitous language. Commonly, the template used for user stories is: [Story Title], As a [Role] I want [Feature or Goal] So that I can get [Benefit or Value]. Whereas, the template to describe the scenarios is as following: Scenario 1: [Title], Given [Context], When [Event], Then [Outcome] (Landauber & Genaid, 2012) (Solis & Wang, 2011) (North, 2006). One interesting

feature of BDD is that each step of the scenario is mapped to one test method. Therefore, to pass a certain scenario, all the test methods (steps) should be passed (Solis & Wang, 2011).

Some of the well-known toolkits for BDD are RSpec, Cucumber, and Friends (Chelimsky, Astels, Helmkamp, North, Dennis, & Hellesoy, 2010). Those toolkits used different mapping rules to map the scenarios to test code (Solis & Wang, 2011).

2.2.2 Architectural Patterns

The selection of an appropriate architecture can be just as significant when developing an application like care process monitoring as the type of software methodology used. An *architectural pattern*, according to (Buschmann, Henney, & Schimdt, 2007), is defined as follows: "A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the way in which they collaborate".

The most well-known architectural patterns for resolving problems arise in the area of data integration are the following patterns: *Extract, Transform, Load* (ETL), *Service Oriented Architecture* (SOA) and finally *Enterprise Service Bus* (ESB) (Giordano, 2010). ETL is an architecture pattern that performs data integration of heterogeneous data sources in order to unify data into one common database like a data warehouse. This architectural pattern is widely used in Business Intelligence processes (De Giusti, Oviedo, & Lira, 2011). SOA is an architectural pattern (Mulik, Ajgaonkar, & Sharma, 2008) that supports and defines the communication between software applications, known as services, for the purpose of building enterprise

applications. Finally, the ESB architectural pattern implements the principles of SOA to integrate applications and systems within or across organizations (Gronli & Bygstad, 2012).

Event-driven architecture (EDA), further explained in section 2.3.5, is considered as an architectural pattern (Preuveneers, Yasar, & Berbers, 2008) that is widely used for designing loosely coupled applications and systems that require a high responsiveness to events emitted from different loosely coupled software components and services (Hemani & Shamsi, 2010). In terms of event processing, EDA can be integrated with Complex Event Processing to form a new architectural pattern that can be combined with the SOA architectural pattern (Pottebaum, Artikis, Marterer, Paliouras, & Koch, 2011). We will refer to this combined pattern as EDA-SOA.

In (Sommerville, 2011), the main architectural patterns used in most systems are identified as layered architecture, repository architecture, client-server architecture and pipe and filter architecture. In a layered architecture, *Model-View-Controller* (MVC) is an example in which the user interface is separated from the business logic. *Repository architecture* is suitable for systems in which their different components are sharing and using the same data source or repository. *Client-server architecture* is a well-known pattern in which the services are running in distributed servers and the multiple clients could call and access the same service through a network using Http request. Finally, in a pipe and filter architecture, the components, which are processes that acts as a filters, are connected to multiple input pipes and multiple output pipes. Multiple filters could be running simultaneously and they perform a particular function once the data is available in their input pipes. This type of architectural pattern is considered as a batch model.

2.2.3 Models

System models are the basis for communications among the development team and between them and the system stakeholders in different stages of development. Different types of graphical models are exploited during the system modeling phase. The most common models used in software development are defined in *UML* (Booch, Rumbaugh, & Jacobson, 2005).

According to (Sommerville, 2011), the popular and most frequent UML models are divided into three main categories which are interaction models, structural models, and behavioral models. When it comes to modeling the business processes of an organization, the most readable and well known graphical modeling notations used for this purpose are UML activity diagram and *Business Process Model and Notation* (BPMN) (OWen & Raj, 2003) (White, 2006) (Kalnins & Vitolins, 2006). The main goal of BPMN is to provide graphical notations understandable and easily readable by technical and non-technical users such as business analysts, developers and managers.

User Requirements Notation (URN) (Amyot, Mussbacher, 2012) is a requirements engineering language that integrates goal models with use case maps of scenarios in order to distill and model the user requirements when building information systems. URN can be used to model both business processes, as well as organizational goals (Weiss & Amyot, 2005) in order to indicate which business processes contribute to satisfying which goals. In addition, URN is considered as important requirements tool to model the requirements of any business process monitoring applications in terms of goals and associated KPIs (Pourshahid A. , Amyot, Chen, Weiss, & Forster, 2007). jUCMNav is an example of a URN tool used for visualizing

requirements of the intended monitoring system in terms of goals and related scenarios that describe the behavior of the system (Roy, Kealey, & Amyot, 2006).

Meta-Modelling

According to (Ernst, 2002), a *meta-model* is defined as "an attempt at describing the world around us for a particular purpose". Meta-models should have clear semantics that define the relationships between the elements in a particular domain of interest. From the meta-model, several models could be instantiated, but those models have to comply with the meta-model.

The use of meta-modeling of healthcare organizations has been increasing in the past few years. There exists numerous works for modelling healthcare systems. Winter el al. (Winter, Brigl, & Wendt, 2003) have proposed a meta-model for hospitals information system (HIS). The goal of their meta-model is to facilitate and support architects and information managers in their work. Their meta-model consists of three layers that can be used as a blueprint for designing data models for hospitals.

Meta-models of healthcare processes have been proposed to facilitate the integration of care processes (Hasselbring & Pedersen, 2005). Such approaches attempt to create a consistent interpretation of data from different systems. Sun et al. (Sun, Rahmaniheris, Kim, Sha, Berlin, & Goldman, 2012) proposed a meta-model that describes the design of an acute care monitoring system that assists clinicians to monitor, make diagnoses and decisions based on available information integrated from patients records and measures generated from different devices. Their meta-model is designed to represent the acute care process in terms of three key elements for patients which are measurements, the diagnosed patient's condition and finally the provided

treatment. Cimellaro et al. (Cimellaro, Reinhorn, & Bruneau, 2011) proposed a meta-model to depict the response of the Hospital Emergency Department. The performance metric chosen to represent the response is the wait time. Their meta-model has the characteristic to assess the ED capacity and its performance in real-time. In addition, the meta-model is generalizable to support different hospital configurations.

2.2.4 Application Framework

According to Schmidt et al. (Schmidt, Gokhale, & Natarajan, 2004), an *application framework* is defined as "an integrated set of software artifacts (such as classes, objects and components) that collaborate to provide a reusable architecture for a family of related applications". It provides class libraries and templates that facilitate developing an application in a specific domain. The idea behind the application framework is to mitigate development effort and hence to increase the productivity and to accelerate the production of a certain application. Often, the components and organization of the application framework are based on common design and architectural patterns. Web application frameworks are growing in popularity and are often based on the MVC architectural pattern. In addition, application frameworks are increasingly characterized by models and model driven architecture (MDA).

Struts is an open source Java web application framework that adheres to the MVC paradigm. In Struts, the model, view and controller are implemented using Java Beans, JSP technology and Servlets respectively (Li, Ma, Feng, & Ma, 2006). One of the key benefits of adopting the Struts framework is the ability to develop large web applications at ease. However, the learning curve is high as users have to spend much time learning the standards of the framework (Feng & Le, 2009).

Grails (Grails, 2009) is an open source web application framework that adheres to the MVC architectural pattern. Because Grails follows the MVC pattern, it is a high productivity framework and provides a powerful development environment which hides a lot of configuration details from the developer. The language that is used in Grails to implement the web applications is called Groovy, which is an interpreted Java-like language. One key feature in Grails is built-in support for testing. It provides three types of testing; unit testing for testing the domain classes and controllers, integration testing, and functional testing that is performed by using several plugins available in Grails (Smith & Ledbrook, 2009).

AndroMDA (AndroMDA, 2012) is an example of an application framework which follows the MDA paradigm. The main idea of this application framework is the ability to transform and generate source code for various platforms and technologies from UML models.

BPEL (Alves, et al., 2007) is for a standard for executing business processes online, which is model based. The business process is modeled, and then a BPM engine is able to execute or interpret the model. BPEL assumes a Service Oriented Architecture in which some or all of the activities might be realized by orchestrating or choreographing calls to web services.

2.2.5 Model-Driven Architecture

The goal of Model Driven Architecture (MDA) is to separate system functionality and specifications that are represented as conceptual models from the details on how the system is implemented for the underlying platform (OMG, 2012). In other words, no matter what technical platform is adopted and changed over time, the models that describe the functionality of the

system are not affected. Rather, those models can be transformed and implemented in multiple technical platforms (Mellor, Scott, Uhl, & Weise, 2004).

There are three types of models in MDA which are (Gardner & Yusuf, 2006) (Schmidt, 2006):

1. Computational Independent Model (CIM). This model is called a domain model. It focuses on the requirements of the system and what the system will perform from the business perspective. The model is demonstrated by a specific business language. At this stage, there is no concern on how the system will be implemented.
2. Platform independent Model (PIM). This model captures the functionality and operations of the system independent of any underlying platform that it might run on. Typically, UML models are leveraged to express PIMs.
3. Platform Specific Model (PSM). This model describes in depth how the system is implemented and should conform to the specifications of a particular technical platform.

For any application developed by exploiting MDA, there should be one PIM and one or more PSM (Hailpern & Tarr, 2006). The ultimate goal of MDA is to transform a PIM to a PSM using a transformation model and definitions. Since there is a strong relationship between a PIM and a PSM, for any modification to a PIM, the PSM will be re-created automatically to follow the changes which occur to the PIM (Tözmal, 2006).

Raghupathi et al. (Raghupathi & Umar, 2008) leverage MDA in healthcare information systems to track patient information. They build a prototype application based on MDA to cope

with the multiple technologies and platforms used in the healthcare system. The application is generated from transforming the PIM, which uses UML class diagrams to model the healthcare process, into several PSMs, by developing transformation rules in healthcare.

2.3. Related Technologies

This section surveys technologies related to care process monitoring, including both performance management frameworks and support for integrating events from the operational process side.

2.3.1 Business Process Management

Business Process Management (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" (Van der Aalst, ter Hofstede, & Weske, 2003). It facilitates the online management and tracking of business processes and their data. Often, BPM is supported by service oriented architecture (SOA). The event-driven architecture described in our approach is complementary to SOA.

Typically, the development of a BPM application should go through several repeated steps in order to enhance and improve business process being conducted. According to (Pourshahid, Amyot, Peyton, Chen, Weiss, & Forster, 2009) (Tchemeube, 2013) (Weske, 2012), the life cycle of BPM consists of the following activities:

1. Define and discover the business process based on the goals set by the organization.
2. Model the business process using business process modeling notations available, such as BPMN, UML and URN.

3. Deploy and execute online business processes using a BPM execution engine.
4. Monitor the business process in order to measure its performance against the goals set based on computing particular performance metrics.
5. Improve and optimize the business process by using the results of performance measurements.

2.3.2 Business Activity Monitoring

Business Activity Monitoring (BAM) is often integrated with BPM in order to supply near real-time information about the results of business processes (Kang & Han, 2008). This near real-time information is often used to track key performance indicators (KPI) which are defined to measure the performance of business processes in terms of achieving business goals. Often KPIs are displayed on a dashboard, and integrated with an alerting mechanism in case of a severe difference between the actual value and the expected value for a KPI (Wetzstein, Ma, & Leymann, 2008).

2.3.3 Business Intelligence (BI)

Business Intelligence (BI) is defined as a "set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision-making" (Evelson, 2008). The main technologies used by BI are data warehouses explained in section 2.1.4, analytical tools, which focuses on analysing to deduce insights into the organizations processes, and finally, visualization and reporting tools that provide reports and dashboards to the end users as a result of analyzing the data (Azvine, Zheng, & Nauck, 2005).

A *Data mart* is considered as a view of a data warehouse. In other words, a single data warehouse might have various data marts created for a specific type of business analysis and focused on a specific subject. The structure of the data mart is created following a star schema design that is optimized for reporting (Saxena & Srinivasan, 2013). The star schema is a dimensional database model in which data is stored into facts and dimensional tables. Each star schema consists of one or more fact tables and several dimensional tables linked to them. Fact table contain records represented as events that are comprised of the primary keys associated to the dimensional tables, as well as measures and quantitative values related to those events. Dimension tables have detailed information required to describe the fact records (Inmon, 2005). For example, the star schema design of the Metrics Data Mart for a hospital care architecture, described in case study 2, consists of a Patient State fact table that contains facts about the duration of states related to patients and several surrounding dimensional tables such as patient, date, state and provider tables.

2.3.4 Service Oriented Architecture (SOA)

A *Web Service* is defined as "a software system designed to support interoperable machine to machine interaction over a network" (W3C, 2012). A web service makes the communication between applications more useful without spending much effort in building a customized service when a similar service is available (Ma, 2005).

According to (Brown, Johnston, & Kelly, 2002) *SOA* is defined as "a way of designing a software system to provide services to either end-user applications or other services through published and discoverable interfaces".

One benefit of leveraging SOA in organizations is that business processes could be designed and implemented as services. Therefore, these business processes could be accessible within the same organization or across organizations (Leymann, Roller, & Schmidt, 2002). SOA also facilitates the automation of business process activities (Rabhi, Yu, Dabous, & Wu, 2007). Another benefit of leveraging SOA is that it supports improvement in a business processes that can improve the overall performance of organizations (Consulting, 2006).

2.3.5 Event Driven Architecture

With respect to business processes, an event represents "state changes of objects within the context of a business process that occur during the execution of a process" (Muhlen & Shapiro, 2010). Event processing is performed on the events that are produced from the execution of a business process in order to monitor the performance of the process and detect particular events by raising alerts to the interested parties. Event processing is divided into two categories, simple event processing and complex event processing. In simple event processing, each occurrence of an event is processed without integration with other events to trigger a specific action (Michelson, 2006). In complex event processing, occurrences of events from distributed event sources are correlated in near real-time to infer a new type of event (Leavitt, 2009).

The main components of event-driven architecture (EDA) are: event consumer, event channel, event processor, and event producer (Levina & Stantchev, 2009). Therefore, monitoring applications built with EDA improves responsiveness to events while they are happening (Niblett & Graham, 2005). Publish/subscribe is a common event-driven model (Eugster, Felber, & Guerraoui, 2003) in which consumers can subscribe to an event channel in order to receive a

particular set of events from an event producer. Often, events are delivered as messages through the use of message-oriented middleware (e.g., a message broker) (Mouttham, Peyton, Eze, & El Saddik, 2009).

2.3.6 Real-time Location Services

A *Real-time location service* (RTLS) is an emerging technology used to locate and track objects in near real-time within an organization. It provides near real-time visibility of the object's movement (Guillemette, Fontaine, & Caron, 2008). Different types of RTLS are available and categories based on the types of tags used for tracking the objects. One type is RFID-based RTLS, in which active RFID tags are used to transmit signals to the readers. Another type is Wi-Fi based RTLS (Schrooyen, et al., 2006), which exploit existing wireless communication in the organization. This type of RTLS leverages Wi-Fi tags attached to the tracked object, which transmit signals to Wi-Fi access points. Ekahau (Ekahau, 2012) is considered as one of the popular Wi-Fi based RTLS vendors in the market. The third type is based on *Global Positioning System* (GPS), in which GPS-enabled devices are tracked (e.g., Vehicle tracking) based on satellite communication. Different domains such as healthcare (Schrooyen, et al., 2006) and supply chain management (Denis, Weyn, Williame, & Schrooyen, 2006) have leveraged RTLS to track and monitor their tagged objects.

RFID, which stands for Radio Frequency Identification, is an efficient technology to locate and track objects in near real-time. Two types of RFID tags exist: active and passive RFID (Asif & Mandviwalla, 2005). Active RFID tags are powered by battery and transmit raw data about their identification and location in a particular time to the RFID reader. This type of tag is sufficient for near real-time tracking of tagged objects. Wi-Fi tags are considered as an Active

RFID system, which exploits the existing wireless infrastructure to transmit radio waves to the access points in order to identify their identity and location in near real-time (Xiang, Chen, Wang, Huang, & Gao, 2004). On the other hand, passive RFID tags are not equipped with a battery and remain inactive until they are powered by the waves of the RFID reader when they are in a short distance from the reader.

GPS is considered as a type of RTLS (Guillemette, Fontaine, & Caron, 2008) (Schrooyen, et al., 2006). It is sufficient to track outdoor objects in near real-time. The communication is taking place between the GPS receiver embedded in the mobile device and the satellite. However, GPS is insufficient to track objects indoors since signals cannot go through the building.

2.3.7 Complex Event Processor (CEP)

A *Complex Event Processing* (CEP) system (Luckham, 2002) processes high volumes of continuous streams of events in near real-time. The main goal of CEP is to process events from distributed sources in order to identify meaningful complex events across these sources by analyzing and correlating basic events (Wang & Jin, 2008). Another goal is to correlate these events to detect and respond in near real-time to any business critical situation (Yao, Chu, & Li, 2011). Furthermore, CEP is responsible for event pattern matching, that is, the CEP engine recognizes in a cloud of events those patterns that are significant for the backend systems (Dunkel, 2009).

Significant research has been made regarding event processing that is focused on processing and detecting complex events based on a continuous stream of RFID events such as

in (Wang, Liu, Liu, & Bai, 2006) (Zang, Fan, & Liu, 2008) (Wu, Diao, & Rizvi, 2006) (Son, Lee, Park, & Kim, 2007). To infer complex events from the streams of basic events received from distributed sources, the following steps are taken (Yao, Chu, & Li, 2011) (Wang, Liu, & Liu, 2009).

1. Event filtering of basic events to remove duplicate events.

2. Correlation of basic events from multiple sources to build a view of a higher-level (complex) event.

3. Complex event creation based on pre-defined rules.

2.3.8 Event Driven Business Process management

Event-Driven Business Process Management (EDBPM) "is a combination of actually two different disciplines: Business Process Management (BPM) and Complex Event Processing (CEP)" (Ammon et al., 2008).

The reference model for EDBPM in (Ammon et al., 2008) is a combinational of BPM, BAM, CEP and Enterprise Service Bus (ESB). A key concept in EDBPM is the creation of two new roles that support the reference model. The first role is related to the workflow of the organization and is performed by the workflow modeler. Their duty is to design new business processes and to improve and re-engineer existing business processes. The second role is related to events, which are generated from the different components of the EDBPM, and performed by the event modeler. Their duty is to identify the event sources within the organization, identify which events should be collected in order to view them in the dashboard, identify to whom the

alerts should be sent and finally write rules to recognize complex events automatically when a pattern of events is detected. The reference model is implemented in the logistics domain.

The event driven business process management approach (Ao, He, Xiao, & Lee, 2010) can also be combined with an RFID capture application. This application tracks RFID tags attached to organizational entities. Whenever an RFID tag is detected at a specific location, an event is generated. Both BPM applications and RFID applications can send events to a common event bus. The CEP collects events from the bus and processes it in order to detect event patterns. This approach has been used to implement an RFID-enabled pallet leasing and tracking system in Singapore (Ao, He, Xiao, & Lee, 2010).

2.4. Related Works on Process Monitoring Applications

We identify four distinctive types of approaches to process monitoring applications (SOA-based CEP, RTLS-based CEP, BPM-based CEP, BPM-based Monitoring Dashboard), which we use to classify related work in the literature on process monitoring applications. The related work we survey in these four categories has been applied specifically to care process monitoring, except for BPM-based CEP. However, our application framework includes support for BPM-based CEP (integrated with RTLS), so it seems reasonable to compare our approach to related work in this category. In our opinion, the works in BPM-based CEP could be applied in a straightforward manner to the types of care processes we worked with in our case studies in chapter 5.

Our application framework is comprised of an application model, an architectural pattern and a software methodology for developing CPMA will be described in chapter 4. In terms of the

application model, none of the related works had an explicit application model that captures the complete requirements of the CPMA in an integrated manner. However, most of the approaches described in the related works either had an explicit model for business processes or a model for a data warehouse for a performance reporting dashboard.

As for the application development methodology, none of the related works mentioned explicitly the methodology the authors followed to build a CPMA. So, we assumed that none of the related works followed an agile test-driven methodology specifically defined for building CPMA as we describe in section 4.3. Instead, it seems reasonable to assume that they followed a generic software engineering approach similar to what we did in developing a CPMA in our initial case study as described in section 5.2, or that methodology was not a focus of their work and that any prototype CPMA described was implemented in an ad hoc manner to demonstrate the architectural approach they advocated.

Ultimately, we grouped the related works into four categories based on the architectural approach that was the focus of the related work. The architectural pattern we present in chapter 4.2 can be seen as a mechanism for unifying and understanding the complementary aspects of the different architectural approaches in the related work.

The first category of process monitoring application related work we have identified is SOA-Based CEP. This type of monitoring application is concerned with leveraging CEP as a core component where the events are received from disparate event sources through SOA. The goal of this approach is to process and correlate those events in real-time to infer meaningful events in order to detect critical situations and report on that in real-time.

The second category is RTLS-Based CEP. This type of monitoring application uses CEP as a core component and mainly focuses on processing, correlating and integrating, in real-time, location events that are generated from RTLS and that are related to the resources participating to the care processes. The inferred events can be further processed and analyzed to trigger alerts and can provide performance reports for care processes.

The third category is BPM-Based CEP. This type of monitoring application uses the concept of CEP to monitor the underlying care process and provide performance reports by processing and correlating events while the care processes are taken place in real time. The execution of the care processes is highly depended on the inferred events generated from CEP.

The final category is BPM-Based Monitoring Dashboard. This type of monitoring application relies entirely on a BPM system that integrates data from disparate information systems while the care process is taking place to provide performance reports that show the state and the status of the care processes at a given time.

We present the related works below in the light of these four categories. In chapter 6, we will compare our application framework approach for engineering care process monitoring applications to each of these categories of related work.

2.4.1 SOA-Based CEP

Middleton et al. (2009) proposed a surveillance portal architecture for monitoring compliance with quality of care policies. The architecture supports a managed process application for monitoring patients who receive care at home. It has significant overlap with our business monitoring architecture. In particular, a message broker routes and collects distributed

event messages from patients and providers, and events are filtered, correlated and logged in an event repository against which statistics and metrics are collected. However, there is no model-based definition of care process monitoring that links business process events and sensor events (RFIDs) in support of metrics for a particular care process. Rather, individual policies are analyzed in an ad hoc manner and monitored by custom-built agents for each policy. Messages are collected on an as needed basis from hospital information systems and patient portals and correlated by policy-specific agents. Their work, in particular, is focused on compliance monitoring of a palliative care processes for patients in the community receiving at home care, clinic care, long term facility care, and even hospital care. The surveillance portal architecture supports performance management reporting by capturing all the events from the distributed hospital information systems in a data mart that supported a performance reporting dashboard. The main focus, though, is on generating alerts when a specific policy is violated.

Vaidehi et al. (Vaidehi, Bhargavi, Ganapathy, & Hemalatha, 2012) proposed a healthcare monitoring architecture for the purpose of monitoring geriatric patients (elderly patients) at their home. Particularly, the objective of their architecture is to generate alerts and warnings to the caregivers and doctors and even for the patients themselves. Complex event processing (CEP) is the main architecture component leveraged for collecting, processing and analyzing continuous sensor data generated from patient's home in order to detect abnormal conditions of the elderly patients. Also, the concept of SOA is exploited for integrating diverse sensor data and making that data available over the Internet for processing by CEP to provide meaningful information to doctors.

Since the authors focused on monitoring elderly patients at home, they assumed that the observed patient should be wearing physiological sensors and RFID reader. In addition, they assumed that the patient's home is equipped with motion and environmental sensors (i.e., cameras and RFID tags) as well. The entire data stream from the sensors is in XML format. In terms of the system architecture, all the generated data from the sensors is sent to the application running at a PC located at the patient's home. Then, the data are available to the central server, where CEP resides, through web service calls. From there, alerts and warnings are sent to the doctors in order for appropriate decision making.

The architecture is designed to generate alerts and warnings related to the elderly patients in near real-time according to the data generated only from the patient's home environment. However, performance reports and tracking metrics are not discussed in the authors' literature. There is no systematic approach to defining a data mart or a performance reporting dashboard. There is no model explicitly relating events to states of a care process. Compared to our architecture (which is presented in chapter 4.2), there are some missing architecture components such as a message broker and a BPM engine. With respect to the evaluation the system, it is still in the early stage of development and hence a considerable amount of future work is needed to fully develop and deploy the system.

Boubeta-Puig et al. (Boubeta-Puig, Ortiz, & Medina-Bulo, 2011) developed a SOA-Based CEP. The main goal of their approach is to detect disease in an early stage, by gathering and processing events from distributed services once the events are available, and hence, minimize the latency in decision making. In their proposed approach, the authors exploit Enterprise Service Bus (ESB) as a message-oriented middleware. In addition to that, rather than

using a data mart and performance reporting dashboard, they leveraged a NoSQL database management system in order to store the published events which were used later on for analysing historical data.

In their work, Boubeta-Puig et al. (2011) developed a case study in the healthcare domain to demonstrate the problem and validate their approach. The key goal of the case study is to detect and alert pandemic and epidemic for avian influenza in near real-time, rather than using the existing tools in this domain which only report on the sudden increase of the avian influenza one week after the fact. In the case study, they defined several complex event patterns for detecting the outbreaks of the avian influenza and each pattern contains specific rules drawn from the World Health Organization (WHO). Once those patterns are detected by CEP, near real-time alerts are sent to the desired consumers.

Their solution focused mainly on sending warning alerts to the interested parties. There is no mention of developing a data mart or a performance reporting dashboard that facilitates monitoring the outbreaks of the avian influenza. In addition, reports are missing and not implemented. Moreover, they do not focus on linking metrics, which can be used for reporting, and event sources to health states of the patients. Because of the access limitation to the hospital and laboratories information systems, the authors created an event generator in order to simulate the source events by simulating randomly the patients and their health states.

2.4.2 RTLS-Based CEP

Wen Yao et al. (Yao, Chu, & Li, 2011) developed an RTLS-Based CEP. They proposed a sensor CEP architecture for processing events from a wide range of sources including RFID tag

readers and embedded sensors in order to update the location and status of business entities (e.g., patients) in response to significant medical events. These distributed events can be combined with patient medical records to detect a critical state change or to infer a semantic event which is sent either to a data warehouse, or healthcare IS, or other applications. Their work monitored surgical care processes.

In their architecture, there is no BPM engine to manage business processes. Instead messages are extracted directly from hospital information systems through SQL queries. There is also no message broker to coordinate and transform events from disparate sources into a single business process event stream. However, the Drools CEP engine filters, correlates and logs basic events in a manner similar to our integrated CEP architecture. Location events are used to track any object within the hospital (e.g., physicians, nurses, patients) by using RFID tag readers.

The main focus in their implementation is patient identification and patient monitoring (e.g., monitoring vital signs). Business process metrics (such as computing wait times for patients in different stages of the business process) and identifying process bottlenecks are not specifically addressed. There are no models that would explicitly link metrics to business processes and the events aggregated to compute metrics.

2.4.3 BPM-Based CEP

Schlegel at al. (2012) proposed an approach to monitor and manage cross-company business processes which they are enacted by multiple service providers using different BPM engines. The idea behind their approach is to build and implement a service platform that monitors and integrates events generated from different BPM engines exploited by different

service providers, as well as events generated from other applications through CEP as a focal component. Events sent from different BPM engines are defined using an XML schema and the event schema is comprised of all the potential events that can be created from different BPMs. This type of monitoring application is not designed to provide analytics to monitor business processes; rather, it is only concerned with managing the execution of cross company business processes enacted by different process execution engines. The authors developed a prototype system and used a scenario in the insurance domain to illustrate and validate their approach.

Janiesch et al. (2012) proposed a BPM-Based CEP architecture. They developed an event-driven architecture for monitoring business processes of an organization in near real-time by extending the notion of Business Activity Management (BAM). Typical BAM systems are designed for observing events during the business process execution and visualizing them in near real-time. In addition, the authors' architecture is designed to make BAM a component that can trigger instant actions according to the analysis of the events. To this end, the authors proposed a solution by integrating BPM with CEP, which is similar to a part of our architecture. In this situation, CEP has the ability not only to process the events from BPM but also to control and keep an eye on BPM based on inferred events. Their conceptual architecture is comprised of three key entities, namely, event producer (BPM and other systems), event processor (CEP) and event consumer (BPM, Visualization and other consumers such as messaging service). RTLS is not an aspect of their architecture.

In order to evaluate their conceptual architecture, they built a proof-of-concept system using a mix of commercial off-the-shelf software (COTS) for BPM, CEP and dashboard, as well as custom components embedded with BPM and CEP. Similar to our approach described in

section 4.2, the communication between the entities of their architecture is based on Apache ActiveMQ with a typical publish/subscribe component (message broker). For testing purposes, they demonstrate their work with a real scenario in the area of supply chain management, specifically, order fulfilment process.

2.4.4 BPM-Based Monitoring Dashboard

Tegegne et al. (Tegegne & Peyton, 2013) proposed a model driven framework for near real-time monitoring of managed processes that supported run time configuration. They focused on engineering managed process applications that captured events from workflow activities and report on performance management of the managed processes. One essential part of their framework is that a CPMA is realized by building different models of different aspects of the CPMA and then having their run-time environment interpret and execute those models similar to the manner in which a BPM engine interprets and executes BPEL. The various elements of the business process monitoring application (e.g., workflow, event, alert, entity, performance indicator) are modeled declaratively in an XML and interpreted dynamically rather than hard-coding them into the application. The runtime environment is composed of an application engine that coordinates between run time services such as workflow and alerting services. Their framework addressed palliative care processes similar to (Middleton et al., 2009) but focused on event collection to measure performance metrics. Events are simply defined as the data submitted by simple forms defined as the steps in a work flow. There is no complex event processor component in their architecture. Alerts are triggered when target values within a form were out of bounds. There is no attempt to integrate the models and capture relationships between them.

Zhang et al. (2009) developed a BPM-Based Monitoring Dashboard. They proposed to leverage workflow management systems to model and enact distributed business processes and to integrate legacy systems of the radiology department of the hospital located in China. They implemented their system using a Yet Another Workflow Language (YAWL). Another team from the same research lab, Zhu et al. (Zhu, Nie, Lu, & Duan, 2010), exploited the work of Zhang et al. (2009) to develop a monitoring dashboard that has the ability to collect data generated from the workflow management system and to integrate it with data from other radiology systems in order to link the process information with patients. The purpose of the dashboard is to enable the managers and the administrators in the radiology department, in near real-time, to monitor the comprehensive states of the radiology processes.

Their workflow-based monitoring dashboard, gained insight into the current executing radiology process by providing charts that show the execution time for each individual task of the process and histograms that present the computation of two key interested metrics; the average examination and reporting turnaround time.

Their workflow-based monitoring dashboard is evaluated and reviewed by the administrators and managers of the radiology department. They reported that the system is beneficial in helping to detect problems especially if the execution time for certain tasks (e.g., examination turnaround time) does not conform to radiology regulations. Moreover, even though the system is unable to provide explicit states for the radiology process, they reported that they could infer the states of the workflow from the dashboard and make decisions. In their proposed approach, the states (i.e., wait times and service times), and the start and the end time for each state are not modeled. Instead, the duration of service states are deduced by interpreting

information displayed in the dashboard. Also, the only wait states considered is the waiting between two tasks in the process. In addition to the above, there are no model to determine the metrics that should be monitored and collected and how those metrics are computed from the radiology processes. Furthermore, the alerts are triggered in an ad hoc manner by the managers and the administrators based on their knowledge. In general, there is no model that linked metrics to radiology workflow states.

2.5. Chapter Summary

This chapter has presented background concepts that are useful to understand the contents of the thesis. We provided background on care process monitoring and managed process applications, and then we explained related concepts such as business analytics, performance management and data warehouse. We covered briefly the approaches and technologies for application development process that are useful to the engineering of a Care Process Monitoring Application (CPMA). We then gave an overview of related approaches and technologies that are important to monitor, process and integrate events from the underlying care processes and to provide analytics and performance management. Finally, we provided a literature review on related works for engineering care process monitoring applications. Representatives of the four main types of alternative approaches identified will be compared in chapter 6 to the application framework approach we advocate in this thesis.

The next chapter will describe the problem and challenges of engineering a care process monitoring application (CPMA) for hospital and community care. In addition, we will provide a gap analysis, and define a set of evaluation criteria used for evaluating our application framework.

Chapter 3. Problem Definition

Our main hypothesis is that an engineering process, based on an application framework, will be repeatable, produce CPMAs of consistent high quality, and require significantly less time, less effort and less complexity compared to current approaches found in the literature or used in health care organizations today. In section 3.1, we give a detailed definition of a CPMA and the current state of engineering for both community and hospital care process monitoring. In section 3.2, we provide a gap analysis to understand where existing approaches to monitor care processes are problematic and deficient. Finally, in section 3.3, we identify a set of evaluation criteria for evaluating and comparing different approaches to engineering CPMAs that will be the basis for evaluating our proposed application framework for CPMAs, and for comparing our approach to related work.

3.1. Engineering of Care Process Monitoring Applications

A Care Process Monitoring Application (CPMA) is a software application that collects and integrates data from various sources while a care process is being provided, in order to provide performance reporting of metrics that are used to measure how well the performance goals and guidelines for the care process are being met. The biggest challenge in engineering a CPMA is to understand what low level data should be collected at what point in the care process from what operational systems in order to compute which metrics.

In section 3.1.1, we define the functionality of a Care Process Monitoring Applications (CPMA) in general distinguishing between two types of care processes: those that deliver care in

the community and those that deliver care within a hospital. The former is explained in more detail in section 3.1.2, while the latter is explained in more detail in section 3.1.3.

3.1.1 Care Process Monitoring Application (CPMA)

Based on our review of related work, and our interaction with domain experts in health care during our three case studies (see chapter 5), a CPMA should provide the following two functions:

1. Track the current state of the care process.

A CPMA should be able to track the current states of different patients, care providers and other resources as the care process is delivered. The CPMA should have the ability to collect, interpret, and log relevant data at each state in a care process from diverse sources within a single organization or across organizations. As a consequence, a CPMA should be able to track particular events that indicate a transition from one state to the next for the care process. A CPMA should be able to determine whether the status of a resource in a specific state is good or not.

2. Report on performance measures for the care processes.

A CPMA should be able to report on performance measures for the care process and communicate how well the overall care process is meeting quality of care goals. It should also provide insight into how well each step of the care process is contributing to meeting a quality of care goal. If a quality of care goal is not being met, it should be possible to see where in the care process there are problems. What states are problematic, what resources are missing or

underperforming should be communicated. It must be possible to explore the performance of the care process along different dimensions (time, service, location, patient, etc.,).

3.1.2 Community Care Process Monitoring

The main goal of community care process is to treat the patients in the community and, as much as possible, avoid emergency room visits, and keep patients healthy enough by managing their pain and symptoms with regular appointments and quick response to emerging situations when necessary.

The key challenge in monitoring care processes delivered in the community is that care providers from different healthcare organizations are involved. When a care process is delivered in the community, each participating organization maintains the relevant data in its own clinical information system. As a result, there is a lack of visibility across care providers regarding the patients' states in a given period of time. In this situation, each healthcare organization can only provide performance management reports and complete analytics towards the part of the care process that it has performed. Reporting on the quality of care provided for the overall community care process can take months of manual work to integrate data from different sources, and some of that data may be inaccurate or incomplete.

A CPMA for community care needs to be able to communicate the following:

1. Identify what patients, healthcare organizations and individual care providers are currently involved in the community care process.
2. Monitor alerts, raised due to emergency situations by patients, and measure the wait time for an appropriate response to the alert.

3. Provide accurate and complete performance management reports across the whole community for the care process to ensure that an appropriate quality of care is achieved.

Current approaches to a CPMA for community care are shown in Figure 3-1 (Kimball & Ross, Health Care, 2002) (Middleton G. , Peyton, Kuziemsky, & Eze, 2009) .

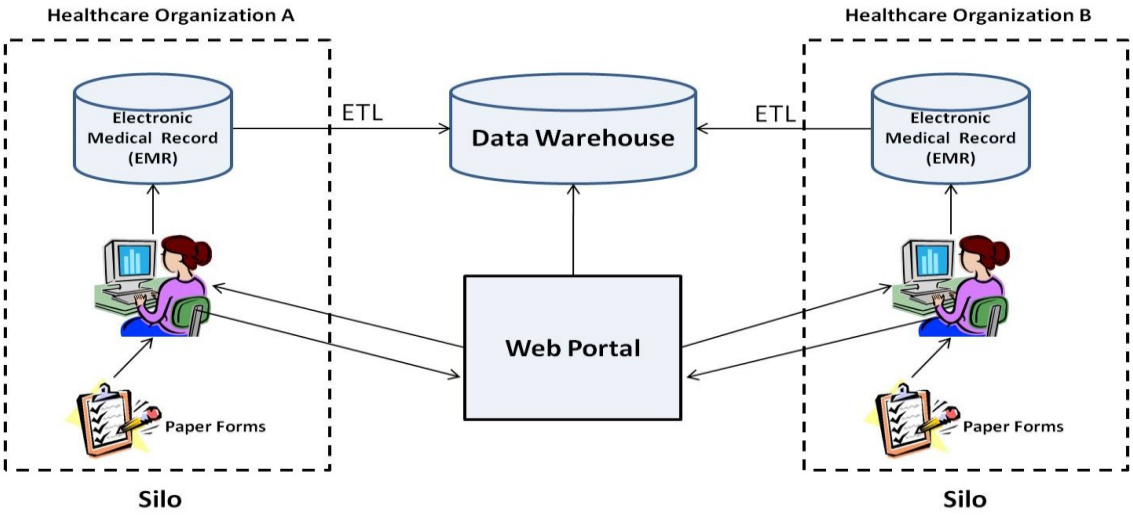


Figure 3-1: Current Approaches for Community Care Process Monitoring

Traditionally, each organization is responsible for maintaining its own electronic medical record (EMR). This process is achieved by entering the data related to the patient, collected from the medical paper forms, into the EMR system. At this point, the information entered is not shared with other healthcare organizations and it is difficult to measure performance of the care process across the community.

There are two possible approaches to address the problem. The first approach is to extract the data from the EMRs through ongoing ETL processes. One of the drawbacks for following this approach is that each EMR must have its own ETL process defined, and there may be

missing information or conflicts in how different EMRs collect information. In the second approach, the data needed to track the care process can be manually entered into a web portal. In this case, there is a burden imposed to enter data for a second time that already exists in the EMR.

In both approaches, the key is to understand the relationship between care processes and their performance measures in order to focus on maximizing the value of performance management provided while minimizing the data entry burden. The essential problems that need to be addressed are:

1. How to capture the relationship between the care process and outcomes?
2. How to identify the minimal data set needed to measure outcomes? This can greatly reduce the ETL or data entry effort.
3. How to ensure that the organizational goals of the care process are linked to data collection forms: either in the EMR or in the web portal?

3.1.3 Hospital Care Process Monitoring

The key challenge in monitoring care processes delivered in a hospital is to ensure that patient-related data is processed efficiently to minimize wait times for services and effectively use resources available in the hospital. This is difficult since patient-related data comes from information systems within and across different departments of the hospital. The processing of data, once it is available in the department information systems, must occur in near real-time.

A CPMA for hospital care needs to be able to:

1. Measure wait times for services and time spent in these services for patients and other resources involved in the care processes.
2. Raise patient-specific and system-wide alerts when wait times or service times approach or exceed specified targets.
3. Provide precise and complete performance management reports, in near real-time, to insure that the quality of care goals are achieved across different departments of the hospital, on patients, rooms, beds, and other resources' information involved in the care processes.

Current approaches to a CPMA for hospital care are shown in Figure 3-2. Typically hospitals have a traditional data warehouse architectures to support performance management reporting (Kimball & Ross, Health Care, 2002). In a traditional data warehouse architecture, the collection of data is a batch process of extracting data from disconnected data sources, and integrating them into a data warehouse view over a period of weeks to give a view of what happened in the past (i.e., historical reports). As a consequence, because of the disjointed nature of the data, it is challenging to provide, in near real-time, fine-grained monitoring of resources participating in different steps of the care processes, as well as the detailed analysis of bottlenecks in the care processes (Mouttham, Peyton, & Kuziemsky, 2011). Only coarse-grained monitoring is available, which is insufficient for monitoring the statuses of stakeholders, such as nurses, physicians, and patients, as well as the status of resources, such as medical equipment, rooms and beds.

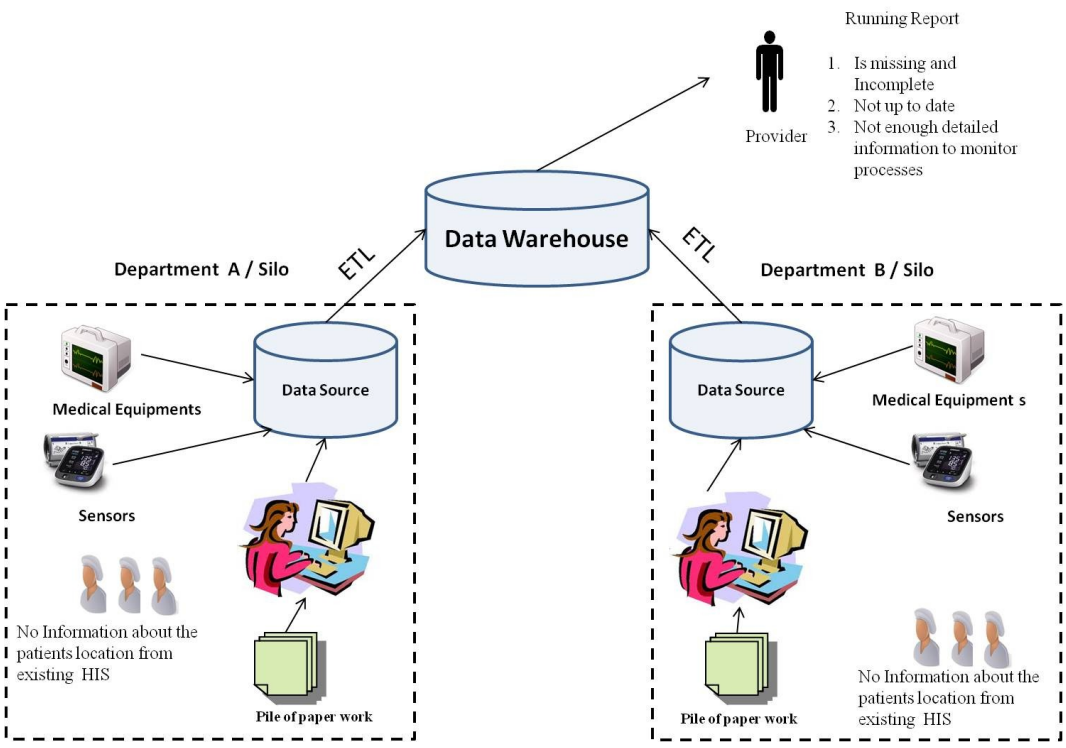


Figure 3-2: Traditional Data Warehouse Adopted by Hospitals

Existing hospital systems are not designed to handle near real-time data. In fact, many departments' records are collected from paper trails and are only recorded in the system hours after the fact. Another potential data source is medical equipment and sensors. Each department in the hospital maintains its local information related to the processes that are enacted in that department.

Given a hospital care process, which spans different units of the hospital and which shows the flow of resources (e.g. patient, physician, nurse, bed, etc.), the fundamental problem a hospital CPMA must address is how to provide near real-time analytics to monitor wait times, individual states and anomalies (alerts) associated to particular resources while the care process is executing. This must be achieved while minimizing the data entry burden for care providers

involved in the delivery of the care process. In addition, to the basic requirements of the CPMA for community care, a CPMA for hospital care must be able to do the following:

1. Precise on-the-fly identification of key events

Identify the start and the end time of different types of wait times for services (patient wait times, operations wait times), as well as the start and end time of services in near real time, in different key points of the care process as events are occurring. This requires identifying what events are needed to determine the start and end time of different types of services. It also requires determining from which sources those events have to be collected.

2. Fine-grained monitoring.

Provide fine-grained monitoring of process states and performance metrics on the fly. Providing such information facilitates measuring wait and service times that are linked to the care process.

3.2. Gap Analysis

We have analyzed current approaches for monitoring hospital and community care processes in use at health care organizations today as well as approaches described in related work. We have also investigated potential technologies and approaches for monitoring care processes, as discussed in chapter 2. Table 3-1 gives an overview of current approaches and their limitations.

Table 3-1: Current Approaches and Limitations

Approach	Limitations
Paper-Based Data Collection and ETL-Based Data Warehouse	1) Paper based data collection followed by electronic data entering into a variety of heterogeneous departmental systems followed by ETL into a central data warehouse is a tedious process that is error prone and only provides coarse-grained monitoring. 2) ETL batch processing on a nightly or weekly basis provides only historical reports and gives a partial, high-level view at best. 3) There is no model that shows the relationship between performance management and the underlying care processes. 4) The analytics that are available are historical and do not reflect current process state
SOA-Based CEP	1) Requires modification of existing software systems and infrastructure to be able to publish and collect events (Event-driven architecture). 2) There is no model that shows the relationship between performance management and the underlying care processes. 3) By itself, does not provide technology needed to instrument care processes to collect events while the care process is taking place instead of paper-based data collection. Therefore, the analytics that are available are historical and do not reflect current process state. 4) Timely performance analytics are not available at point of care for care providers. 5) Methodology to follow in engineering CPMA is not as well-defined as for ETL data warehouse.

BPM-Based CEP	1) Requires interfaces to existing software systems and infrastructure to be able to instrument care processes online. 2) There is no model that shows the relationship between performance management and the underlying care processes though a custom model can be built for each process. 3) Care processes are instrumented to collect events while care process is taking place but requires custom coding to publish events for reporting. Events are limited to explicit actions of care providers or data collected from existing systems (historical data). Therefore reporting on the current state of a process, especially with regard to wait times, is limited and coarse-grained. 4) Timely performance analytics are not integrated at point of care for care providers. 5) Methodology to follow in engineering CPMA is not as well-defined as for ETL data warehouse.
RTLS-Based CEP	1) Does not require interfaces to existing software systems and infrastructure to be able to instrument care processes online but there is also no defined model of the care process. 2) There is no model that shows the relationship between performance management and the underlying care processes though a custom model can be built for each process. 3) Events are limited to location of resources (patients, providers, equipment). Therefore reporting on current state of process is limited and coarse-grained. 4) Timely performance analytics are not integrated at point of care for care providers. 5) Methodology to follow in engineering CPMA is not as well-defined as for ETL data warehouse.
RTLS and BPM-Based CEP	1) Requires custom work to integrate with existing software systems. 2) The relationship between the performance management and the underlying care processes is not clear. 3) Methodology to follow in engineering CPMA is not clearly defined.

In summary, the traditional ETL data warehouse approach has a well-defined architecture and methodology, but the relationship between performance measures of quality of care goals and the low level details of the care processes stored in operational systems is not obvious. More importantly, it does not process data in near real-time. The other approaches present innovations in architecture that address how to process data in near real-time, but the relationship between performance measures and operational systems is equally not obvious. A methodology for developing a CPMA with these technologies is not as clearly defined as in the traditional approach. Our thesis aims to address this gap with an application framework for engineering CPMAs that includes:

1. A model to clearly express the relationship between performance measures and operational processes.
2. A general architecture pattern for care process monitoring that sets the architectural context within which the CPMA must operate in order to leverage emerging technologies like BPM, CEP, RTLS as well as BI and online forms.
3. A methodology to define systematically the steps needed to build a CPMA.

3.3. Evaluation Criteria

To address the problems described in sections 3.1 and 3.2, we need a set of evaluation criteria to evaluate and assess approaches for engineering CPMAs. The evaluation criteria below are represented as check lists that can be leveraged to analyze any framework or proposed solution for addressing the problem of engineering CPMAs. The evaluation criteria identified in this section were determined based on:

1. Careful analysis of related work in the literature survey related to monitoring business processes.
2. Gap analysis of the current practices in hospital and community care against the potential technologies and approaches, including what is needed to be met.
3. Medical accreditation requirements and government regulations (Accredidation Canada, 2013).
4. Feedback and discussions, on a weekly basis, with a domain expert who was a member of the hospital team we collaborated with in our case studies and a member of our research team.
5. Periodic reviews of the CPMA from the hospitals we collaborated with (Osler Hospital and Bruyère Hospital) as well as reviews from representatives from other hospitals that came to our lab (the Ottawa Hospital, Queensway Carleton, and visitors from King Mongkut University in Thailand).
6. Experiences we had early in our case study work described in chapter 5 concerning both hospital care processes and community care processes. As we built our CPMAs, we learned what aspects of the engineering were difficult that should be improved.

The evaluation criteria are used in chapter 6 to evaluate our proposed application framework and compare it against the related works described in chapter 2.

3.3.1 Engineering Effort

Engineering effort is concerned with the energy and time required to develop a CPMA. It also includes consideration of to what extent existing CPMAs can reused when building new

ones. The level of complexity involved to specify, implement and test such systems is also relevant.

1. **Requirements Effort**: The requirements gathering and documentation for any CPMA should be in terms of goals, metrics, care process states, events and their sources. The benefit of using URN modeling tools to model the requirements of the business process monitoring applications in terms of goals and the associated KPIs is well known from the literature (Pourshahid et al., 2007). However, from the experience we had in the case studies, the effort needed for gathering requirements and the amount of time needed to document the requirements depends on how systematically the requirement elements (i.e., Goals, Metrics and States) are collected from the domain expert.
2. **Level of Coding**: This measures the coding complexity of the desired CPMA. Traditional web applications are hardcoded and built from scratch, and the level of coding is quite high. In contrast, the level of coding for developing CPMAs depends on how much of existing code is leveraged. In other words, the level of coding for developing any CPMA is low when it can be configured based on a model definition. This criterion derives from the related work (Tegegne & Peyton, 2013) in which the authors assure that complexity of coding is an essential criteria that should be taken into account when it comes to implement managed care process applications. The potential values for this criterion is either low level or high level coding.
3. **Tool Support**: Utilizing development tools to build a CPMA can be efficient, to some extent, and can conserve time and effort for software engineers; however, these

tools could be complex and challenging since they require a special kind of configuration in order to establish the development environment. The source of this criterion comes from the literature (Liu, 2010) and the related work (Tegegne & Peyton, 2013).

4. **Learning Curve**: This concerns the time that the developers of CPMA require to learn particular tools, programming languages and the development environment. It has the influence on delivering CPMA at the agreed time. The source of this criteria is drawn from the related work (Boubbeta-Puig, Ortiz, & Medina-Bulo, 2011) in which they argue that learning curve is a significant aspect to consider when developing a care process monitoring tool. The potential values for this learning curve criterion to develop CPMA are low and high.
5. **Event Processing and Metrics Specification**: This effort concerns the steps to design and define different types of events, as well as the effort to define the relationships between events and metrics. This effort consists of designing rules on how to create inferred events. This criterion is derived from the related work (Yao, Chu, & Li, 2011) ,where they discussed how event definition and complex event rules for inferring meaningful events are important for developing monitoring applications that collect events from distributed sources within healthcare organizations. The feedback we received from the domain expert confirms the value of exploiting basic and inferred events to compute the performance metrics.
6. **Implementation Effort**: This measures and shows what efforts, tools and software are required to implement the desired CPMA. The effort for configuring any CPMA, which exploits emerging technologies, should be taken into consideration. From the

experience we had with the case studies, we found that the technologies and the tools leveraged in the implementation play also a critical role on the amount of time required to configure a CPMA.

7. **Modeling Effort**: The importance of leveraging UML models in different phases of the software development is well known in the literature (Sommerville, 2011) (Booch, Rumbaugh, & Jacobson, 2005). This criterion is concerned with what efforts are required for modeling CPMA in terms of tools or techniques support.
8. **Testing Effort**: This describes what tools or techniques are leveraged to generate simulated events for testing CPMAs before the real integration of a CPMA within the enterprise. As stated in the related works (Yao, Chu, & Li, 2011) (Boubbeta-Puig, Ortiz, & Medina-Bulo, 2011), it is important to leverage event generator tools to simulate events for the purpose of testing the monitoring application due to the restrictions on accessing the existing information systems in the healthcare organizations.

3.3.2 Application Features for Care Process Monitoring

The following evaluation criteria are identified as features of the CPMA that the engineering approach should support.

1. **Integrated Care Process Forms**: Existing clinical practice, both in hospitals and in the community, records progress through a care process via forms. It is essential for a CPMA to be able to collect data from the care process forms in near real-time in order to report on those data. This criterion was derived from the experience we had in the

case studies and from feedback and discussions from the domain experts, as well as from the periodic reviews of CPMA with the Bruyère and Osler hospitals.

2. **Near Real-Time Data Integration**: The integration of health data across heterogeneous and disparate Health Information Systems (HIS), in near real-time, is a significant concern for healthcare organizations, and is required to monitor the effectiveness of care process and ensure improvement in healthcare delivery. In addition, in the related work (Zhang, Lu, Nie, Huang, & Van der Aalst, 2009), the authors emphasize that data integration is essential for providing an integral view of the overall care processes that are enacted by different source systems.
3. **Performance Management Reports**: A complete reporting tool should be supported by CPMA to measure the effectiveness of care processes against the healthcare organizational goals. The importance for this criterion is discussed in the related work (Middleton G. , Peyton, Kuziemsky, & Eze, 2009).
4. **Usability at Point of Care**: It is important that a CPMA should be usable and be able to provide performance reports at the point of care as a reference while care is ongoing. This was emphasized by the reviews and feedback from both Bruyère and Osler hospitals. In addition, our domain expert confirmed that this criterion is essential, especially as any CPMA should be effective by allowing the managers and the clinicians to drill down to a specific care process details, such as state changes and performance measures for each resource/patient/provider in a particular care process.
5. **Tracking of Current State and Status of Individuals in Care Process**: This feature is essential for a CPMA. Being able to track what step or stage of a care process individual patients, for example, are currently at and provide information on that step

(i.e., time spent in that step) is critical to measure performance of the care process against goals, as well as to identify where in the care process bottlenecks are occurring. The related work (Zhu, Nie, Lu, & Duan, 2010) discussed the importance of having a monitoring tool to track the current state of the care process. In addition, the domain expert argued that this criterion should be fulfilled by any CPMA.

6. **Near Real-Time Alerts**: It is important for any CPMA to raise alerts, in near real-time, to flag a violation in a certain regulation or to flag that particular states exceed the target which is defined according to the healthcare organizations regulations. This criterion is derived from the medical accreditation criteria for palliative care patients (Accredidation Canada, 2013), as well as the review of the CPMA from Osler Hospital.

3.3.3 Application Features for Hospital Care Process Monitoring

All the evaluation criteria mentioned in the previous section are relevant application features for hospital care process monitoring. In addition, the following three criteria are of specific importance to hospital care processes, which are typically more complex and specify more steps in more detail that must be accomplished in a short amount of time.

1. **Near Real-time Performance Reporting**: It is important for CPMA to monitor the performance of the care process while it is taking place. Responsiveness within hours or a day is usually acceptable for community care, but responsiveness within seconds or a minute is required for many hospital processes. The domain expert emphasized that near-real time performance reporting should be a requirement for any CPMA and all of the hospital reviews of our case studies confirmed this.

2. **Event Collection**: Collecting events from electronic systems in addition to care process forms, as they are happening due to the execution of care processes, are required for any CPMA to monitor details of care processes to detect critical situations. All the hospitals we interacted with confirmed the need to integrate with complex enterprise architectures. The importance of event collection is well known in the related work (Yao, Chu, & Li, 2011).
3. **Metrics Granularity**: Levels of metrics granularity supported are fine-grained and coarse-grained metrics. In terms of identifying and measuring fine-grained metrics, according to our gap analysis requires collecting, integrating and processing events from heterogeneous source systems. The time a patient spent from triage to discharge is an example of a coarse-grained metric; whereas measures of the time for every service (i.e., physician assessment) and patient wait time for such services are examples of fine-grained metrics. The domain expert strongly recommended that the CPMA should provide fine-grained metrics to discover bottlenecks in the flow of care processes and help in root cause analysis. This was validated by both Osler Hospital and Ottawa Hospital during their review.

3.3.4 Application Definition

The fundamental design step in developing a monitoring application is to define and configure the way the application should be operated. The following are the core elements that should be identified for developing any CPMA.

1. **Identification of checkpoints and states in care process**: Based on the feedback we received from the domain expert, one of the main steps to defining any CPMA is to

define for what states which resources (entities) should be monitored in the care process. The value of defining states is to measure the fine-grained metrics associated to those states.

2. **Identification of metrics**: According to the discussions with the domain expert, it is essential to define the performance metrics that should be tracked by a CPMA to monitor the performance management of the care process.
3. **Identification of event sources**: According to the experience we had with the case studies, it is necessary to define the source systems where the events come from. By configuring a CPMA, in terms of event sources, the application will be able to track the source systems and monitor the required events to update the resources' states in a particular care process.
4. **Relationship between performance management and care process**: Since there is a gap between the performance management and the underlying operational care processes, as we experienced in the hospital and the community care process monitoring case studies, the definition of any CPMA should bridge the gap and it should show how performance, in terms of metrics, is measured from the checkpoints and states of the care process.
5. **Relationship between care process and sources of events**: Based on the discussion and feedback received from the domain expert concerning the problem on how to monitor the details of care process, in terms of events, the definition of any CPMA should be able to show how the events that determine the start and the end of each care process state, which represent the operational care process, can be linked to their actual source healthcare information system.

6. **Model-defined CPMA**: According to (Raghupathi & Umar, 2008), model-driven development is invaluable for modeling and developing complex healthcare care processes in addition to defining the application models. But, at a minimum, since the integration of a CPMA into the enterprise architecture is complex, it is important that once the relationship between data architecture of the enterprise, business processes and the performance management have been captured in the requirement models the implementation should conform to those models.

3.3.5 Architectural Requirements

In order to build a thorough care process monitoring application that fulfilled the requirements discussed in section 3.1.1, the following architectural requirements are needed to support building the desired CPMA.

1. **Event Driven Architecture (EDA)**: Behavior of any CPMA should be event driven to support responsiveness in which it consumes, processes and produces events. This is well known in the literature (Niblett & Graham, 2005) for any monitoring systems that react to events. In addition, the related work (Boubbeta-Puig, Ortiz, & Medina-Bulo, 2011) emphasizes the significant of EDA for building care process monitoring applications that require near real-time event processing and integration to provide analysis and identify abnormal situations towards the care processes.
2. **Event Integration and Filtering**: The importance for monitoring applications to perform event integration and filtering for the purpose of inferring anomalies that require immediate attention and rapid response or new event types that measure the performance of care processes is widely known from the literature (Luckham, 2002)

and the related works (Vaidehi, Bhargavi, Ganapathy, & Hemalatha, 2012) (Boubeta-Puig, Ortiz, & Medina-Bulo, 2011). In addition, according to the results of gap analysis, event integration and filtering from distributed sources is a key requirement for any CPMA in order to define, infer and determine that start and the end time of care process states.

3. **Integrate with Event Sources**: From the experience we had in the case studies, a core requirement for any CPMA is to integrate with the event sources of the enterprise. It is significant to monitor event sources in order to keep track of the resources' states in the care processes and to monitor the performance management.
4. **Integrate with Business Process Management (BPM) Architecture**: The importance of integrating the business process monitoring application with BPM in order to capture, process and intrgrate the events as they are happening when the business processes are enacted by the BPM engine in addition to control the execution of business process is well known from the related work (Janiesch, Matzner, & Müller, 2012). As a result from the gap analysis, monitoring care processes using BPM only is not adequate to provide fine-grained monitoring. Therefore, any CPMA should be integrated with BPM to provide fine-grained metrics and monitoring, particularly, if the events generated from the BPM are integrated with other event sources in the enterprise. In addition to the above, the domain expert emphasized the value of integrating any CPMA to BPM.
5. **Integrate with Performance Management Architecture**: It is well known from the related work (Janiesch, Matzner, & Müller, 2012) that business process monitoring applications should visualize the performance management, as a result of processing

the events collected from the underlying business processes, in the monitoring dashboard. In addition, from the experience we gained in the case studies, any CPMA should be able to bridge the gap between the operational care processes and the performance management by providing a dashboard that graphically presents the performance of the underlying care processes.

6. **Integrate with Real-Time Location Service (RTLS)**: It is well known from the literature (Boulos & Berry, 2012) that an RTLS architecture is useful to monitor and pinpoint the locations of individuals and to gain insights into the statuses of those individuals in the hospital. However, developing monitoring applications to collect and process RTLS events only is not advantageous for fine-grained monitoring of care processes. As a result of the gap analysis and feedback from the domain expert, it is a requirement for any CPMA to integrate with RTLS architecture to define fine-grained metrics if the location events are integrated with other type of events.

3.3.6 Application Development Process and Tools

Care process monitoring applications are complex to build because data has to be integrated from disparate sources and integrated with care processes to monitor and manage them effectively. During the application development for any CPMA, there should be techniques or tools available to support different phases of the application development process.

1. **Tool or Technique to Test CPMA Prior to Enterprise Integration**: According to our experience we had in the case studies, it is important to adopt a tool or technique in the testing phase of the application development process before the actual deployment and integration of CPMA in the real enterprise environment. The tool

should be able to simulate events of any type from any source systems in the enterprise. In addition, it is well known (Beck, 2003) that test driven with test scenarios is a fundamental technique in the agile development process that adapts to the rapidly changing requirements and reduces the amount of time required to maintain the application.

2. **Tool or Technique for event processing and integration rules**: It is important to have a tool or a technique that supports writing event processing and integrations rules as part of the implementation phase of developing CPMA. The related works (Janiesch, Matzner, & Müller, 2012) (Boubbeta-Puig, Ortiz, & Medina-Bulo, 2011) (Yao, Chu, & Li, 2011) underline the value of leveraging event processing and integration rules tools to support and streamline writing the complex event rules.
3. **Tool or Technique to translate care process events to performance metrics**: To support the implementation phase of developing a CPMA, in particular, for implementing performance metrics, the domain expert confirmed that there should be a technique maps between the details of the care processes, in terms of events, and the corresponding performance metrics based on the goals that should be satisfied towards the desired care process.
4. **Tool for visualizing the requirements**: It is well established in the literature with respect to URN (Roy, Kealey, & Amyot, 2006) and UML (Sommerville, 2011) that for complex systems, it is important for stakeholders to be able to visualize requirements. In terms of developing CPMA, it is critical to grasp the relationship between performance management, business processes and the data architecture of the enterprise.

5. **Support for Agile development process**: The importance of adopting an agile development process for rapid software delivery is widely known in the literature (Cohen, Lindvall, & Costa, 2004). As discussed in the problem definition that development of any CPMA should be agile and rapid, particularly, it should support responsiveness to the changing requirements of the care processes (i.e., add new resource types, add new care process state typse, remove existing metric types).

3.4. Chapter Summary

In this chapter, we first discussed the key challenges and problems that exist in engineering CPMA in hospital and community care. Then, we analyzed the current approaches leveraged to monitor care process in the hospital and the community care and their limitations. we also analyzed the current approaches to monitor care processes exploited in the related works and their limitations. Finally, we developed a set of evaluation criteria drawn from the following: analysis of the related works, gap analysis, discussions and feedback of the domain experts, periodic reviews of CPMA with Osler and Bruyère hospitals and experiences we had in our case studies.

In the next chapter, we will present our application framework for monitoring care processes that is comprised of three key elements: the state-based meta-model, the Care Process Monitoring (CPM) architectural pattern, and a methodology to develop a CPMA.

Chapter 4. Application Framework for Care Process Monitoring

In this chapter we present our application framework for care process monitoring. The three key elements of the application framework are:

1. A state-based application meta-model for defining any care process monitoring application (CPMA);
2. a general architecture pattern for care process monitoring that addresses how to integrate a CPMA into the enterprise architecture supporting health care processes whether they are community-based or hospital-based;
3. and a software development methodology for building a CPMA based on the application meta-model and the general architectural pattern.

The application meta-model is an object model of the elements common to any CPMA that captures the relationships between elements involved in the care process, elements involved in measuring the performance of the care process, and elements in the enterprise architecture that support the care process. The general architecture pattern integrates events collected from enterprise sources, while the care process is taking place, and delivers them to a performance management reporting database as a sequence of care process states and associated metrics for measuring performance. Finally, the methodology is an agile one, based on behavior-driven

development, which defines the steps to follow in developing a CPMA from requirements to testing to implementation using the application meta-model and general architectural pattern.

4.1. State-Based Application Meta-model

Our application meta-model is a simple but generic object model that captures the essential elements of any CPMA. The main objective of defining a meta-model is to identify and standardize the elements used to define a CPMA at a logical level in terms of its information structures so that the relationship between information that characterizes the states of a care process and the metrics that measure performance of the care process are captured. The application meta-model defines the information model that the CPMA maintains as performance management reports are generated from care process states while the care process is taking place. The processing that a CPMA must perform is complicated by the fact that the enterprise infrastructure that supports the care process is often somewhat disconnected from the performance management reporting infrastructure. Section 4.2 will address this issue with a general architectural pattern for integrating a CPMA that connects care process infrastructures with performance management reporting.

We will not, in this thesis, define an execution semantics for the meta-model so that the behavior of the CPMA is precisely specified, although that is a worthwhile area for future research. Also, we will not, in this thesis, create a model driven architecture (MDA) in which the implementation of a CPMA might be generated from an application model, although this is a worthwhile area for future research as well. Our focus is to define an application meta-model that can be used to clearly and concisely define the essential requirements or blueprint for any CPMA as an application model to guide design and implementation in a systematic fashion.

The application model is fundamental to designing a CPMA since it:

- Defines what events from what sources should be collected for the care process.
- Defines how the states of the care process should be monitored in relation to those events.
- Defines what metrics are computed from what events or states.

In the following sections, we first discuss the mapping between an operational care process, which is represented by a simplified state model, and the strategic performance management that ensures the goals of the care process are achieved. Then we introduce some diagrams and notations that help visualize this mapping for typical CPMAs for care processes that can either be community-based or hospital-based. Finally, in the last section, we define a generic application meta-model that provides a common, structured and systematic representation for any CPMA in order to support those diagrams and notations.

4.1.1 Mapping Operational Processes to Strategic Performance Management

The main purpose of a care process monitoring application is to give insight into how care processes are performing while they are taking place. To do this, one needs to link a representation of performance with a representation of the data captured in information systems that communicate what is taking place in a care process.

Traditional approaches to strategic performance management compute metrics from historical data. That historical data is populated by ETL processes that extract data from a wide variety of operational data stores and then transforms and loads it into a common data model in a

data warehouse in order to support performance reports. However, this only supports historical analysis of processes after they have completed.

There are a number of approaches that can be taken in order to support monitoring of care processes while they are taking place. The essential element of these approaches is that the data needed to monitor processes is available on-line. In community care processes, participants in the care process often fill out forms on-line in a web application that stores the information in a reporting database. In a hospital environment, there is often a service-oriented architecture (SOA) in which operational systems can be accessed through a web service interface. A Business Process Management (BPM) engine can orchestrate processes that cut across several operational systems and be an integrated source of some process data. Increasingly, event driven architectures are also put in place which can pull and push events directly from all data sources within a hospital including operational data stores, specialized equipment, real-time location systems, and BPM systems.

The essential task in all these approaches is to bridge the gap between a performance data model for reporting on performance in terms of metrics and an operational data model of processes. In particular, to support process monitoring while processes are taking place, one must receive data in "small chunks" as individual processes are unfolding. Figure 4-1 depicts the situation.

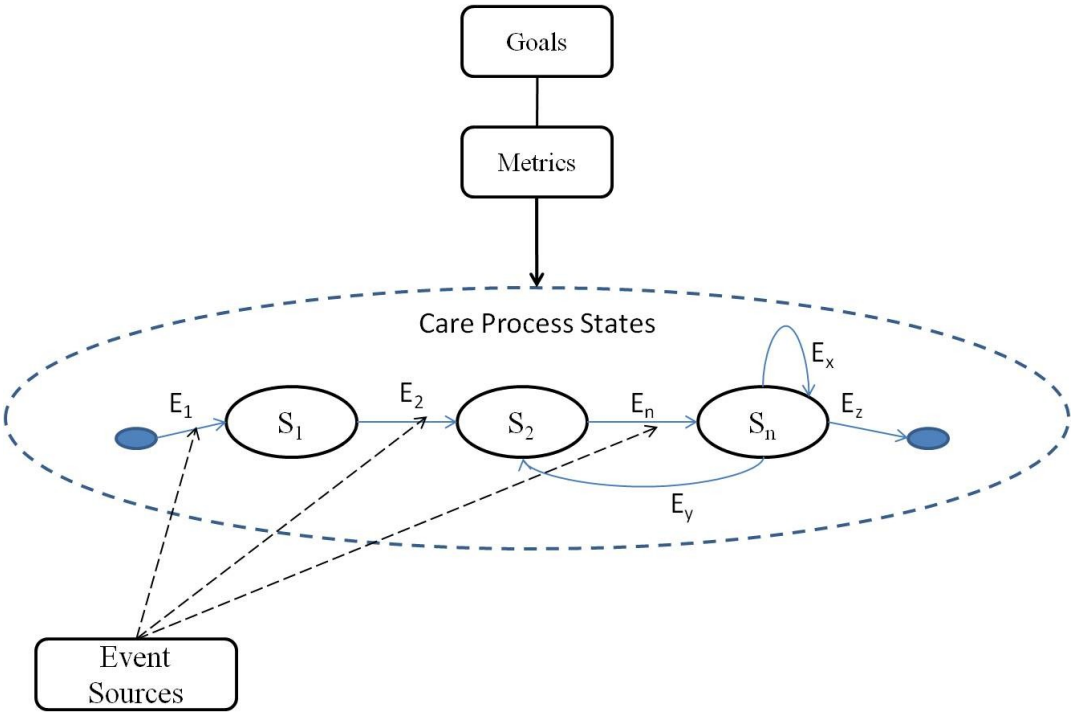


Figure 4-1: Mapping Operational Care Process to Performance Measurement

The care process is represented in terms of the key states that we wish to monitor. As the process takes places, "small chunks" of data are received from the various sources and interpreted as events that mark the transition from state to state within the process. Using this representation of the care process in terms of states and events, one can compute metrics that measure the performance of the process against goals of care. In the case of a simple community care process, a "small chunk" may be the data entered in an electronic form. In the case of a sophisticated hospital care process, the "small chunk" may be a message received from an existing operational system via a web service. In either case, the "small chunk" is data that represents an event in the care process.

4.1.2 Modeling Applications for Community Care Process Monitoring

The purpose of modeling a community care process is to design the monitoring application which provides analytics while the care process is taking place. Since the target is to monitor key steps in the community care process that provide information regarding the performance of the process, the community care process can be represented in terms of patient states and forms that capture the transition from one patient state to another patient state. Those forms are filled in to gather data with respect to the patients in the community at each step of the care process. Figure 4-2 shows a simplified care process that is used to manage palliative patients in the community. The palliative care process is designed to register palliative patients to monitor their care at the end of their life with regular scheduled consults in order to manage their pain and symptoms with the emphasis on delivering care at the patients' home without scheduling a visit to the hospital. A full version of the palliative care process states along with the forms is covered in chapter 5. Once the patient is referred to the palliative care program, he will be in wait-for-triage state until an appointment is scheduled for him. In this case, the patient will be in the wait-for-scheduled consultation. After that, repeated consultations take place.

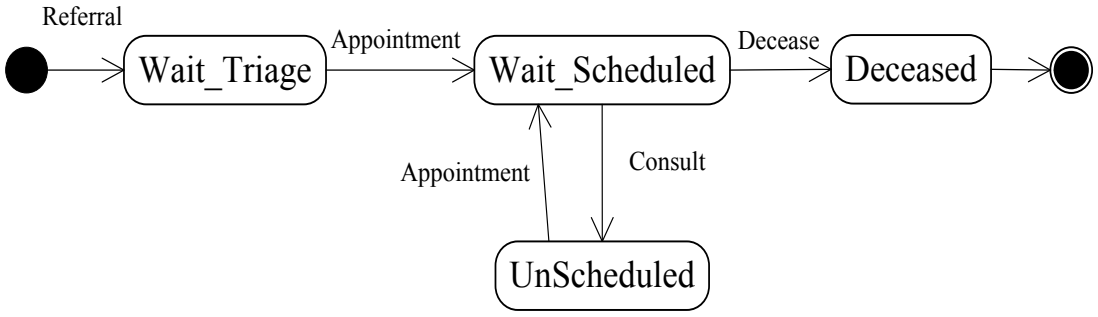


Figure 4-2: A Simplified Palliative Care Process

After selecting the key community care process states, the next step is to model the performance of the community care process. This can be achieved by linking the performance management, expressed as metrics, to the key community care process states. In this case, the metrics utilize the data collected from specific forms shown in the previous figure to measure whether goals set by the community care program are met or not.

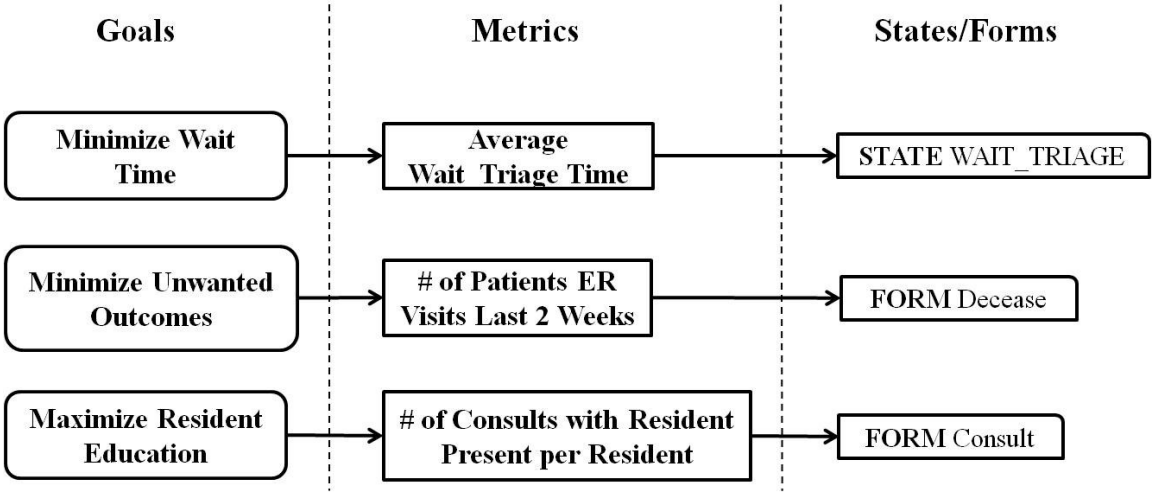


Figure 4-3: Mapping Goals to Metrics to Forms

Figure 4-3 depicts the mapping between goals to metrics to forms for a few example metrics (a complete analysis of all metrics is discussed in detail in Chapter 5). In order to ensure wait time is minimized for a first consult, Average Wait_Triage Time is measured from computing the duration of the state WAIT_TRIAGE. The duration of this state is computed by measuring the start time of the form Referral and the start time of the form Appointment as shown in Figure 4-2. In order to ensure that the care process is successful for minimizing visits to the hospital, there is a check on the form that is filled out when the patient is deceased to indicate whether or not the patient visited the emergency room of a hospital in the last 2 weeks before death. In order to ensure that the care process is integrated with medical training programs, there

is a check box on the form that is filled out when a consultation takes place to indicate whether or not a resident (medical student) was present during the consult.

The essential design task when building a CPMA for monitoring community care processes is to:

1. Determine what metrics best communicate the main goals for the community care process (e.g. Average Wait Triage Time);
2. Determine what states in the community care process are the key elements to measure the progress towards the ultimate goal.
3. Determine what forms required to trigger the key states in step 2 in which the attributes of those forms are used to compute the metrics.

4.1.3 Modeling Applications for Hospital Care Process Monitoring

The approach to modeling for hospital care process monitoring is similar to that for community care, but the situation is more complex. Hospital care processes can cut across many different departments and many different information systems. This provides richer and more varied sources of data for monitoring the care process, but this complicates the information system task of integrating and processing that data to produce reports. There is not a simple form at each step in the process to collect the data into a single departmental system. Instead, we have to define events that specify the data which is extracted or generated from different information systems to create an event trace of the process as it takes place across many different departments. This is further complicated by the fact that the integration and processing of data must happen in near real-time. Hospital care processes take place as a sequence of steps that

occur over a period of hours, whereas community care processes take place as a sequence of steps that occur over a period of days.

The concept for modeling the hospital care process is quite similar to that for community care where a state transition diagram is used. But rather than employing forms to trigger states, events (received as messages through a web service) are the basis for state transitions. In addition one needs to specify the source system from which the events originated. In this situation there are two possibilities. The first possibility is that events, in the state diagram, are mapped to only one source system, while the second possibility is that events are inferred by integrating messages emitted from multiple source systems. Figure 4-4 depicts the case which shows a small subset of the patient states in a hospital care process for cardiac care. The cardiac care process is enacted to manage the patients as they arrive in the hospital and triaged for cardiac care. There are two such consultations with the patients to determine first what tests need to be performed and second to determine what procedures need to be performed. At the end of the care process, the cardiac patient is discharged. In the cardiac care process, it is important that the time from triage to procedure performed is minimized as some procedures must be done within, say 90, minutes to be effective. A full discussion of the complete process with all its states is provided in section 5.3.3.

To illustrate how events are mapped from the source systems, consider the following example. In Figure 4-4 events can come from:

- a Real-Time Location Service (RTLS) that indicates the location of patients, transporter, nurses and physicians as they move around the hospital, or

- a Business Process Management engine (BPM) that provides an interface to the various systems in the hospital and guides the physicians and nurses with respect to the next steps to be performed.

For example, integrating events BedNotAvailable and OrderBed generated from BPM with event PhysicianOutED generated from RTLS will infer the state WAIT_FOR_BED_CW.

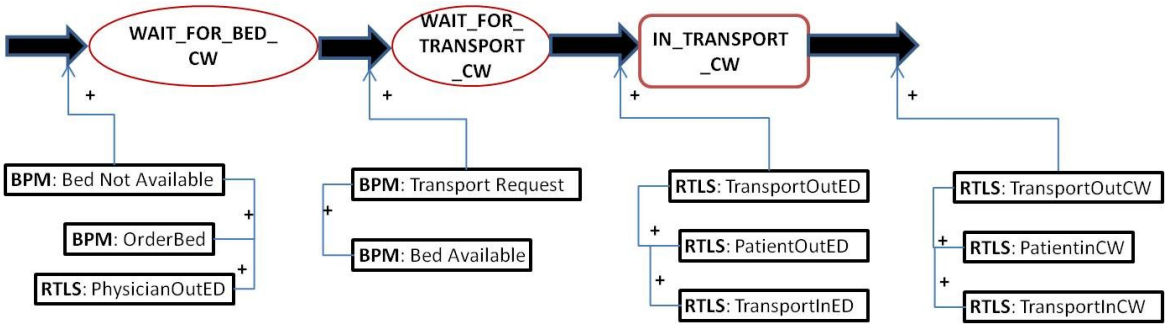


Figure 4-4: Snapshot of Hospital Care Process States for Cardiac Patient

Again, as in modeling the community care process monitoring application, the next step to design the hospital care process monitoring application is to model the performance of the hospital care process. The method is similar to that for community care with a minor modification. Rather than linking the metrics to forms, metrics are linked to states or single events. In other words, those metrics are measured either from the duration of particular states (the time difference between two events that trigger and exit the state) or from the specific event that triggers a hospital care process state. Figure 4-5 shows the goal model. For instance, the goal that the hospital aims to achieve is to monitor the wait times for beds and to ensure that the value for the wait times is within an acceptable range set by the hospital. To do so, according to the goal model, the Care Process Monitoring Application (CPMA) should monitor, compute and update the metric *Average Wait_Time for Bed* by measuring the duration of the state

WAIT_FOR_BED_CW. The integration of the events BedNotAvailable, OrderBed and PhysicianOutED determines the start time of the state. Similarly, the combination of the events TransportRequest and BedAvailable determine the end time of the state. The aforementioned events are shown in the previous Figure 4-4.

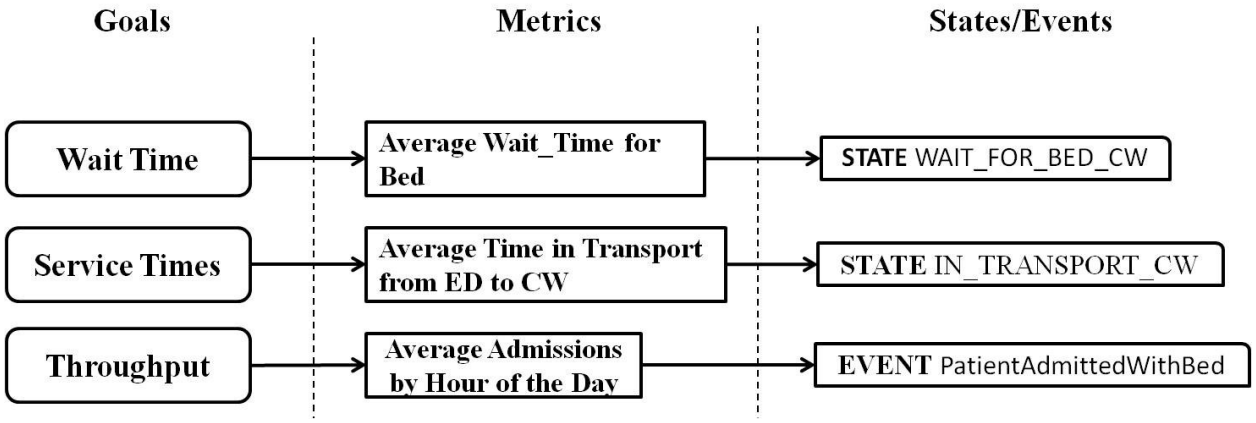


Figure 4-5: Mapping Goals to Metrics to Events

The essential design task when building a CPMA for monitoring hospital care processes is to:

1. Determine what metrics best communicate the main goals for the care process (e.g., wait times and service times).
2. Determine what steps and states in the care process are the key elements to measure in order to determine progress towards the ultimate goal.
3. Determine what operational events can be integrated from what event sources in order to infer and define care process states (BPM, RTLS, other systems).

4.1.4 Application Meta-model for Care Process Monitoring Applications

We can see from the last two sections that community care and hospital care processes are defined in very similar terms. The community care process focuses on forms rather than events, and the processing of events and their mapping to states and metrics is more complex for hospital care processes; however, a form can be viewed as simply one type of source for events. As well, we could expect there to be a continuous gray scale spectrum from very simple community care monitoring to very complex hospital care monitoring. A hospital without sophisticated infrastructure might have very simple form-based monitoring, while a community might have very sophisticated infrastructure which supports event-based monitoring. As such, we have defined a generic application meta-model for care process monitoring that can be used to model any care process monitoring application along that gray scale spectrum.

Our meta-model addresses care processes with goals to be satisfied in terms of measuring metrics, wait times, service times and states related to particular resources (e.g., patient, physician, room, bed, asset, etc.,) participating in the care process, as well as event sources that generate events and trigger states. This meta-model is defined as a logical object model that captures key data entities and their relationships. The meta-model is designed to be generic and is not meant to be restricted to the specific case studies we have used to evaluate it. The Appendix contains the complete application models that were built for our two cases studies using the application meta-model we define here.

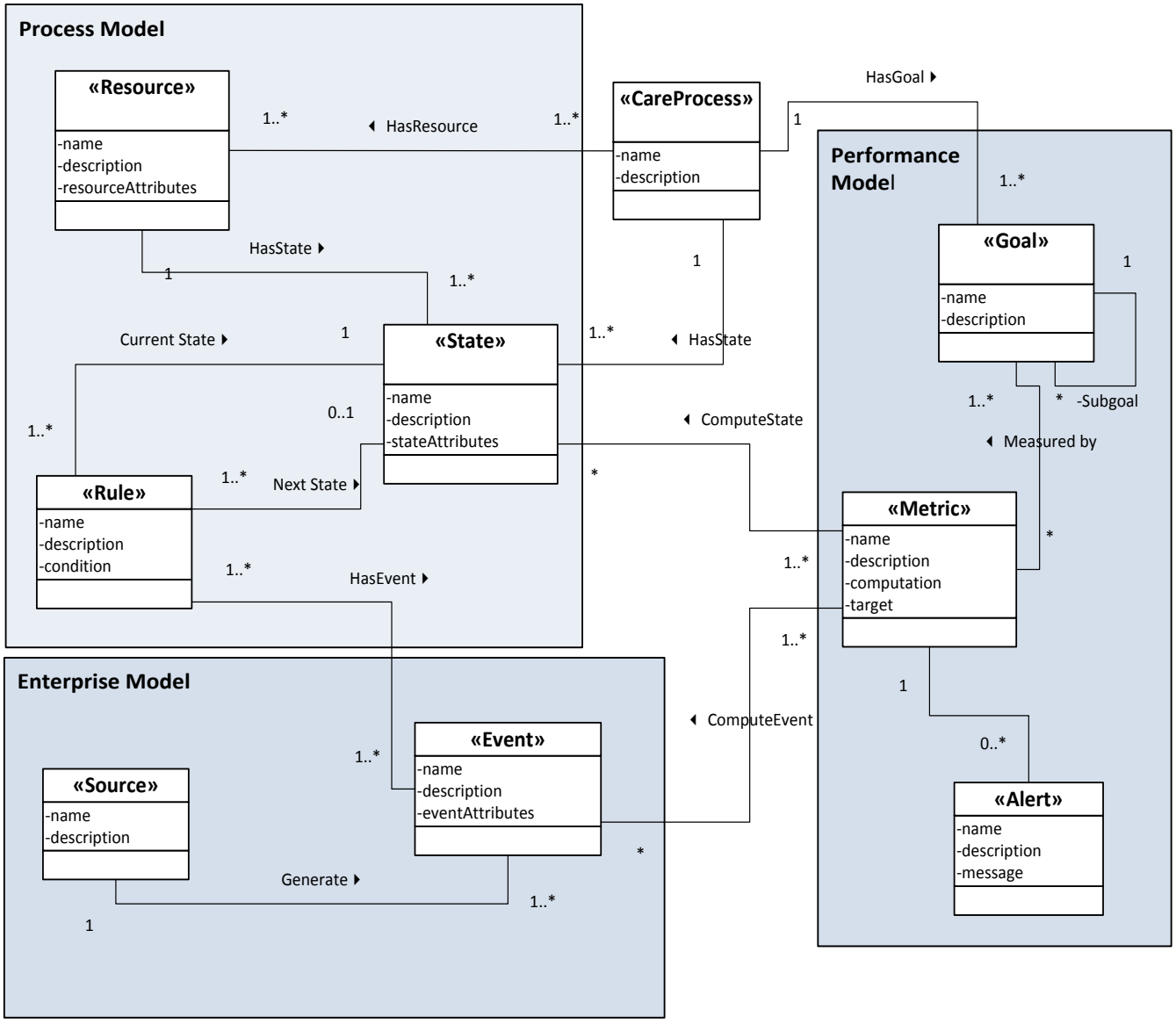


Figure 4-6: Meta-model for Care Process Monitoring Application

As shown in Figure 4-6, the meta-model defines a care process monitoring application in terms of a process model, a performance model and an enterprise model. The process model defines the care process in terms of states to be monitored, resources, and rules that specify the transition from state to state as events are received from sources as defined by the enterprise model. The performance model measures how well the goals for the care process are being

achieved in terms of metrics computed from the monitored states and events for the process. Alerts are defined to flag when targets are not being met.

The Care Process object names the care process to be monitored and provides a description. It identifies the states to be monitored, the goals to be achieved, and the resources involved in the care process.

The Resource object is either the recipient of care (patient), care provider (e.g., nurse, or physician) or a physical resource (e.g., bed, room, or piece or equipment like a ventilator or X-ray machine). It can be either a human resource (e.g., patient, or even physician, or nurse) or a physical resource (e.g., a bed, or room, or even a test). The essential information about a resource is a name that uniquely identifies it, but there may also be a list of attributes that provide additional information about the resource.

The State object is essential to measure the performance of the care process in which a metric can be computed from the duration of particular state(s). It is uniquely identified by its name and explained by its description. Each state has a list of attributes. For instance a state can have an attribute called timestamp that captures the time when a particular resource enters that state. The rules associated for each state define how to transit from the current state to the next state or define how to update the current state.

The Rule object is identified and demonstrated by its name and its description. Processing incoming events for the purpose of inferring new events or triggering next state or updating the current state are defined by rules with a certain syntax. Generally, the syntax for each rule is composed of a condition field which contains the current state and the event(s) that should be

monitored to trigger the next state. For example, in the cardiac care case study (see section 5.3), the following rule syntax shows how to infer the state WAIT_FOR_TRANSPORT-CW in response to the events TransportRequest and BedAvailable.

If CURRENT_STATE is WAIT_FOR_BED_CW
AND EVENT TransportRequest
AND EVENT BedAvailable
Then NEXT_STATE is WAIT_FOR_TRANSPORT_CW

The Event object is illustrated by its description and every event type is uniquely identified by its name. Each event must have a list of attributes that provide a full description of that event. For example, an event name PatientInED generated from RTLS has a list of attributes, namely, patientId, locationId and timestamp. Events are important to compute metrics and trigger resources' states while the care process is taking place.

The Source object identifies the service, or server, or data source from which particular event types originate. Each event source is identified by its name. For example, a BPM engine, RTLS, and CEP are names of event sources that can all contribute to the pool of generated events. Moreover, the description of the source event might show how it generates events or it might describe the format of the event generated.

The Goal object defines some criteria or aspect that will be used to communicate how well the care process is being performed. The metric(s) associated with a goal will give some indication as to what degree the goal is being achieved. The name uniquely identifies the goal, while the description explains the meaning and significance of the goal as it related to performance of the care process. For each goal (e.g., reduce wait time), there is one or more

associated metrics which measure or quantify the performance of the care process in achieving the goal. Goals can also be decomposed into sub goals. However, the main purpose of goals in our application meta-model is not to provide complex analysis of organizational objectives. The intent is only to provide a means of organizing the metrics that will be reported in a CPMA in a manner that links to more sophisticated goal analysis that can be done in notations like URN.

The Metric object is uniquely identified by its name, and explained by its description. It is specified by a computation which expresses how to calculate or measure it from specific events or states for a care process. An important issue regarding the metric object is to specify the target, which identifies what values are acceptable, what values are cautionary (typically when the values are near acceptable range) and what values flag out of range, for each metric. Alerts can be associated with a metric to flag how well the metric is meeting its target.

The Alert object can define a message or visual cue that can be sent or displayed to indicate the status of a metric. For instance, as the metric, average wait time for bed in the cardiology ward approaches and exceeds the target wait time, a message could be displayed.

Implicit in the application meta-model, is the assumption that the CPMA will be receiving events and forms data from an event driven architecture and logging states, events and forms data to report metrics in a performance reporting dashboard.

4.2. An Architectural Pattern for Care Process Monitoring

In this section, we introduce an architectural pattern for care process monitoring and discuss how that application architecture integrates within the typical enterprise context for both community care process and hospital care process monitoring. We also discuss different

approaches to building a care process monitoring application using the application meta-model and architectural pattern.

4.2.1 Architectural Pattern

Based on our work with health organizations to build CPMAs and our gap analysis of existing approaches, we have identified the following architectural pattern for care process monitoring. It is defined as follows:

Pattern Name: Care Process Monitoring (CPM)

The CPM architectural pattern describes how to monitor a care process and how to integrate events from various event sources in different healthcare departments or organizations in order to provide near-real time performance management.

Context:

The CPM pattern is useful for healthcare organizations that want to manage the performance of care processes as they are taking place with support of diverse information systems and require fine grained monitoring of particular resources participating in that care process in order to pinpoint bottlenecks and determine root causes.

Problem:

1) How to integrate CPMA into a complex enterprise architecture?
2) How to bridge the disconnect between the enterprise support of care processes and the performance management architecture?

3) How to integrate event data from disparate sources within or across healthcare organizations to obtain detailed analysis of the care processes to measure the metrics precisely in near real-time?

Forces:

The following forces must be considered when developing such care process monitoring applications:

1) Events are available as they are taking place.
2) Processing of events is scalable and performed in near-real time.
3) Events are integrated from different data sources in the enterprise.
4) Current state of care process is required both for individual patients and aggregated across all patients along many different dimensions.
5) Performance management reports are required in enough detail to pinpoint troubles pots within a care process and identify root causes.
6) Performance management reports must be available to care providers and administrators while the process is taking place.

Related Patterns:

The typical architecture pattern for care process monitoring applications used in hospitals is ETL (see section 2.2.2 (Giordano, 2010)), which is used for providing historical performance management as opposed to near-real time performance management. In some simple cases, hospital and community care processes use a three-tiered web architecture (Sommerville, 2011)

for a CPMA that uses online forms to collect information manually. In more sophisticated cases, there is an SOA pattern (see section 2.2.2 (Mulik, Ajgaonkar, & Sharma, 2008)) integrating a large number of separate systems that support a care process. To deliver events from disparate sources in near-real time, EDA (see section 2.2.2 (Preuveneers, Yasar, & Berbers, 2008)) is the most relevant pattern. The challenge for a CPMA is to be able to transform those events into the states that characterize them and store them in a data base so that a data mart pattern (Jarke, Lenzerini, Vassiliou, & Vassiliadis, 2010) can be used for performance reporting. Ultimately, CPM is a pattern that can achieve the integration from data sources to data warehouse in near real-time using EDA to deliver events that are interpreted in terms of care processes states and logged to a data mart.

Solution:

The solution is crafted to satisfy the aforementioned forces in the given context. In particular, EDA-SOA satisfies the force 1, Care Process Monitoring Engine (CPME) satisfies the forces 2, 3 and 4. Finally, Performance Reporting Dashboard satisfies forces 5 and 6. Figure 4-7 shows the essential elements of an architectural pattern for care process monitoring that integrates a Care Process Monitoring Application into the enterprise architecture for the care process. It is comprised of: Event Sources, CPME, Performance Reporting Dashboard, Metrics Data Mart and Data Warehouse.

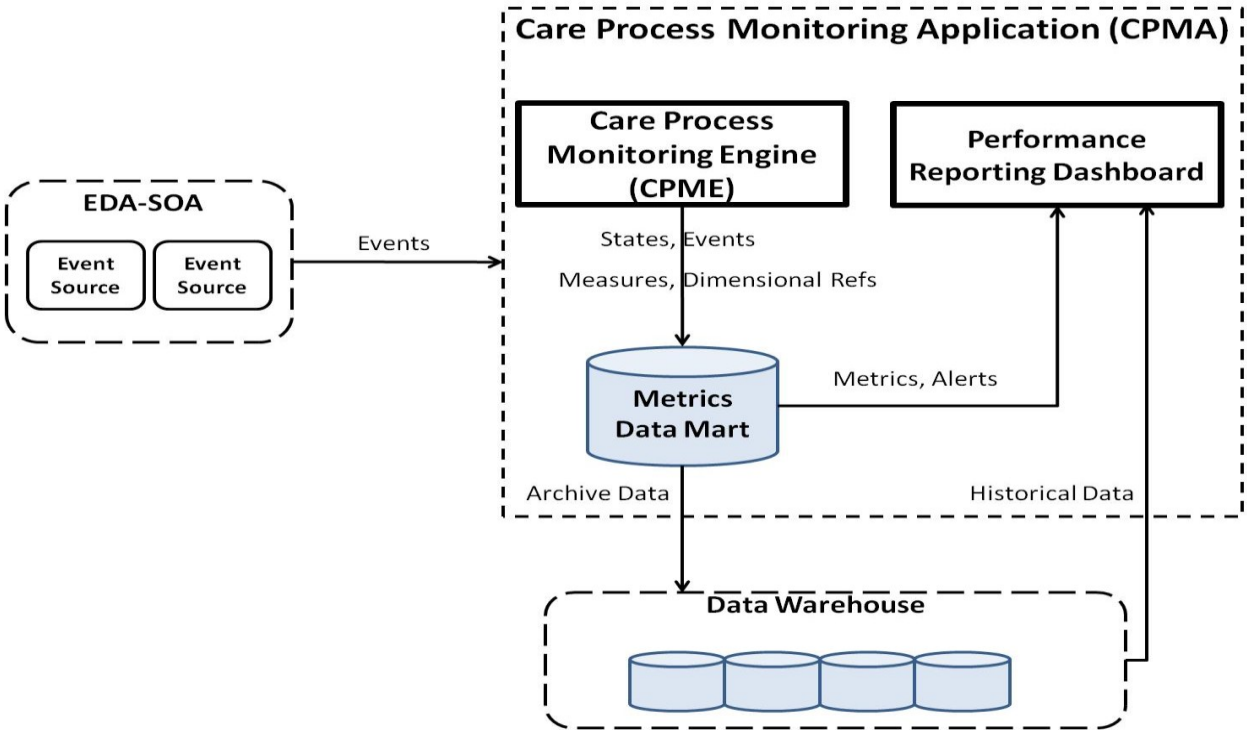


Figure 4-7: CPM Architectural Pattern

1) CPMA

CPMA is the new component that needs to be built. It processes events generated from the EDA-SOA architecture to keep track of the current states and the related events of the resources participating in the care process. As well, CPMA should be able to report on the performance of the care process in terms of metrics based on the collected events and the duration of states.

2) EDA-SOA

EDA-SOA is the existing infrastructure of the healthcare organizations' enterprise that supports the care process. In EDA-SOA architecture, the source systems are able to generate events and to communicate with CPMA through web services.

3) Data Warehouse

Data Warehouse is the existing infrastructure that is integrated into CPMA. It is updated by CPMA based on the data stored in the Metrics Data Mart. As well, it provides CPMA with historical data to support performance management reporting for the care process.

4) Metrics Data Mart

The Metrics Data Mart is an OLAP database optimized for reporting. It is implemented as a star schema with two fact tables (one for states and one for events) that store the measures needed for performance reporting, as well as references to the dimension members relevant to the facts. The dimension members are stored in look-up tables (dimension tables) to facilitate performance reporting along different dimensions such as Resources (Patient, Rooms, Physicians, Nurses, etc.,), Time (hour, day, month, year) and Demographics (Age, Gender, Region). Figure 4-8 is an example of a start schema that records the facts regarding patients' states at a given point in time. It contains a fact table called patient_state_fact that holds all state transition of patients and the duration for each state. This fact table is surrounding by a number of dimension tables such as state_dimension, patient_dimension and time_dimension.

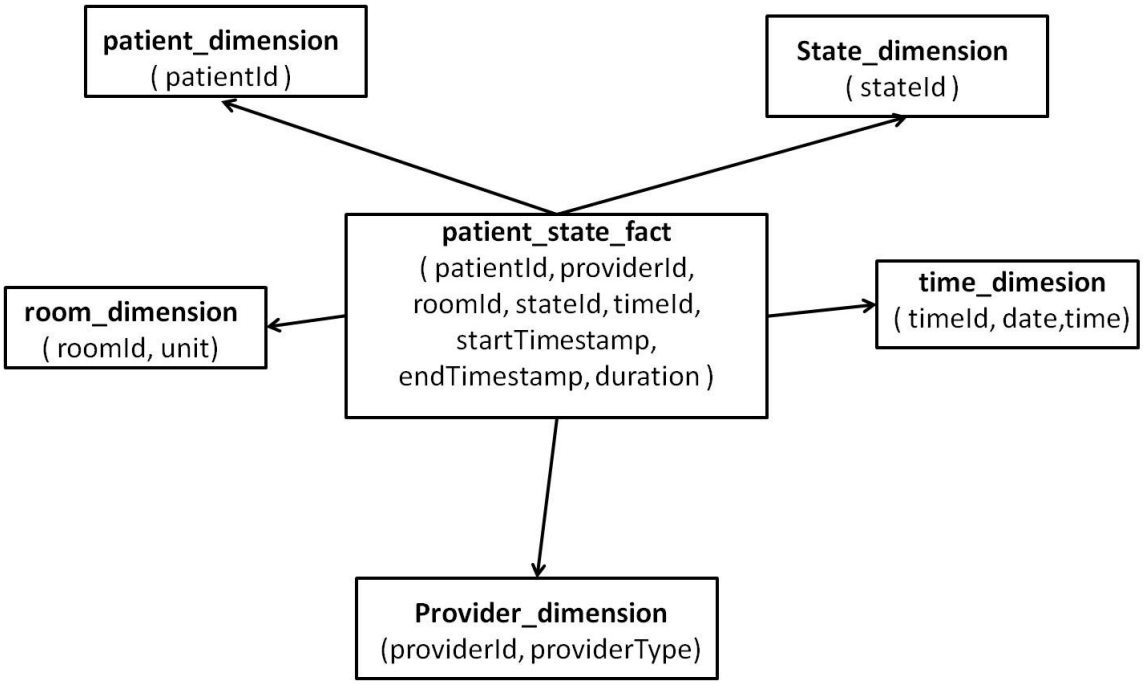


Figure 4-8: Star Schema for Hospital Care Process

5) Performance Reporting Dashboard

The Performance Reporting Dashboard offers performance management reporting for the users of the monitoring application (e.g., Case manager, Physician and Nurse) by displaying charts and tables that depict the performance metrics. This component can be implemented either by using any Business Intelligence (BI) tool, such as the IBM Cognos 10 Business Intelligence Suite, or from scratch using a web application frameworks such as Grails. Performance metrics can be displayed on a patient by patient basis or on an aggregate basis with drill up and drill down features along dimensions, and they can be broken down into a step by step analysis according to the sequence of states and events that have occurred. Alerts can be displayed as messages to flag where metrics are not on target, or a color coding scheme (green – safely on target, yellow –warning on boundary, and red - out of range) can be used to highlight in reports

the status of metrics associated with states or resources in the care process. Example screen shots of this type of reporting are shown in chapter 5 for each of the case studies.

6) Care Process Monitoring Engine (CPME)

The CPME is a critical component in the CPM architectural pattern. It collects particular events from particular event sources in order to update and track states for the key resources that are participating in the care process, and later, those outcomes are displayed in the performance reporting dashboard. It creates a log of all care process events and states as they occur in the Metrics Data Mart, while the process is taking place. This log includes measures that are calculated as events and states are processed as well as dimensional references that allow the states and events to be aggregated along different dimensions.

Figure 4-9 shows the internal architecture of CPME and its key components: Message Handling, State Inferencing, Care Process Model, and Database Mapping.

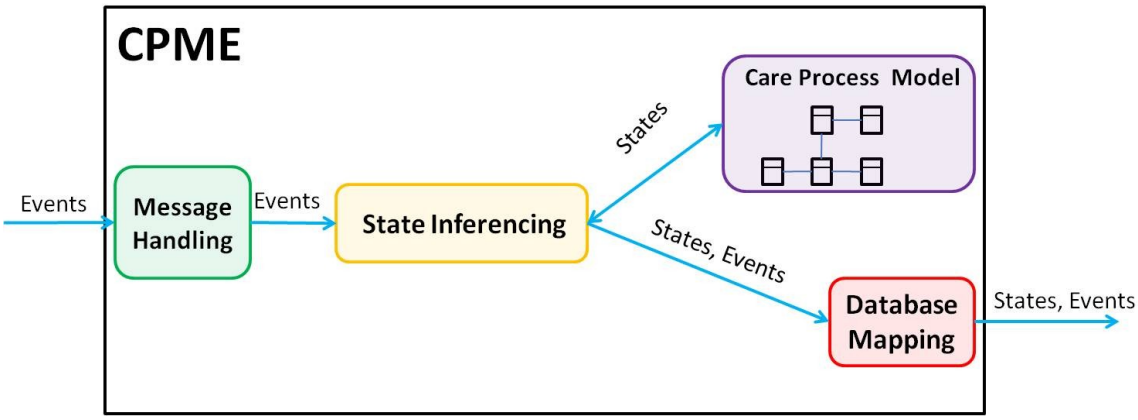


Figure 4-9: Internal Architecture of CPME

Upon receiving an event from one of the event sources, the Message Handling component parses the message and identifies the event type. Accordingly, the event is routed to the State

Inferencing component which triggers state transitions and updates in the Care Process Model based on the events that have been seen, their attributes, and the current state of the Care Process Model. The State Inferencing engine also logs new events and states to the Database Mapping component as they occur.

Based on the application meta-model:

- The Care Process Model component is an in-memory representation of the current state of all active processes in terms of States and Resources.
- Rules are implemented in the State Inferencing Component, and
- The Data Base Mapping component is a Data Access Object (DAO) (Alur, Malks, & Crupi, 2003), which stores the measures and dimension references for each state and event fact in the Metrics Data Mart so that the Metrics and Alerts can be computed and displayed in the Performance Reporting Dashboard.

Interactions:

EDA-SOA and CPMA: Event sources in the EDA-SOA send events to the CPMA directly as a web service or indirectly via a Message Middleware.

Performance Reporting Dashboard and Metrics Data Mart: The Performance Reporting Dashboard performs queries against the Metrics Data Mart for a particular resource using any BI tool or custom web-based component that can query the appropriate state and event fact tables and their associated dimension tables.

CPME and Metrics Data Mart: Within the CPMA, the care process monitoring engine (CPME) sends information on events and states to the Metrics Data Mart which logs the events and states into fact tables to store the measures and relevant dimension references for each event and state.

Examples:

We will discuss how the CPM architectural pattern can be applied and integrated into community care and hospital care enterprise architectures in sections 4.2.3 and 4.2.4 respectively. In addition, a complete set of examples along with implementation issues will be discussed in our case studies in chapter 5.

4.2.2 Model-based Instantiation of CPM Architectural Pattern

The application meta-model presented in section 4.1 can be used as a basis for instantiating the CPM architecture pattern. In particular, our engineering approach has been to instantiate the CPM architectural pattern as a CPMA engine that can monitor any care process that has a care process application model defined by the application meta-model. There are three ways in which the meta-model can be leveraged when instantiating the CPM architectural pattern.

1. Use the application meta-model and the CPM architectural pattern as a guide to the design and implementation of a generic CPMA that processes events and states according to rules and stores them in a generic metrics data mart for care processes.
2. Define an XML representation of the application meta-model so that any particular application model can be stored, interpreted, and executed by a generic CPMA.

3. Have a model driven architecture approach that generates the CPMA code for a specific care process from the application model for the care process defined using the application meta-model.

Of course, these options are not mutually exclusive. In some respects, they are complementary. It is quite common for model-driven approaches (3. above) to have a target application framework (1. above) into which the application specific code is generated (AndroMDA, 2012). An XML representation of the application model (2. above) is always useful as an intermediate format. In our case studies, we have done prototype implementations using 1. and 2. above. Toolsmithing in support of 1, 2, and 3 above, and formal transformations that would be needed for 3, have not been a focus of our thesis work although they will be rich areas to mine for future work. The focus for this research is to lay the foundation for understanding and characterizing what a CPMA is, which can be leveraged in future research. We do that by identifying an architectural pattern for care process monitoring, along with an application meta-model and methodology that bridges the gap between the EDA-SOA support for online care processes and OLAP database architecture support for measuring performance. The validity of using these contributions for simplifying and automating CPMA development is evaluated by the case studies we present in chapter 5, where prototype implementations of a CPMA for a hospital care process and a community care process are made using our approach.

4.2.3 Integration into Community Care Architecture

Currently, healthcare organizations (e.g. hospitals, clinics, long term facilities) that provide care to patients in the community have different information systems that do not easily communicate with each other. Therefore, it is difficult for organizations, who share care in the

community, to track and obtain the updated states for patients in the community over a period of time (e.g., over two weeks, over one month). As well, it is hard for them to gain an accurate status of the patients for a given time period. Hence, the inconsistency of information maintained by different healthcare organizations negatively impacts the ability to report and track metrics for performance.

One potential solution for integrating CPMA into a community care enterprise architecture is to have online forms shared by all healthcare organizations that are interested in tracking a specific type of patients in the community (e.g. Palliative patients). The online forms can deliver the data collected to the CPMA as events. The data values used in those forms are directly mapped to the measures and dimension references (members) defined in the Metrics Data Mart.

Figure 4-10 shows how the online forms are related to the CPMA. Each time a form is saved it corresponds to an event that is stored in the Metrics Data Mart. The data that is filled in for each field on the online form corresponds either to a measure or to a dimensional lookup table.

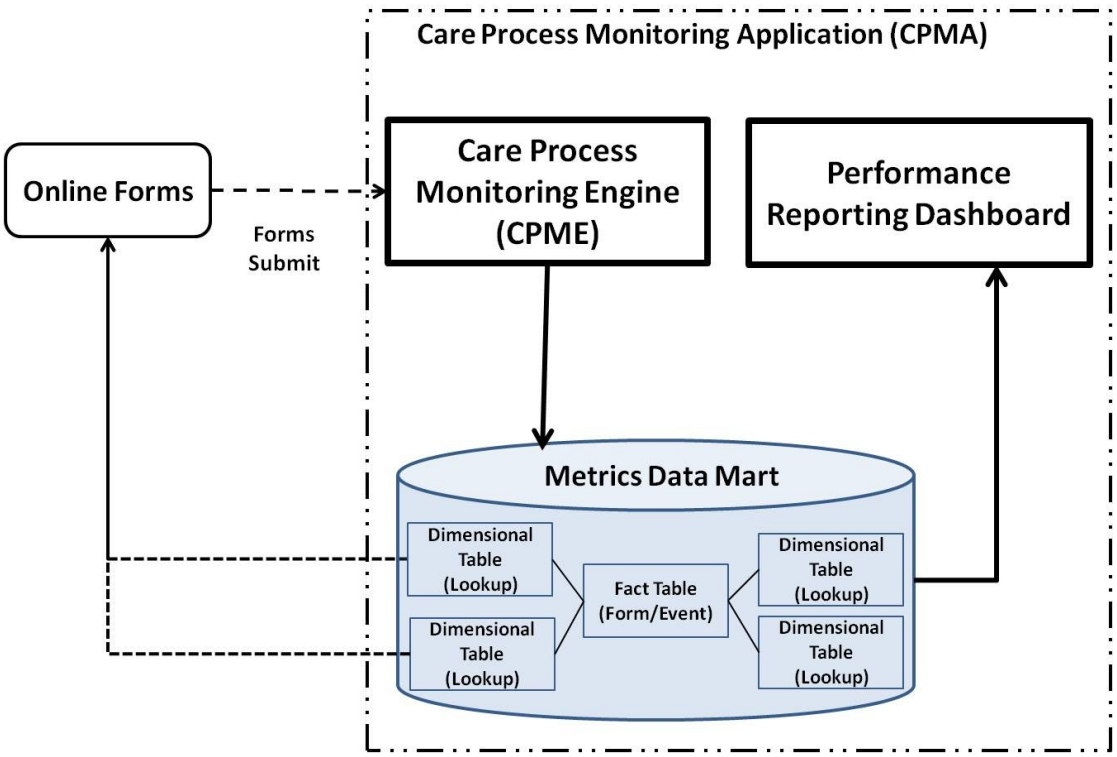


Figure 4-10: Online Forms Integration with CPMA

Another potential solution is to integrate a CPMA into the existing community care enterprise architecture as a care process registry. For example, identifying and tracking all patients in the community who are identified and being treated as palliative care patients. Note that many physicians and healthcare organizations in the community already each have their own Electronic Medical Record (EMR) application. As shown in

Figure 4-11, the integration process could be achieved by building an interface between the existing EMRs application and CPMA. So, whenever a new record, related to a specific patient, is entered in any of the EMR applications in the community, a corresponding event is generated and sent to CPMA in which the CPME will handle the event. In this respect, the patients across the community could be tracked and their states could be updated as events are happening.

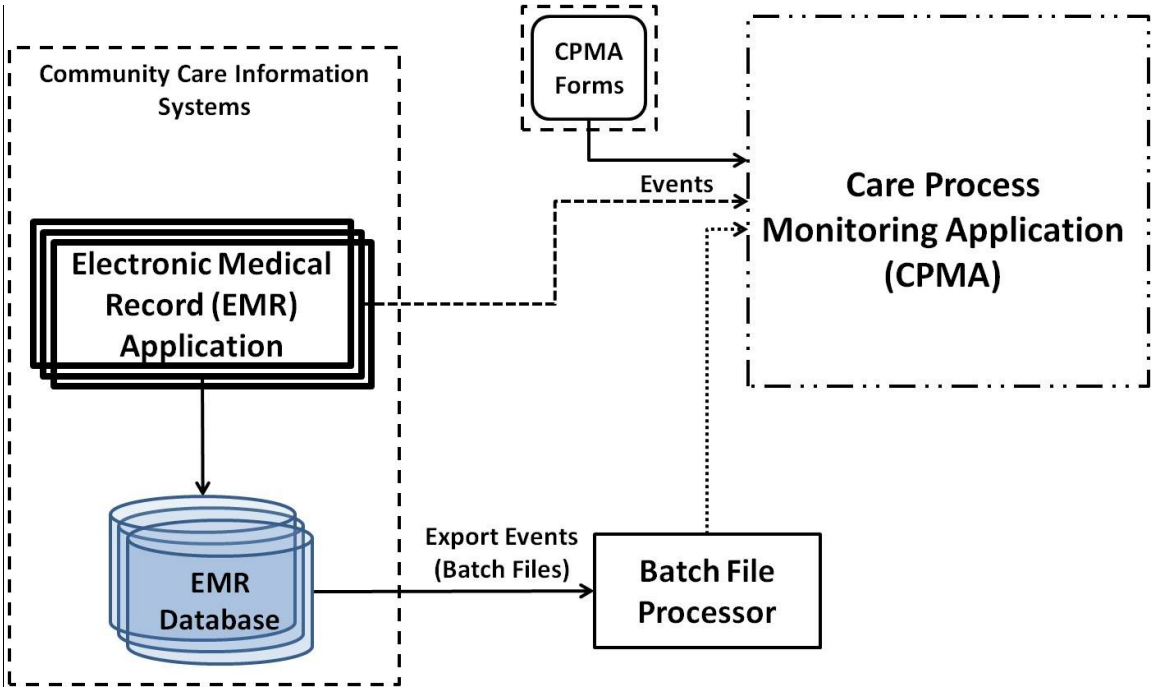


Figure 4-11: CPMA integrated into Community Care Architecture

Figure 4-11 also shows that one could export events out of different EMR database as batch files over fixed time interval. To be effective, the time interval needs to be short (once every 24 hours) so that the registry is kept abreast of events as the care process is taking place.

4.2.4 Integration into Hospital Care architecture

Currently, different hospital departments (emergency, cardiac, x-ray, pharmacy, etc.,) and often each has different information systems that do not easily communicate with each other. Since care processes can span across different departments, it is complex to figure out the current precise states of the resources involved in the process.

One potential solution for integrating CPMA into hospital care enterprise architecture is to have a Message Middleware publish events that a CPMA could subscribe to as depicted in Figure 4-12.

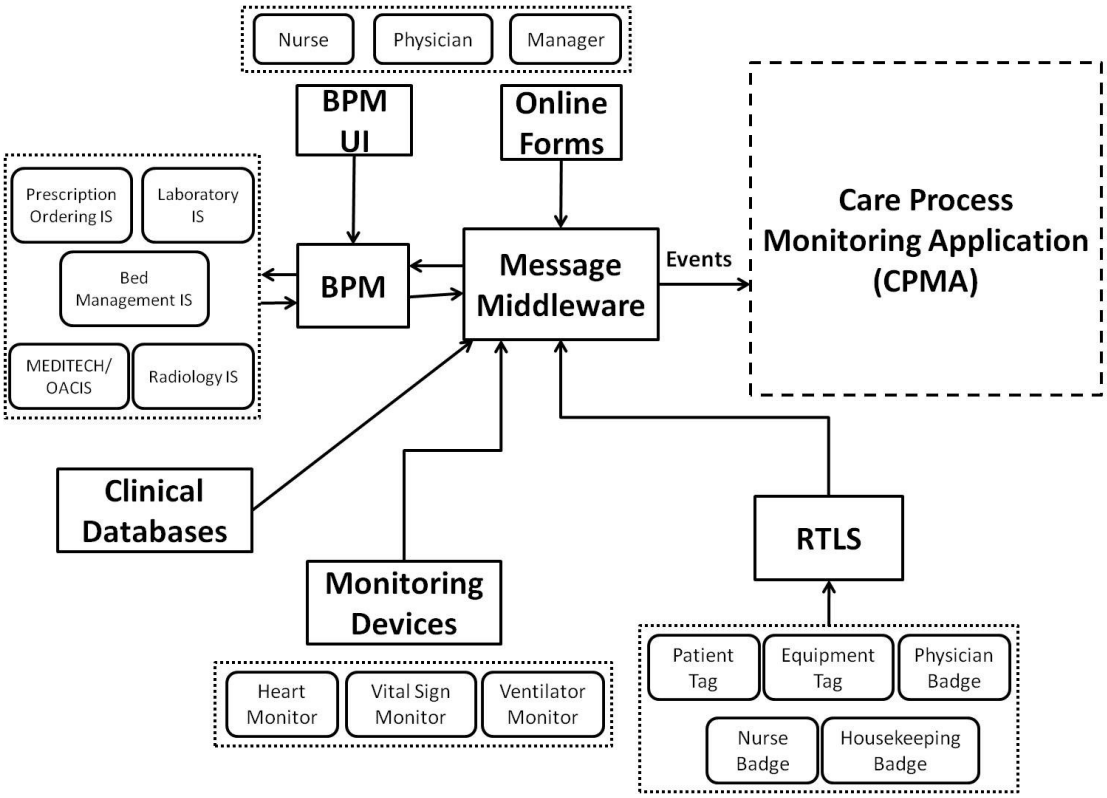


Figure 4-12: CPMA integrated into Hospital Care Architecture

The BPM server interacts with existing clinical information systems to support detailed work flow support of care process according to well-defined care process models. Each care process is connected to one or more health information systems to accomplish a certain task. A BPM guides each care provider in the steps to accomplish their tasks in real time. Note that a complete business process model must be in place that defines every step in the care process. However, for purposes of monitoring and evaluation of the process, only a portion of the business process model needs to be monitored by CPMA.

The care providers (Manager, Physician and Nurse) use the BPM web user interface (BPM UI) running at their portable devices to accomplish the required tasks. As a result of completing their tasks, the underlying hospital information system is updated and a

corresponding event is generated and can be communicated to other components via the Message Middleware. Also, the care process can receive events which can serve to trigger or coordinate or inform steps within the care process. Besides the BPM UI, clinicians can also use Online Forms independent of the BPM to enter information for particular events.

The RTLS server communicates events that identify the location of patients, equipment and care providers based on RFID readers in each room and RFID tags worn by the patients and RFID badges worn by care providers. The output of the server is filtered to send only the critical location events needed to identify state changes relevant to outcomes for the care process.

Monitoring devices are part of the hospital enterprise architecture. Those devices, such as heart, vital signs and ventilator monitors, are attached to the patients and monitor their hearts, blood pressure and pulse rate, etc. This type of device emits continuous events in a specific format to the Message Middleware.

The message middleware is the communication component in the architecture. It is in charge of transforming the events from all sources in a variety of formats into a single stream of events in a standardized format. The emitter of the event sends messages to the Message Middleware, and it will deliver the event to the subscribers. The entities (e.g., BPM, CPMA) who are interested in a particular event should register with the Message Middleware to receive the events. All the published events have to be registered in the Message Middleware. The major benefit of this component is that the publisher does not have to be concerned with the event format of the subscriber. In other words, the publisher publishes the event using its standard format and the Message Middleware converts the format in order to be acceptable by the

subscriber. Another benefit of the Message Middleware is its ability to pull data from clinical databases in the hospital and generate a stream of events to the subscribers.

As shown in Figure 4-12 , events are collected by the CPMA, from a BPM server, a RTLS server, as well as potentially directly from any of the existing systems in the hospital such as EHR and bed management via the Message Middleware. The output of the CPME is displayed in the Performance Reporting Dashboard which displays resources' states, durations and wait times of the states, as well as charts of hourly metrics and trends for overall patient flow. Note that all components connected to the Message Middleware are connected via web services technology.

4.3. Software Methodology for Developing a CPMA

In this section, we define an agile, behavior-driven methodology for developing a CPMA that leverages the application meta-model and general architectural pattern.

4.3.1 Overview

The software methodology for developing CPMA is an iterative process consisting of requirements analysis, test driven development based on test scripts originating from both candidate scenarios and the application model, and finally implementation and deployment to the desired enterprise architecture. Multiple versions of the application model are created when they are validated by the care process experts and the enterprise architect, as well, when there are errors in defining the requirements during the development process. As a consequence, new versions of CPMA are created once the application model is modified, or, once the tests against

CPMA have failed. The activity diagram show in Figure 4-13 identifies the steps taken to create the main artifacts of the CPMA.

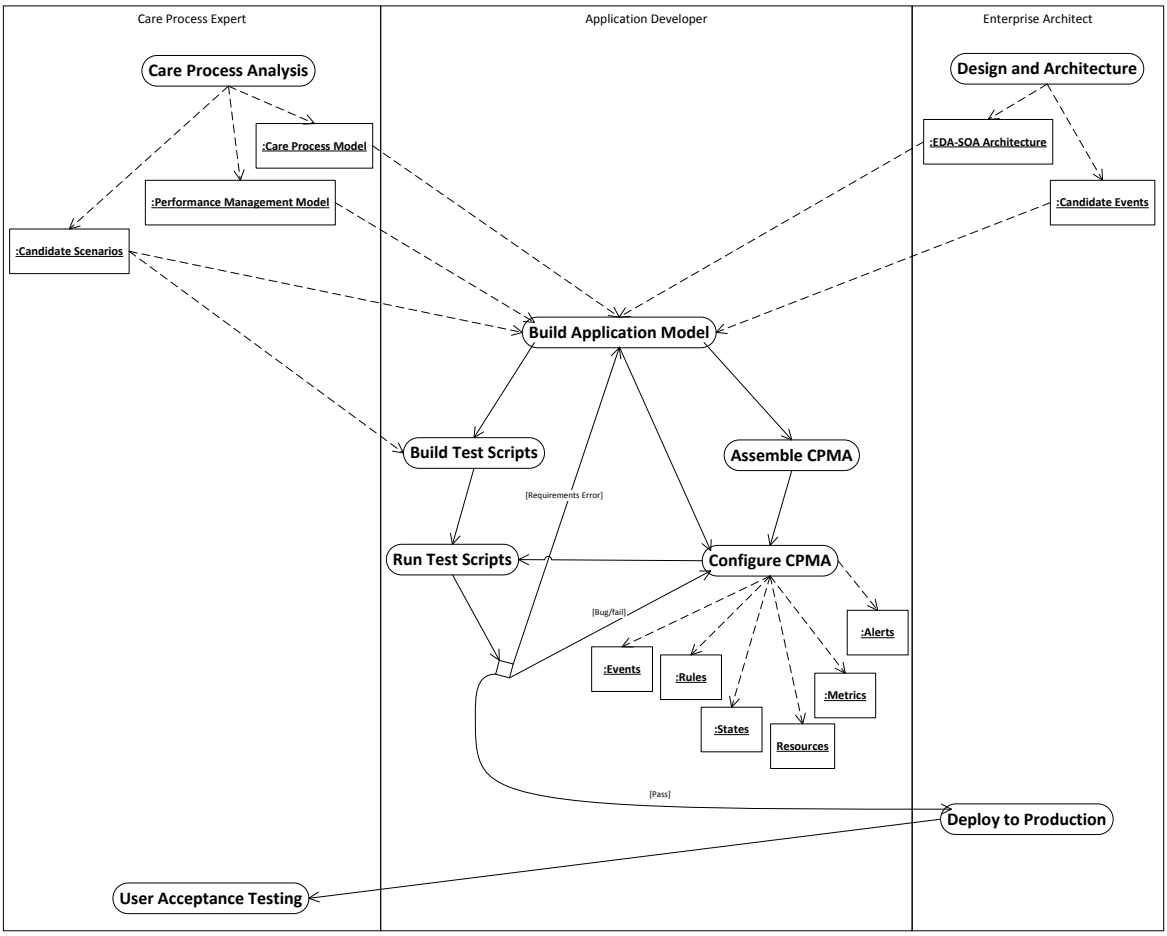


Figure 4-13: Software Methodology for Developing CPMA

The development of CPMA starts when the detailed information from the care process experts and the enterprise architect are available to the application developer who will then build the application model. Thereafter, the application developer can conduct two separate tasks at the same time; build the test scripts and select and build the components of CPMA. Then, the configuration and implementation of CPMA is driven incrementally by the test scripts. Finally, the care process expert follows the candidate scenario set by him to conduct the user acceptance

test after the CPMA is deployed into the healthcare organizations' enterprise by the enterprise expert.

4.3.2 Roles and Objectives

During the development of CPMA, the application developer works closely with the care process experts and the enterprise architect to leverage the artifacts produced by them. The following describes the roles and objectives for each of the participants involved in the development process of CPMA.

The **Care Process Expert** conducts a detailed analysis towards a particular care process. As a consequence, he/she provides to the application developer with two main models and a set of candidate scenarios to show how the care process is enacted. The first model is the performance management model in which the goals of monitoring a particular care process along with associated metrics are identified. The second model is the care process model in which the desired care process is represented as main steps or states along with expected events that trigger those states.

The **Enterprise Architect** is responsible for the EDA-SOA architecture available in the enterprise in which the web services interface to the event sources and the EDA (candidate events generated from those event sources in near real-time) are defined and provided to the application developer. In addition, the format of each event type produced from each particular event sources is determined.

The **Application Developer** is responsible for several activities in the software development methodology. Indeed, there are three parts to the role of application developer. The

first part is to conduct requirements analysis for the purpose of building the application model. The second part is to perform testing based on the candidate scenarios and the application model. The last part is to implement CPMA, which is developed based on the application model. Those three roles could be achieved by the collaboration of three different people in the team, or, it could be achieved by one or more persons.

4.3.3 Requirements Analysis for Care Process Monitoring Applications

This is the initial phase of the software methodology for developing a CPMA. It consists of three major activities conducted by three different roles. Particularly, those activities are Build Application Model, Care Process Analysis and Design and Architecture. The following describes each of the requirements activities.

Build Application Model is achieved by the application developer and considered as the fundamental part of the requirement analysis. Indeed, the task of building the application model is highly dependent on the artifacts (candidate scenarios, care process model and performance management model) that resulted from the care process analysis accomplished by the care process experts, as well as the artifacts (EDA-SOA architecture and candidate events) that resulted from the design and architecture task achieved by the enterprise architect. Basically, the application model is driven by the candidate scenarios in which the basic steps (states) are identified. After that, the application model is updated to show the following: which events generated in the enterprise systems trigger those states, which metrics are calculated from which events and states, the computation of the metrics, the alerts generated, the sources for each event and the rules to update the current states and the rules to transit from one state to another. Then, the application model is refined several times, with collaboration and feedback from the care

process experts and the enterprise architect, to include the inferred events and states, as well as, the attributes of all the events and states. A complete application model used to configure the aspects of CPMA for hospital and community care processes is depicted in Appendices A and B.

Care Process Analysis is accomplished by the care process expert. The outcomes of this activity are three artifacts leveraged for building the application model. The first artifact is the care process model represented as key steps (states) that should be monitored to measure the performance of the overall care process. The second artifact is the performance management model that identifies the goals the healthcare organization wishes to achieve towards a particular care process expressed in terms of metrics. Having this information, the application developer has to figure out which states and events are used to measure those metrics. The last artifact is the candidate scenarios to show precisely what information from which sources should be captured during the execution of the care process. In effect, the core benefits of the candidate scenarios for the application developer is to infer new events not provided by the enterprise architecture, infer new states not provided by the care process experts and infer rules to update and trigger states. All the artifacts produced in this activity could be developed while the application model is built.

Design and Architecture activity is executed by the enterprise architect to provide information about the existing enterprise architecture with which CPMA will be integrated. This activity produces a complete picture about the EDA-SOA architecture that supports monitoring the care process in terms of the available event sources and what are the candidate events and its format that might be generated from those sources that are used to compute metrics or used to define states which are used to compute metrics. The event sources could be systems within the

existing hospital IT infrastructure or new system components integrated within the current hospital system such as BPM, RTLS or even online forms. According to the analysis of this activity, the application developer has to figure out the missing inferred events and states by writing specific rules.

4.3.4 Test Driven Development with Scenarios

As shown in the software methodology activity diagram in Figure 4-13, the test driven development phase consists of three tasks performed by two different actors. Build test scripts and run test scripts activities are conducted by the application developer, while user acceptance testing is accomplished by the care process experts. In fact, building test scripts and user acceptance testing activities are driven by candidate scenarios. The following describes the aforementioned activities.

Build Test Scripts is performed according to the candidate example scenarios. However, this activity will be started once the application developer completed building the initial application model. The objective of building test scripts is to achieve the following:

1. Simulate diverse event sources within the hospital and the community enterprise architecture.
2. Define the expected output test results, in the form of reports (metrics), to be shown in the Performance Reporting Dashboard.
3. Define the collection of expected events that should be monitored, collected and processed by CPME.

The following is a snapshot of the test script build for a Hospital-based care process monitoring that simulates CEP and BPM events to test CPME.

```
<?xml version="1.0" encoding="utf-8"?>
<oslerTestScript>
  <testName> Test Script to test SME</testName>
  <testDescription>Contains all the input events for PFM. Ordered by timestamp. Sources simulate coming from either BPM or
CEP</testDescription>
  <!-- Input events -->
  <event name="WaitForConsultation1" sourceSuffix="CEP">
    <Patient_ID>Pa123456</Patient_ID>
    <Location_ID>AssessmentRoom</Location_ID>
    <timestamp>2012-03-12 01:12:00</timestamp>
  </event>
<event name="WaitForBed" sourceSuffix="CEP">
    <Patient_ID>Pa123456</Patient_ID>
    <Unit_ID>CW</Unit_ID>
    <timestamp>2012-03-12 02:07:00</timestamp>
  </event>
  <event name="WaitForTransport" sourceSuffix="CEP">
    <Patient_ID>Pa123456</Patient_ID>
    <Unit_ID>CW</Unit_ID>
    <timestamp>2012-03-12 02:26:00</timestamp>
  </event>
  <event name="DischargeCompleted" sourceSuffix="CEP">
    <Patient_ID>Pa123456</Patient_ID>
    <Unit_ID>CW</Unit_ID>
    <timestamp>2012-07-22 17:49:00</timestamp>
  </event>
  <event name="BedCleanUpStarted" sourceSuffix="CEP">
    <HouseKeeping_ID>HK444444</HouseKeeping_ID>
    <Location_ID>R107</Location_ID>
    <Unit_ID>CW</Unit_ID>
    <timestamp>2012-03-14 18:00:00</timestamp>
  </event>
</oslerTestScript>
```

Run Test Script is an activity achieved before integrating the actual CPMA into the EDA-SOA enterprise architecture. To this end, we developed a testing component called Test Console as an option to facilitate the task of running test scripts automatically and it has the capability to act like any event source within the enterprise architecture. Figure 4-14 is a snapshot of the test console that shows a list of events that can be run and sent to CPMA one at a time. Another option for testing CPMA is to walk through the contents of the test scripts manually by tracing each of the events with their attributes and figure out whether the

corresponding outputs could be possibly computed and satisfied. In the latter, this manual testing is performed by the application developer in case the CPMA is not configured at the time the test scripts are available. Test failure might result when running the test scripts. In this case, the cause of the error might be in the configuration of the CPMA or might be an error in requirements. In the former, the configuration should be updated to resolve the problem; in the latter, the application model should be updated to resolve the problem.

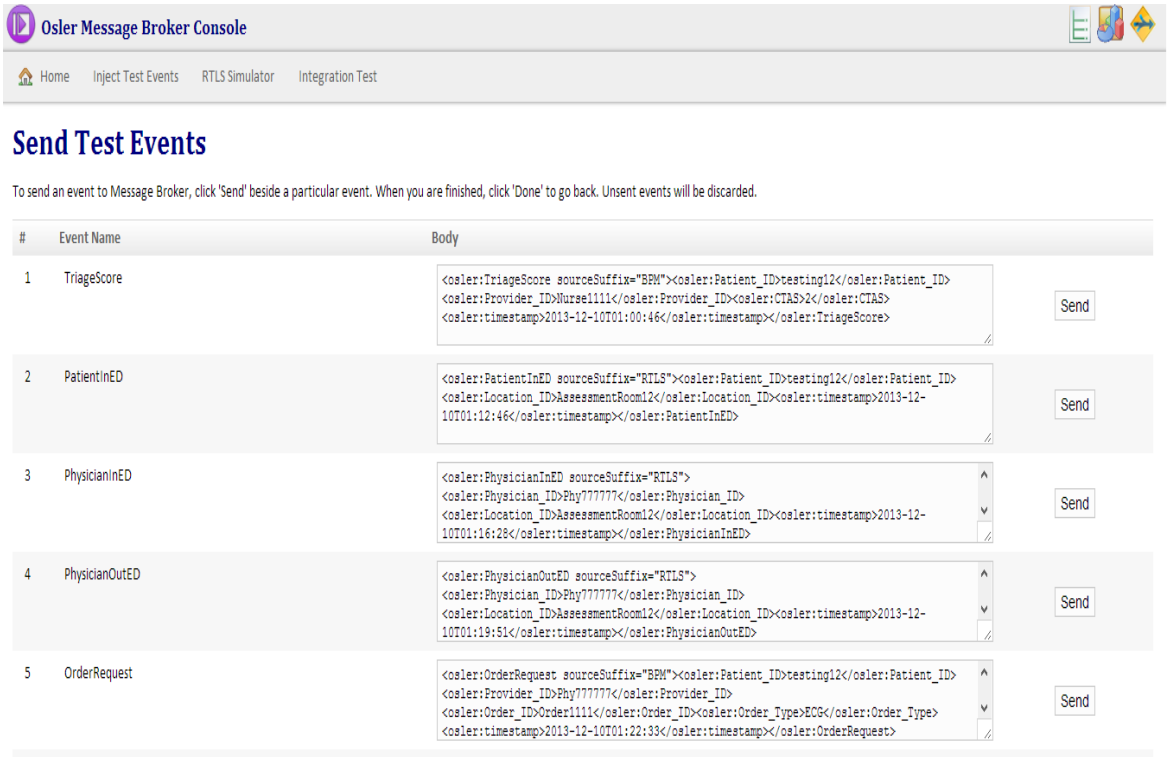


Figure 4-14: Test Console

User Acceptance Testing is the last activity in the testing phase and even the last activity in the software development methodology for developing CPMA. As stated before, this process is accomplished by the care process experts and is driven by the candidate scenarios. In this situation, the care process experts use the actual CPMA integrated into the enterprise architecture to walk through the candidate scenarios on which the test scripts were built and test both whether

they are able to get the expected results and whether the CPMA integrates properly into their work processes and information infrastructure.

4.3.5 Implementation and Deployment

As depicted in the software methodology diagram shown in Figure 4-13, this phase of development focuses on implementing and deploying the CPMA into the enterprise architecture. The implementation stage consists of assembling and configuring CPMA. While in the deployment stage, development to production activity is achieved in order to integrate CPMA into the EDA-SOA enterprise architecture. The following illustrates the activities to implement and deploy CPMA.

Assemble CPMA activity is performed by the application developer in which the appropriate tools, technologies and custom components to build and implement different pieces of the CPMA (CPME, Metrics Data Mart, and Performance Reporting Dashboard) are selected. Usually, Commercial-Off-The-Shelf software is leveraged for this purpose. For example, IBM Cognos Business Intelligence Suite could be exploited to build and implement the Performance Reporting Dashboard component of CPMA and IBM Websphere Business Events could be used for rules.

Configure CPMA is a straightforward process. This activity is performed after the CPM pattern is instantiated and parallel to the running test scripts activity. In other words, configuring CPMA is driven by running test scripts via test console. However, configuring the CPMA is concerned with defining six artifacts, which are events, rules, states, resources, metrics and alerts. Those artifacts are explicitly defined in the application model. Events and states and their

associated attributes are defined and enumerated in the event and the state dimensions (lookup tables) respectively in the Metrics Data Mart. Rules are either implemented using a specific tool in CPME, such as IBM Websphere Business Events, which can be leveraged to write complex rules to infer new events, or implemented in the state handlers of the State Inferencing component. Each state handler has a set of rules in order to infer new states based on the current state of the resource and a set of received events. The following shows an example rule for transitioning a patient to the state WAIT_FOR_BED_CW by integrating source events PhysicianOutED generated from RTLS, OrderBed and BedNotAvailable generated from BPM to infer the state WAIT_FOR_BED_CW.

If CURRENT_STATE is IN_PHYS_RE_ASSES
AND EVENT PhysicianOutED
AND EVENT OrderBed
AND EVENT BedNotAvailable
Then CURRENT_STATE is *WAIT_FOR_BED_CW***.**

Reports are implemented to display Metrics in the Performance Reporting Dashboard. They can be implemented using basic SQL queries and Excel, or an enterprise reporting tool like IBM Cognos Business Intelligence Suite.

Deployment into Production activity is only conducted when testing CPMA via Test Console is passed. Basically, this task is concerned with integrating CPMA into the EDA-SOA architecture. Therefore, it requires some sort of production configuration ensuring that CPMA is able to receive the expected events from the expected event sources.

4.4. Chapter Summary

In this chapter, we proposed an application framework for monitoring care processes. It encompasses three main elements. The first element is a state-based application meta-model that defined the fundamental elements and the relationships among them that were used to define the CPMA. The second element is the Care Process Monitoring (CPM) architectural pattern that addressed the problem on how to integrate the CPMA into the hospital or community enterprise architecture, and the last element is the software engineering methodology for developing and deploying the CPMA.

In the next chapter, we will present three different case studies that use our application framework to build two different CPMAs, one for hospital care and the other for community care, in collaboration with the staff of two different hospitals located in Ontario.

Chapter 5. Care Process Monitoring Case Studies

Our application framework for care process monitoring applications was developed iteratively over a period of 3 years during the development of two different applications (one in community care and one in hospital care) in collaboration with clinical and managerial staff from two different healthcare organizations in Ontario. During that time, 3 different case studies, each resulting in a prototype CPMA, were developed, evaluated and validated. In the first case study, we used a preliminary version of our application framework which had a well-defined architecture and engineering approach but did not have an application meta-model. Our experience with the first case study led to creation of the complete application framework, presented in chapter 4, which was used in the second and third case studies.

In section 5.1, we explain the role of the thesis researcher in the case studies and the nature of his participation. In section 5.2, the details of the first case study are provided in which a prototype CPMA for cardiac care in a hospital setting was built in collaboration with IBM and William Osler Hospital. Then, Section 5.3 describes the second case study in which the CPMA from the first case study was refined and extended and re-engineered using our complete application framework. Finally, section 5.4 describes our final case study in which a prototype CPMA for palliative care in a community setting was built in collaboration with Bruyère Hospital and the Champlain LHIN (Local Health Integration Network). At both Osler and Bruyère hospitals, there were existing information systems in place that provided some capability for monitoring cardiac care and palliative care processes, respectively. So, the

evaluation of our application framework used in each case study included an evaluation of the quality and efficacy of the CPMA produced in comparison to the existing information systems.

5.1. The Roles of the Thesis Researcher in the Care Process Monitoring Case Studies

The thesis researcher was a software engineering researcher on two different large teams (Cardiac care team and Palliative care team) conducting research in health informatics domain.

- The Cardiac care team consisted of the following participants: two university professors from University of Ottawa including the thesis researcher's supervisor, two PhD students including the thesis researcher, who was the Enterprise architect interface to IBM, and the other PhD student, Alain Mouttham, who was the care process expert interface to Osler Hospital, three master's thesis students and three master's project students.
- The Palliative care team consisted of the following participants: two professors from University of Ottawa, including the thesis researcher's supervisor who was the Enterprise architect interface to LHIN, and Dr. Craig Kuziemsky, who was the Care Process expert interface to Bruyère Hospital, one PhD student (the thesis researcher), two master's thesis students and 5 master's project students.

The thesis researcher was a participant in the software methodology for developing CPMA in all case studies. After he gained experience from his participation in the methodology and architecture in the initial case study, the thesis researcher became the sole researcher who focused on creating and developing the CPMA application framework, which consists of an application meta-model, an architectural pattern and a methodology for developing CPMA. The

development team in the second and the third case studies agreed to adopt the application framework proposed by the thesis researcher instead of following the traditional software engineering approach to building the CPMA. However, there are two master's theses focused on the detailed implementation and design of CPME component. The first master thesis (Baffoe, 2013) focused on the specific design and implementation of CPME implemented for the second cardiac care case study. The second master thesis, which is still in progress, focuses on the specific design and implementation of CPME (Quickforms service) implemented for palliative care case study.

Regarding the first case study, the thesis researcher participated in the architecture design and the methodology for developing CPMA. In general, he was involved in the testing and part of the implementation phase; in particular, he was responsible for the following activities: Build Test Scripts, Run Test Scripts and Configure CPMA (Configure rules). For the second and the third case studies, the thesis researcher was in charge of defining the application framework (application meta-model, architectural pattern and the methodology) that should be followed by the cardiac care and palliative care teams. Specifically, the thesis researcher was solely responsible for the creation of the application model leveraged by the developers of the two teams in order to implement different pieces of the CPMA. In addition to the above, the thesis researcher was consulted during the testing of CPMA, as well as during the implementation of the rules.

Concerning the aforementioned two master theses, the students built their CPMA based on and adhered to the contents of the application model that was created by the thesis researcher and following the application framework created by the thesis researcher as well.

5.2. Initial Cardiac Care Process Monitoring Case Study

A prototype CPMA for cardiac care in a hospital setting was built in collaboration with IBM and Osler Hospital as we developed our ideas on how best to engineer a CPMA. In this initial case study, I was part of a research team that proposed an architecture for leveraging emerging technologies to monitor care processes in near-real time. There was no concept of an application meta-model, and the team followed a classical software engineering approach to building the CPMA. This initial case study was defined by performing the following:

1. Selected a well-defined clinical pathway at Osler Hospital for Acute Coronary Syndrome (ACS) as the care process that would be monitored by our prototype CPMA.
2. Conducted several meetings and interviews with a domain expert to document and analyze the existing cardiac patient flow and define a complete example use case scenario.
3. Understood the gaps in existing information technology for monitoring the patient flow and investigated emerging technologies and approaches to address in an initial architecture for hospital care process monitoring.
4. Developed a CPMA for Acute Coronary Syndrome (ACS) at Osler Hospital using a traditional software engineering methodology.
5. Created a demonstration environment that would allow domain experts and clinicians to walk through the cardiac care scenario using the prototype CPMA we built.

6. Evaluated and reviewed results with domain expert based on survey of 80 staff at Osler Hospital and 10 IBM staff who walked through the prototype in the demonstration environment.

The objective of the case study was to engineer a CPMA for cardiac care with a focus on measuring wait times in very fine-grained detail throughout the cardiac care process in order to identify bottlenecks. More specifically, the following types of wait times were measured:

1. Patient wait times. For instance, the wait time for consultation with physician, the wait time for admission, and the wait time for discharge.
2. Service wait times. For instance, how long it takes to get a test result (e.g., ECG).
3. Operation wait times for housekeeping and patient transport.

5.2.1 Existing Cardiac Care Environment

The existing environment to monitor the cardiac care process and to provide performance management reports at Osler Hospital is entirely based on the ETL-Date Warehouse architecture pattern. Information collected from medical papers forms, while the cardiac care process is taking place, is entered manually, hours or even days after the fact, in the disconnected information systems of different departments of the hospital. The ETL processes to populate the hospital data warehouse take days and sometimes weeks to incorporate a view of all data that is available for the care process, and the data that is available is not fine-grained enough to have a complete view of all steps in the process. Hence, the architecture is unable to provide a near real-time view of what is happening at each step in the care process, while the care is taking place. Also, the reports, which can be created from the data warehouse, are unable to pinpoint the

individual states of the cardiac patients, key care providers and the beds in the cardiac care process. In particular, the reports cannot pinpoint and measure the individual wait times of cardiac patients and service times in each step of the care process. Only high level wait times and service times are available (e.g., wait time from triage to procedure, service time for procedure, wait time from procedure to discharge).

5.2.2 Care Process Monitoring Application (CPMA)

To address the limitations in the existing cardiac care environment, we developed a Care Process Monitoring Application (CPMA) that has the capability to provide fine-grained monitoring of the cardiac patients and other resources such as rooms and beds in terms of states and events related to those resources participated in the cardiac care process from the time the cardiac patient arrives at the Emergency Department to the time the patient is discharged.

The CPMA that was developed specifically for this case study has the following functionalities:

1. Care Process Performance Reports (Metrics): Key metrics and statistics are reported in near-real time in order to quantify performance (see Figure 5-1).
2. Detailed Care Process States and Wait Times: Each of the states identified in the care process model is characterized and its duration quantified in near-real time, with a range of acceptable values defined to drive alerts (see Figure 5-1 and Figure 5-2).
3. Infer Complex Events: These are defined by integrating care process events from the BPM with RTLS events to extract measures and aggregate them to provide 1) and 2) above.

4. Location tracking of the key care providers and cardiac patients in the hospital. The current locations of the cardiac patient and the key care providers are tracked on a floor map representing the layout of different units in the hospital (see Figure 5-4).

Figure 5-1 is a sample near real-time report generated by CPMA which shows the states and its duration along with the target duration for each state the cardiac patient has been through so far in the ACS clinical pathway. As shown in Figure 5-1, the cardiac patient is in TRIAGED state when he arrived in ED for 2 seconds (within the target range) and then the patient went through several states until he arrived to the bed in Cardiology Ward (CW) in which the state is changed to IN_BED_CW. The color coding in Figure 5-1 represents the following:

1. The state flagged with a red color indicates that the state exceeds its target range.
2. The state flagged with a yellow color indicates that the state is near overdue.
3. The state flagged with a green color indicates that the state is within acceptable target.
4. The state with no color coding has not time limit specified.

Patient States

Search:

State	Start Time	End Time	Duration (mins)	Target (mins)
IN_BED_CW	2013-03-02 11:13:00.0	N/A	50	N/A
IN_TRANSPORT_CW	2013-03-02 11:02:00.0	2013-03-02 11:13:00.0	11	15
WAIT_FOR_TRANSPORT_CW	2013-03-02 10:39:00.0	2013-03-02 11:02:00.0	23	15
WAIT_FOR_BED_CW	2013-03-02 10:13:10.0	2013-03-02 10:39:00.0	26	480
IN_BED_ED	2013-03-02 10:12:00.0	2013-03-02 10:13:10.0	2	N/A
IN_PHYS_RE_ASSESS	2013-03-02 10:00:00.0	2013-03-02 10:12:00.0	12	N/A
WAIT_FOR_PHYS_RE_ASSESS	2013-03-02 09:40:00.0	2013-03-02 10:00:00.0	20	30
WAIT_FOR_ORDERS_EXECUTION	2013-03-02 08:19:00.0	2013-03-02 09:40:00.0	81	30
IN_BED_ED	2013-03-02 08:15:00.0	2013-03-02 08:19:00.0	4	N/A
IN_PHYS_INIT_ASSESS	2013-03-02 08:12:00.0	2013-03-02 08:15:00.0	3	N/A
WAIT_FOR_PHYS_INIT_ASSESS	2013-03-02 08:10:00.0	2013-03-02 08:12:00.0	2	30
TRIAGED	2013-03-02 08:08:00.0	2013-03-02 08:10:00.0	2	30

Showing 1 to 12 of 12 entries

Overall Duration: 3 hours 1 mins

Figure 5-1: Patient States Duration against Target Duration

Figure 5-2 shows the average time patients spend in each step in the cardiac care process provided by the CPMA, which can be used to help identify systemic problems. Providing near real-time reports as in Figure 5-1 and charts as in Figure 5-2 is difficult and requires tedious work to integrate data from distributed data sources. For each state, events or combinations of events need are identified to determine start, end and duration of states precisely.

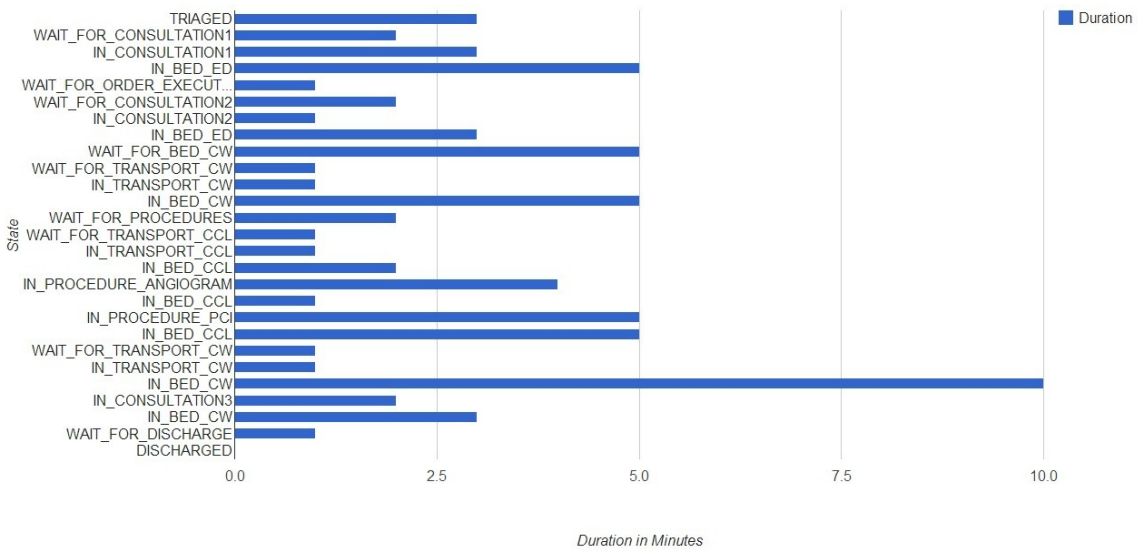


Figure 5-2: Average Time Patients Spend in Each Step of the Cardiac Care Process

Patients transition from one state to another state within a particular care process as events occur. While the cardiac patient goes through the ACS clinical pathways, several events related to the patient are published that are either source events (i.e. from BPM, RTLS) or complex events (inferred by a CEP). Figure 5-3 is a snapshot from a CPMA application that shows an event list for a particular patient.

Upon admission, patients are each given a smart wireless sensor tag that can detect the infrared beams of beacons placed in rooms and that uses Wi-Fi to communicate its location information to a RTLS. Care providers are also given smart sensor tags to wear to identify their location as well. Figure 5-4 shows a sample map of the location of care providers and cardiac patients. This map shows the ED room, triage space, assessment room on the left side of the map. The right side of the map shows three rooms, 105, 106, and 107. Two of these rooms are occupied with patients, and one room is available.

Patient Info:Pa111118 Room: R101

Patient States Chart | Patient States | Events Received

Events Received

Search:

Event data received	Received Time
ProceduresScheduled	2013-04-30 14:30:00.0
PatientInCW	2013-04-30 14:25:00.0
PatientOutED	2013-04-30 14:07:00.0
PatientTransportRequest	2013-04-30 13:41:00.0
PatientAdmittedWithBed	2013-04-30 13:29:00.0
PhysicianOutED	2013-04-30 08:50:00.0
PhysicianInED	2013-04-30 08:05:00.0
PatientAdmittedWithNoBed	2013-04-30 08:03:10.0
OrderRequestCompleted	2013-04-30 07:40:00.0
OrderRequestCompleted	2013-04-30 07:10:00.0
OrderRequestCompleted	2013-04-30 07:05:00.0
OrderRequest	2013-04-30 06:51:00.0
OrderRequest	2013-04-30 06:50:00.0
OrderRequest	2013-04-30 06:49:00.0
PhysicianOutED	2013-04-30 06:45:00.0
PhysicianInED	2013-04-30 06:26:00.0
PatientInED	2013-04-30 06:10:00.0
Triage	2013-04-30 06:01:00.0

Showing 1 to 18 of 18 entries

Figure 5-3: Event List Related to a Particular Cardiac Patient

A demonstration environment was set up at Osler Hospital, and 80 staff (plus 12 from IBM) walked through simulated scenarios and reviewed the overall architecture, the BPM-supported care process with integrated RTLS support, as well as the timely reporting provided by the CPMA. 12 demo sessions were conducted at Osler Hospital and the validation was very positive based on the survey answered by 80 participants of Osler staff. The survey showed that 82% of the participants were of the opinion that these technologies and architecture would improve patient flows and would be better than what exists today. In addition, more than 50% believed that the technology should ideally be deployed within the next 2 years (Mouttham A. , A Framework for Real-Time Analytics and Decision Support in Patient Flow Management.

Invited Talk, 2013) (Amyot, An architecture and a platform for real-time, location-based patient flow monitoring. Invited talk, 2012).

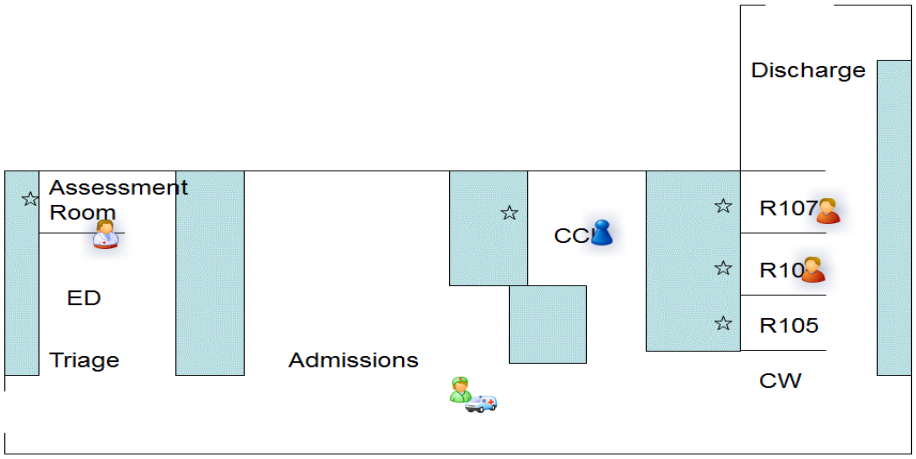


Figure 5-4: Patients and Key Care Provider's Location Map

5.2.3 Model

There was no single model built in the initial case study. Rather a disjoint collection of models was used to understand the requirements and to implement CPMA. The collection of models showed the following:

1. Sequence of key cardiac care wait and service states that should be monitored in order to measure their duration.
2. Detailed Business process model of the complete ACS Clinical Pathway.
3. A detailed interaction diagram that listed the expected interactions between each component of the architecture as the care process took place to identify what events would be delivered to the CPMA.

The first model built is the sequence of key ACS service and wait states, since the focus is to monitor key steps in the cardiac care process rather than the whole care process. This model is shown in Figure 5-5 below.

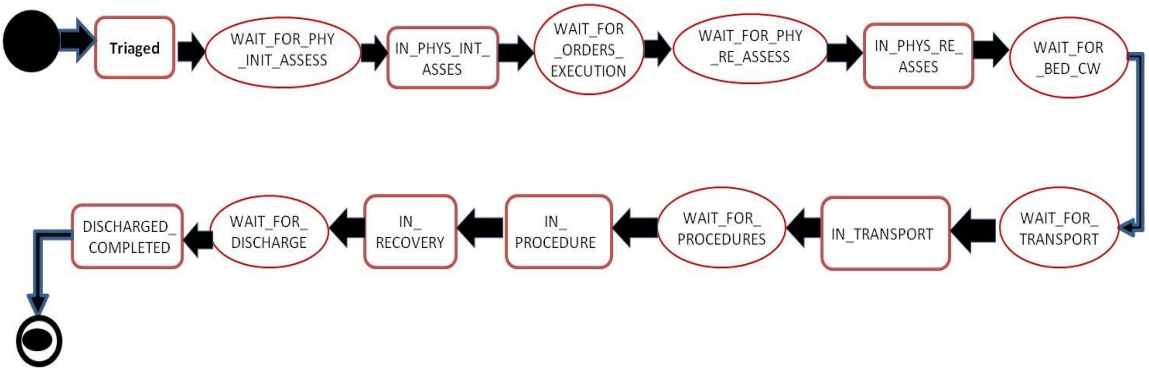


Figure 5-5: Care Process States for Cardiac Patient

The transition from one care process state to another will be inferred from the occurrence of either a BPM event, an RTLS event, or a CEP event (see the architecture diagram in the next section). The duration for each state is calculated by taking the absolute difference between the timestamp of the event that induces the transition of the current state and the timestamp of the next event that causes the transition to the next state in the model.

The second model built is the complete business process model that represents the ACS Clinical Pathway. This model was implemented by a Master student, Renaud Tchemeube (Tchemeube, 2013). Figure 5-6 is a subset of cardiac patient flow, which is modeled using IBM Business Process Manager, which shows what complex events should be received in order to continue executing the care process.

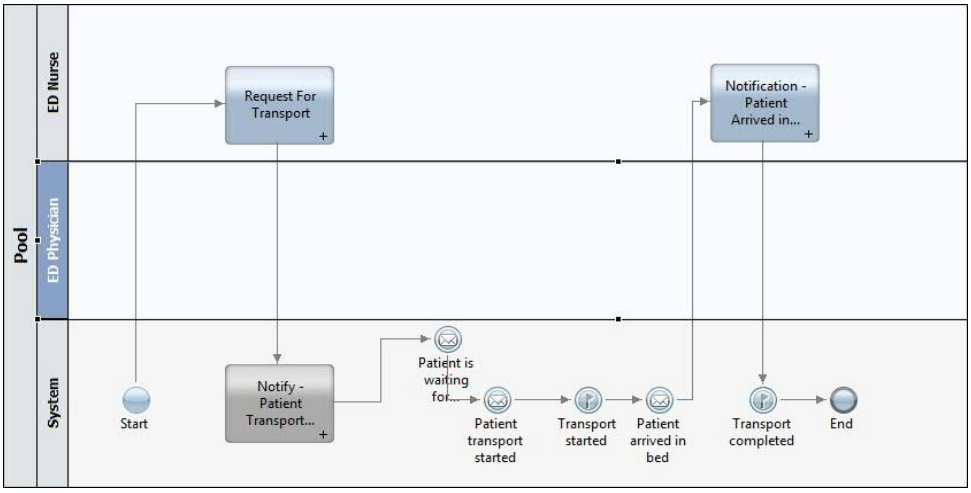


Figure 5-6: Subset of Cardiac Patient Flow Model

Finally, a complete interaction diagram, as shown in Figure 5-7, was created to show the interactions between RTLS, BPM, CEP and PFM. The PFM (Patient Flow Monitor) was the name given to the CPMA which monitored events from RTL, BPM and CEP and displayed metrics for monitoring the cardiac care process.

Figure 5-7 shows the stream of events that the PFM would be processing at each step in the cardiac care process. The Patient is triaged in the Emergency Department (ED) by a Nurse, and then the Patient enters the assessment room where he/she waits for the Physician. At each step there is event messaging to the PFM. For example, in the next step, when the Physician is in the assessment room in order to diagnose the Patient, the event PatientInED and PhysicianInED are published by the RTLS and consumed by CEP. Thereafter, the Physician orders ECG and blood test by accessing the web interface to BPM via mobile device (e.g. Tablet or Smart Phone). As a consequence, the events OrderRequest(ECG), OrderRequest (Blood Sample) and OrderRequest (BloodAnalysis) are emitted from BPM and consumed by CEP. When the Physician leaves the assessment room, the CEP starts processing the received events by applying

a specific rule to produce the complex event (inferred event) ConsultationCompleted which is consumed by the BPM and PFM.

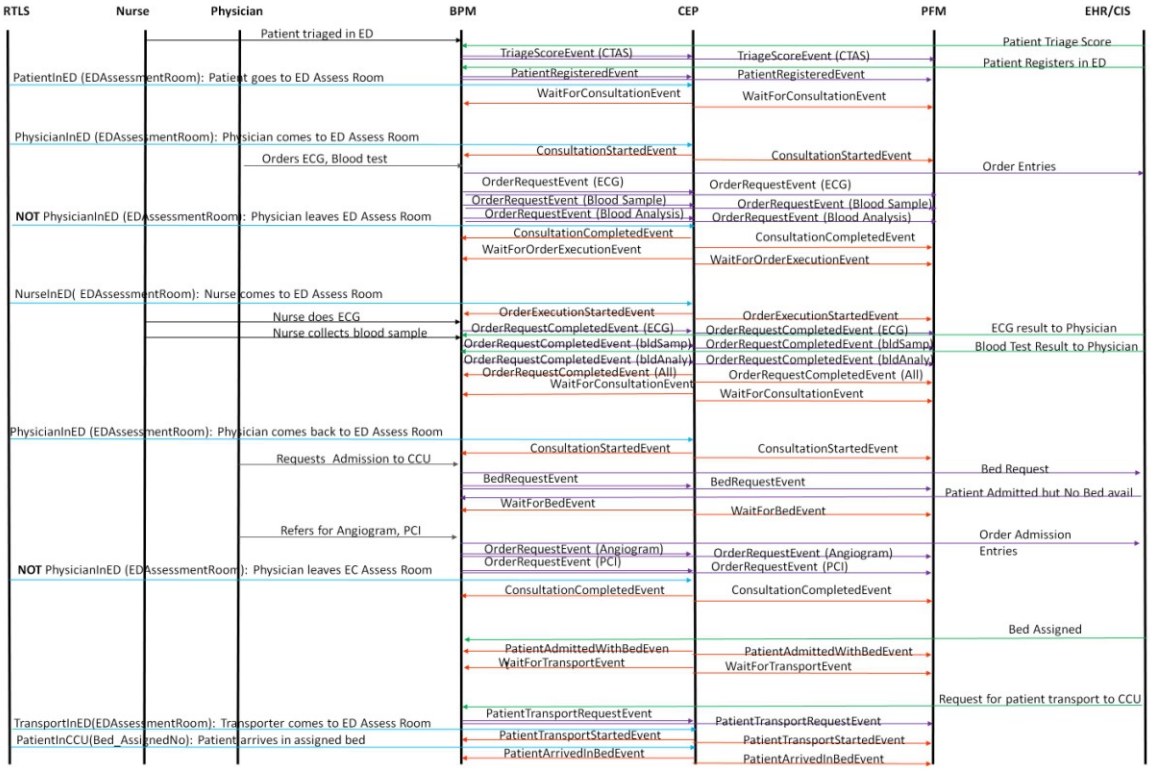


Figure 5-7: Event Interaction Diagram

5.2.4 Architecture

We partnered with IBM to use existing technology from their product stack: IBM WebSphere Message Broker, Websphere Business Events for complex event processing (CEP) and IBM Business Process Manager (BPM). Figure 5-8 shows the architecture for the initial case study.

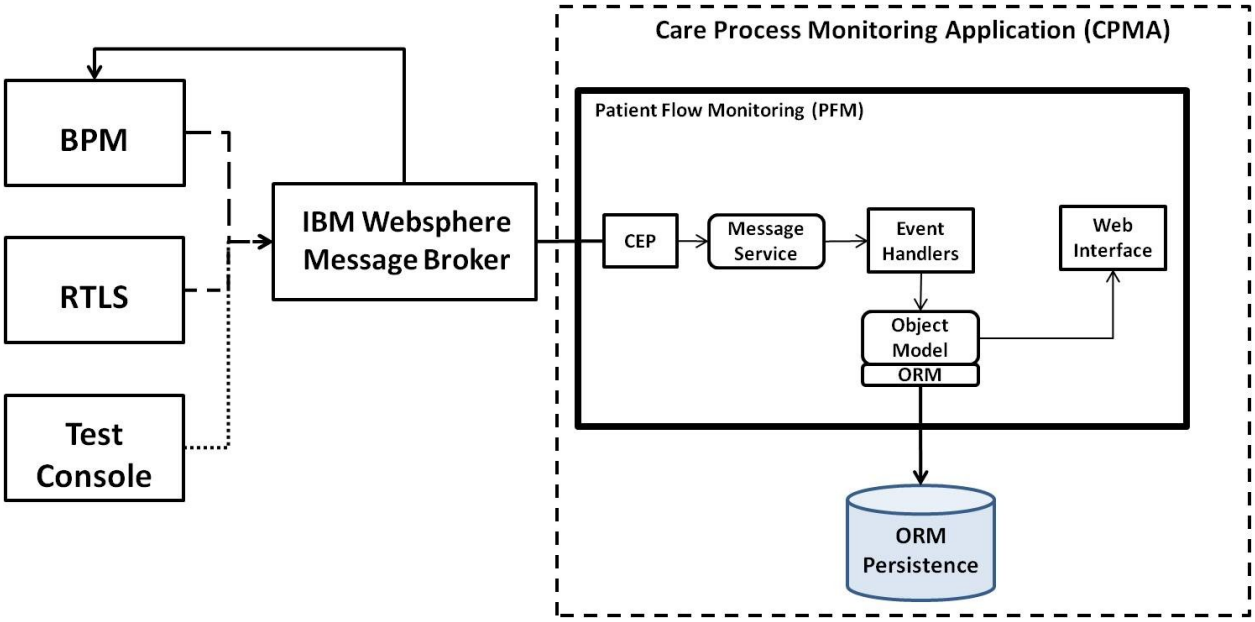


Figure 5-8: Architecture of CPMA for Initial Case Study

As shown in Figure 5-8, events are collected by CPMA, from a BPM server and a RTLS server, through IBM Message Broker. A Test Console is also part of the architecture, from which events from any source can be simulated in order to test the CPMA.

The IBM Message Broker is the communication component in the architecture. It is in charge of transforming the events from all sources in a variety of formats into a single stream of events in a standardized format. We implemented this component using IBM WebSphere Message Broker.

The BPM server, which was implemented using IBM Business Process Manager, guides each care provider in the steps to accomplish their tasks in near real-time. Note that a complete care process model must be in place that defines every step in the care process. However, for purposes of monitoring and evaluation of the process, only a subset of the care process model

needs to be monitored by the state monitoring service. The CPMA receives messages from the BPM server when events occur that should be monitored.

The RTLS server tracks patient and care provider's locations as they enter and leave rooms in the hospital wearing the smart sensor tags. The RTLS server sends the CPMA only the critical location events needed to identify state changes relevant to outcomes for the care process.

The Test Console is used to test the system with a complete simulation of our scenario with events of any type from any source (i.e., RTLS, BPM or other system) and to send them to the CPMA via the Message Broker. The simulated events are generated by providing a XML formatted test script to the console, which can be used to send events manually one at a time or in batch.

The CPMA was implemented as a custom-built web application in Grails, called Patient Flow Monitoring (PFM) (Luo, 2012). PFM processes the incoming events from IBM Message Broker, creates a log of all events as they are processed, infers events via a CEP component that determines the start and end states and hence infers patient states, updates the current object model as events occur and finally logs the current states and events objects related to patients in an Object-Relational Mapping (ORM) database as events are processed and the object model is updated.

The CEP is part of PFM, which performs pattern matching to recognize patterns of events that can be integrated to infer more complex events. CEP integrates events generated from BPM and RTLS through the Message Broker based on rules implemented in the CEP engine. These inferred events mark the state transitions in the care process. In our architecture, CEP was

implemented using IBM WebSphere Business Events. For example, the *WaitForConsult1* state can only be inferred by combining the event generated from BPM, which is *Triage*, with the event generated from RTLS, which is *PatientInED*.

The message service parses the message generated from CEP and delegates the message to the appropriate event handler which in turn updates the object model. Event handlers are hard coded and each one processes a particular event type to infer states, update resource information and measure metrics concerning wait and service times.

Reports on metrics as well as patient location, patient status and room status are displayed in the web interface. The display is updated as the object model is updated. It displays, in near real-time, patient states, durations and wait times, as well as charts of hourly metrics and trends for overall patient flow. The care providers can use the web interface to determine bottlenecks in the cardiac patient flows, as the care process is taking place. As a consequence, the provider can respond quickly to a critical situation (e.g., long wait times in ED for admission to Cardiology Ward) in proactive and reactive manner.

5.2.5 Methodology

In this section we highlight some of the major steps to develop the CPMA for Osler Hospital in terms of requirement analysis, test-driven development implementation and deployment.

Requirements Analysis

The interaction diagram (Figure 5-7 in section 5.2.3) was used to determine what type of events would be received by the CPMA, from which it would have to identify service and wait

states. In particular, an in depth analysis was done to determine what rule processing the CEP would have to do, and what event attributes would be needed. Table 5-1 below defines the events and processing for the CPMA. The event table lists the events identified using the interaction diagram. It shows the name of the source event, event type, event attributes and, more importantly the rules that infers the complex events. These rules are implemented in the CEP (described under implementation below).

The total effort required for requirements analysis was quite high, especially when the time taken to develop the three models is included. In fact, as we built the above table, each of the three models constantly needed to be updated and revised as we encountered problems in the fine details. It took about four months of painstaking work from the initial gathering or requirements until our event table and three models (service and wait states, care process model, and architecture component interactions) were finalized.

Table 5-1: Events Definition and Rules

Event	Integration	Source Event	Event Type	Event Source	Attributes
		PatientInED	Location	RTLS	PatientID, LocationID, Timestamp
		TriageScore	Care Process Event	BPM	PatientID, ProviderID, CTAS, timestamp
WaitForConsultation	PatientInED Follows TriageScore		Inferred	CEP	PatientID, timestamp
		PhyscianInED	Location	RTLS	PhysicianID, LocationID, timestamp
ConsultationStarted	PatientInED AND PhyscianInED Follows WaitForConsultation		Inferred Event	CEP	PatientID, PhysicianID, Location_ID timestamp
ConsultationCompleted	**Not** PhyscianInED AND Follows ConsultationStarted		Inferred Event	CEP	Patient_ID, PhysicianID, LocationID, timestamp

		OrderRequest	Care Process Event	BPM	PatientID, ProviderID, OrderType, Order_ID, timestamp
WaitForOrderExecution	Occurrences of OrderRequest Is 1 and Follows ConsultationStarted		Inferred Event	CEP	PatientID, timestamp
		OrderRequestCompleted	Care Process Event	BPM	PatientID, ProviderID, OrderType, OrderID, timestamp
OrderExecutionCompleted	Occurrences of OrderRequestCompleted Is 3 **AND** Follows WaitForOrderExecution		Inferred Event	CEP	PatientID, ProviderID, timestamp
		BedRequest	Care Process Event	BPM	PatientID, ProviderID, UnitID, timestamp
		PatientAdmittedWithNoBed	Care Process Event	BPM	PatientID, timestamp
WaitForBed	Occurrences of BedRequest IS 1 **AND** Follows PatientAdmittedWithNoBed		Inferred Event	CEP	PatientID, UnitID, timestamp
		PatientAdmittedWithBed	Care Process Event	BPM	PatientID, BedID, timestamp
		PatientTransportRequest	Care Process Event	BPM	PatientID, TransportID, UnitID, timestamp
WaitForTransport	PatientTransportRequest Follows WaitForBed **AND** Follows PatientAdmittedWithBed		Inferred Event	CEP	PatientID, UnitID, timestamp
PatientTransportStarted	**NOT** PatientInED **AND** Follows WaitForTransport		Inferred Event	CEP	PatientID, TransportID, UnitID, timestamp
		PatientInCCU	Location	RTLS	PatientID. LocationID, Timestamp
PatientArrivedInBed	PatientInCCU **AND** Follows PatientTransportStarted		Inferred Event	CEP	Patient_ID, Bed_ID, Unit_ID, timestamp

Test Driven Development with Scenarios

Once the table of events was finalized, we were able to use it to define a series of test scripts that could be run by the test console that would simulate the candidate scenarios provided by the care process expert and enable us to test each of the components in the architecture (RTLS, BPM, CEP and CPME). Below is a sample of a test script that simulated source events from BPM and RTLS to test that the CEP was behaving correctly, i.e., inferring the appropriate complex events. The test console and test scripts were effective and invaluable mechanisms for testing the CPMA (and other components). Based on the event table, scripts were created and used to drive the development of each of the components. During case study 1, they were very effective to highlight gaps in the requirements analysis at an early stage of implementation which caused us to go back and update all three models and the test table. The creation and running of test scripts could be done in a matter of days. But the overall testing effort was quite high due to errors and inconsistencies in the requirements, so weeks of testing effort was required before requirements could be finalized.

```
<?xml version="1.0" encoding="utf-8"?>
<oslerTestScript>
 <testName>Test CEP</testName>
 <testDescription>Contains all the events for CEP (InputCEPTestXML). Ordered by timestamp. Sources simulate coming from either RTLS or BPM.
 </testDescription>
 <!-- Input events -->
 <event name="TriageScore" sourceSuffix="BPM">
  <Patient_ID>Pa123456</Patient_ID>
  <Provider_ID>Nurse1111</Provider_ID>
  <CTAS>2</CTAS>
  <timestamp>2012-03-12 01:00:46</timestamp>
 </event>
 <event name="PatientInED" sourceSuffix="RTLS">
  <Patient_ID>Pa123456</Patient_ID>
  <Location_ID>AssessmentRoom12</Location_ID>
  <timestamp>2012-03-12 01:12:46</timestamp>
 </event>
 <event name="PhysicianInED" sourceSuffix="RTLS">
  <Physician_ID>Phy777777</Physician_ID>
  <Location_ID>AssessmentRoom12</Location_ID>
  <timestamp>2012-03-12 01:16:28</timestamp>
 </event>
 <event name="OrderRequest" sourceSuffix="BPM">
  <Patient_ID>Pa123456</Patient_ID>
  <Provider_ID>Phy777777</Provider_ID>
  <Order_ID>Order1111</Order_ID>
  <Order_Type>ECG</Order_Type>
  <timestamp>2012-03-12 01:22:33</timestamp>
```

```
 </event>
 <event name="BedRequest" sourceSuffix="BPM">
  <Patient_ID>Pa123456</Patient_ID>
  <Provider_ID>Pro654321</Provider_ID>
  <Unit_ID>CCU</Unit_ID>
  <timestamp>2012-03-12 02:08:16</timestamp>
 <event name="PatientTransportRequest" sourceSuffix="BPM">
  <Patient_ID>Pa123456</Patient_ID>
  <Provider_ID>Nurse745774</Provider_ID>
  <Unit_ID>CCU</Unit_ID>
  <timestamp>2012-03-12 02:26:13</timestamp>
 <event name="PatientInCCU" sourceSuffix="RTLS">
  <Patient_ID>Pa123456</Patient_ID>
  <Location_ID>Bed207</Location_ID>
  <timestamp>2012-03-12 02:35:47</timestamp>
 </event>
</oslerTestScript>
```

Implementation and Deployment

In case study 1, the entire CPMA was built as a custom application that was hard-coded specifically for monitoring the ACS clinical pathway at Osler Hospital. The key implementation steps were the following:

- Custom development of CPME in Grails with event handlers and object model,
- Implementation of CEP rules for inferring events,
- Configuration of GRAILS (Grails, 2009) ORM for database persistence and
- Development of custom interface in GRAILS with custom-code SQL queries and reports using Google charts.

The event handlers parsed the event messages from the various architecture components (RTLS, BPM, and CEP) and updated the object model to track the current state for each patient.

The CEP rules were defined and implemented using WebSphere Business Events (Mandi, 2008). Figure 5-9 shows the rule that identifies the start of the WaitForBed wait state. Note that the rule has to do special processing with artificial "Syn" events to manage rule side effects and other interactions with other CEP rules. The rule generated a WaitForBedSyn event that is used

by rules later in the process, and it depends on a "Syn" event related to the 2nd physician consultation which should have taken place before the patient is actually in a wait-state.

```
Interaction Set: Patient Wait For Bed  Related by Patient.Patient_ID

In response to: PatientAdmittedWithNoBed From WFM

        Immediately
                All Occurrences of BedRequest Is 1
                AND
                If All Occurrences of ConsultationStarted2Syn  Is 1

                Then: Immediately WaitForBed (WFM)
                      : Immediately WaitForBedSyn (CEP)
```

Figure 5-9: CEP Rule for Generating WaitForBed Event

The mapping for each object (event, states) maintained in the object model of CPME to ORM Persistence database is automatic via the Grails ORM module (Hibernate) and is not optimized for reporting.

Each report is custom coded and specific to the ACS care process using queries that are complex and which are not optimized to handle volumes. The Google charting tool is adequate, but does not support navigation or complex reporting.

The implementation effort was quite high and required much time to build CPMA since the development team followed the traditional software engineering methodology. A team of two graduate students, including the thesis researcher, spent about 8 months hard-coding the monitoring application using Grails software including the event handlers, writing ad hoc SQL queries in Grails for reports and developing CEP rules using the IBM Websphere Business Events commercial off-the-shelf CEP engine. Any error or misinterpretation in the requirements required a review of custom code, CEP rules and SQL queries in order to understand how to fix

the CPMA. This process is time consuming and effects on the total time required to deliver CPMA. As a consequence, the development process did not adapt quickly to changes in requirements.

5.2.6 Summary

In this section, we described the approach and efforts used in the first case study. The resulting CPMA and hospital care architecture for CPM was a huge improvement over the existing environment as was validated by the survey of 80 Osler staff. However, the engineering approach was error prone and required a great deal of effort to produce a hard-coded application specification to cardiac care that was brittle and difficult to maintain. In particular, none of the three models used in requirements analysis seemed to be effective in clearly defining the requirements. In the next section, we describe a follow-up case study that used our full application framework for care process monitoring based on a single application meta-model. It was successful in dramatically reducing the engineering efforts required and dramatically improving the quality of the CPMA built.

5.3. Cardiac Care Process Monitoring Case Study Using the Application Framework

In this case study, which we call case study 2, we used our complete application framework to re-implement the CPMA from case study 1. A complete application model (see Appendix) was built to create a concise, complete and consistent specification of the CPMA. The high-level CPM architecture pattern, which was used in case study 1, remained the same in terms of the enterprise architecture in which the CPMA was embedded, but the architecture of the CPMA itself was refined, and a different set of components was used to assemble the CPMA.

The methodology defined in section 4.3 was followed. A Master student, Shirley Baffoe, was the application developer who implemented the CPMA as part of her Master's thesis (Baffoe, 2013), using the application model that we defined and the test scripts that we created. Her primary focus was the development of a new State Monitoring Engine (SME) component to replace the CEP used in case study 1. Another Masters student, Amanpal Bhatia, helped with the implementation of the Data Mapping and Metrics Data Mart as part of his Masters Project (Bhatia, 2013).

This case study was defined by performing the following steps:

1. Used the same ACS Clinical Pathway from case study 1, and reused the same event-driven architecture (BPM, RTLS, Message Broker) to provide the events monitored by the CPMA. We interacted with the same domain experts for analysis.
2. Built a complete, concise and consistent application model to specify the CPMA that was validated by our domain experts.
3. Re-implemented the CPMA using our framework. A M.Sc. student, Shirley Baffoe, was the lead application developer. She used our application model and architectural pattern, and followed our development methodology. She implemented a new CPME, in which the CEP was replaced with an SME with an interface to a Metrics Data Mart (built by a Master Student, Amanpal Bhatia).
4. Tested the new version of the CPMA by re-running the test scripts developed in case study 1 and measuring performance to compare.

5. Walked through the demonstration environment from case study 1 with domain experts, as well as hospital staff and technical experts from two new hospitals (TOH, Queensway Carleton) to validate the experience with the new CPMA.

5.3.1 Existing Cardiac Care Environment

The existing environment was what had been achieved with the CPMA from case study 1. Although the CPMA from case study 1 was successful in monitoring the care process in near real-time, there were issues with the way the CPMA had been engineered and difficulties with maintaining it. Any minor modification in the events, states or even the specifications of the metrics require painstaking manual effort and custom coding to review all the rules and event handlers and hard-coded reports to examine if they were influenced by the modification. The system was not declaratively configurable, so to monitor any similar process would require creating and custom coding a brand new CPMA for that process. In addition, the reports are hard coded and custom queries against the ORM persistence database are handcrafted specifically to the cardiac care process to provide simple reports of wait and service times related to cardiac patients.

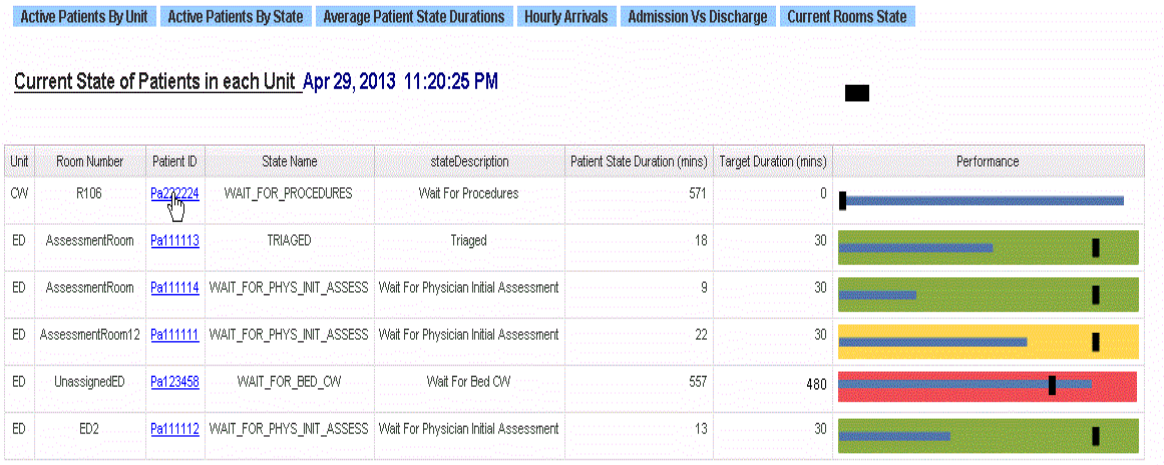
5.3.2 Care Process Monitoring Application (CPMA)

From an end-users point of view, the new CPMA that was built for case study 2 has almost the same functionality and capability as the previous CPMA. It provides fine-grained monitoring of cardiac patients, providers, beds and other resources, from the time the cardiac patient arrives at the Emergency Department to the time the patient is discharged.

The key improvement in CPMA for case study 2 is the reporting mechanism, which is more sophisticated, flexible and provides navigation, filtering, drill down and drill up capabilities. This is because the reports leverage a metrics data mart that is optimized for reporting wait and service times and we use a professional enterprise reporting dashboard that was built using IBM Cognos Report Studio.

More importantly, the reports, dashboard and star schema used in the new CPMA are generic and can be used to report on wait and service times for any care process, not just cardiac care. To do so, the only part to change in the metrics database is the lookup tables (dimensional tables) that provide the list of states, events and resources that should be monitored for a particular care process.

Figure 5-10 is an example of a dashboard that is auto-refreshed every 30 seconds in the new CPMA. It shows briefly the current states and the corresponding states duration along with the target duration of active patients in a particular unit and in a particular room within the unit. In addition, the chart in the dashboard shows the performance of each active patient's state based on a comparison of the measured state's duration and the target duration.



Unit	Room Number	Patient ID	State Name	stateDescription	Patient State Duration (mins)	Target Duration (mins)	Performance
CW	R106	Pa222224	WAIT_FOR_PROCEDURES	Wait For Procedures	571	0	
ED	AssessmentRoom	Pa111113	TRIAGED	Triaged	18	30	
ED	AssessmentRoom	Pa111114	WAIT_FOR_PHYS_INIT_ASSESS	Wait For Physician Initial Assessment	9	30	
ED	AssessmentRoom12	Pa111111	WAIT_FOR_PHYS_INIT_ASSESS	Wait For Physician Initial Assessment	22	30	
ED	UnassignedED	Pa123458	WAIT_FOR_BED_CW	Wait For Bed CW	557	480	
ED	ED2	Pa111112	WAIT_FOR_PHYS_INIT_ASSESS	Wait For Physician Initial Assessment	13	30	

Figure 5-10: Current Patients' States Dashboard

The CPMA provides care providers with drill down, drill up, and filter capabilities. When the care provider clicks on a particular patient (e.g. Pa222224), this will allow him to navigate to another report that shows in details all the patient states (current states and historical states) from the time he is being triaged up to the current state. This is shown in Figure 5-11where the care provider can filter the patient states by unit and measured duration alert.

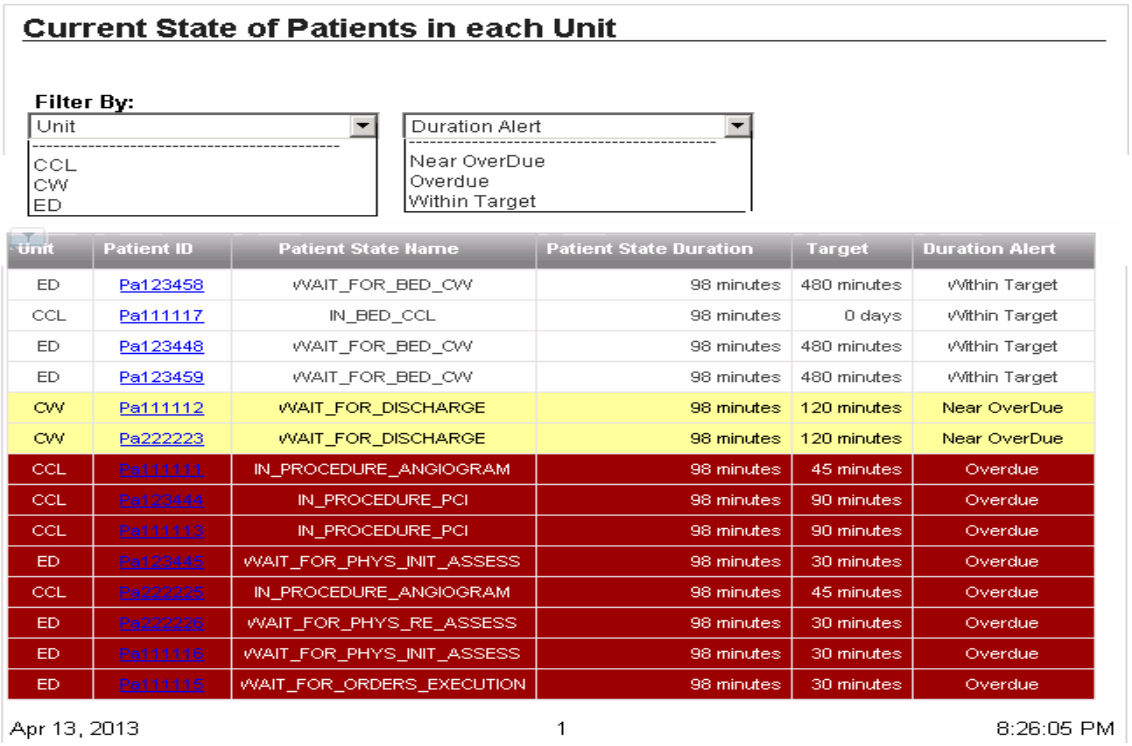
Patient State Details for : Pa222224

Unit | Duration Alert

Unit	Provider ID	Patient State	State Duration	Target Duration	Bullet Chart	Start Timestamp	End Timestamp
CW	Not Defined Num	Wait For Procedures	569 minutes	0 days		Apr 29, 2013 1:49:00 PM	
CW	Not Defined Num	In Bed CW	10 minutes	0 days		Apr 29, 2013 1:39:00 PM	Apr 29, 2013 1:49:00 PM
CW	Tra1234	In Transport CW	15 minutes	15 minutes		Apr 29, 2013 1:24:00 PM	Apr 29, 2013 1:39:00 PM
ED	Nurse121212	Wait For Transport CW	15 minutes	15 minutes		Apr 29, 2013 1:09:00 PM	Apr 29, 2013 1:24:00 PM
ED	Not Defined Num	Wait For Bed CW	46 minutes	480 minutes		Apr 29, 2013 12:23:00 PM	Apr 29, 2013 1:09:00 PM
ED	Phy777777	In Bed ED	21 minutes	0 days		Apr 29, 2013 12:21:00 PM	Apr 29, 2013 12:23:00 PM
ED	Phy777777	In Physician Re-Assessment	2 minutes	0 days		Apr 29, 2013 12:19:00 PM	Apr 29, 2013 12:21:00 PM
ED	Nurse121212	Wait For Physician Re-Assessment	10 minutes	30 minutes		Apr 29, 2013 12:09:00 PM	Apr 29, 2013 12:19:00 PM
ED	Phy777777	Wait For Orders Execution	80 minutes	30 minutes		Apr 29, 2013 10:49:00 AM	Apr 29, 2013 12:09:00 PM
ED	Phy777777	In Physician Initial Assessment	14 minutes	0 days		Apr 29, 2013 10:16:00 AM	Apr 29, 2013 10:30:00 AM
ED	Not Defined Num	Wait For Physician Initial Assessment	26 minutes	30 minutes		Apr 29, 2013 9:50:00 AM	Apr 29, 2013 10:16:00 AM
ED	Nurse745774	Triaged	16 minutes	30 minutes		Apr 29, 2013 9:34:00 AM	Apr 29, 2013 9:50:00 AM

Figure 5-11: Detailed Patient States Report

Another similar report that shows the current states of active patients in each unit of the hospital is shown in Figure 5-12. This type of report gives the care provider the ability to filter the report to show the current patients' states in a particular unit of the hospital and at the same time filter the report based on the measured duration alert.



Current State of Patients in each Unit

Filter By:

Unit: CCL, CW, ED

Duration Alert: Near OverDue, Overdue, Within Target

Unit	Patient ID	Patient State Name	Patient State Duration	Target	Duration Alert
ED	Pa123458	WAIT_FOR_BED_CW	98 minutes	480 minutes	Within Target
CCL	Pa111117	IN_BED_CCL	98 minutes	0 days	Within Target
ED	Pa123448	WAIT_FOR_BED_CW	98 minutes	480 minutes	Within Target
ED	Pa123459	WAIT_FOR_BED_CW	98 minutes	480 minutes	Within Target
CW	Pa111112	WAIT_FOR_DISCHARGE	98 minutes	120 minutes	Near OverDue
CW	Pa222223	WAIT_FOR_DISCHARGE	98 minutes	120 minutes	Near OverDue
CCL	Pa111111	IN_PROCEDURE_ANGIOGRAM	98 minutes	45 minutes	Overdue
CCL	Pa123444	IN_PROCEDURE_PCI	98 minutes	90 minutes	Overdue
CCL	Pa111113	IN_PROCEDURE_PCI	98 minutes	90 minutes	Overdue
ED	Pa123445	WAIT_FOR_PHYS_INIT_ASSESS	98 minutes	30 minutes	Overdue
CCL	Pa222225	IN_PROCEDURE_ANGIOGRAM	98 minutes	45 minutes	Overdue
ED	Pa222226	WAIT_FOR_PHYS_RE_ASSESS	98 minutes	30 minutes	Overdue
ED	Pa111116	WAIT_FOR_PHYS_INIT_ASSESS	98 minutes	30 minutes	Overdue
ED	Pa111115	WAIT_FOR_ORDERS_EXECUTION	98 minutes	30 minutes	Overdue

Apr 13, 2013 1 8:26:05 PM

Figure 5-12: Sample Report Filtered by Unit and Duration Alert

5.3.3 Model

The main focus in case study 2 was to validate the improvements to the engineering of the CPMA that would be realized by using our application framework and in particular our application meta-model. The first step in developing the new CPMA was to create an application model using our meta-model from chapter 4.1. The complete application model that was built can be found in the Appendix A. The first step in building the model was to identify the key states and steps and identify the events that would be used to mark the transition from state to

state. This is shown in Figure 5-13 below. At each point we can see what event or combination of events was used to identify the start of a state. Also, for each event, we identified whether it originated from the BPM or the RTLS. For example, PhysicianInED from the RTLS is enough to known that Physician assessement or re-assessment has started, but all three of PhysicianOutED from RTLS, OrderBed from BPM and BedNotAvailable from BPM are needed to know when Wait_For_BED_CW has started.

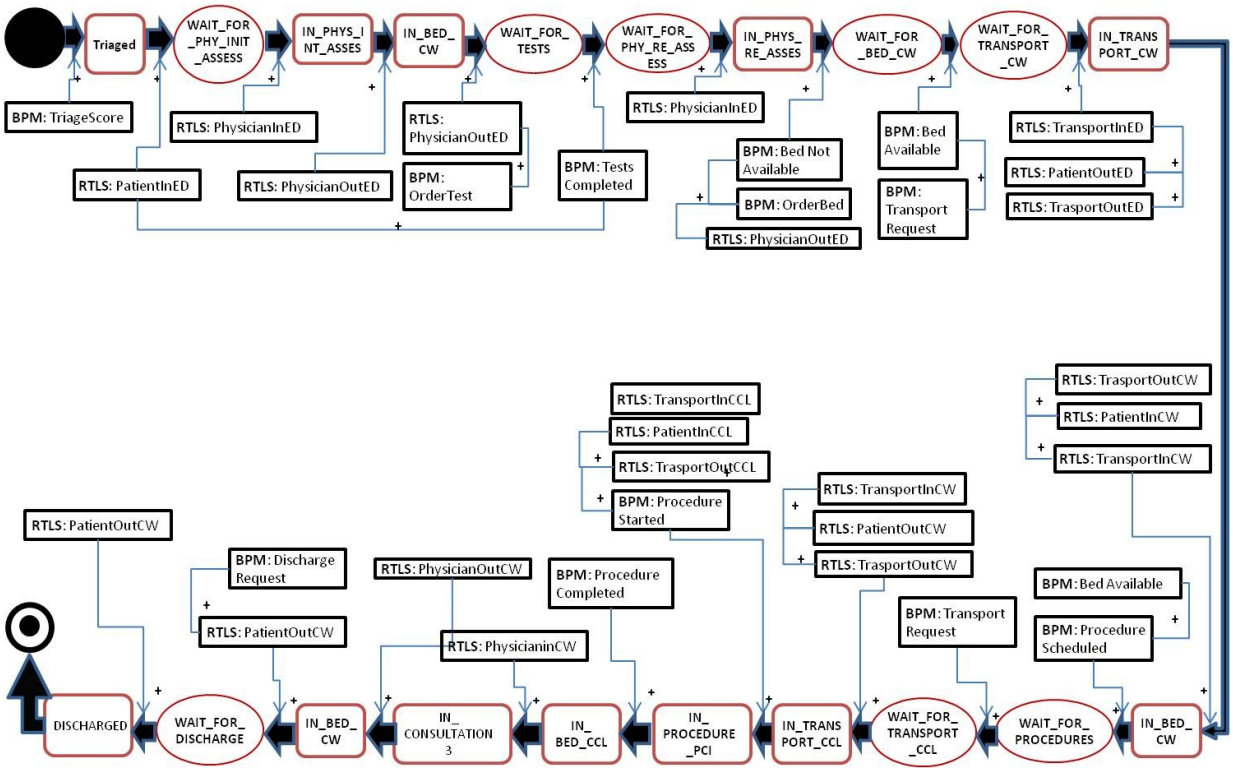


Figure 5-13: Care Process States with Events

The second step in the modeling process is to determine what metrics are needed to measure what goals and map those metrics to the states or events that will be used to compute them. According to Figure 5-14, one of the goals the hospital aims to monitor is the wait time. The objective is to keep the wait times within acceptable range and in case the ultimate value of

specific wait time exceeds the acceptable range, an associated alter is raised. For example, the metric *Average Wait_Time for Bed* is computed by measuring the duration of the state WAIT_FOR_BED_CW. Another goal the hospital wishes to monitor is the throughput of patients in terms of patient arrivals, admissions and discharges. To measure this goal, the following metrics have to be measured, *Average_Arrivals by Hour of the Day*, *Average_Admissions by Hour of the Day* and *Average_Disacharges by Hour of the Day*. Those metrics are basically computed by counting single event for each metric as shown in Figure 5-14 below.

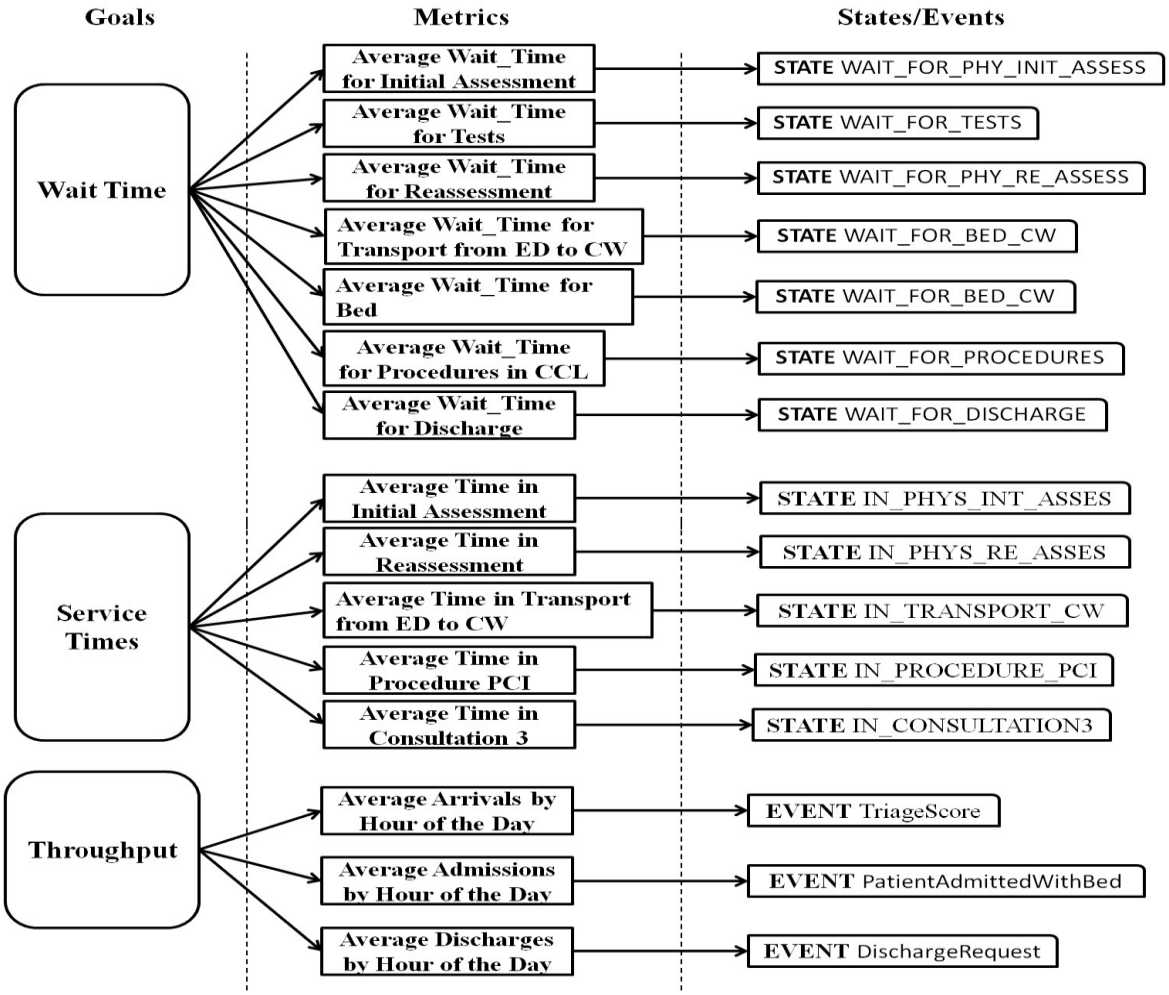


Figure 5-14: Linking Goals to Metrics to States or Events

The final step in building the application model is to elaborate precisely what attributes are defined for each state and event in order to provide the information needed for the metrics computation, and to clearly define with a rule each state transition. For example in Metric Average Wait-Time for Bed, we need to compute the average duration (endTime-startTime) for each Wait_FOR_BED_CW state. The definitions of the above metric and the state that computed the metric Average Wait-Time for Bed is shown below.

Metric: Average Wait_Time for Bed
Description: "This metric measure the average time patients wait for bed in Cardiology Ward over period of time"
Computation: AVERAGE STATE: WAIT_FOR_BED_CW. duration Over Period of Time
State: WAIT_FOR_BED_CW
Events:
Target: 480 minutes
Alert: OVERDUE, NEAR_OVERDUE

State: WAIT_FOR_BED_CW
Description: "This state indicates that the patient is waiting for bed in Cardiology Ward"
StateAttributes: patientID, unitID, startTime, endTime

An example of a state inference rule is the one below that infers that patient is in the WAIT_FOR_BED_CW (waiting for bed in cardiac ward) state. This rule defines the state transition from the current state IN_PHY_RE-ASS based on integrating the source events OrderBed, BedNotAvailable and PhysicianOutED..

-Rule: Patient Wait for Bed in CW
Description: "This rule describes how to infer the state WAIT_FOR_BED_CW"
Condition: IF EVENT: OrderBed AND EVENT: BedNotAvailable AND EVENT PhysicianOutED
CurrentState: IN_PHYS_RE_ASSES
NextState: WAIT_FOR_BED_CW

5.3.4 Architecture

The architecture for case study 2 is similar to the architecture for case study 1 to some extent. The surrounding components around CPMA are the same as in case study 1. But, in terms of the internal components of CPMA, it is quite different. There is an SME with state handlers

and a Metrics Data Mart with a Performance Reporting Dashboard (using IBM Cognos Business Intelligence Suite 10) in the new CPMA that replaces the CEP with event handlers, ORM Database and Web Interface in the old CPMA. Figure 5-15 shows the architecture of the new CPMA. In terms of the general architecture pattern articulated in chapter 4.2, the key difference is the Metrics Data Mart. This conforms to the general pattern, whereas the ORM Database did not, and provides the extra benefits that users experience with the second CPMA. Both the CEP from the first CPMA and the SME from the second CPMA are alternative implementations of the CPME component in the general architecture pattern. They provide the same functionality in terms of enabling the end-user experience, but there are advantages to the SME in terms of how the two CPMAs are engineered differently.

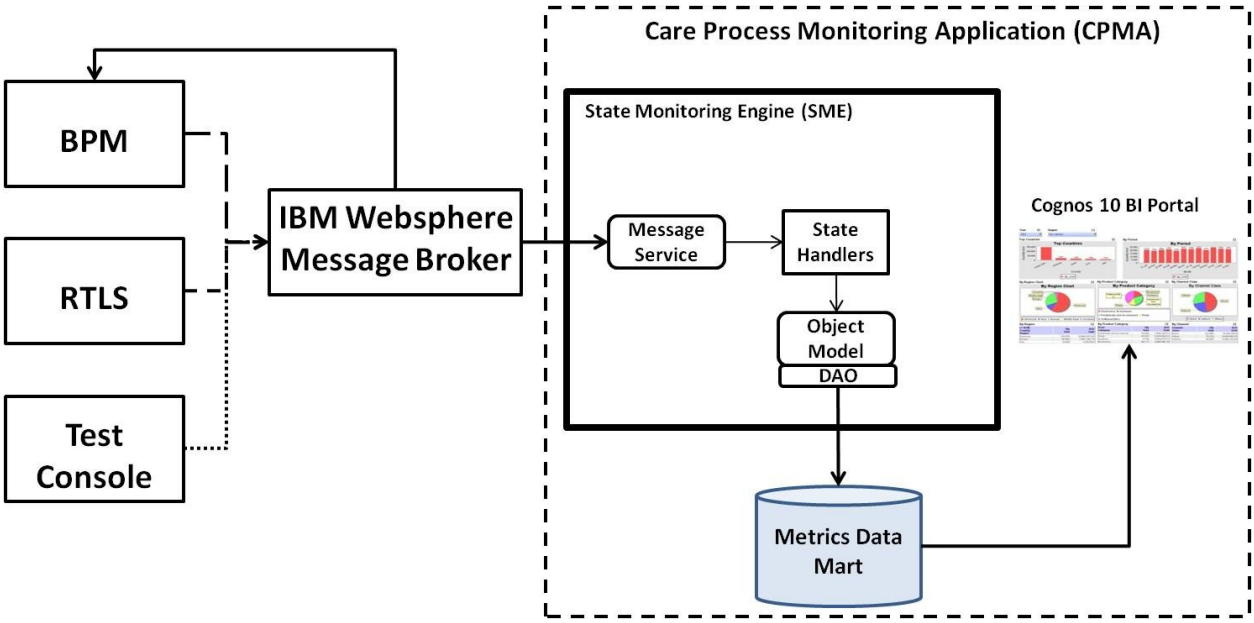


Figure 5-15: Architecture for Second Case Study

The core functions of the SME are the following:

1. Process source events generated from BPM, RTLS via Message Broker.
2. Infer resources' states based on state inference rule that consists of the current state of the resource plus combination of source events received related to that resource.
3. Update the resources' current state and events related.
4. Log the current events and states of the resources as facts in the Metrics Data Mart.

The Message Service receives events from Message Broker and routes those events to the appropriate state handler after retrieving the current state of that resource.

Each state handler subscribes to number of events and as each one receives its subscribed events, the object model is updated and event facts are logged to the Metrics Data Mart. The state handler processes subscribed events using state inferencing rules that determine what event or combination of events identifies the start of a new state (and the ending of the current state). On each state transition a state fact is logged to the Metrics Data Mart.

The Data Access Object (DAO) logs state and event attributes to the state and event fact tables in the Metrics Data Mart database as events are processed and states inferred by the state handlers. The Metrics Data Mart organizes the fact tables into a standard star schema.

The Performance Reporting Dashboard component of the CPM architectural pattern (shown in Figure 4-7 in section 4.2.1) was implemented as an IBM Cognos 10 BI portal. It is driven from the Metrics Data Mart. It provides navigation, filtering, drill down and drill up capabilities.

5.3.5 Methodology

While there were refinements of the architecture between the first CPMA and the second CPMA, which more closely aligned it with the architectural pattern articulated in chapter 4.2, there were significant differences in how the CPMA was engineered. In case study 2, we leveraged the full application framework described in chapter 4 and explicitly followed the methodology described in 4.3.

Requirements Analysis

As described in chapter 4.3.3, requirement analysis consisted of three major activities conducted by three different roles.

The fundamental activity is **Build Application Model.** This was done by the thesis researcher. A complete application model for case study 2 is shown in Appendix A. It was assembled from the EDA-SOA architecture and associated events that did not change from case study 1, as well as the care process and performance management models and candidate scenario that also did not change from case study 1. Figure 5-13 and Figure 5-14 in the previous section 5.3.3, show precisely what events will be monitored at what states in the care process in order to compute the metrics needed for performance reporting.

Building the application model for case study 2 depended on the artifacts from the activities **Care Process Analysis** and **Design and Architecture**. The Care Process Analysis activity was performed by Alain Mouttham, who led the overall research project on patient flow monitoring that our research is a part of and who worked as a researcher as part of the clinical team at Osler Hospital. The Design and Architecture activity was the responsibility of the thesis

researcher's supervisor, Prof. Peyton, who interfaced with Alain Mouttham, University of Ottawa support staff, and technology experts at IBM.

The effort to document the requirements as a reference to the developers to guide them implementing the new CPMA was much less complex and took much less time in the second case study than in the initial case study. It took two to three weeks to document and verify the requirements instead of four months. This is because all the requirements elements (goals, metrics, alerts, states, events, rules and sources) are precisely defined in a concise format represented in a single application model that requires much less effort to fix when the requirements are changed during the development process. From the experience we had in the initial case study, it was clear that the developers were confusing about the precise details of event order, state attributes, and metrics and those elements were misinterpreted by them and the developers had always to refer to the domain expert. Rather, in the second case study, the developers had to refer to a single application model that showed how it corresponded to configuration points in the CPMA.

Test Driven Development with Scenarios

The desired behavior of the new CPMA is defined by the same test scripts that were developed in the first case study from the candidate scenarios provided by the domain expert. Those test scripts were written by the thesis researcher and reviewed by the care process experts to ensure that they adhered to the requirements. They were also used by the thesis researcher to validate the application model was built and ensure the necessary precision in event and state attributes as well as rule and metric definitions. Figure 5-16 shows a snippet of a test script. The

test scripts are used to refine the following aspects in the application model in order to verify the CPMA that was implemented:

1. Define all the events that should be logged and captured by the CPMA. Figure 5-16 shows the expected event names such as TriageScore and PatientInED events.
2. Define the source for each event. For example, in Figure 5-16, the source of TriageScore event is BPM and the source for event PatientInED is RTLS.
3. Define the order of the events based on a certain scenario. Figure 5-16 shows the sequence of events that should be tested.
4. Define the expected complex events when a particular event is generated.

Once the CPMA was assembled and configured, the same Test Console component used in case study 1, was used to test the CPMA as a standalone application before it was deployed and integrated into the hospital enterprise architecture (our demonstration environment) for acceptance testing. The console can simulate events of any type from any source (i.e. RTLS, BPM or other systems). Those simulated events are generated by providing a test script, which is derived from scenario, to the console. The test script is an XML-based formatting which shows the sequence of events along with the event sources and source events. Figure 5-16 is a sample of the test script.

```
<?xml version="1.0" encoding="utf-8"?>

<oslerTestScript>
  <testName>Aladdin's Test Script For The Extension Scenario</testName>
  <testDescription>Contains all the events for CEP (InputCEPTestXML). Ordered by timestamp. Sources

  <!-- Input events -->
  <event name="TriageScore" sourceSuffix="BPM">
    <Patient_ID>Pa123456</Patient_ID>
    <Provider_ID>Nurse1111</Provider_ID>
    <CTAS>2</CTAS>
    <timestamp>2012-03-12 01:00:46</timestamp>
  </event>
  <event name="PatientInED" sourceSuffix="RTLS">
    <Patient_ID>Pa123456</Patient_ID>
    <Location_ID>AssessmentRoom12</Location_ID>
    <timestamp>2012-03-12 01:12:46</timestamp>
  </event>
  <event name="PhysicianInED" sourceSuffix="RTLS">
    <Physician_ID>Phy777777</Physician_ID>
    <Location_ID>AssessmentRoom12</Location_ID>
    <timestamp>2012-03-12 01:16:28</timestamp>
  </event>
  <event name="PhysicianOutED" sourceSuffix="RTLS">
    <Physician_ID>Phy777777</Physician_ID>
    <Location_ID>AssessmentRoom12</Location_ID>
    <timestamp>2012-03-12 01:19:51</timestamp>
  </event>
```

Figure 5-16: Sample of the Test Script

The effort and complexity to creating test scripts and running them via the Test Console is, of course, identical to Case Study1, since we simply reused what was created for Case Study 1. However, the amount of time taken to achieve a CPMA that passed all tests was much less, and the amount of time needed to debug failed tests and maintain tests was much less than Case Study 1. This was due to the fact that the requirements were finalized and completely captured in the application model. There was less confusion and less concern that the test script might be inaccurate as it was very easy to compare it to the application model and very easy to review the application model to ensure requirements were clear and accurate.

Implementation and Deployment

The first step in implementation is to perform the **Assemble CPMA** activity (as defined in chapter 4.3) in which the CPMA components and the software used to implement those components are determined. The SME, state handler, Data Access Object (DAO) and Cognos 10 BI Portal were chosen to be used as the CPMA components CPME, Database Mapping, Performance Reporting Dashboard respectively. The SME was built over a period of 3 months

by Shirley Baffoe as part of her master's thesis work. It was tested as part of the testing of the overall CPMA using the test scripts. The Metrics Data Mart and Performance Reporting Dashboard are generic for any care process as they are based on a generic event fact table and a generic state fact table.

The second step is to **Configure CPMA**. This includes defining six artifacts, which are events, rules, states, resources, metrics and alerts. Events, states, resources and their associated attributes are defined and enumerated in the event, state and resources dimensions (lookup tables) respectively in the Metrics Data Mart. All the events and states that should be monitored by CPMA are defined in the application model shown in Appendix A.

Rules are configured in the state handlers to define the state transition of a particular resource. Each rule is configured by defining the current state of a particular resource and the related source events received by SME that should be integrated to infer the next state. Table 5-2 shows a subset of the rules to infer a patient's states in the ACS Clinical pathway. This subset was derived from a Master thesis written by the Master student, Shirley Baffoe (Baffoe, 2013), who used our application model in Appendix A as a guide to write the rules. In particular, she followed the rules described in the application model and illustrated them in a table. To see the full table, refer to her Master thesis (Table 4, page 60) (Baffoe, 2013)..

Table 5-2: Rule and Patient State Mapping

Rules which Transition Patient State			**Inferred Patient State**
Current Patient State	***Source Event(s)***	***Event Source***	
Start	TriageScore	BPM	Triaged
Triaged	PatientIn	RTLS	Wait for Phys. Initial Assess.
Wait for Phys. Initial Assess.	PhysicianIn	RTLS	In Phys. Initial Assessment
In Phys. Initial Assessment	PhysicianOut	RTLS	In Bed ED
In Bed ED	OrderTests	BPM	Wait for Tests

Reports are implemented to display metrics using the IBM Cognos 10 BI Portal. The metrics specifications are depicted in the application model. Since the IBM Cognos 10 BI Portal provides navigation, drill down and drill up functionalities and the custom queries supported by Cognos are running against Metrics Data Mart that was created based on star schema, therefore the engineering effort for building Performance Reporting Dashboard component requires less time and less effort than building reports from scratch using custom code and custom SQL queries.

For case study 2, there is an improvement, compared to the initial case study, in terms of the time needed to implement CPMA. In effect, it is hard to measure how much improvement can be attributed to the learning effect (Giovagnoli & Romano, 2004) when the same developers repeat the same task. However, based on the results of case study 2, the significant improvement in implementing CPMA is attributed to our application framework that has a generic Metrics Data Mart and a generic state handler. Therefore, custom code is not needed to implement CPMA as in the initial case study. Once the components of the hospital care CPMA is built and assembled (SME, DAO, Metrics Data Mart) it only takes two to three weeks for two developers

to implement, test and deploy the CPMA by implementing the state handler rules, computing the metrics and writing the reports for the Performance Reporting dashboard.

In case study 2, the amount of code needed to implement the CPMA is approximately 190 lines of code (i.e., about 10 lines of code for each state handler) and the number of rules needed is 19, one rule per state. Whereas in case study 1, the amount of code needed to implement CPMA is about 2000 lines of custom code and the total number of rules needed is approximately 54 rules; 27 event handlers' rules and 27 CEP pre-processing rules, to monitor the same ACS clinical pathway as in the second case study.

If a different CPMA was required to monitor service and wait times for a different care process, it would be a matter of days to implement and test the state handler rules since the application framework is reusable The metrics and reports would require no work, other than to update the lookup table that lists event names and state names in the Metrics Data Mart. The main effort would be in analyzing the care process and designing the EDA-SOA to build the application model that defined what events were available to monitor what states to compute what metrics.

5.3.6 Summary

In this section, we have leveraged our application framework to re-implement CPMA. The architecture of CPMA in the second case study conformed exactly to the CPMA architecture pattern articulated in section 4.2.1. However, there was huge improvement in engineering CPMA where a single application model which adhering to the application meta-model was leveraged to define and configure CPMA. Another improvement was defining a specific methodology that

should be followed to develop CPMA. The effort required to build CPMA is much less than building CPMA in the initial case study. It took several days to build the initial application model, and test scripts. Once the components of CPMA were built and assembled, it only took three weeks to implement, test and deploy into the production, which including keeping the application model up to date as requirements were analyzed and refined until the application was completed.

In the next section, we describe another case study related to community care process monitoring which used our full application framework. The goal is to validate our application framework and determine whether applying our framework can provide similar results for community care process monitoring as it did in case study 2 for hospital care process monitoring.

5.4. Community Care Process Monitoring Case Study

In this case study, we applied our complete application framework to build a CPMA for community care to explore how it differed from hospital care. A complete application model (see Appendix B) was built to create a complete specification and configuration of a CPMA for palliative care. A Master student, Austin Chamney, was the application developer who implemented the CPMA as part of his Master's thesis using the application model that we defined in section 5.4.3.

This case study was conducted by performing the following:

1. Collaborated with the Palliative Pain and Symptom Management Consultation Services (PPSMCS) team at Bruyère Hospital, through our care process expert Dr.

Craig Kuziemsky, in order to understand the palliative care process and performance management requirements.

2. Collaborated with the IT staff at Bruyère Hospital and Community Care and Access Center (CCAC) Ontario through our enterprise architect Prof. Liam Peyton in order to understand the enterprise architecture context for community-based palliative care.
3. Built a complete, concise and consistent application model to specify the CPMA that was validated by our domain expert.
4. Implemented a CPMA for palliative care using our application framework. A M.Sc. student, Austin Chamney, used our application model and architectural pattern, and followed our development methodology. He implemented a new CPME, in which support for simple, online AJAX forms to log care process events was integrated with an interface to a Metrics Data Mart.
5. Tested the new CPMA with test scripts and scenario-based walk-through with the key clinical and management staff at Bruyère Hospital.
6. Deployed the CPMA to a beta environment in which members of the PPSMCS team could compare the new CPMA with an existing CPMA for palliative care built as a traditional web application.

5.4.1 Existing Palliative Care Environment

Initially, the environment to monitor the palliative care process and to provide performance management report at Bruyère Hospital was entirely based on paper forms. When the case managers need reports for accreditation and program accountability, they have to collect data manually from paper forms related to all palliative patients in a certain period of time and

the data is analyzed and reported on in an Excel spreadsheet. To address this situation, a traditional web application called Palliative Care Information Systems (PAL-IS) version 1.0 was built as an online registry of palliative care patients and online-forms to record their information and care.

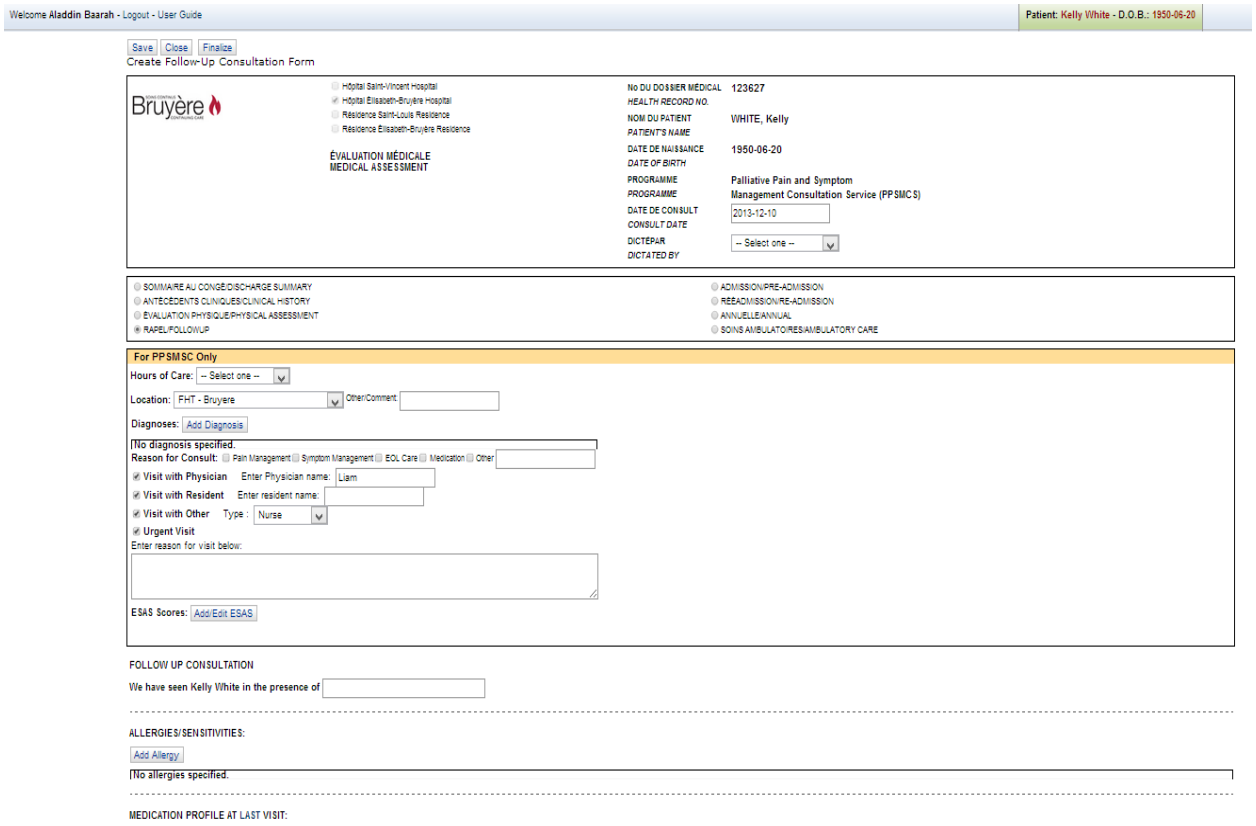


Figure 5-17: PAL-IS 1.0 FollowUp Consultation Form

PAL-IS can be considered a CPMA for palliative care monitoring, although most of the reports supported by PAL-IS 1.0 were not used, and many key reports were not supported. The application was found to be impractical for use in the field because of the complexity of the forms and huge data entry burden which required filling in over a hundred fields of some forms for a complete detailed record of care provided, even though most fields were not used in reports.

For example, Figure 5-17 shows only a part of a complex online form with over 100 fields required (including medications, allergies etc.).

In terms of performance management reports, there were several issues:

1. PAL-IS 1.0 provides many reports that were not being used.
2. Many of the forms' fields were not being used for reporting.
3. There was a poor understanding of relationship between forms and reports. It was hard to determine which data should be collected from which forms, to compute what metrics displayed in what reports.

5.4.2 Care Process Monitoring Application (CPMA)

The focus of our CPMA for palliative care (called PAL-IS 2.0) was to provide simple online forms and reporting (accessible by phones, tablets and laptops) that collected only the minimal set of fields directly relevant to reporting. Figure 5-18 shows an example of a PALIS 2.0 CPMA Referral form with only 20 fields. This form had information about the source of the referral, when the referral was taken place, when the care delivery will be occurred, the location of the patient and what was the primary diagnosis of the referred palliative patient (i.e., cancer or not cancer).

Figure 5-18: Referral Form

Performance reports in terms of metrics provided by the new PAL-IS 2.0 CPMA were optimized and the queries to provide reports were less complex compared to the queries developed in the PAL-IS 1.0. Figure 5-19 is an example of a monthly report that showed information about patients' consultations, as well as information on the type of diagnosis for the referred patients. In the Consultation with Providers report, the Average Waiting Time which corresponded to the metric *Average Wait_Triage Time* was computed from the form referral showed above, particularly, from the time difference between the delivery time and the request time. The same applies for the Diagnosis Report where the corresponding metrics *# of Patients Cancer* and *# of Patient Non Cancer* were computed from the referral form, in particular from the attribute Primary Diagnosis.

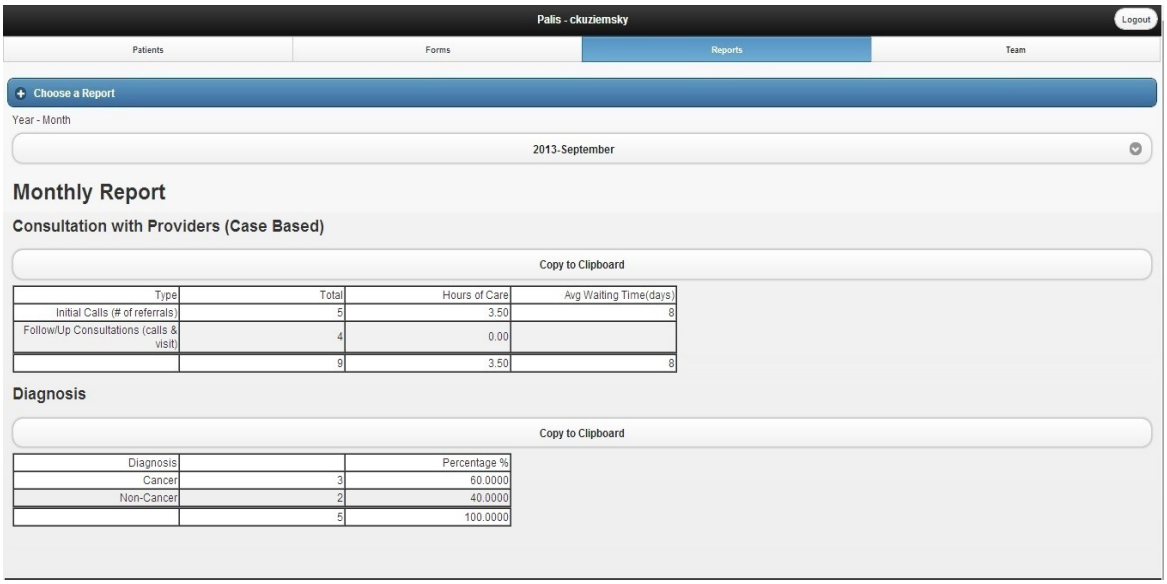


Figure 5-19: Monthly Report Showing Metrics Computation

5.4.3 Model

The main focus in building the new PAL-IS 2.0 CPMA was to define the complete application model that configures CPMA and which conforms to the application meta-model described at section 4.1.4. The complete application model that was built for PALI-IS 2.0 CPMA to monitor the palliative process can be found in Appendix B.

The first step to build the application model was to identify the key states in the palliative care process that should be monitored and identify the forms that would be used to transit from one state to other state. The palliative care process states, which are inferred by the thesis researcher as a result of conducting several meeting with the key stakeholders, were validated by the domain expert, Dr. Craig Kuziemsky. Figure 5-20 shows the state transition diagram that identifies the key patient states for the palliative care process and identifies when forms are used to collect data.

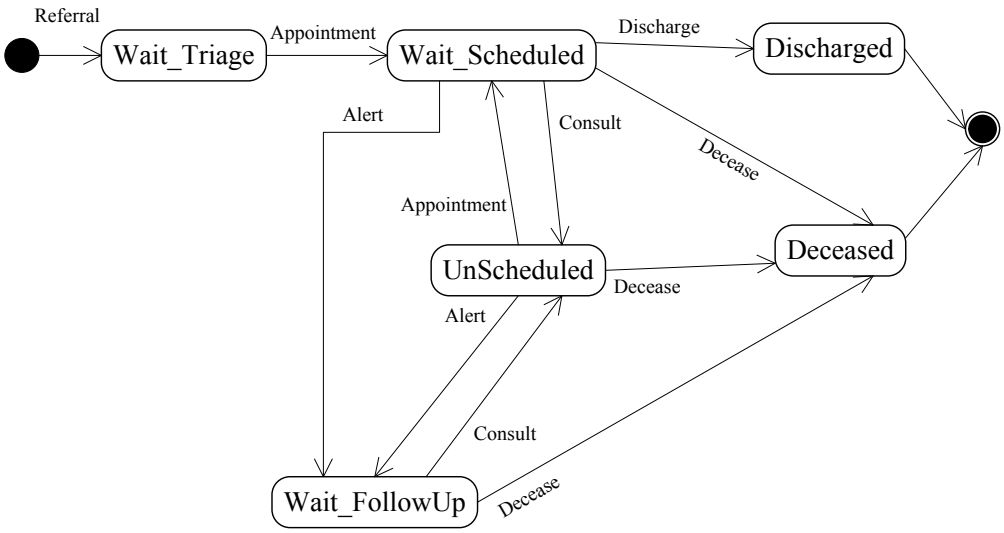


Figure 5-20: Palliative Care Process State

The process begins with a "Referral" form from a physician or facility. Then, an "Appointment" form schedules a consult. If the patient's condition is stable, then a "Consult" form is followed by an "Appointment" form for the next regularly scheduled consult. This is repeated until there is either a "Decease" form or a "Discharge" form (if no longer considered terminally ill). But if something goes wrong, an "Alert" form captures the issue and the patient waits for a follow up consultation (recorded by a "Consult" form). After any consult the patient is in an Unscheduled state until the "Appointment" form is filled in and then they are in a Wait_Scheduled state.

The second step to build the application model is to identify what metrics are needed to measure what goals by linking those metrics with states or forms. Figure 5-21 depicts this situation.

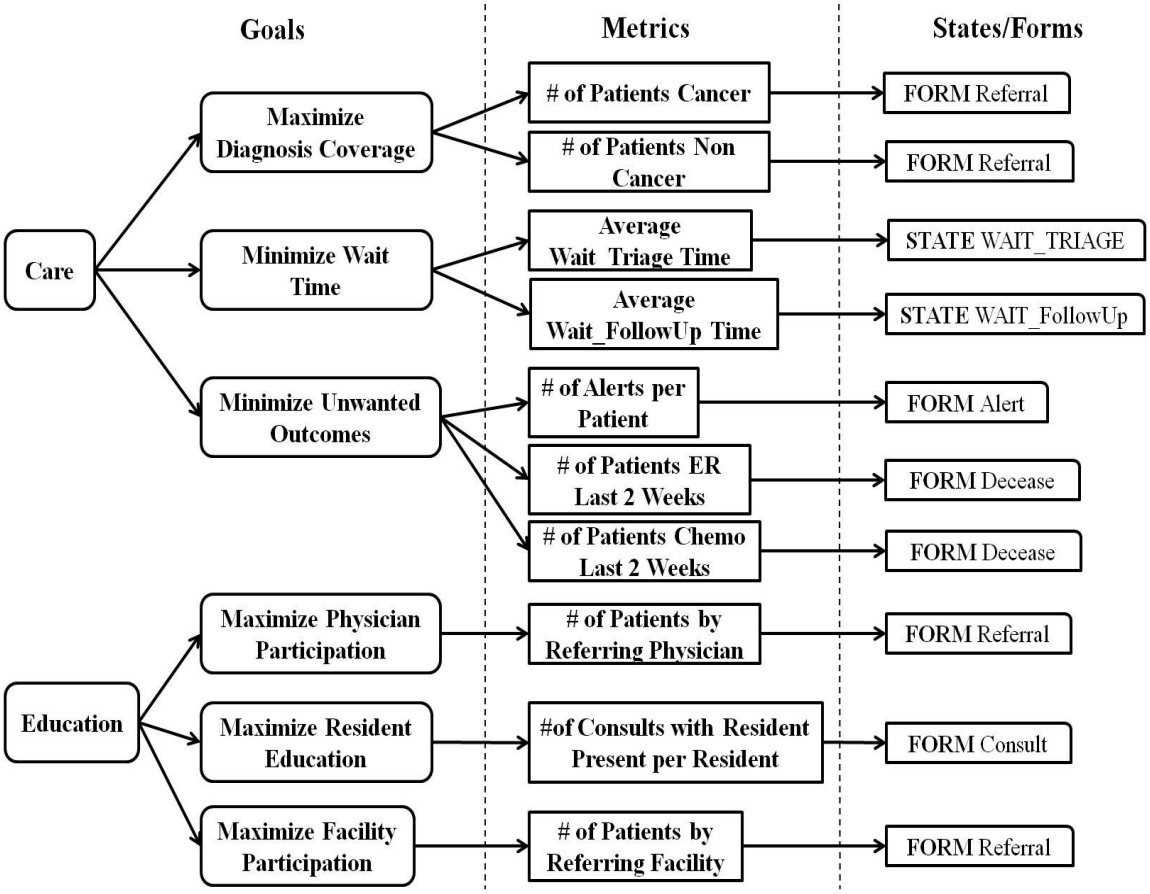


Figure 5-21: Linking Goals to Metrics to State or Forms

Each metric can be calculated either by counting particular forms or by computing the duration of particular states. According to Figure 5-21, the first set of goals ("Care") is related to understanding the quality and coverage of care provided. To ensure coverage of patients' population, a "Referral" form captures the data for "# of Patients Cancer" and "Non Cancer" (with drill down into diagnosis, gender, and age). To ensure "Wait Time" is minimized, the "Average Wait_Triage" time (from "Referral" to "Appointment") is measured as well as "Average Wait_Followup Time" taken to respond to alerts (from "Alert" to "Consult"). Finally, "Minimize Unwanted Outcomes" is measured to ensure "# of Alerts" is minimized (patients

should be stabilized) and to track the number of unnecessary interventions during a patient's last days ("Decease" form).

The second set of goals is used to measure how effective the program is in promoting palliative care to facilities and physicians so the number of referrals by physician and facility is captured. As well, the "# of Consults With Resident Present" is tracked ("Consult" form).

The final step in building the application model is to determine accurately what attributes are defined for each state or form in order to provide information needed for metrics computation. In terms of state transition, the rules to trigger states are straightforward with a one to one correspondence between events (Forms) and state transitions. The following example, drawn from the application model shows the definition and the computation of the Metric *Average Wait_FollowUp Time*.

- Metric: Average Wait_FollowUp Time
Description: "This metric measure the average time the palliative patients wait when abnormal condition occurred until a consult occurs"
Computation: AVERAGE STATE: WAIT_FOLLOWUP. duration Over Period of Time
State: WAIT_FOLLOWUP
Events:
Target: <=4hours
Alert: WAIT_FOLLOWUP_WARNING, WAIT_FOLLOWUP_UNACCEPTABLE

-State: WAIT_FOLLOWUP
Description: "This state shows that the palliative patient is waiting for unscheduled follow-up consultation when an abnormal condition occurred"
StateAttributes: patientID, startTime, endTime

In order to compute the metric defined above, we need to compute the average duration (endTime-startTime) for the state WAIT_FOLLLOWUP. From the palliative care process model shown in Figure 5-20, the duration is computed by measuring the start time of the form Alert and the start time of the form Consult. The following rule definition shows that the form (Alert) triggers the state WAIT_FOLLOWUP

-Rule: Patient Wait for Unscheduled FollowUp Consultation
Description: "This rule describes how the patient transits to state WAIT_FOLLOWUP for unscheduled consultation due to an abnormal condition occurred"
Condition: IF EVENT: Alert
CurrentState:
NextState: WAIT_FOLLOWUP

5.4.4 Architecture

The architecture of the new PAL-IS 2.0 CPMA adheres to the general CPM architecture pattern described in section 4.2.1. In general, the source of the events is the online forms that are accessible and filled out by the care providers in the community rather than BPM forms or RTLS events as in our first two case studies.

The detailed architecture for building PAL-IS 2.0 CPMA is depicted in Figure 5-22. The online forms are integrated with a Performance Reporting Dashboard component in the exact same browser-based interface. The data collection Forms and performance management Reports in are built in HTML independently of the Metrics Data Mart. CPME (Quickforms service) links the community accessible browser-based CPMA to the Metrics Data Mart by providing lookup values for form fields and query results for reports in JSON format. The Quickforms client extends the jQuery mobile libraries with functions to replace the lookup values in a wide variety of controls (drop down lists, radio buttons etc.) with a list of values in JSON format from a dimension table in the database. Similarly, there are controls which provide the data for an HTML table or chart with a list of query results in JSON format.

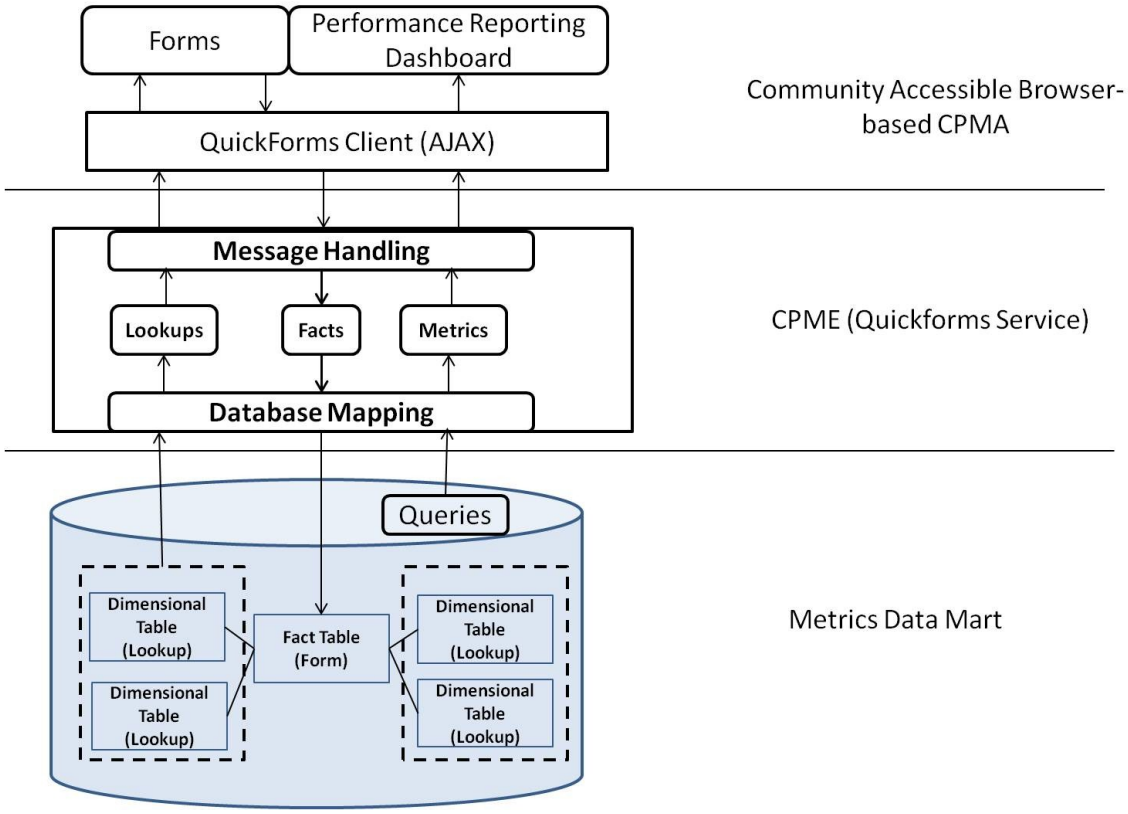


Figure 5-22: PAL-IS 2.0 CPMA Architecture

The CPME (Quickforms Service) was implemented by a Master student, Austin Chamney. As shown in Figure 5-22, CPME has the exact same interfaces (i.e., conformed to the CPME pattern) as case studies 1 and 2. It receives events from the Forms via a Quickforms client which in turns sends them to the CPME through the Message Handling that stores a data record (fact) to the Metrics Data Mart every time a form is submitted via Database Mapping. The CPME (Quickforms service) has the same internal components of CPME architecture pattern shown in section 4.2.1, except for the State Inferencing component which is missing. It is not needed due to the fact that the rules shown in the application model are straightforward in which each event (form) maps to a particular state transition.

The Metrics Data Mart model is a classic star schema optimized for reporting. In this case study, the Metrics Data Mart is more complex than in case study 2 because for each Form that records an event during the care process there is a Fact Table which stores that event record. Each field on the Form has its own Dimension Table that provides a lookup table of possible values for the field. Often, the lookup values are classified into a hierarchy to support drill down in reports. For example, a date field might be classified into day, months and years, so that one can see the number of referrals rolled up by day, month or year. Similarly, a Lookup Table of referring facilities might be classified by type of facility and geographic area. The Queries used to report metrics are stored in the Database as well.

Queries for metrics can be defined based on our application model in terms of Fact Tables for form events independently of the community accessible browser-based CPMA or CPME (Quickforms service). HTML Form designers (the application developers) can build forms and reports that are convenient to use. In our case, forms were designed using the jQuery Mobile JavaScript libraries to ensure compatibility across all device types and sizes.

In summary, the architecture of PAL-IS 2.0 CPMA still conforms to the high level CPM architecture pattern but there were some additional extensions required to fit the nature of the community care process in healthcare. First, forms and performance reports are integrated into the Performance Reporting Dashboard. Second the Metrics Data Mart is more complex in which there is one fact table for each event. Finally, the Metrics Data Mart master data such as the lookups is integrated into CPME to support online forms and ensure that the data is collected for reporting. However, the CPME is still evolving and the extensions made for PAL-IS 2.0 CPMA architecture might be integrated into CPM pattern.

5.4.5 Methodology

There were significant differences between case study 2 and this case study in terms of the supporting enterprise environment, as well as how the CPME that was implemented. But both CPMAs that were created conformed to the general architecture pattern, and the manner in which they were engineered was similar in that both explicitly followed the methodology described in section 4.3. The interesting aspects to highlight is how the same methodology addressed the rather different natures of the community care process in this case study, versus the hospital care process in case study 2.

Requirements Analysis

In this case study, we followed a very similar approach to case study 2 but the application model resulting from conducting the requirements analysis is quite different from the application model from case study 2 and reflected the nature of the community care process. The variation was due to the following:

1. The rules for state transitions in the community care process were straightforward and only required a single event (form) to transition into a new state instead of a combination of events as in case study 2.
2. The metrics are quite different and mostly were not state based. Rather, most of the metrics' calculations were based on counting attributes from specific forms.
3. All events came from online forms that record a state in the care process.

Again, as described in section 4.3, requirements analysis was performed by executing three different activities done by three different roles. The key activity was **Build Application**

Model that was achieved by the thesis researcher. A complete application model for case study 3 can be found in Appendix B. Figure 5-20 and Figure 5-21 show accurately what forms will be monitored at what states in the care process in order to measure the metrics needed for performance reporting. The **Care Process Analysis** activity was performed by Dr. Craig Kuziemsky, while the **Design and Architecture** activity was the responsibility of the thesis researcher's supervisor, Prof. Peyton, who interfaced with both Bruyère Hospital IT and CCAC IT.

It took two to three weeks to document and verify the requirements instead of four months. This is because all the requirements elements (goals, metrics, alerts, states, events, rules and sources) are precisely defined in a concise format represented in a single application model that requires much less effort to fix when the requirements are changed during the development process. From the experience we had in the initial case study, it was clear that the developers were confused about the precise details of event order, state attributes, and metrics and so those elements were misinterpreted by them; consequently, the developers always had to refer to the domain expert frequently. In the second case study, the developers had to refer to a single application model that showed how it corresponded to configuration points in the CPMA.

The effort for analyzing and documenting the requirements was low for case study 3 and was similar to that of case study 2. The requirements were gathered and documented within two to three weeks. This is much less than the time taken to gather and analyze requirements for PALIS 1.0 which took 4 months initially.

Test Driven Development with Scenarios

The behavior of the new PAL-IS 2.0 CPMA was defined in the test scripts that were created by the thesis researcher based on the candidate scenarios provided by the domain expert, Dr. Craig Kuziemsky, who interfaced with Bruyère Hospital. Those test scripts were validated by the same domain expert.

Figure 5-23 shows a snippet of a test script written according to the behavior driven development (BDD) format (Given-When-Then steps) which highlights what attributes needed to be filled in to verify that the metrics were being computed properly.

```
Scenario: New Patient
Given Referral Form is Recieved
When I fill in “Patient ID”
And I fill in "Referral Date"
And I fill in "Delivery Date"
And I fill in “Referral Source”
And I fill in “Referral Physician”
And I fill in “Primary Diagnosis”
And submit the form
Then event Referral should be logged
And the metric "Average Wait_Triage Time" should be updated
And the metric “Total # of Patients Cancer” should be updated
And the metric “Total # of Patients by Referring Physician” should be updated
And the metric “Total # of Patients by Referring Facility” should be updated
```

Figure 5-23: Snippet of Test Script using BDD Format (North, 2006)

In this case study, the test console was not used to automatically run the test scripts as in case studies 1 and 2. This was because the basic scenarios used for writing test scripts were much less complex than the basic scenarios in case study 1. In addition to that, the online forms were the only source of events, easy to use and only a few fields (attributes) were required to be filled in. Once PAL-IS 2.0 CPMA was assembled and configured, instead of using test console, a database script was first used to populate example patient data, then, a test plan was created with

test scripts that should be followed manually by the application developer based on scenarios provided by the care process expert, Dr. Craig Kuziemsky.

It is worth noting that the validation of test scripts in this case study was not achieved through simulating events by test console as in the first and the second case studies. Rather, test scripts were validated by explicitly walking through it with the key clinical staff at Bruyère Hospital and the test scripts were modified accordingly.

The overall effort for testing was similar to case study 2, but the test scripts written for case study 3 were simpler than the test scripts written for case study 2 and there is no need to run them automatically through a test console as in the previous case studies. The amount of time took for writing test scripts and run them manually with the clinical staff was about 3 weeks. Comparing to PAL-IS 1.0, testing effort in PAL-IS 2.0 CPMA is much less.

Implementation and Deployment

In this case study, there were two main activities, **Assemble CPMA** and **Configure CPMA**, performed by the application developer for implementing the new PAL-IS 2.0 CPMA. The initial step was to determine the components of PAL-IS 2.0 CPMA and the software used to implement them. An alternative implementation of CPME was built as a Quickforms service that fitted the nature of the community care process. The Quickforms service was implemented by a Master student, Austin Chamney. It had interfaces with the metrics data mart to log events as well an interface with the online forms and reports dashboard that were tied together as a one component.

The **Configure CPMA** activity was more complex to some degree than in case study 2 as we had to create and build online forms using HTML. Even though the implementation was straightforward, since the definition of the forms, its attributes and the corresponding events were clearly identified in the application model, the activity required detailed analysis of the lookup tables. Unlike case study 2, there was no generic solution for the metrics data mart, instead the metrics data mart was complex where a fact table and an associated lookup table for each attribute in the form had to be created for each form. For the performance reporting dashboard, rather than using IBM Cognos Business Intelligence 10 suite as in case study 2, Microsoft Excel was supported via a simple mechanism for copy/pasting result sets displayed in the CPMA.

Once the components were assembled (Quickforms service, Message Handling, Database Mapping and Metrics Data mart), the amount of time it took the application developer to configure those components, test the PAL-IS 2.0 CPMA and deploy into the enterprise architecture of the hospital was two to three weeks and was similar to case study 2.

Many thousand lines of code are required to implement PAL-IS 1.0 over the course of one year. In contrast, only few hundred lines of code is needed (i.e., implement 6 simple forms) to implement PAL-IS 2.0 CPMA over the course of two to three weeks. Thus, it took less time to implement PAL-IS 2.0 CPMA compared to the initial PAL-IS 1.0. The improvement, in terms of time needed to implement PAL-IS 2.0 CPMA, is attributed to the generic application framework and is not attributed to the learning effects. The only effort required whenever the application model changed due to requirements change was to modify the lookup tables relating to a specific form instead of changing custom code as in PAL-IS 1.0. Therefore, compared to PAL-IS 1.0, the

implementation of PAL-IS 2.0 CPMA is much less complex, in terms of number of lines of code needed.

5.4.6 Summary

In this section, we have shown how our complete application framework could be applied in different domains in healthcare, particularly, in community care process. There were significant differences between case study 2 and this case study in terms of the supporting enterprise environment, as well as the CPME that was implemented. However, the effort and the amount of time required to build the application model, build the test scripts, and configure the new PAL-IS 2.0 CPMA was quite similar to case study 2. As a result, the application framework approach is viable for both community care and cardiac care.

However, compared to PAL-IS 1.0, the effort required to develop the new CPMA was much less complex, in terms of number of lines of code needed, thus, the amount of time taken to develop the new PAL-IS 2.0 CPMA was reduced from one year to 2 months.

5.5. Chapter Summary

In this chapter, we developed three different case studies for two different applications (one for community care and one for hospital care). We showed how our application framework evolved in each case study. The result of each case study was a CPMA prototype that was evaluated and validated by the clinical staff of Osler and Bruyère hospitals.

In the initial case study, the application meta-model was not developed at the time of conducting the case study, and the high level of the CPM architectural pattern was used. In terms of application development, we followed a traditional approach to build CPMA. The main goal

for this case study was to measure wait times and service times of the cardiac patients as the cardiac care process was taken place.

In the second case study, we re-implemented CPMA by leveraging our complete application framework. In particular, the application model was instantiated from the application meta-model in order to define the specifications of the new CPMA. We summarized quantitatively the engineering effort required to build the new CPMA.

In the last case study, we applied our complete application framework to build CPMA for community care. The application model was exactly the same as in the second case study, even the high level architectural pattern was the same. However, the architectural pattern was evolved and refined. The results in terms of engineering effort are quite similar to the second case study.

Then Next chapter will present the evaluation of our application framework in each of our case studies and compare them to the ETL-Based Data Warehouse. The comparison is based on the evaluation criteria we defined in section 3.3. Then, we will use the same evaluation criteria to compare our application framework against the related works described in section 2.4.

Chapter 6. Framework Evaluation

The evaluation of our application framework for care process monitoring is based on the three case studies described in Chapter 5. Recall that our application framework was developed iteratively during the development of two different applications: one related to a hospital care process (ACS clinical pathway) and the other related to a community care process (palliative care). First, we use the evaluation criteria defined in section 3.3 to compare the three different versions of the application framework applied in the case studies with the ETL-based Data Warehouse approach typically seen in health care organizations today. After that, we use the evaluation criteria from section 3.3 to compare our application framework to the related works identified in section 2.4. Those related works are: SOA-Based CEP, RTLS- Based CEP, BPM-Based CEP and BPM-Based Monitoring Dashboard. Then in the final section, we discuss the limitations and assumptions of our thesis.

6.1. Evaluation of Application Framework across Case Studies

In this section we evaluate our application framework prototype from each of the three case studies that we described in chapter 5 and as well as the typical ETL-Based Data Warehouse approach that was in place at the two hospitals where we performed our case studies. The ETL-Based Data Warehouse approach is described in chapter 3.1 as well as in the current environment for Case Study 1(Osler Hospital) and Case Study 2 (Bruyère Hospital). It is also described in the background section of chapter 2. Our analysis of ETL-Based Data Warehouse also draws on the work of Alain Mouttham, who used the Osler Data Warehouse to create historical reports on

cardiac patient flow monitoring as part of the analysis work that was performed before Case study 1 started (Mouttham, Kuziemsky & Peyton, 2011). Case study 1 was done very early in the development of our application framework. There was no application meta-model and the architectural pattern for care process monitoring was still not well defined. In case study 2, regarding the extended hospital care process monitoring, and in case study 3, regarding the community care process monitoring which is the new version of PAL-IS 2.0 CPMA, the application framework was complete and the same application meta-model was used for both. Both case studies used the same general architectural pattern for care process monitoring, but in case study 2 we used the more sophisticated State Monitoring Engine that was integrated into the hospital's event driven architecture, while in case study 3 we used a simplified CPME (Quickforms Service) that was optimized to handle the simple online forms of our community care process.

In order to highlight the comparison of our application framework in each of the three case studies, we color code each table cell as follows:

1. Green color code indicates that the evaluation criteria for a specific case study is completely satisfied (**GOOD**/Complete).
2. Yellow color code indicates that the evaluation criteria for a specific case study is partially satisfied (**FAIR**/Partial).
3. Red color code indicates that the evaluation criteria for a specific case study is not satisfied (**POOR**/Incomplete).

6.1.1 Engineering Efforts

Table 6-1 evaluates ETL-based Data Warehouse and each version of our framework from the three case studies in terms of engineering efforts required to build the monitoring applications. The evaluation shows how it is difficult to develop a monitoring application in ETL-Based Data Warehouse approach and in the first case study, and how the enhanced version of our application framework simplifies the development of monitoring applications in case study 2 and 3.

The efforts required for requirements in terms of gathering information and documentation for ETL-Based Data Warehouse and the initial case study is high and takes months of painstakingly work to develop four separate documents which are power point slides to represent and define the care processes, interaction diagram to show where the data could be collected, goal models, and finally definitions of required performance metrics. In the second case study the requirement efforts is low, compared to the initial case study, because, once the requirement elements (goals, metrics, states, events and their sources) are captured and analyzed, the amount of time it takes to document those requirements elements in a single application model is approximately a couple of weeks. In the same manner, the requirements effort is low for case study 3 and the amount of time needed to complete documenting the requirements is much less, compared to the previous version of PAL-IS (web form-based portal). The requirements for case study 3 were gathered and documented within two to three weeks.

Table 6-1: Comparison in Terms of Engineering efforts

Criteria	ETL-Based Data Warehouse	Case Study 1	Case Study 2	Case Study 3
Requirements Efforts	High, ad hoc, and difficult to maintain.	High, ad hoc, 4 months to document and difficult to maintain	Low, model based, 1 graduate student, 2-3 weeks to document and easy to maintain	Low, model based, 1 graduate, 2-3 weeks to document and easy to understand
Level of Coding	Low Level: hard coded ETL, but high level BI reports	Low level hard-coded and low level custom reports.	High level framework for both data integration and BI reports	High level framework for both data integration and BI reports
Tool Support	Standard ETL and data warehouse tools	Good tool support but custom coded CPMA: Websphere Business events, Websphere Lombardi, Grails, RTLS	SME engine but state handlers custom coded; data warehouse tools	Quickforms Framework and data warehouse tools
Learning Curve	High: ETL	High: IBM tools and custom coding	Low to moderate: Data warehouse tools, plus custom handlers in Grails	Low to moderate: Simple structured Ajax forms and data warehouse tools
Event Processing and Metrics Specifications	ETL. Coarse Grained. Historical reports only.	Ad hoc, not modeled, two graduate students 2-3 weeks	Modeled, Thesis researcher, 2-3 days	Modeled, Thesis researcher, 2-3 days
Implementation Effort	Custom-coded ETL processes, writing custom SQL queries in BI dashboard tool	Hard-coded GRAILS application code and SQL queries for reports, with custom-code CEP rules and Event handlers. 2 graduate students including the thesis researcher, 8 months	Configurable, model-based SME, Custom handlers (in simple GRAILS syntax) for each state in process, BI dashboard tool, one graduate student, 2-3 weeks	Configurable, model-based HTML Forms and custom SQL queries, one graduate student, 2-3 weeks
Modeling Effort	Complex: Data models disconnected from process	Complex: Business process model, interaction diagram, state transition diagram, 2 graduate students including thesis researcher, 2- 3 months	Simple: Application model, one graduate student (thesis researcher), 2-3 weeks	Simple: Application model, one graduate student (thesis researcher), 2-3 weeks
Testing Effort	Moderate: ETL and custom reports.	Moderate: test console but not model-driven reports, and more errors with event code to work through,2 graduate students including the thesis researcher, 1- 3 months	Simple: writing test script user scenarios, running the scripts using test console, one graduate student (thesis researcher), 1 -3 weeks	Simple: writing test script user scenarios, running the scripts manually, one graduate student (thesis researcher), 1-3 weeks

Coding level for developing ETL-Based Data Warehouse is low. In the initial case study, the level of coding required to engineer CPMA is low due to the hard-coding of the main components of CPMA as well as customizing the performance reports. However, in the second

and the thied case studies, the level of coding is high. This is true due to the fact that the proposed application framework applied in these case studies supports high level data integration from various event sources configured through CPMA application model. Furthermore, BI reports supported by the application framework require high level code to customize the reports through the BI tool.

Since ETL-Based Data Warehouse focused on providing analytics towards care processes, the supporting tools leveraged for building such application are standard ETL and data warehouse. However, the tools for developing CPMA in the initial case study are IBM tools for BPM and CEP, RTLS tool for location based monitoring and finally open source Grails for monitoring the care process. In case study 2, the supporting tool needed to process and integrate events to monitor the states of the resources is SME engine, but a custom code is required for each state defined in the application model. Furthermore, a data warehouse tool is needed to provide performance management reports. Finally, in the last case study, the supporting tool needed to process the online forms and provide analytics is the Quickforms service. In addition, a data warehouse tool is necessarily required for performance management.

The learning skills required building ETL-Based Data Warehouse and a monitoring application in the initial case study is both high in different ways. In the former, it is required to learn ETL as is the major process for data integration and analytics. While in the latter, it is essential to learn IBM tools in addition to learning programming language to write custom code. In fact, in case study 2, the learning curve is low to moderate, compared to the first case study; since it is only required to learn data warehouse tools for providing performance reports in addition to learn how to custom code the state handlers supported by SME for each state defined

in the CPMA application model. Since the CPMA developed for case study 3 is a web based analytical application, it is essential to learn how to create simple online forms, as well, to learn the data warehouse tools for reporting. Therefore, the learning curve is low to moderate.

Event processing and metrics specifications in ETL-Based Data Warehouse are achieved through ETL but not all required information is available and correct. Therefore, only coarse grained data is provided by this approach. For the initial case study, the design of events is performed in an ad hoc manner in which the relationship between events and the related performance metrics is graphically represented in a power point slide deck. In case study 2, the specifications for event processing rules and metrics computations are documented in a single application model. In both case study 2 and case study 3, these final detailed specifications were done comfortably in a couple of days as a final pass after all the other details of the application model had been completed.

The implementation efforts in ETL-Based Data Warehouse involve writing SQL queries against data warehouse using BI tool, assuming that ETL process is already performed. On the other hand, in the initial case study, a team of two graduate students including the thesis researcher spent about 8 months to hard-coded the monitoring application using Grails software including the event handlers, writing ad hoc SQL queries in Grails against an ORM database, and developing CEP rules using IBM Websphere Business Events. For case study 2, there is a huge improvement, compared to the initial case study, in terms of the time needed to implement the monitoring application. In fact, in only took one graduate student two to three weeks to implement the application, once the SME and Metrics Data Mart were assembled. Implementation involved configuring the SME in terms of what events, states should be

monitored, specifying the lookup tables for the states and the events as well, writing a custom coded state inference handler for each state of the care process according to CPMA application model. Finally, in case study 3, once Quickforms service was available, it required only one graduate student who spent two to three weeks to implement configurable HTML forms with their contents based on the application model, specifying the lookup tables for the forms attributes and writing custom SQL queries against data warehouse. In effect, adopting the application framework in this case study leads up to a huge improvements, compare to the old version of community care monitoring (PAL-IS 1.0) which was implemented by two graduate students and took them about 1 year.

The modeling effort required for building ETL-Based Data Warehouse is complex because of the separation of data model from the process model. In the same manner, the effort for modeling the CPMA in the initial case study is complex, as well, because specific models were needed to model the monitoring application where one graduate student and the thesis researcher spent couple of months to model the application. On the other hand, in case study 2, the modeling is much different, compared to the initial case study, and the effort required modeling the monitoring application is simple and basically was achieved by creating a single application model that defines the application. The construction of the model was accomplished by the thesis researcher and only took him couple of weeks. Similarly, the effort required to model the application for the last case study was straightforward and, as in the case study 2, the thesis researcher was responsible for constructing the application model which had completed in couple of weeks.

Testing efforts regarding ETL-Based Data Warehouse application is complex because it requires testing both ETL process and the custom BI reports. Testing was different in the initial case study. Even though test console was leveraged for testing the monitoring application and the creation and running of test scripts could be done in a matter of days, the overall testing effort was quite high due to errors and inconsistencies in the requirements. This process took about couple of months and was performed by two graduate students. In case study 2 the existing test scripts were reused and run. Overall testing was low as requirements were clearly documented in the application model so there were fewer errors and inconsistencies. This process was done by the thesis researcher and required about 2 weeks to accomplish. Finally, in case study 3, the test scripts were simpler than the test scripts written for case study two and the scripts were run manually by walking through the scripts with the key clinical staff at Bruyère Hospital.

6.1.2 Application Features for Care Process Monitoring

Table 6-2 evaluates ETL-based Data Warehouse and each version of our framework from the three case studies in terms of the application features provided for care process monitoring according to the criteria identified in chapter 3.

It is important that the forms used in the care process be tightly linked to the reports which measure performance, but this is only achieved in case study 3 for community care process using the Quickforms service. In case study 2, there is a tight link between the events collected from the care process and the performance reports, but the forms used in the BPM supported process are only implicitly connected to those events. In the ETL-Based Data Warehouse application there is even less of a link between care process forms and performance reports as the data stored in the various IT systems has to be extracted, transformed and loaded

(ETL) before it is reported on. This also means that data integration can only support historical reports, whereas each of the three case studies supported data integration in near-real time so that reports were available while the process was taking place.

Table 6-2: Evaluation of support for Application Features for Care Process Monitoring

Criteria	ETL-based Data Warehouse	Case Study 1	Case Study 2	Case Study 3
Integrated Care Process Forms	Yes, but not linked to reports.	Implicit in BPM, but must translate to events for reporting.	Implicit in BPM, but must translate to events for reporting	Yes, tightly linked to reports
Near Real-time Data Integration	Historical Only, ETL support	Near Real Time, Complex Event Processor	Near-Real Time, Event-based State Monitoring Engine (SME)	Forms Only, Near Real-time, Quick Forms service
Performance Management Reports	Data warehouse using BI tools	Ad hoc queries and hard-coded reports against ORM database	Data warehouse using BI tools	Data warehouse using BI tools
Usability at point of care	Historical reports only	Custom coded near-real time reports at point of care	Model-based near-real time reports at point of care	Model-based near-real time reports at point of care
Track Current State and Status of Individuals	No	Yes	Yes	Yes
Near Real-time Alerts	No	Yes	Yes, simple flagging.	Yes, Simple flagging

Performance management reports are supported for ETL-Based Data Warehouse, as well as all monitoring applications developed for the case studies. In ETL-Based Data Warehouse and the last two case studies, the performance management reports are generated by applying BI tools against the information stored in the data warehouse. However, in the initial case study, the performance management reports are hard coded and are available through running ad hoc queries against ORM modeled database. This kind of database is not optimized for reporting because it stores only the information about the objects defined in the object model of the monitoring application.

In ETL-Based Data Warehouse, the application is not particularly useful at point of care for care providers since it can only provide historical reports, and not reports of what is currently taking place. With respect to the initial case study, the monitoring application is usable for the care providers and managers are point of care and the performance reports are available in near real-time as the care process is executed, but those reports are custom coded and not model-based. On the contrary, in the second and the last cast studies, the performance reports are modeled and measured according to the application model and are available at point of care for care providers and managers in near real-time, since the events are collected as they are happening. This provides a rich, easy to navigate, standardized interface for care providers.

Similarly, the ETL-Based Data Warehouse has no information on the current states and status of the individual patients or resources within the care process. On the other hand, all case studies support tracking and monitoring the current states and status of patients and resources in near real-time.

Alerts are supported, in near real-time, in all the case studies. The alerts are simply flagged to highlight violations if a particular care process state or performance metric exceeds the acceptable range.

6.1.3 Application Features for Hospital Care Process Monitoring

Table 6-3 summarizes evaluation based on supporting application features for hospital care process monitoring. Although the framework in case study 2 was only used for a community care process, it is evaluated, since it could be used for hospital care process monitoring based solely on the use of online forms with no integration with event driven architecture. This is

important, since most hospitals today currently have only limited support for event driven architecture.

Table 6-3: Comparison in Terms of Application Features for Hospital Care Process Monitoring

Criteria	ETL-Based Data Warehouse	Case Study 1	Case Study 2	Case Study 3
Near Real-time Performance Reporting	No. Historical performance reporting	Yes	Yes	Yes. (Based on data collection from forms only)
Event Collection	ETL batch processing	IBM Message Broker	IBM Message Broker	Quickforms service
Metrics Granularity	High Level	Fine Grained	Fine Grained	Fine Grained

ETL-Based Data Warehouse does not support near real-time performance reporting because of the delay in persisting data in the data warehouse. However, all the cases studies had the capability to provide near real-time performance reporting due to the fact that the care process events and states are persisted in the database as they are happening.

In ETL-Based Data Warehouse, event collection is performed through ETL batch processing from diverse source systems into a data warehouse. Often this is restricted to data collected via online forms, but it could include event data extracted from patient monitors, for example. In the first and the second case study, we used a message broker and message queue to capture events. In the last case study, specialized Quickforms service was used strictly for optimized form-based collection.

Concerning metric granularity, ETL-Based Data Warehouse provides high-level metrics, but not fine-grained metrics, as the architecture consolidates and integrates data from various disconnected information systems days or even weeks after the fact and the data that is available in the data warehouse is not fine-grained enough to have a complete view of all the steps in the care process. Whereas, the architecture leveraged in all case studies has the ability to collect and

integrate events from various sources, as they are taking place, and to infer care process states in order to compute fine-grained metrics for the care process.

6.1.4 Application Definition

The criteria described in the table below address how easy it is to define the care process monitoring application (requirements). In particular, it is important to be able to understand and articulate the relationship between overall performance management and the step by step details of the care process itself.

Table 6-4: Comparison in Terms of Application Definition

Criteria	ETL-Based Data Warehouse	Case Study 1	Case Study 2	Case Study 3
Identification of checkpoints and states in care process	Not defined	Defined in an ad hoc manner	Care process states are defined by a single consistent application model	Care process states are defined by a single consistent application model
Identification of metrics	Defined in ad hoc manner	Defined in an ad hoc manner	Defined by a single consistent application model	Defined by a single consistent application model
Identification of event sources	Not defined	Defined in an ad hoc manner	Defined by a single consistent application model	Defined by a single consistent application model
Relationship between performance management and care process	Relationship is established in an ad hoc manner	Relationship is established in an ad hoc manner	Relationship is established by a single consistent application model	Relationship is established by a single consistent application model
Relationship between care process and sources of events	Relationship is not established	Relationship is established in an ad hoc manner	Relationship is established by a single consistent application model	Relationship is established by a single consistent application model.
Model-defined CPMA	Performance management model only	Not model-defined	CPMA Application model	CPMA Application Model

In an ETL-Based Data Warehouse the performance management model is usually well represented and understood, but the relationship to the step by step details of the care process is usually not captured. Often the sources of data for the performance management model is not well understood either. In the first case study, these relationships and sources were eventually

identified and understood through the process of building the CPMA, but in an ad hoc manner, and there was no application model that could represent the information and relationships clearly. The Application meta-model was developed in case studyies 2 and 3 to close the gap.

6.1.5 Architectural Requirements

Table 6-5 summarizes the architectural requirements needed to build the monitoring application in each case study and compares them to the architecture components supported in ETL-Based Data Warehouse.

Table 6-5: Comparison in terms of Architectural Requirements

Criteria	ETL-Based Data Warehouse	Case Study 1	Case Study 2	Case Study 3
EDA-SOA	Not supported	Yes, complex event processing	Yes, State Monitoring Engine	Yes, simple event processing (forms)
State Inferencing	Not supported	Yes, CEP	Yes, state based approach in SME	Yes, but very simple for forms
Integrate with Event sources	Yes, but not in real-time.	Yes	Yes	Yes, forms only
Integrate with Business Process Management Architecture	Not integrated	Yes	Yes	Not integrated
Integrate with Performance Management Architecture	Yes, but not in real time	Yes, near real-time	Yes, near real-time	Yes, near real-time
Integrate with Real-Time Location Service	Not supported	Supported	Supported	Not Integrated

Event Driven Architecture (EDA) is not supported in ETL-based data warehouse, nor is event integration and filtering, but both were supported in all case studies. In the initial and the second case studies, consuming, processing and producing events are done by CEP and SME respectively. In the last case study, each form corresponded to a state transition so event processing was simple, but supported by the Quick Forms service.

For ETL-Based Data Warehouse, integration with event sources is possible using batch-based ETL processes, but it is often not included. In the last case study, only online forms were used so there was no integration with event sources.

CPMA developed for the initial and the second case studies supports integration with Business Process Management (BPM) through the EDA. Care processes must be implemented in the BPM so that they generate the necessary events. It is not automatic from the BPM forms themselves. BPM was not integrated in case study 3, since that technology is more applicable in hospitals and less likely to be used in a community care process that cuts across organizations.

Integration to performance management is supported in ETL-Based Data Warehouse but not in real-time since the data warehouse is populated after the fact. All the case studies supports integration with the performance management in near real-time for displaying the performance metrics associated to a particular care process as the process is taking place.

ETL-Based Data Warehouse does not support integration with RTLS. While, in the last case study, RTLS is not integrated since the monitoring application is built only to process online forms. On the other hand, CPMA developed for the initial and the second case studies support and require integration with RTLS to track the location of the resources involved in the care process and to provide fine grained care process monitoring. Custom code was needed to convert the data from the RTLS into the events sent to the care process monitoring application.

6.1.6 Application Development Process and Tools

Table 6-6 shows and compares what tools or techniques were available to support different phases of the application development process.

Table 6-6: Comparison in Terms of Application Development Process and Tools

Criteria	ETL-Based Data Warehouse	Case Study 1	Case Study 2	Case Study 3
Tool or Technique to test CPMA prior enterprise integration	None	Test console	Test console	Forms and Data scripts
Tool or Technique for event processing and integration rules	None	Custom CEP rules and Event handlers	Custom State handlers	Quickforms Service (simple events only)
Tool or Technique to translate care process events to performance metrics	ETL+ Custom SQL queries	ORM, Custom SQL queries	DAO + BI tool (model-based OLAP queries)	DAO+ BI Tool (model-based OLAP queries)
Tool or Technique for visualizing the requirements	Ad hoc	Ad hoc	Application Model	Application model
Support agile development process	No. Traditional software engineering of ETL processes and BI reports.	Partial. Traditional software engineering of CPMA, but Behavior driven development of CEP rules and reports.	Yes. Application Model with Behavior Driven development	Yes. Application Model with Behavior Driven development

In the second and third case studies, a test console was used to simulate expected events and forms prior to integration with the enterprise architecture where the process took place. In particular, using behavior driven development the critical user scenarios were identified and simulated, t o ensure that performance would be accurately measured. There is no ability to do this with ETL-based Data Warehouse. In the last case study, database script was used to populate example patient data, then, a test plan was created with test scripts that should be followed manually by filling the online forms based on scenarios.

The event integration rules are trivial for the last case study. Moreover, there is no support to identify and manage integration rules in ETL-based Data Warehouse; it is achieved as a side effect of custom coding ETL scripts and SQL queries. Event integration was not done in the last case study due to the simplicity of the process and the forms. In the first case study, it was an onerous task to custom code correlation of events corresponding to the state of the

process in CEP rules and a event handlers for each event. In the second case study, this task was greatly simplified. One only had to encode a standardized state handler for each state in the process.

In ETL-Based Data Warehouse, ETL process translates the care process events to a data record in the data warehouse. Then custom SQL queries are running against the data warehouse to compute the performance metrics. For the initial case study, the object model is transformed automatically as a record in the corresponding table through the use of an ORM (Object-Relational Mapping technology). Then, custom SQL queries are run against the ORM data base to calculate specific performance metrics, but this is not optimal for performance reports. In the second and the last case studies, a standardized DAO is used by the state handlers to log events and states in a straightforward manner into a star schema database optimized for performance reporting. Reports are created using model-based OLAP queries running against the star schema database, which supports navigation through reports using drill up and drill down functionality.

In ETL-Based Data Warehouse and the initial case study, there is no tool or technique that supports visualizing requirements to the development team of the monitoring application. Rather, the requirements are documented in a text document and manually leveraged by the developers to build the application. In the second and the third case studies, the requirements are captured in an application model to visualize the relationship between goals, metrics, care process states, events and event sources. However, there is no tool support, so the model and its diagrams must be built by hand.

An Agile development process is not supported to build the application ETL-Based Data Warehouse. Instead, a traditional software process is followed for implementing ETL process and the performance reports. In the first case study an agile development process is only partially through the use of the test console. However, in the second and the last case studies, an agile development process is fully supported in two ways. First, scenarios and test scripts using the test console are created to support behavior-driven development. Second, the application model is utilized to pre-build much of the CPMA including the star schema database and SME engine, so that development is largely a matter of configuration and specification of process state handlers as they are needed.

6.2. Comparison of Application Framework with Related Works

In this section, we compare our application framework with the four categories of related work that were identified in chapter 2. The comparison is based on a composite of all the related works listed for a particular category. As long as at least one related work supported a feature or evaluation criteria, then we deemed that the entire category supported the feature or evaluation criteria for the purposes of the comparison.

SOA-Based CEP (Boubeta-Puig, Ortiz, & Medina-Bulo, 2011) (Middleton G. , Peyton, Kuziemsky, & Eze, 2009) (Vaidehi, Bhargavi, Ganapathy, & Hemalatha, 2012) is concerned with monitoring care processes by integrating services from heterogeneous healthcare applications with CEP. This approach processes events generated from services in order to send alerts to the desired services when a particular complex event patterns is detected. It can also support a data mart and performance reporting dashboard based on logging all events processed by CEP.

On the other hand, RTLS-Based CEP (Yao, Chu, & Li, 2011) focuses on using RFID tags to track patients and assets in the hospital and integrate those RFID location events using a commercial CEP to infer critical situation and to generate alerts. BPM-Based CEP (Janiesch, Matzner, & Müller, 2012) (Vaidehi, Bhargavi, Ganapathy, & Hemalatha, 2012) focuses on leveraging CEP to process events generated from BPM, as well to oversight the business processes. Tracking KPIs is accomplished by displaying them in a dashboard. Commercial off-the-shelf software is leveraged to implement CEP, BPM and the dashboard.

Finally, BPM-Based Monitoring Dashboard (Zhu, Nie, Lu, & Duan, 2010) (Tegegne & Peyton, Application framework support for process-oriented software development, 2013) focuses on capturing events from workflow activities (tasks) and integrates data from other information systems in order to assist the managers and the administrators to monitor the comprehensive states of the workflow on the fly.

Similar to section 6.1, in order to highlight the comparison of our application framework with the four categories of related works, we color code each table cell as follow:

1. Green color indicates that the evaluation criteria for a particular category is completely satisfied (**GOOD**/Complete).
2. Yellow color indicates that the evaluation criteria for a particular category is partially satisfied (**FAIR**/Partial).
3. Red color indicates that the evaluation criteria for a particular category is not satisfied (**POOR**/Incomplete).

6.2.1 Engineering Efforts

Table 6-7 below summarizes the comparison between our Application Framework approach against the other approaches in the related works, SOA-Based CEP, RTLS-Based CEP, BPM-Based CEP and BPM-Based Monitoring Dashboard with respect to the engineering efforts.

Table 6-7: Comparison with the Related Works in Terms of Engineering Efforts

Criteria	SOA- Based CEP	RTLS- Based CEP	BPM- Based CEP	BPM-Based Monitoring Dashboard	Application Framework
Requirements Effort	High, disjointed traditional requirement techniques	High, disjointed traditional requirement techniques	High, disjointed traditional requirement techniques	Low, model based	Low, model based
Level of Coding	Low level: custom coded CEP engine, hard-coded Event simulator	Low level: hard-coded RFID middleware, custom coded CEP engine, hard-coded event simulator and dashboard	Low level: custom coded for Commercial off-the-shelf CEP engine, BPM engine and Dashboard, hard coded for messaging middleware	High level: Configure application with models	High level framework for both data integration and BI reports
Tool support	Open source Esper for CEP, EPL for event processing, but no high level tools.	Open Source Drools for modeling business processes, CEP rules and events, but no high level tools.	SAP NetWeaver for BPM, SAP Sybase Aleri for CEP, SAP BusinessObjects Xcelsius for Dashboard, Active MQ for messaging middleware	XML editor to edit the models represented as XML files but no high level tools	State handlers, custom code for CEP; data warehouse and BI reporting tools; AJAX forms linked to data warehouse using Quickforms.
Learning Curve	High: Programming Language, Configure and custom coded Open source software for CEP, Writing Complex event rules	High: Programming Language, Configure and custom coded Open source software for CEP, Writing Complex event rules	High: A wide variety of complex tools to learn and integrate	Low: XML Models for simple form-based work flows	Low to moderate: Data warehouse tools, plus custom state handlers
Event Processing and Metrics Specifications	Not modeled, ad hoc	Not modeled, ad hoc	Not modeled, ad hoc	Modeled	Modeled

Implementation Efforts	Configuring and customizing open source software for CEP	Configuring and customizing open source software for implementing care process and CEP	Configuring and customizing Commercial off-the-shelf software to implement BPM, CEP and Monitoring Dashboard	Configurable model-based application	Configurable, model-based CPMA
Modeling Efforts	Complex: Events modeling, Event Integration rules modeling, data modeling	Complex: Business process modeling, Events Modeling, Event Integration rules modeling	Complex: Business process modeling, Events Modeling, CEP modeling, Event Integration rules modeling	Simple: Application Model	Simple: CPMA application model
Testing Effort	Complex: Custom event processing and complex rules	Complex: Custom event processing and complex rules	Complex: Custom event processing and complex rules	Moderate: Configure and executing separate XML model files	Simple: writing test script user scenarios, running the scripts using test console

Requirements efforts to define CPMAs in all the related approaches except for BPM-Based Monitoring Dashboard is complex compared to our approach since traditional techniques are adopted in a disjoint fashion to define different features of the application (business process, event processing, metrics). A single integrated CPMA application model simplifies this process considerably. The requirement efforts for BPM-Based Monitoring Dashboard are low, in part, because event processing is less complex and focused solely on monitoring the steps of a defined business process.

Developing the monitoring applications requires integration and configuration of many different components often with a large amount of custom low-level code. SOA-Based CEP, RTLS-Based CEP and BPM-Based CEP approaches leverage commercial off-the-shelf software, but they are often complex with their own proprietary languages for writing custom code to configure. Also, SOA-Based CEP and RTLS-Based CEP key components were custom built in

their entirety. BPM-Based Monitoring Dashboard use high level coding and only requires configuring the monitoring application using XML files. These XML files are largely manageable because the models and CPMAs are simply defined (form-based workflow). Similarly, our CPMA approach is considered as a high-level framework. There is custom coding of the rules for individual state handlers, but the rules are a simple statement of state inferencing from observed events.

Low-level tools are leveraged to support developing monitoring applications for all the related approaches except for BPM-Based Monitoring Dashboard. In particular, SOA-Based CEP and RTLS-Based CEP leverages, open source tool for implementing CEP. BPM-Based CEP is the most well supported in terms of tools as there are commercially available tools for every component of the architecture, although integrating and coordinating those separate tools can be challenging. BPM-Based Monitoring Dashboard provides only an XML editor to edit the models. However, the run-time environment is well integrated. Our run-time environment is similarly well-integrated but our in-house developed CPMEs are lacking in tool support.

The learning curve required in order developing the monitoring applications SOA-Based CEP, RTLS-Based CEP and BPM-Based CEP is high because these approaches require custom coding. Moreover, those approaches require learning and configuring commercial off-the-shelf or open source software to develop different components of the monitoring application. The learning curve in BPM-Based Monitoring Dashboard is low and requires learning simple XML models. In our application framework, the learning curve is low to moderate because it only requires learning data warehouse tools for performance reporting and learning how to write state handler rules.

The specification of event processing and metrics calculation is not model-based for the approaches SOA-Based CEP, RTLS-Based CEP and BPM-Based CEP. Instead, computing metrics from event processing is achieved in an ad hoc manner by defining the custom SQL queries that should be running against a particular database. On the contrary, our application framework and BPM-Based Monitoring Dashboard use application models for the specification of business process, event processing and metrics.

The implementation effort required to build monitoring applications using SOA-Based CEP, RTLS-Based CEP and BPM-Based CEP approaches is high and complex as these approaches must integrate a mix of commercial off-the shelf software, open source software and custom-code components. In contrast, the implementation effort of applications in BPM-Based Monitoring Dashboard and our application framework is low as the design of the CPMA has been systematized to the point where implementation is largely about model-based configuration.

The modeling effort is complex for SOA-Based CEP, RTLS-Based CEP and BPM-Based, because these approaches need to model all the desired events and model the event in rules using CEP engines, as well as model the business processes using BPM engine. In BPM-Based Monitoring Dashboard, the modeling effort is simple because the models are simple. Our approach benefits from a single integrated application model.

Because of the need to write many test cases for the purpose of testing the event processing engine and ensuring that complex event rules are working properly, the effort required to test the monitoring applications developed in the related works SOA-Based CEP, RTLS-Based CEP, BPM-Based CEP is complex. In contrast, the effort requires to test our

approach is straightforward because it is only required creating and running test script scenario via test console. Whereas in BPM-Based Monitoring Dashboard, the testing effort is based on the effort for configuring and executing the application model.

6.2.2 Applications Features for Care Process Monitoring

Table 6-8 below is the summary comparisons between our approach against SOA-Based CEP, RFID-Based CEP, BPM-Based CEP and BPM-Based Monitoring Dashboard in terms of care process monitoring application features.

Table 6-8: Comparison with Related Works in Terms of Care Process Monitoring Features

Criteria	SOA- Based CEP	RTLS-Based CEP	BPM- Based CEP	BPM- Based Monitoring Dashboard	Application Framework
Integrated Care Process Forms	Yes, but not linked to reports	Yes, but not linked to reports	Yes	Yes	Yes
Near Real-Time Data Integration	Near Real-time, Messaging Middleware	Near Real-time, CEP engine	Near Real-time, messaging middleware	Near Real-time, integrated run-time environment	Near-Real Time, State Based CPME
Performance Management Reports	Data warehouse	Data warehouse	BAM	ORM database	Data warehouse
Usability at Point of Care	Historical reports only	Yes-	Yes	Yes	Yes
Track Current State and Status of Individuals	Alerts only	Yes	Yes	Alerts only	Yes
Near Real-Time Alerts	Yes	Yes	Yes	Yes	Yes

BPM-Based CEP and BPM-Based Monitoring Dashboard supports care process forms provided by BPM. However, RTLS-Based CEP supports care process forms but is not linked directly to reports. The same applies for SOA-Based CEP. On the other hand, our application framework can support both types of care process forms, the first one is the web form-based that

is tightly linked to reports and the second one is the care process forms implicit in BPM and used for reporting once it is converted to event stored in the database.

The monitoring applications developed for the related approached leveraged different techniques for data integration. Both SOA-Based CEP and BPM-Based CEP used Message-oriented middleware. For RTLS-Based CEP, the CEP engine has the ability to integrate date from various data sources. On the other hand, in BPM-Based Monitoring Dashboard, there is an integrated run-time environment. In our approach, data integration is accomplished by CPME in which events are integrated from the available event sources in the healthcare organization to infer and update states related to the resources participating in the care process.

All the monitoring applications developed in the related approaches support performance management reports. In SOA-Based CEP, RTLS-Based CEP and our application framework there is a well-defined data warehouse for which any off-the-shelf enterprise reporting and BI tool can be used. In BPM-Based CEP and BPM-Based Monitoring Dashboard either an ORM or proprietary database format is used which can be problematic for external reporting tools (although both approaches have a rich set of built in reports).

SOA-Based CEP provides only historical reports which may not be useful at point of care. However, system usability at point of care in RTLS-Based CEP is supported in which the complete resource (i.e., patient) information (i.e., patient EHR) can be retrieved, for example during the operation room, to avoid critical situations. Even though BPM-Based CEP is not implemented in healthcare domain, usability at point of care is possible because of the existence of a monitoring dashboard in their architecture that collects information from the underlying

business process. In BPM-Based Monitoring Dashboard, the application monitoring has the ability to drill down into a particular care process instance related to a particular resource and collect information about that resource at point of care. In the same degree, our approach provides monitoring reports for a particular resource on the dashboard at point of care in near real-time.

Only RTLS-Based CEP monitoring application, BPM-Based CEP and our approach have the ability to track the current states and status of resources participating in the care process. For the remaining approaches, the monitoring applications are not designed to track the current states of the resources. The remaining approaches, SOA-Based CEP and BPM-Based Monitoring Dashboard, are focused on raising alerts.

Alerts are triggered automatically by the monitoring applications developed in all the related approaches. In our approach, the monitoring application has the ability to flag alerts displayed in the dashboard when the duration of particular metrics and states exceeds the threshold, therefore, the managers can take immediate action.

6.2.3 Application Features for Hospital Care Process Monitoring

In this section we compare our CPMA with other related works, SOA-Based CEP, RTLS-Based CEP, BPM-Based CEP and BPM-Based Monitoring Dashboard. Table 6-9 summarizes the comparison in terms of application features that support monitoring hospital care processes.

Table 6-9: Comparison between CPMA and the Related Works in Terms of Hospital Care Process Monitoring Features

Criteria	SOA- Based CEP	RTLS- Based CEP	BPM- Based CEP	BPM- Based Monitoring Dashboard	Application Framework
Near Real-time Performance Reporting	Yes	Yes	Yes	Yes	Yes
Event Collection	Yes, Messaging Middleware	Yes, CEP	Yes, CEP	Yes, BPM engine	Yes, CPME
Metrics Granularity	Fine grained	Fine grained	Fine grained	Fine grained	Fine grained

All the monitoring applications developed in the related approaches have the power to provide near real-time performance reporting since they leverage technologies that are able to collect and process the required events as they are happening and store them in the database. The same applies to the monitoring application developed using our application framework.

Event collection from various event sources is supported by the monitoring application developed for the related approaches. But, each approach collects events differently. In SOA-Based CEP and BPM-Based CEP, event collection is achieved through message-oriented middleware. For RTLS-Based CEP, the event collection is performed directly by CEP. In BPM-Based Monitoring Dashboard, the event collection is accomplished by BPM engine and transforms them to data in the corresponding legacy systems. In our application framework, CPME is taking charge of collecting events from diverse event sources in the hospital enterprise.

Since all the related approaches in addition to our application framework support integration with diverse event sources within the enterprise, the corresponding monitoring applications have the ability to provide fine-grained monitoring of the processes.

6.2.4 Application Definition

Table 6-10 below summarizes the comparison between our approach against the other approaches in the related works in terms of the application definition.

Table 6-10: Comparison between CPMA and the Related Works in Terms of Application Definition

Criteria	SOA-based CEP	RTLS-based CEP	BPM-based CEP*	BPM-based Monitoring Dashboard	Application Framework
Identification of checkpoints and states in care process	Alerts and metrics are defined but no business process	Alerts and metrics are defined but no business process	All steps of business process have equal weight.	All steps in process have equal weight.	Application model identifies only the critical states in the care process that will be monitored
Identification of metrics	Defined in an ad hoc manner	Defined in an ad hoc manner	Defined in an ad hoc manner	Defined by XML model files.	Defined by a single consistent application model.
Identification of event sources	Defined in CEP engine in an ad hoc manner	Defined in CEP engine in an ad hoc manner	Defined in CEP engine in an ad hoc manner	Simple forms in the workflow process.	Defined by a single consistent application model.
Relationship between performance management and care process	Relationship is established in an ad hoc manner.	Relationship is established in an ad hoc manner.	Relationship is established in an ad hoc manner	Relationship is established in an ad hoc manner	Relationship is established by a single consistent application model.
Relationship between care process and sources of events	Relationship is established in an ad hoc manner	Relationship is established in an ad hoc manner	Relationship is established in an ad hoc manner	Relationship is simple as workflow forms are the source.	Relationship is established by a single consistent application model.
Model-defined CPMA	Data model	Event model	BPM model	Separate XML Models define CPMA.	CPMA Application Model

In SOA-based CEP and RTLS-based CEP, there are no care process states defined. In BPM-based CEP and BPM-based Monitoring Dashboard a complete business process is defined, but all steps in that process have equal weight. It is only in our CPMA application model that the critical states to be monitored in a care process are identified.

Identification of metrics and event sources is done in an ad-hoc manner in SOA-Based CEP, RTLS-Based CEP and BPM-Based CEP. In, BPM-Based Monitoring Dashboard metrics

are identified in XML model files and event source identification is systematic but simplistic since all events come from work-flow online forms. In our approach, metrics and event sources are systematically identified and organized in a single integrated application model.

The relationship between performance management and care process, as well as the relationship between care processes and the sources of events for SOA-Based CEP, RTLS-Based CEP is not well established since there is no model of care process. In BPM-Based CEP, although there is a business process model, the relationship between event sources and metrics is not included in that model and is managed on an ad-hoc basis. In BPM-Based Monitoring Dashboard, the relationship between processes and event sources is trivial since all events come from workflow forms. Care process and metrics are linked in the sense that metrics are defined on the events logged but there is no modeled relationship between the two. In our approach, the relationships between different aspects of care process monitoring are captured in a single integrated application meta-model.

6.2.5 Architectural Requirements

Developing monitoring applications requires leveraging a set of modern technologies which support monitoring and measuring the performance of care processes. Table 6-11 below depicts the comparison of our engineering approach against other approaches related to the related works.

The monitoring applications developed for all case studies, as well our approach, are designed to consume, process and generate events as they are happening from the event sources. Therefore, an event driven architecture is supported by all. SOA-Based CEP, RTLS-Based CEP

and BPM-Based CEP leverage a CEP engine to process and generate complex events. In the same manner, our application framework used CPME to process events in order to infer and update current states of the resources. BPM-Based Monitoring Dashboard only supports simple event processing by logging data provided by work flow forms. On the other hand, in our application framework, event integration and filtering rules are written and embedded within CPME.

Since all related approaches in addition to our approach are event driven based, it is a requirement to integrate with event sources in the enterprise. The number of event sources integrated with the monitoring applications developed in the related approaches depends on the architectural design of the application.

Table 6-11: Comparison with the Related Works in terms of Architectural Requirements

Criteria	**SOA- based CEP**	**RTLS-based CEP**	**BPM-based CEP**	**BPM-Based Monitoring Dashboard**	**Application Framework**
Event Driven Architecture	Yes, Complex Event Processing	Yes, Complex Event Processing	Yes, Complex Event Processing	Yes, simple event processing	Yes, CPME
Event Correlation and Filtering	Yes, Rules embedded within CEP Engine	Yes, Rules embedded within CEP engine	Yes, Rules embedded within CEP engine	Simple	Yes, CPME
Integrate with Event sources	Yes	Yes	Yes	Yes	Yes
Integrate with Business Process management Architecture (BPM)	Not Integrated	Not Integrated	Yes	Yes	Yes
Integrate with Performance management Architecture	Yes, near real-time	Yes, Integrated with data warehouse	Yes, near real-time	Yes	Yes
Integrate with Real-time Location Service (RTLS)	Not Integrated	Integrated, using RFID tags and RFID reader, and RFID middleware	Not Integrated	Not Integrated	Yes

The monitoring applications developed in the related approaches, as well as in our approach, have the capability to integrate with performance management architecture to provide measurement for performance metrics.

SOA-Based CEP and RTLS-Based CEP are not designed to integrate with BPM architecture. The only approaches that support integration with RTLS are RTLS-Based CEP and our application framework. In the former, a hard-coded RFID middleware is developed to filter RTLS events. It is worth noting that BPM-Based CEP is designed to support integration with different types of event sources. However, the RTLS is not integrated with the architecture.

6.2.6 Application Development Process and Tools

In this section we compare our CPMA with other related works; SOA-Based CEP, RTLS-Based CEP, BPM-Based CEP and BPM-Based Monitoring Dashboard, in terms of what tools are required during the application development of the monitoring application. Table 6-12 summarizes the comparison.

There exists a tool to test the monitoring application prior to integration with the real environment adopted by SOA-Based CEP and RTLS-Based CEP through developing a hard-coded event simulator. In the same manner but using different technology, our application framework has the capability to test the monitoring application using a test console.

CEP engine is the tool that supports writing and modeling event integration rules leveraged by the related approaches; SOA-Based CEP, RTLS-Based CEP and BPM-Based CEP. In our approach, the proposed CPME supports writing custom state handler rules for event integration.

Table 6-12: Comparison with the Related Works in Terms of Application Development Process

Criteria	SOA-Based CEP	RTLS-Based CEP	BPM-Based CEP	BPM-Based Monitoring Dashboard	Application Framework
Tool or Technique to test CPMA prior to enterprise integration	Event Simulator	Event Simulator	None	None	Test console
Tool or Technique for event processing and integration rules	CEP engine	CEP engine	CEP engine	None	Custom State handler rule
Tool or Technique to translate care process events to performance metrics	Data Warehouse Logging	Data Warehouse Logging	CEP output adapter to translate complex events in a form of KPI displayed in the dashboard	ORM Logging	Data Warehouse Logging
Tool for visualizing requirements	No	No	No	No	CPMA application model
Support agile development process	Not Mentioned	Not Mentioned	Not Mentioned	Not Mentioned	Behavior driven development based on integrated application model

Translation of events to performance metrics for SOA-Based CEP and RTLS-Based CEP and our application framework is accomplished by logging to an events fact table in a data warehouse. In addition, our application framework has support for logging the critical states to a states fact table. For BPM-Based CEP, there is a hard-coded adapter service developed in the CEP engine to translate the complex events to KPIs that are directly displayed in the dashboard. For BPM-based Monitoring Dashboard, events are logged to an ORM database which is less straightforward to report on.

None of the related approaches leverages a tool or technique to visualize the requirements. Similarly, in our approach application framework, there is no tool support adopted

to visualize the requirements for the stakeholders and the development team. Instead, we exploited the application model as a technique to define requirements in an integrated, detailed manner.

None of the related approaches discussed development methodology or agile test-driven approaches. Our approach supports systematic behavior-driven agile development based on an integrated, detailed application meta-model used to define requirements.

6.3. Result, Limitations and Assumptions

Our research adopts a design-oriented research methodology where the aim was to show that there is a gap in current approaches to engineering CPMAs and to show that there is potential for our application framework to address the gap. The major gap identified in this thesis is, in general, how difficult and time-consuming it is to engineer a care process monitoring application in a systematic and repeatable fashion. To address the gap, we have developed an application framework for CPMAs and built CPMA prototypes to show whether it is possible to provide evidence that our approach can help address the gap in less time, less effort and with less complexity. We have demonstrated in two specific care processes at two specific healthcare organizations how our approach can address the gap. Generally, applying our application framework towards the two specific cases showed a great potential where only about one month of configuration and testing were needed to implement a CPMA whereas similar applications had required one year of development by two to three developers.

In the community care CPMA for palliative care processes at Bruyère Hospital, the existing CPMA PAL-IS 1.0 was developed by a team of three developers (masters students

working on projects or thesis) as a custom-built web application and took approximately a year to build, but it was hard to maintain. For the new PAL-IS 2.0 CPMA, it was developed using our application framework (as described in our third case study in chapter 5.4) and the Quickforms service took a master student about four weeks to build. Personally, I had the opportunity to participate in fixing bugs in both applications in the latter stages of development after each CPMA was deployed to Bruyère Hospital. I found it much easier and it took less time and effort to fix bugs in the new PAL-IS 2.0 CPMA, compared to the old PAL-IS 1.0.

Similarly, in the hospital care CPMA for Cardiac patients at Osler Hospital, it took a team of three developers (master students working on thesis or project) about one year to build the initial CPMA (described in our initial case study in chapter 5.2) that did not leverage our full application framework. However, with a full application framework defined including an application model, SME-based CPME and Metrics Data Mart, it took one master student about four weeks to build the CPMA described in our second case study in Chapter 5.3. Personally, I was involved in the testing of both CPMAs and can attest that there was much less confusion and it took less time and effort to understand and debug the expected behavior of the second CPMA and configure its rules.

However, even though we have shown that our approach can help address the gap when it was applied against the two specific care process we studied, it is not proven that our approach will be viable for any healthcare organization and for any care process. In addition, it is not proven that IT staff at hospitals can repeat the steps of configuring and maintaining the monitoring application. More systematic experimentation is needed including:

1. Comprehensive clinical trials at a bigger selection of hospitals with bigger selection of different types of care processes
2. The use of typical IT staff to deploy and configure the monitoring application rather than researchers and without direct support from them. This requires better tool support, and better documentation of our application framework.

Although our application framework on how to build a care process monitoring application might be exploited by other IT staff and graduate students by reading our thesis, more work is needed to instrument and document the methodology and release a standardized CPME as an open source product. There is still ongoing research on the best implementation of the CPMA.

There are some factors that might impede adoption of CPMA by medical staff and the hospital which require more study. A cost benefit analysis of implementing the emerging technologies should be done. As well, a careful assessment should be done of the technical skills required to implement and use the CPMA using our application framework.

Theoretically, in order to make our approach viable for monitoring any care processes in any hospital or healthcare organizations, our approach assumes the following:

1. The desired care process will have a well-defined performance management model, in terms of goals and related metrics, and a well-defined care process model in terms of states and events. It is also necessary that events are available and can be acquired while the care process is taking place.

2. An appropriate EDA-SOA technology infrastructure exists or can be built either in the hospital or in the community and is able to provide all the events required to measure the performance of the care process as defined in the application model.
3. The complexity of integrating with enterprise architecture is manageable.

6.4. Chapter Summary

In this chapter, we have presented the results of evaluating our application framework against the three case studies developed in chapter 5 based on the criteria defined and discussed in section 3.3. The results showed that the engineering effort to engineer CPMAs in a systematic, reusable fashion, in the second and the third case studies when adopting our application framework required less time, less effort and less complexity.

In addition to above, we have presented the results of comparing our application framework to the related works described in section 2.4. The evaluation criteria defined in section 3.3 were used for the comparison.

Chapter 7. Conclusion and Future Work

In this chapter, we summarize the contributions of this thesis described in section 7.1. Then, in section 7.2, we discuss the future works that make our application framework more generic, model driven and applicable for any care process.

7.1. Conclusions

In this thesis, we have proposed an application framework for monitoring care processes enacted by a single hospital or multiple healthcare organizations in the community. The main objective of the framework is to facilitate and make it simpler to develop CPMAs for healthcare organizations, whether they have minimal IT infrastructure and require measuring and monitoring a few simple goals for a simple care, or they have a complex infrastructure and require measuring and monitoring every aspect of the process.

We have demonstrated, in our case studies, the viability of our application framework to help close the gap of engineering CPMA in less time, less effort with less complexity and requiring less sophisticated IT skills. Time and effort was reduced by an order of magnitude from over a year of effort to a few weeks of effort in both the CPMAs developed: one for hospital care and one for community care.

Here are the main research contributions for our thesis and their significance:

- **A state-based application meta-model of care process monitoring**

The application meta-model enables one to instantiate an application model of any CPMA by defining the elements of the monitoring applications; goals, performance metrics, care process states, events and event sources, as well as defining the relationship among those objects. The application meta-model bridges the EDA-SOA architecture in which care processes take place with the Data Warehouse – Performance Reporting architecture which measures the performance of reports, and enables health care domain experts, IT architecture architects, and CPMA developers to view and highlight the critical elements of the CPMA and their relationships in a single model.

- **A care process monitoring (CPM) architectural pattern**

The CPM architectural pattern uses a CPMA as a gateway from the EDA-SOA architecture or care processes to the Performance Management architecture for reporting. In particular, we developed a Metrics Data Mart with a predefined star schema and performance reporting dashboard that could be used to monitor wait and service times for any process. In addition, we showed that the essential event processing required for a CPMA was that of state inferencing.

- **A behavior driven development methodology based on our meta-model and architectural pattern**

The methodology provides guidance for the developers to build and implement CPMA based on an application model and scenarios to define the expected behavior. With the model and test scripts well-defined and grounded in a single consistent application model, development is largely simplified to the tasks of assembling and configuring the CPMA (event processing rules, and metrics queries) without any complex custom coding. Development time, effort, complexity and skills had an order of magnitude reduction.

In conclusion, our application framework is comprised of an architectural pattern that supports any of the architectural approaches used in the four categories of the related works we used as described in section 2.4 to compare and evaluate our approach. In addition, we defined a clear and a concise application model that none of the related works had and we defined an agile test driven methodology to assemble and configure CPMA that none of the related works mentioned.

7.2. Future Work

Although we have demonstrated the viability of our approach to improve the engineering of care process monitoring applications, better case studies are needed to fully validate it. More clinical trials with CPMAs for different types of processes at additional hospitals and community organizations would help further establish the generality of our application framework. More case studies are also needed to validate adoption and skills required by the developers using our framework.

In that regard, better tool and documentation support is needed so that developers and IT staff in health care organizations are able to use the framework to develop and maintain CPMAs on their own. In particular, tool support for building, viewing and analyzing the model and managing test scripts based on the model would be useful.

It would also be interesting to define an execution semantics for the meta-model and explore the creation of a CPMA engine that could interpret or execute application models in a way similar to how BPEL engines execute business processes. Alternatively a Model Driven Architecture (MDA) approach could be explored, as well as tool support to generate the CPMA from its application model.

REFERENCES

Accredidation Canada. (2013). Retrieved September 25, 2013, from Hospice Palliative and End-of-Life Services: http://www.accreditation.ca/accreditation-programs/qmentum/standards/hospice-palliative-and-end-of-life-services/

Aked, M. (2003, November 25). *Risk reduction with the RUP phase plan.* Retrieved September 25, 2013, from IBM: http://www.ibm.com/developerworks/rational/library/1826.html#N100E4

Alur, D., Malks, D., & Crupi, J. (2003). *Core J2EE Patterns Best Practices and Design Strategies* (Second ed.). Prentice Hall.

Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., et al. (Eds.). (2007, April 11). Web Services Business Process Execution Language Version 2.0 OASIS Standard. OASIS. Retrieved September 25, 2013, from http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf

Ammon, R., Emmersberger, C., Springer, F., & Wolff, C. (2008). Event- Driven Business Process Management and its Practical Application Tacking the Example of DHL. *1st International Workshop on Complex Event Processing for Future Internet.*

Ammon, v., Emmersberger, C., Greiner, T., Paschke, A., & Springer, F. W. (2008). Event-Driven Business Process Management. *International Conference on Distributed Event-Based Systems.*

Amyot, D. (2012). An architecture and a platform for real-time, location-based patient flow monitoring. Invited talk. *Third Kuwait Conference on e-Services and e-Systems.* Kuwait.

Amyot, D., & Mussbacher, G. (Eds.). (2012, 10). ITU-T, Recommendations Z.151 : User Requirements Notation (URN) - Language definition. (2.0). Geneva,, Switzerland. Retrieved September 25, 2013, from http://www.itu.int/rec/T-REC-Z.151/en

AndroMDA. (2012, July 03). Retrieved September 25, 2013, from http://www.andromda.org/index.html

Ao, Y., He, W., Xiao, X., & Lee, E. (2010). A Business Process Management Approach for RFID Enabled Supply Chain Management. *IEEE Conference on Emerging Technologies and Factory Automation* (pp. 1-7). Toulouse: IEEE.

Armellin, G., Betti, D., Casati, F., Chiasera, A., Martinez, G., & Stevovic, J. (2010). Privacy Preserving Event Driven Integration for Interoperating Social and Health Systems. In *Secure Data Management* (Vol. 6358, pp. 54-69). Springer Berlin Heidelberg.

Asif, Z., & Mandviwalla, M. (2005). Integrating the Supply Chain with RFID: A Technical and Business Analysis. *Communications of the Association for Information Systems , 15*, 393-427.

Azvine, B., Nauck, D., & Ho, C. (2003). Intelligent Business Analytics — A Tool to Build Decision-Support Systems for eBusinesses. *BT Technology Journal , 21* (4), 65-71.

Azvine, b., Zheng, C., & Nauck, D. (2005). Towards real-time business intelligence. *BT Technology Journal , 23* (3), 214-225.

Baarah, A., Mouttham, A., & Peyton, L. (2012). Architecture of an Event Processing Application for Monitoring Cardiac Patient Wait Times. *International Journal of Information Technology and Web Engineering (IJITWE) , 7* (1), 1-16.

Baarah, A., Mouttham, A., & Peyton, L. (2011). Improving Cardiac Patient Flow Based On Complex Event Processing. *2011 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT),* (pp. 1-6). Amman: IEEE.

Baffoe, S. (2013). A Generic BI Application for Real-time Monitoring of Care Processes. Published Master's Thesis. Retrieved September 25, 2013, from https://www.ruor.uottawa.ca/fr/handle/10393/24245

Baffoe, S., Baarah, A., & Peyton, L. (2013). Inferring State for Real-Time Monitroing of Care Processes. *5th International Workshop on Software Engineering in Health Care.* San Francisco.

Beck, K. (2003). *Test Driven Development: By Example.* Addison-Wesley Longman.

Beck, K., & Andress, C. (2004). *Extreme Programming Explained: Embrace Change* (2nd Edition ed.). Addison-Wesley Professional.

Bellenger, D., Pawlowski, O., & Westhuis, J. (2010). An Extensible Event Processing Architecture for RFID-Based Tracking and Tracing. *European Workshop on Smart Objects: Systems, Technologies and Applications* (pp. 1-7). Ciudad Real: IEEE.

Bhatia, A. (2013). *Transformation of a Web Application for Patient Monitoring into a Real-time Business Intelligence Portal.* Master's Report, University of Ottawa, Electrical Engineering and Computer Science (EECS), Ottawa.

Biju, S. (2008). Agile Software Development. *E- Learning , 5* (1), 97-102.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The Unified Modeling Language User Guide. Addison-Wesley Professional.

Boubbeta-Puig, J., Ortiz, G., & Medina-Bulo, I. (2011). An approach of early disease detection using CEP and SOA. *Third International Conference on Advanced Service Computing*, (pp. 143-148). Rome, Italy.

Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I. (2011). An Approach of Early Disease Detection using CEP and SOA. *The Third International Conferences on Advanced Service Computing* (p. 143 to 148). Rome: ThinkMind.

Boulos, M., & Berry, G. (2012). Real-time locating systems (RTLS) in healthcare: a condensed primer. *International Journal of Health Geographics , 11* (25).

Brown, A., Johnston, S., & Kelly, K. (2002). *Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications.* A Rational Software. IBM .

Buschmann, F., Henney, K., & Schimdt, D. (2007). *Pattern Oriented Software Architecture* (Vol. 5). John Wiley & Sons.

Care Process Models. (2013). Retrieved September 25, 2013, from Intermountain Healthcare: https://intermountainphysician.org/clinical/Pages/Care-Process-Models-(CPMs).aspx

Chaudhuri, S., & Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. *ACM SIGMOD Record , 26* (1), 65-74.

Chelimsky, D., Astels, D., Helmkamp, B., North, D., Dennis, Z., & Hellesoy, A. (2010). *The RSpec Book: Behaviour Driven Development with Rspec, Cucumber, and Friends.* Pragmatic Bookshelf.

Chieu, T. C., & Zeng, L. (2008). Real-time performance monitoring for an enterprise information management system. *IEEE International Conference on e-Business Engineering* (pp. 429-434). Xian: IEEE.

Cimellaro, G. P., Reinhorn, A. M., & Bruneau, M. (2011). Performance-based metamodel for healthcare facilities. *Earthquake Engineering & Structural Dynamics , 40* (11), 1197–1217.

Cohen, D., Lindvall, M., & Costa, P. (2004). An Introduction to Agile Methods. *Advances in Computers* , 1-66.

Cohen, M. (2009). *Succeeding with Agile: Software Development Using Scrum.* Pearson Education.

Consulting, E. (2006). Issues and Best Practices for the BPM and SOA Journey [White Paper]. Retrieved September 25, 2013, from http://www.waria.com/Documents/Issues_and_Best_Practices_for_the_BPM_and_SOA_Journey.pdf

De Giusti, M., Oviedo, N., & Lira, A. (2011). Extract, Transform and Load architecture for metadata collection. *6th International Symposium on Digital Libraries.* Porto Alegre.

Denis, T., Weyn, M., Williame, K., & Schrooyen, F. (2006). Real Time Location System using WiFi [White Paper]. Turnhout. Retrieved September 25, 2013, from http://www.productivet.com/docs-2/RTLS_Belgium_White_Paper.pdf

Dunkel, J. (2009). On Complex Event Processing for Sensor Networks. *International Symposium on Autonomous Decentralized Systems* (pp. 1-6). Athens: IEEE.

Dyck, W. V., Vertes, G., Palaniappan, M., Gassull, D., Jain, P., Schulthess, D., et al. (2012). Acute coronary syndrome: What is the cost-effectiveness of prevention, point-of-care technology and telemonitoring? *Health Policy and Technology , 1* (3), 173–177.

Ekahau. (2012). Retrieved September 25, 2013, from http://www.ekahau.com/

El Baz, N., Middel, B., van Dijk, J., Oosterhof, A., Boonstra, P., & Reijneveld, S. (2007). Are the outcomes of clinical pathways evidence-based? A. *Journal of Evaluation in Clinical Practice , 13* (6), 920–929.

Erdem, G., Geisler, &., & Flather, M. (2010). What Goes Into a Major Acute Coronary Syndrome Trial and What Will Future Trials Look Like? *Current Cardiology Reports , 12* (4), 348-355.

Ernst, J. (2002). *What is Metamodeling.* Retrieved September 25, 2013, from Infogrid: http://infogrid.org/trac/wiki/Reference/WhatIsMetaModeling

Eugster, P. T., Felber, P. A., & Guerraoui, R. K.-M. (2003). The Many Faces of Publish/Subscribe. *ACM Computing Surveys , 35* (2), 114-131.

Evelson, B. (2008). *Topic Overview: Business Intelligence.* Forrester.

Feng, X., & Le, T. (2009). Construction of B2B Electronic Commerce System Based on Apache Struts Framework. *International Conference on Services Science, Management and Engineering* (pp. 221-224). IEEE.

Ferris, F., Balfour, H., Bowen, K., Farley, J., Hardwick, M., Lamontagne, C., et al. (2002). *A Model to Guide Hospice Palliative Care: Based on National Principles and Norms of Practice.* Ottawa: Canadian Hospice Palliative Care Association (CHPCA).

Field, M., & Lohr, K. (Eds.). (1990). *Clinical practice guidelines: directions for a new program.* National Academies Press.

Forrester. (2008). *Enabling Dynamic Business Applications with BPM and SOA.* IBM.

Gardner, T., & Yusuf, L. (2006, March 14). *Explore model-driven development (MDD) and related approaches: A closer look at model-driven development and other industry initiatives.* Retrieved September 25, 2013, from http://www.ibm.com/developerworks/library/ar-mdd3/

Gattnar, E., Ekinci, O., & Detschew, V. (2011). A Novel Generic Clinical Reference Process Model for Event-Based Process Times Measurement. In *Business Information Systems Workshops* (Vol. 97). Springer Berlin Heidelberg.

Ghattas, J., Peleg, M., Soffer, P., & Denekamp, Y. (2010). Learning the Context of a Clinical Process. In *Business Process Management Workshops* (pp. 545-556). Springer Berlin Heidelberg.

Giordano, A. (2010). *Data Integration Blueprint and Modeling: Techniques for a Scalable and Sustainable Architecture.* IBM Press.

Giovagnoli, A., & Romano, D. (2004). Optimal experiments in the presence of a learning effect: a problem suggested by software production. *Statistical Methods & Applications , 13* (2), 227-239.

Grails. (2009). Retrieved September 25, 2013, from http://grails.org/

Gronli, T.-M., & Bygstad, B. (2012). A Successful Implementation of Service Oriented Architecture. *26th International Conference on Advanced Information Networking and Applications Workshops* (pp. 41 - 46). IEEE.

Guillemette, M. G., Fontaine, I., & Caron, C. (2008). Hybrid RFID-GPS Real-Time Location System for Human Resources: Development, Impacts and Perspectives. *41st Hawaii International Conference on System Sciences* (pp. 406-415). Washington: IEEE.

Gunter, T., Terry, N., & Powell, J. (2005). The Emergence of National Electronic Health Record Architectures in the United States and Australia: Models, Costs, and Questions. *Journal of Medical Internet Research* .

Habib, J. L. (2010). EHRs, Meaningful Use, and a Model EMR. *Drug Benefit Trends , 22* (4), 99-101.

Hailpern, B. T., & Tarr, P. (2006). Model-driven development: The good, the bad, and the ugly. *IBM Systems Journal , 45* (3), 451- 461.

Hammond, S., & Umphress, D. (2012). Test Driven Development: The State of the Practice. *50th Annual Southeast Regional Conference* (pp. 158-163). Tuscaloosa: ACM.

Hasselbring, W., & Pedersen, S. (2005). Metamodelling of Domain-Specific Standards for Semantic Interoperability. *Third Biennial Conference on Professional Knowledge Management. 3782*, pp. 557–559. Kaiserslautern: Springer Berlin Heidelberg.

Haugen, D., Nauck, F., & Caraceni, A. (2011). The core team and the extended team. In *Oxford Textbook of Palliative Medicine.* Oxford University Press.

Hemani, A., & Shamsi, J. (2010). Foundations of a generic design for complex event processing. *International Conference on Information and Emerging Technologies* (pp. 1 - 6). Karachi: IEEE.

Hevner, A., March, S., Park, J., & Ram, S. (2004). Design science in information systems research. *Journal of MIS Quarterl , 28* (1), 75-105.

Inmon, W. H. (2005). *Building the data warehouse* (4th ed.). Indianapolis: Wiley Publishers.

Janiesch, C., Matzner, M., & Müller, O. (2012). Beyond process monitoring: a proof-of-concept of event-driven business activity management. *Business Process Management Journal , 18* (4), 625 - 643.

Janzen, D., & Saiedian, H. (2005). Test-Driven Development:Concepts, Taxonomyand Future Direction. *Computer , 38* (9), 43 - 50.

Jarke, M., Lenzerini, M., Vassiliou, Y., & Vassiliadis, P. (2010). *Fundamentals of Data Warehouses* (Second ed.). Springer.

Jeffries, R., & Melnik, G. (2007). TDD: The Art of Fearless Programming. *IEEE Software , 24* (3), 24-30.

Kalnins, A., & Vitolins, V. (2006). Use of UML and Model Transformations for Workflow Process Definitions. *7th International Baltic Conference on Databases and Information Systems* , (pp. 3-14). Vilnius.

Kang, J. G., & Han, K. H. (2008). A Business Activity Monitoring System Supporting Real-Time Business Performance Management. *Third International Conference on Convergence and Hybrid Information Technology. 1*, pp. 473- 478. Busan: IEEE.

Kaplan, R. S., & Norton, D. P. (1996). *The Balanced Scorecard: Translating Strategy into Action* (1st ed.). Harvard Business Press.

Kemper, H.-G., Rausch, P., & Baars, H. (2013). Business Intelligence and Performance Management: Introduction. In *Business Intelligence and Performance Management* (pp. 3-10). Springer London.

Kierkegaard, P. (2011). Electronic Health Record: Wiring Europe's healthcare. *computer law & security review , 27*, 503-515.

Kimball, R., & Ross, M. (2002). Health Care. In *The Data Warehouse Toolkit* (2nd ed., pp. 255-275). Wiley Computer Publishing.

Kimball, R., & Ross, M. (2008). *The Data Warehouse Lifecycle Toolkit.* USA: Wiley.

Ko, R., Lee, S., & Lee, E. (2009). Business process management (BPM) standards: a survey. *Business Process Management , 15* (5), 744-791.

Kronz, A. (2006). Managing of Process Key Performance Indicators as Part of the ARIS Methodology. In *Corporate Performance Management* (pp. 31-44). Springer Berlin Heidelberg.

Kruchten, P. (2004). *The Rational Unified Process: An Introduction.* Addison-Wesley Professional.

Kuziemsky, C., Liu, X., & Peyton, P. (2010). Leveraging Goal Models and Performance Indicators to Assess Health Care Information Systems. *7th International Conference on the Quality of Information and Communications Technology* (pp. 222-227). Oporto: IEEE.

Kuziemsky, C., Williams, J., & Weber-Jahnke, J. (2011). Towards electronic health record support for collaborative processes. *3rd Workshop on Software Engineering in Health Care* (pp. 32-39). ACM.

Landauber, M., & Genaid, A. (2012). Connecting User Stories and code for test development. *Third International Workshop on Recommendation Systems for Software Engineering* (pp. 33 - 37). Zurich: IEEE.

Larman, C., & Basili, V. (2003). Iterative and incremental developments. a brief history. *Computer , 36* (6), 47 - 56.

Leau, Y., Loo, W., Tham, W., & Tan, S. (2012). Software Development Life Cycle AGILE vs Traditional Approaches. *International Conference on Information and Network Technology*, *37*, pp. 162-167.

Leavitt, N. (2009). Complex-Event Processing Poised for Growth. *Computer , 42* (4), 17-20.

Leggat, G. S., Bartram, T., & Stanton, P. P. (2011). High performance work systems: the gap between policy and practice in health care. *Journal of Health Organization and Management , 25* (3), 281-297.

Levina, O., & Stantchev, V. (2009). Realizing Event-Driven SOA. *Fourth International Conference on Internet and Web Applications and Services* (pp. 37-42). Venice: IEEE.

Leymann, F., Roller, D., & Schmidt, M.-t. T. (2002). Web services and business process management. *IBM Systems Journal , 41* (2), 198 - 211 .

Li, J.-M., Ma, G.-S., Feng, G., & Ma, Y.-Q. (2006). Research on Web Application of Struts Framework Based on MVC Pattern. *International Workshop on Advanced Web and Network Technologies, and Applications. 3842*, pp. 1029-1032. Harbin: Springer Berlin Heidelberg.

Li, W., Liu, K., Yang, H., & Yu, C. (2013, May 28). Integrated clinical pathway management for medical quality improvement – based on a semiotically inspired systems architecture. *European Journal of Information Systems advance* .

Liu, X. (2010). *A Requirement Engineering Framework for Assessing Health Care Information Systems (Master Thesis).* Ottawa, Ontario, Canada: University of Ottawa.

Luckham, D. (2002). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Boston, Massachusetts, USA: Addison-Wesley.

Luo, R. (2012). *Patient Flow Monitoring Application in the Osler Project.* Master's Report, University of Ottawa, Electrical Engineering and Comouter Science (EECS), Ottawa.

Ma, K. J. (2005). Web Services: What's Real and What's Not? *IT Professional , 7* (2), 14- 21.

Mandi, R. (2008). Empowering the business to sense and respond: Delivering Business Event Processing with IBM WebSphere Business Events [White Paper]. Retrieved September 25, 2013, from ftp://ftp.software.ibm.com/software/integration/wbe/5565_Empowering-the-Business-US-white-paper.pdf

Martin, R. (2003). *Agile Software Development: Principles, Patterns, and Practices.* NJ, USA: Prentice Hall.

Mawilmada, P. K. (2011). Impact of a data warehouse model for improved decision-making process in healthcare. Published Master's Thesis. Australia: Queensland University of Technology. Retrieved September 25, 2013, from http://eprints.qut.edu.au/47532/1/Pubudika_Mawilmada_Thesis.pdf

Mellor, S. J., Scott, K., Uhl, A., & Weise, D. (2004). *MDA Distilled: Principles of Model-Driven Architecture.* Boston: Addison-Wesley.

Michelson, B. (2006). *Event-Driven Architecture Overview Event-Driven SOA Is Just Part of the EDA Story.* Boston: Patricia Seybold Group.

Middleton, G., Peyton, L., Kuziemsky, C., & Eze, B. (2009, August 25-27). A framework for continuous compliance monitoring of eHealth Processes. *World Congress on Privacy, Security, Trust and the Management of e-Business.* , pp. 152-160.

Middleton, G., Peyton, L., Kuziemsky, C., & Eze, B. (2009). A Framework for Continuous Compliance Monitoring of eHealth Processes. *World Congress on Privacy, Security, Trust and the Management of e-Business* (pp. 152- 160). Saint John: IEEE.

Mouttham, A. (2013). A Framework for Real-Time Analytics and Decision Support in Patient Flow Management. Invited Talk. *IBM IMPACT 2013.* Las Vegas.

Mouttham, A., Peyton, L., & Kuziemsky, C. (2011). Leveraging Performance Analytics to Improve Integration of Care. *3rd Workshop on Software Engineering in Health Care* (pp. 56-62). Honolulu: ACM.

Mouttham, A., Peyton, L., & Kuziemsky, C. (2011). Leveraging Performance Analytics to Improve Integration of Care. *SEHC,11* (pp. 56-62). Waikiki, Honolulu: ACM.

Mouttham, A., Peyton, L., Eze, B., & El Saddik, A. (2009). Event-Driven Data Integration for Personal Health Monitoring. *Journal of Emerging Technologies in Web Intelligence , 1* (2), 110-118.

Muhlen, M. z., & Shapiro, R. (2010). Business Process Analytics. In *Handbook on Business Process Management 2* (pp. 137-157). Springer Berlin Heidelberg.

Mulik, S., Ajgaonkar, S., & Sharma, K. (2008). Where Do You Want to Go in Your SOA Adoption Journey? *IT Professional , 10* (3), 36 - 39.

Neal, H. (2008, November 14). *EHR vs EMR – What's the Difference?* Retrieved September 25, 2013, from The Profitable Practice: http://profitable-practice.softwareadvice.com/ehr-vs-emr-whats-the-difference/

Niblett, P., & Graham, S. (2005). Events and service-oriented architecture: The OASIS Web Services Notification Specifications. *IBM Systems Journal , 44* (4), 869-886.

NIH, N. I. (2006). Electronic Health Records Overview [Report]. Virginia: The MITRE Corporation. Retrieved September 25, 2013, from http://www.himss.org/files/HIMSSorg/content/files/Code%20180%20MITRE%20Key%20Components%20of%20an%20EHR.pdf

Nikiforova, O., & Nikulsins, V. (2008). Integration of MDA Framework into the Model of Traditional Software Development. *Eighth International Baltic Conference on Databases and Information Systems V. 187*, pp. 229-239. Tallinn: IOS Press.

North, D. (2006). Introducing BDD. *Better Software Magazine* .

OMG. (2012). *MDA.* Retrieved September 25, 2013, from OMG: http://www.omg.org/mda/

OWen, M., & Raj, J. (2003). *BPMN and Business Process Management Introduction to the New Business Process Modeling Standard [White Paper]*. Popkin Software.

Peffers, K., Tuunanen, T., Gengler, C., Rossi, M., Hui, W., Virtanen, V., et al. (2006). The Design Science Research Process: A Model For Producing And Presenting Infor-mation Systems Research. *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology*, (pp. 83-106).

Pottebaum, J., Artikis, A., Marterer, R., Paliouras, G., & Koch, R. (2011). Event Definition for the Application of Event Processing to Intelligent Resource Management. *International Conference on Information Systems for Crisis Response and Management.* Lisbon.

Pourshahid, A., Amyot, D., Chen, P., Weiss, M., & Forster, A. J. (2007). Business Process Monitoring and Alignment: An Approach Based on the User Requirements Notation and Business Intelligence Tools. *10th workshop of requirement engineering*, (pp. 80–91). Toronto.

Pourshahid, A., Amyot, D., Peyton, L. G., Chen, P., Weiss, M., & Forster, A. (2009). Business process management with the user requirements notation. *Electronic Commerce Research , 9* (4), 269-316.

Preuveneers, D., Yasar, A.-U.-H., & Berbers, Y. (2008). Architectural Styles for Opportunistic Mobile Communication: Requirements and Design Patterns. *International Conference on Mobile Technology, Applications, and Systems.* Ilan: ACM.

Rabhi, F., Yu, H., Dabous, F. T., & Wu, S. Y. (2007). A service-oriented architecture for financial business processes: A case study in trading strategy simulation. *Information Systems and e-Business Management , 5* (2), 185-200.

Raghupathi, W., & Umar, A. (2008). Exploring a model-driven architecture (MDA) approach to health care information systems development. *International Journal of Medical Informatics , 77*, 305-314.

Rational. (2003). Rational Unified Process Best Practices for Software Development Teams [White Paper]. Retrieved September 25, 2013, from http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Reichert, M., & Lenz, R. (2007). IT support for healthcare processes – premises, challenges, perspectives. *Data & Knowledge Engineering , 61* (1), 39–58.

Roy, J.-F., Kealey, J., & Amyot, D. (2006). Towards Integrated Tool Support for the User Requirements Notation. *5th International Workshop on System Analysis and Modeling: Language Profiles. 4320*, pp. 198-215. Kaiserslautern: Springer Berlin Heidelberg.

Sacks, M. (2012). Web Testing Practices. In *Pro Website Development and Operations* (pp. 27-43). Apress.

Saxena, R., & Srinivasan, A. (2013). Business Intelligence. In *Business Analytics A Practitioner's Guide* (Vol. 186, pp. 85-99). Springer New York.

Scacchi, W. (2002). Process Models in Software Engineering. *Encyclopedia of Software Engineering* .

Schlegel, T., Vidackovic, K., Dusch, S., & Seiger, R. (2012). Management of interactive business processes in decentralized service infrastructures through v. *Journal of King Saud University –Computer and Information Sciences , 24*, 137-144.

Schmidt, D. C. (2006). Model-Driven Engineering. *IEEE Computer , 39* (2), 25-31.

Schmidt, D. C., Gokhale, A., & Natarajan, B. (2004). Leveraging Application Frameworks. *ACM Queue , 2* (5), 66-75.

Schrooyen, F., Baert, I., Truijen, S., Pieters, L., Denis, T., Koen, W., et al. (2006). Real time location system over WiFi in a healthcare environment. *Journal on Information Technology in Healthcare , 4* (6).

Shahin, A., & Mahbod, M. (2007). Prioritization of key performance indicators: An integration of analytical hierarchy process and goal setting. *International Journal of Productivity and Performance Management , 56* (3), 226 - 240.

Singh, R., & Bakshi, A. (2013). Need of Agile Development. *International Journal of Recent Technology and Engineering , 2* (1), 59-61.

Smith, G., & Ledbrook, P. (2009). *Grails in Action.* Manning Publications.

Soeken, M., Wille, R., & Drechsler, R. (2012). Assisted Behavior Driven Development Using Natural Language Processing. *50th International Conference on Objects, Models, Components, Patterns. 7304*, pp. 269-287. Prague: Springer Berlin Heidelberg.

Solis, C., & Wang, X. (2011). A Study of the Characteristics of Behaviour Driven Development. *37th EUROMICRO Conference on Software Engineering and Advanced Applications* (pp. 383-387). Oulu: IEEE.

Sommerville, I. (2011). *Software Engineering 9.* Addison-Wesley.

Son, B. K., Lee, J. H., Park, K. L.-G.-C., & Kim, S. D. (2007). An Efficient Method to Create Business Level Events Using Complex Event Processing Based on RFID Standards. *5th IFIP WG 10.2 International Workshop. 4761* , pp. 1-10. Santorini Island: Springer.

Sun, M., Rahmaniheris, M., Kim, C., Sha, L., Berlin, R., & Goldman, J. M. (2012). *Towards a Systematic Software Architecture for Acute Care Support.* University of Illinois at Urbana-Champaign, Department of Computer Science. IDEALS.

Tchemeube, R. B. (2013, September 25). Location-Aware Business Process Management for Real-time Monitoring of Patient Care Processes. Unpublished Master Thesis. Ottawa, Ontario, Canada. Retrieved from http://www.ruor.uottawa.ca/en/bitstream/handle/10393/24336/Bougueng_Tchemeube_Renaud_2013_thesis.pdf?sequence=3

Tegegne, A., & Peyton, L. (2013). Application framework support for process-oriented software development. *Internation Journal of Electrical Business , 10* (3), 232-253.

Tegegne, A., & Peyton, L. (2011). Model-Based Engineering of a Managed Process Application Framework. *5th International Conference on E-Technologies: Transformation in a Connected World. 78*, pp. 173-188. Les Diablerets: Springer Berlin Heidelberg.

Tözmal, R. R. (2006, January). *Model Driven Architecture- Test Methods and Tools.* Retrieved September 25, 2013, from Master Thesis: http://www.bth.se/fou/cuppsats.nsf/all/3e3f43e707ebbb36c125710d004fd0e5/$file/MasterThesis-MDA.pdf

Vaidehi, V., Bhargavi, R., Ganapathy, K., & Hemalatha, C. S. (2012). Multi-sensor based in-home health monitoring using Complex Event Processing. *International Conference on Recent Trends In Information Technology* (pp. 570 - 575). Chennai: IEEE.

Vaishnavi, V., & Kuechler, B. (Eds.). (2012, November 11). Design Science Research in Information Systems. Association for Information Systems. Retrieved September 25, 2013, from http://www.desrist.org/design-research-in-information-systems/

Van der Aalst, W., ter Hofstede, A., & Weske, M. (2003). Business Process Management: A Survey. *International Conference on Business Process Management. 2678*, pp. 1-12. Eindhoven: Springer Berlin Heidelberg.

Vanhaecht, K., Panella, M., Zelm, R. v., & Sermeus, W. (2010). An overview on the history and concept of care pathways as complex interventions. *International Journal of Care Pathways , 14* (3), 117-123.

W3C. (2012). *WEB OF SERVICES*. Retrieved September 25, 2013, from http://www.w3.org/standards/webofservices/

Wang, F., Liu, S., & Liu, P. (2009). Complex RFID event processing. *VLDB , 18* (4), 913-931.

Wang, F., Liu, S., Liu, P., & Bai, Y. (2006). Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams. *10th International Conference on Extending Database Technology. 3896* , pp. 588-607. Munich: Springer.

Wang, G., & Jin, G. (2008). Research and Design of RFID Data Processing Model Based on Complex Event Processing. *International Conference on Computer Science and Software Engineering. 5*, pp. 1396-1399. Wuhan: IEEE.

Weiss, M., & Amyot, D. (2005). Business Process Modeling with URN. *International Journal of E- Business Research , 1* (3), 63-90.

Weske, M. (2012). *Business Process Management: Concepts, Languages, Architectures.* Springer.

Wetzstein, B., Ma, Z., & Leymann, F. (2008). Towards Measuring Key Performance Indicators of Semantic Business Processes. *11th International Conference on Business Information Systems. 7*, pp. 227–238. Innsbruck: Springer Berlin Heidelberg.

White, S. (2006). Process Modeling Notations and Workflow Patterns [White Paper]. Retrieved September 25, 2013, from http://www.omg.org/bp-corner/bp-files/Process_Modeling_Notations.pdf

Winter, A., Brigl, B., & Wendt, T. (2003). Modeling hospital information systems (part 1): the revised three-layer graph-based meta model 3LGM2. *Journal of Mehods of Information Medicine* , 544-551.

Wu, E., Diao, Y., & Rizvi, S. (2006). High-Performance Complex Event Processing over Streams. *ACM SIGMOD international conference on Management of data* (pp. 407-418). Chicago: ACM.

Xiang, Z. S., Chen, J., Wang, H., Huang, J., & Gao, X. (2004). A wireless LAN-based indoor positioning technology. *IBM Research and Development , 48* (5.6), 617 - 626 .

Yao, W., Chu, C., Li, Z., & Mullen, T. (2008). Levarging Complex Event Pprocessing for RFID Applications: A Case Study in Hospitals. *The 39th National Conference of Decision Sciences Institues* (pp. 4341-4346). Baltimore: The E-print Network.

Yao, W., Chu, C.-H., & Li, Z. (2011). Leveraging complex event processing for smart hospitals using RFID. *Journal of Network and Computer Applications , 34* (3), 799–810.

Yina, W. (2010). Application of EHR in Health Care . *Second International Conference on Multimedia and Information Technology* (pp. 60 - 63). Kaifeng: IEEE.

Zang, C., Fan, Y., & Liu, R. (2008). Architecture, implementation and application of complex event processing in enterprise information systems based on RFID. *Information Systems Frontiers , 10* (5), 543-553.

Zhang, J., Lu, X., Nie, H., Huang, Z., & Van der Aalst, W. (2009). Radiology information system: a workflow-based approach. *International Journal of Computer Assisted Radiology and Surgery , 4* (5), 509-516.

Zhu, Q., Nie, H., Lu, X., & Duan, H. (2010). Radiology Workflow-Based Monitoring Dashboard in a Heterogeneous Environment. *3rd International Conference on Biomedical Engineering and Informatics. 6*, pp. 2494- 2498. Yantai: IEEE.

APPENDIX

Appendix A: Cardiac Care (ACS) CPMA Application Model

-CareProcess: Clinical Pathway (ACS)

Description: "This clinical pathway describes the flow of ACS for cardiac patients in the hospital from when the patients enter ED to when the procedure is performed until when the patients are discharged"

Goals: Wait Time, Service Time, Throughput

Resources: Patient, Physician, Nurse, Transporter, Housekeeping, Room

States: Triaged, WAIT_FOR_PHY_INIT_ASSESS, IN_PHYS_INT_ASSES, IN_BED_ED, WAIT_FOR_ORDERS_EXECUTION, WAIT_FOR_PHY_RE_ASSESS, IN_PHYS_RE_ASSES, WAIT_FOR_BED_CW, WAIT_FOR_TRANSPORT_CW, IN_TRANSPORT_CW, IN_BED_CW, WAIT_FOR_PROCEDURES, WAIT_FOR_TRANSPORT_CCL, IN_TRANSPORT_CCL, IN_BED_CCL, IN_PROCEDURE_ANGIOGRAM, IN_PROCEDURE_PCI, IN_CONULTATION3, WAIT_FOR_DISCHARGE, DISCHARGED_COMPLETED, BED_ASSIGNED, WAIT_FOR_BED_CLEANUP, IN_CLEANUP, BED_AVAILABLE.

GOALS

-Goal: Wait Time

Description: "The hospital aims to reduce the current wait times they are experiencing in different units and keep the wait time in acceptable target range set by the hospital"

Metrics: Average Wait_Time for Initial Assesement, Average Wait_Time for Tests, Average Wait_Time for Reassessment, Average Wait_Time for Bed, Average Wait_Time for Transport from ED to CW, Average Wait_Time for Procedures in CCL, Average Wait_Time for Transport from CW to CCL, Average Wait_Time for Discharge, Average Wait_Time for Bed Cleanup.

-Goal: Service Time

Description: The hospital aims to measure the current service time in real time"

Metrics: Average Time in Initial Assessment, Average Time in Reassessment, Average Time in Transport from ED to CW, Average Time in Transport from CW to CCL, Average Time in Procedure Angiogram, Average Time in Procedure PCI, Average Time in Consultation 3

-Goal: Throughput

Description: "The hospital aims to measure the arrivals to the hospital, admissions to the Cardiac Ward, and monitor the discharge in real time in order to increase the throughput"

Metrics: Average Arrivals by Hour of the Day, Average Arrivals by Day of the Week, Average Admissions by Hour of the Day, Average Admissions by Day of the Week, Average Discharge by Hour of the Day, Average Discharge by Day of the Week

METRICS

- Metric: Average Wait_Time for Initial Assessment

Description: "This metric measures the average time patients wait for the initial assessment with the physician once he get admitted to ED"

Computation: AVERAGE STATE: WAIT_FOR_PHY_INIT_ASSESS. duration Over Period of Time

State: WAIT_FOR_PHY_INIT_ASSESS

Events:

Target: 30 minutes

Alert: OVERDUE, NEAR_OVERDUE

-Metric: Average Wait_Time for Tests

Description: "This metric measures the average time patients wait for orders to be executed after the physician request for tests over a period of time"

Computation: AVERAGE STATE: WAIT_FOR_TESTS. duration Over Period of Time

State: WAIT_FOR_TESTS

Events:

Target: 30 minutes

Alert: OVERDUE, NEAR_OVERDUE

-Metric: Average Wait_Time for Reassessment

Description: "This metric measures the average time patients wait for reassessment process in the second consultation in ED after all the orders executions have been completed over a period of time"

Computation: AVERAGE STATE: WAIT_FOR_PHY_RE_ASSESS. duration Over Period of Time

States: WAIT_FOR_PHY_RE_ASSESS

Events:
Target: 30 minutes

Alert: OVERDUE, NEAR_OVERDUE

-Metric: Average Wait_Time for Bed
Description: "This metric measures the average time patients wait for bed in Cardiology Ward over period of time"
Computation: AVERAGE STATE: WAIT_FOR_BED_CW. duration Over Period of Time
State: WAIT_FOR_BED_CW
Events:
Target: 480 minutes

Alert: OVERDUE, NEAR_OVERDUE

-Metric: Average Wait_Time for Transport from ED to CW
Description: "This metric measures the average time patients wait for transportation from ED to Cardiology Ward when they get admitted to CW over a period of time"
Computation: AVERAGE STATE: WAIT_FOR_TRANSPORT_CW. duration Over Period of Time
State: WAIT_FOR_TRANSPORT_CW
Events:
Target: 15 minutes

Alert: OVERDUE, NEAR_OVERDUE

-Metric: Average Wait_Time for Transport from CW to CCL
Description: "This metric measures the average time patients wait for transportation from Cardiology Ward to Cardiac Cath lab for procedures over a period of time"
Computation: AVERAGE STATE: WAIT_FOR_TRANSPORT_CCL. duration Over Period of Time
State: WAIT_FOR_TRANSPORT_CCL
Events:
Target: 15 minutes

Alert: OVERDUE, NEAR_OVERDUE

-Metric: Average Wait_Time for Procedures in CCL
Description: "This metric measures the average time patients wait for procedures to be performed in Cardiac Cath Lab over period of time"
Computation: AVERAGE STATE: WAIT_FOR_PROCEDURES. duration Over Period of Time
State: WAIT_FOR_PROCEDURES
Events:
Target: N/A
Alert:

-Metric: Average Wait_Time for Discharge
Description: "This metric measures the average time the patients wait for discharge from Cardiology Ward over a period of time"
Computation: AVERAGE STATE: WAIT_FOR_DISCHARGE. duration Over Period of Time
State: WAIT_FOR_DISCHARGE
Events:
Target: N/A
Alert:

-Metric: Average Wait_Time for Bed Cleanup
Description: "This metric measures the average wait times for beds to be cleaned up over a period of time"
Computation: AVERAGE STATE: WAIT_FOR_BED_CLEANUP. duration Over Period of Time
State: WAIT_FOR_BED_CLEANUP
Events:
Target: 30 minutes

Alert: OVERDUE, NEAR_OVERDUE

- Metric: Average Time in Initial Assessment
Description: "This metric measures the average time patients spend in the initial assessment with the physician in ED"
Computation: AVERAGE STATE: IN_PHYS_INT_ASSES. duration Over Period of Time
State: IN_PHYS_INT_ASSES
Events:
Target: N/A
Alert:

- Metric: Average Time in Reassessment
Description: "This metric measures the average time the patients spend in the reassessment process with the physician in ED after all their orders execution have been completed"
Computation: AVERAGE STATE: IN_PHYS_RE_ASSES. duration Over Period of Time
State: IN_PHYS_RE_ASSES
Events:
Target: N/A
Alert:

- Metric: Average Time in Transport from ED to CW
Description: "This metric measures the average time the patients spend in the transportation process from ED to Cardiology Ward"

Computation: AVERAGE STATE: IN_TRANSPORT_CW. duration Over Period of Time
State: IN_TRANSPORT_CW
Events:
Target: 15 minutes

Alert: OVERDUE, NEAR_OVERDUE

- Metric: Average Time in Transport from CW to CCL
Description: "This metric measures the average time the patients spend in the transportation process from Cardiology Ward to Cardiac Cath Lab"
Computation: AVERAGE STATE: IN_TRANSPORT_CCL. duration Over Period of Time
State: IN_TRANSPORT_CCL
Events:
Target: 15 minutes

Alert: OVERDUE, NEAR_OVERDUE

- Metric: Average Time in Procedure Angiogram
Description: "This metric measures the average time the patients spend in the procedure Angiogram in Cardiac Cath lab"
Computation: AVERAGE STATE: IN_PROCEDURE_ANGIOGRAM. duration Over Period of Time
State: IN_PROCEDURE_ANGIOGRAM
Events:
Target: 45 minutes

Alert: OVERDUE, NEAR_OVERDUE

- Metric: Average Time in Procedure PCI
Description: "This metric measures the average time the patients spend in the procedure called PCI in Cardiac Cath Lab"
Computation: AVERAGE of IN_PROCEDURE_PCI. duration Over Period of Time
State: IN_PROCEDURE_PCI
Events:
Target: 90 minutes

Alert: OVERDUE, NEAR_OVERDUE

- Metric: Average Time in Consultation 3
Description: "This metric measures the average time the patients spend with the physician in the last consultation"
Computation: AVERAGE of STATE: IN_CONULTATION3. duration Over Period of Time
State: IN_CONULTATION3
Events:
Target: N/A
Alert:

- Metric: Average Arrivals by Hour of the Day
Description: "This metric measures the average arrivals of patients by hour of the day"
Computation: AVERAGE COUNT EVENT: TriageScore each hour of day
State:
Events: TriageScore
Target: N/A
Alert:

- Metric: Average Arrivals by Day of the Week
Description: "This metric measures the average arrivals of patients by day of the week"
Computation: AVERAGE COUNT EVENT: TriageScore each day of the week
State:
Events: TriageScore
Target: N/A
Alert:

- Metric: Average Admissions by Hour of the Day
Description: "This metric measures the average admissions of patients by hour of the day"
Computation: AVERAGE COUNT EVENT: PatientAdmittedWithBed each hour of day
State:
Events: PatientAdmittedWithBed
Target: N/A
Alert:

- Metric: Average Admissions by Day of the Week
Description: "This metric measures the average admissions of patients by day of the week"
Computation: AVERAGE COUNT EVENT: PatientAdmittedWithBed each day of the week
Events: PatientAdmittedWithBed
Target: N/A
Alert:

- Metric: Average Discharge by Hour of the Day
Description: "This metric measures the average discharge of patients by hour of the day"
Computation: AVERAGE COUNT EVENT: DischargedCompleted each hour of day
State:
Events: DischargeRequest

Target: N/A
Alert:

- Metric: Average Discharge by Day of the Week
Description: "This metric measures the average discharge of patients by day of the week"
Computation: AVERAGE COUNT EVENT: DischargedCompleted each day of the week
State:
Events: DischargeRequest
Target: N/A
Alert:

ALERTS

-Alert: OVERDUE
Description: "This type of alert is triggered whenever the measure of the metric is greater than or equals the target"
Message: "UNACCEPTABLE. The Metric is Overdue**"**

-Alert: NEAR_OVERDUE
Description: "This type of alert is triggered whenever the measure of the metric greater than 66% of target and less than target "
Message: "WARNING. The Metric is Near Overdue**"**

RESOURCES

-Resource: Patient
Description: "This is the Cardiac patient, who is been given a care in the hospital since he/she arrives ED until he/she is discharged"
ResourceAttributes: patientID
States: Triaged, WAIT_FOR_PHY_INIT_ASSESS, IN_PHYS_INT_ASSES, IN_BED_ED, WAIT_FOR_ORDERS_EXECUTION, WAIT_FOR_PHY_RE_ASSESS, IN_PHYS_RE_ASSES, WAIT_FOR_BED_CW, WAIT_FOR_TRANSPORT_CW, IN_TRANSPORT_CW, IN_BED_CW, WAIT_FOR_PROCEDURES, WAIT_FOR_TRANSPORT_CCL, IN_TRANSPORT_CCL, IN_BED_CCL, IN_PROCEDURE_ANGIOGRAM, IN_PROCEDURE_PCI, IN_CONULTATION3, WAIT_FOR_DISCHARGE, DISCHARGED

-Resource: Physician
Description: "The resource is in charge of providing consultations, diagnosis, ordering test, ordering bed for the cardiac patient in the clinical pathway, as well, responsible for discharging the cardiac patient"
ResourceAttributes: physicianID
States:

-Resource: Nurse
Description: "The resource is responsible for triaging the cardiac patients and taking blood samples, as well as ordering fortransport"
Resource Attributes: nurseID
States:

-Resource: Transport
Description: "This resource is responsible to transport the cardiac patients to Cardiology Ward once the patient get admitted and he is responsible for transporting the patient to Cardiac Cath lab and back again to Cardiology Ward"
ResourceAttributes: transporterID
States:

-Resource: Housekeeping
Description: "The housekeeping is responsible for cleaning the beds in the CW after a patient is discharged"
ResourceAttributes: houseKeepingID
States:

-Resource: Room
Description: "This resource is monitored in order to increase throughput by informing the manager about an empty bed to be cleaned up once the patient has been discharged"
ResourceAttributes: roomID, unitID
States: BED_ASSIGNED, WAIT_FOR_BED_CLEANUP, IN_CLEANUP, BED_AVAILABLE.

STATES

-State: Triaged
Description: "This state indicates that the patient is in triaged"
StateAttributes: patientID, providerID, startTime, endTime

-State: WAIT_FOR_PHY_INIT_ASSESS
Description: "This state indicates that the patient is waiting for the initial assessment with the physician in ED assessment room"
StateAttributes: patientID, unitID, startTime, endTime

-State: IN_PHYS_INT_ASSES
Description: "This state indicates that the patient is in the initial assessment phase with the physician in ED assessment room"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: IN_BED_ED

Description: "This state indicates that the patient is remaining in bed after the initial assessment"
StateAttributes: patientID, roomID, unitID, startTime, endTime timestamp

-State: WAIT_FOR_ORDERS_EXECUTION
Description: "This state indicates that the patient is waiting for the orders to be executed by the nurse or the lab"
StateAttributes: patientID, startTime: startTime, endTime

-State: WAIT_FOR_PHY_RE_ASSESS
Description: "This state indicates that the patient is waiting for reassessment in the second consultation after the orders execution have been completed"
StateAttributes: patientID, providerID, unitID, startTime, endTime

State: IN_PHYS_RE_ASSES
Description: "This state indicates that the patient is in the reassessment phase in the second consultation"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: WAIT_FOR_BED_CW
Description: "This state indicates that the patient is waiting for bed in Cardiology Ward"
StateAttributes: patientID, unitID, startTime, endTime

-State: WAIT_FOR_TRANSPORT_CW
Description: "This state indicates that the patient is waiting for transportation to Cardiology Ward after he has been admitted"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: IN_TRANSPORT_CW
Description: "This state indicates that the patient is in transportation with the transporter to take him to Cardiology Ward"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: IN_BED_CW
Description: "This state indicates that the patient is remaining in bed in Cardiology Ward"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: WAIT_FOR_PROCEDURES
Description: "This state indicates that the patient is waiting for procedures to be performed in Cardiac Care Lab"
StateAttributes: patientID, unitID, startTime, endTime

-State: WAIT_FOR_TRANSPORT_CCL
Description: "This state indicates that the patient is waiting for transportation to Cardiac Care Lab for procedures"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: IN_TRANSPORT_CCL
Description: "This state indicates that the patient is in transportation with the transporter to take him to Cardiac Care Lab"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: IN_BED_CCL
Description: "This state indicates that the patient is remaining in bed in Cardiac Care Lab"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: IN_PROCEDURE_ANGIOGRAM
Description: "This state indicates that the patient is under the procedure Angiogram process"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: IN_PROCEDURE_PCI
Description: "This indicates that the patient is under the procedure PCI process"
StateAttributes: patientID, providerID, unitID, startTime, endTime

-State: IN_CONULTATION3
Description: "This state indicates that the patient is in the last consultation with the physician after he completed the procedures"
StateAttributes: patientID, providerID, startTime, endTime

-State: WAIT_FOR_DISCHARGE
Description: "This state indicates that the patient is waiting for discharge from Cardiology Ward"
StateAttributes: patientID, unitID, startTime, endTime

-State: DISCHARGED
Descrption: "This state indicates that the patient is discharged from Cardiology Ward**"**
StateAttributes: patientID, unitID, startTime

-State: BED_ASSIGNED
Description: "This state indicates that the bed is assigned to a particular patient"
StateAttributes: roomID, startTime, endTime

-State: WAIT_FOR_BED_CLEANUP
Description: "This state indicates that the bed is waiting for cleanup after the patient has been discharged and left the bed"
StateAttributes: roomID, startTime, endTime

-State: IN_CLEANUP
Description: "This state indicates that the bed is in cleanup process by the housekeeping"
StateAttributes: roomID, providerID, startTime, endTime

-State: BED_AVAILABLE
Description: "This state indicates that the bed is available to be assigned for the next cardiac patient"
StateAttributes: roomID, providerID, startTime, endTime

RULES

-Rule: Patient Wait For Initial Assessment with the Physician

Description: "This rule describes how to infer state WAIT_FOR_PHY_INIT_ASSESS"
Condition: IF EVENT: PatienInED AND EVENT: TriageScore
CurrentState: Triaged
NextState: WAIT_FOR_PHY_INIT_ASSESS

-Rule: Patient Starts the Initial Assessment with the Physician
Description: "This rule describes how to infer the state IN_PHYS_INT_ASSES"
Condition: IF EVENT: PhysicianInED
CurrentState: WAIT_FOR_PHY_INIT_ASSESS
NextState: IN_PHYS_INT_ASSES

-Rule: Patients Wait for Tests After the Physician Orders Tests
Description: "This rule describes how to infer the state WAIT_FOR_TESTS"
Condition: IF EVENT: OrderTest
CurrentState: IN_PHYS_INT_ASSES
NextState: WAIT_FOR_TESTS

-Rule: Patient Wait for Reassessment with the Physician
Description: "This rule describes how to infer state WAIT_FOR_PHY_RE_ASSESS"
Condition: IF EVENT: PatientInED AND EVENT: TestsCompleted
CurrentState: WAIT_FOR_TESTS
NextState: WAIT_FOR_PHY_RE_ASSESS

-Rule: Patient is in Reassessment Session with the Physician
Description: "This rule describes how to infer the state IN_PHYS_RE_ASSES"
Condition: IF EVENT: PhysicianInED
CurrentState: WAIT_FOR_PHY_RE_ASSESS
NextState: IN_PHYS_RE_ASSES

-Rule: Patient Completes Reassessment Session with the Physician and Remains in Bed in ED
Description: "This rule describes how to infer the state IN_BED_ED"
Condition: IF EVENT: PhysicianOutED
CurrentState: IN_PHYS_RE_ASSES
NextState: IN_BED_ED

-Rule: Patient Wait for Bed in CW
Description: "This rule describes how to infer the state WAIT_FOR_BED_CW"
Condition: IF EVENT: OrderBed AND EVENT: BedNotAvailable AND EVENT PhysicianOutED
CurrentState: IN_PHYS_RE_ASSES
NextState: WAIT_FOR_BED_CW

-Rule: Patient Wait for Transport to CW
Description: "This rule describes how to infer the state WAIT_FOR_TRANSPORT_CW"
Condition: IF (EVENT: BedAvailable AND EVENT: TransportRequest
CurrentState: WAIT_FOR_BED_CW
NextState: WAIT_FOR_TRANSPORT_CW

-Rule: Patient is in Transportation CW
Description: "This rule describes how to infer state IN_TRANSPORT_CW"
Condition: IF EVENT: PatientOutED AND EVENT: TransportOutED EVENT: TransportInED
CurrentState: WAIT_FOR_TRANSPORT_CW
NextState: IN_TRANSPORT_CW

-Rule: Patient Arrived In Bed CW
Description: "This rule describes how to infer state IN_BED_CW"
Condition: IF EVENT: PatientInCW AND EVENT: TransportOutED AND EVENT: TransportInED
CurrentState: IN_TRANSPORT_CW
NextState: IN_BED_CW

-Rule: Patient Wait for Procedures in CCL
Description: "This rule describes the state WAIT_FOR_PROCEDURES"
Condition: IF EVENT: ProceduresScheduled
CurrentState: IN_BED_CW
NextState: WAIT_FOR_PROCEDURES

-Rule: Patient Wait for Transport to CCL
Description: "This rule describes how to infer the state WAIT_FOR_TRANSPORT_CCL"
Condition: IF EVENT: PatientTransportRequest.Unit_ID = CCL
CurrentState: WAIT_FOR_PROCEDURES
NextState: WAIT_FOR_TRANSPORT_CCL

-Rule: Patient in Transportation to CCL
Description: "This rule describes how to infer the state IN_TRANSPORT_CCL"
Condition: IF EVENT: PatientOutCW AND EVENT: TransportOutCW EVENT: TransportInCW
CurrentState: WAIT_FOR_TRANSPORT_CCL
NextState: IN_TRANSPORT_CCL

-Rule: Procedure Angiogram is Started for the Patient
Description: "This rule describes how to trigger state IN_PROCEDURE_ANGIOGRAM once an event ProcedureStarted is received"
Condition: IF EVENT: ProcedureStarted. Procedure_Type = Angiogram AND EVENT: TransportInCCL

AND EVENT: PatientInCCL AND EVENT: TransportOutCCL
CurrentState: IN_BED_CCL
NextState: IN_PROCEDURE_ANGIOGRAM

-Rule: Procedure Angiogram is Completed and Patient In Bed CCL
Description: "This rule describes how to trigger state IN_BED_CCL once an event ProcedureCompleted is received"
Condition: IF EVENT: ProcedureCompleted. Procedure_Type = Angiogram
CurrentState: IN_PROCEDURE_ANGIOGRAM
NextState: IN_BED_CCL

-Rule: Procedure PCI is Started for the Patient
Description: "This rule describes how to trigger state IN_PROCEDURE_PCI once an event ProcedureStarted is received"
Condition: IF EVENT: ProcedureStarted. Procedure_Type = PCI
CurrentState: IN_BED_CCL
NextState: IN_PROCEDURE_PCI

-Rule: Procedure PCI is Completed and Patient In Bed CCL
Description: "This rule describes how to trigger state IN_BED_CCL once an event ProcedureCompleted is received"
Condition: IF EVENT: ProcedureCompleted. Procedure_Type = Angiogram
CurrentState: IN_PROCEDURE_ANGIOGRAM
NextState: IN_BED_CCL

-Rule: Patient Wait for Transport Back to CW
Description: "This rule describes how to infer the state WAIT_FOR_TRANSPORT_CW"
Condition: IF EVENT: PatientTransportRequest.Unit_ID = CW
CurrentState: IN_BED_CCL
NextState: WAIT_FOR_TRANSPORT_CW

-Rule: Patient in Transportation to CW
Description: "This rule describes how to infer the state IN_TRANSPORT_CW"
Condition: IF EVENT: PatientOutCCL AND EVENT: TransportOutCCL EVENT: TransportInCCL
CurrentState: WAIT_FOR_TRANSPORT_CW
NextState: IN_TRANSPORT_CW

-Rule: Patient Arrived Back in Bed CW
Description: "This rule describes how to infer the state IN_BED_ED"
Condition: IF EVENT: PatientInCW AND EVENT: TransportInCW
AND EVENT: TransportOutCW
CurrentState: IN_TRANSPORT_CW
NextState: IN_BED_CW

-Rule: Patient in Last Consultation with Physician
Description: "This rule describes how to infer the state IN_CONSULTATION3"
Condition: IF EVENT: PhysicianInCW
CurrentState: IN_BED_ED
NextState: IN_CONSULTATION3

-Rule: Patient Completes the Last Consultation and In Bed CW
Description: "This rule describes how to infer thestate IN_BED_ED"
Condition: IF EVENT: PhysicianOutCW
CurrentState: IN_CONSULTATION3
NextState: IN_BED_ED

-Rule: Patient Wait for Discharge
Description: "This rule describes how to infer the state WAIT_FOR_DISCHARGE"
Condition: IF EVENT: DischargeRequest
CurrentState: IN_BED_ED
NextState: WAIT_FOR_DISCHARGE

-Rule: Patient is Discharged from CW
Description: "This rule describes how to infer the state DISCHARGED_COMPLETED"
Condition: IF EVENT: PatientOutED
CurrentState: WAIT_FOR_DISCHARGE
NextState: DISCHARGED

-Rule: Bed is Assigned
Description: "This rule describes how to trigger state BED_ASSIGNED once the event BED_AVAILABLE is received"
Condition: IF EVENT: PatientAdmittedWithBed
CurrentState: BED_AVAILABLE
NextState: BED_ASSIGNED

-Rule: Bed is Waited For CleanUP
Description: "This rule describes how to infer the state WAIT_FOR_BED_CLEANUP"
Condition: IF EVENT: BedCleanUpRequest
Action: EVENT: WaitForBedCleanUp
CurrentState: BED_ASSIGNED
NextState: WAIT_FOR_BED_CLEANUP

-Rule: Bed is in Cleaning Process
Description: "This rule describes how to infer the state IN_CLEANUP"
Condition: IF EVENT: HouseKeepingInCW

CurrentState: WAIT_FOR_BED_CLEANUP
NextState: IN_CLEANUP

-Rule: Bed is Available for Admitted Patient
Description: "This rule describes how to infer the state BED_AVAILABLE"
Condition: IF EVENT: HouseKeepingOutCW
CurrentState: IN_CLEANUP
NextState: BED_AVAILABLE

EVENTS

-Event: PatientInED
Description: "This event is generated by RTLS and triggered when the patient enters one of the assessment rooms in ED"
EventAttributes: patientID, locationID, startTime

-Event: PatientOutED
Description: "This event is generated by RTLS and triggered when the patient steps out the assigned assessment room in ED"
EventAttributes: patientID, locationID, startTime

-Event: PatientInCW
Description: "This event is generated by RTLS and triggered when the patient enters one of rooms in Cardiology Ward"
EventAttributes: patientID, locationID, startTime

-Event: PatientOutCW
Description: "This event is generated by RTLS and triggered when the patient steps out from the assigned room in Cardiology Ward"
EventAttributes: patientID, locationID, startTime

-Event: PatientInCCL
Description: "This event is generated by RTLS and triggered when the patient enters one of rooms in Cardiac Cath Lab"
EventAttributes: patientID, locationID, startTime

-Event: PatientOutCCL
Description: "This event is generated by RTLS and triggered when the patient steps out from the assigned room in Cardiac Cath Lab"
EventAttributes: patientID, locationID, startTime

-Event: PhysicianInED
Description: "This event is generated by RTLS and triggered when the physician enters one of the assessment rooms in ED"
EventAttributes: physicianID, locationID, startTime

-Event: PhysicianOutED
Description: "This event is generated by RTLS and triggered when the physician steps out from the assessment room in ED"
EventAttributes: physicianID, locationID, startTime

-Event: PhysicianInCW
Description: "This event is generated by RTLS and triggered when the physician enters one of the rooms in Cardiology Ward"
EventAttributes: physicianID, locationID, startTime

-Event: PhysicianOutCW
Description: "This event is generated by RTLS and triggered when the physician steps out the room in Cardiology Ward"
EventAttributes: physicianID, locationID: CWRoom, startTime: timestamp

-Event: TransporterInED
Description: "This event is generated by RTLS and triggered when the transporter enters one of the assessment rooms in ED"
EventAttributes: transporterID, locationID, startTime

-Event: TransportOutED
Description: "This event is generated by RTLS and triggered when the transporter steps out from the assessment room in ED"
EventAttributes: transporterID, locationID, startTime

-Event: TransportInCW
Description: "This event is generated by RTLS and triggered when the transporter enters one of the rooms in Cardiology Ward"
EventAttributes: transporterID, locationID, startTime

-Event: TransportOutCW
Description: "This event is generated by RTLS and triggered when the transporter steps out from the room in Cardiology Ward"
EventAttributes: transporterID, locationID, startTime

-Event: TransportInCCL
Description: "This event is generated by RTLS and triggered when the transporter enters one of the rooms in Cardiac Cath Lab"
EventAttributes: transporterID, locationID, startTime

-Event: TransportOutCCL
Description: "This event is generated by RTLS and triggered when the transporter steps out from the room in Cardiac Cath Lab"
EventAttributes: transporterID, locationID, startTime:

-Event: HouseKeepingInCW
Description: "This event is generated by RTLS and triggered when the house keeping enters one of the rooms in Cardiology Ward to clean a bed when the patient is discharged"
EventAttributes: houskeepingID, locationID, startTime

-Event: HouseKeepingOutCW
Description: "This event is generated by RTLS and triggered when the house keeping steps out from the room in Cardiology Ward"
EventAttributes: houskeepingID, locationID, startTime

-Event: TriageScore
Description: "This event is generated by BPM when the nurse triage the patient before got admitted to ED"

EventAttributes: patientID, providerID, CTAS, startTime

-Event: OrderRequest
Description: "This event is generated by BPM when the physician request for order to the patient during the initial consultation"
EventAttributes: patientID, providerID, orderID, orderType, startTime

-Event: OrderRequestCompleted
Description: "This event is generated by BPM when the order requested by the physician is completed by nurse or lab"
EventAttributes: patientID, providerID, orderID, orderType, startTime

-Event: BedRequest
Description: "This event is generated by BPM when the physician request a bed for patient in CW"
EventAttributes: patientID, providerID, unitID, startTime

-Event: PatientAdmittedWithNoBed
Description: "This event is generated by BPM after the physician request for a bed in CW. This event indicates that the patient is admitted in CW but there is no bed available in the ward"
EventAttributes: patientID, unitID, startTime

-Event: ProceduresScheduled
Description: "This event is generated by BPM after the physician request for procedures for the patient in the second consultation. This event indicates that the procedures are scheduled for a patient"
EventAttributes: patientID, proceduresScheduledTime, unitID, startTime

-Event: PatientAdmittedWithBed
Description: "This event is generated by BPM after the physician request for a bed in CW. This event indicates that the patient is admitted in CW when there is a bed available in the ward"
EventAttributes: patientID, unitID, startTime

-Event: PatientTransportRequest
Description: "This event is generated by BPM when the nurse request to transport the admitted patient to CW or request to transport to CCL for procedure or transport back to CW when the patient completed the procedures"
EventAttributes: patientID, providerID, unitID, startTime

-Event: ProcedureStarted
Description: "This event is generated by BPM when the procedure for a patient in CCL is started"
EventAttributes: patientID, providerID, procedureType, startTime

-Event: ProcedureCompleted
Description: "This event is generated by BPM when the procedure for a patient in CCL is completed"
EventAttributes: patientID, providerID, procedureType, startTime

-Event: DischargeRequest
Description: "This event is generated by BPM when the physician request for a discharge for a patient from CW"
EventAttributes: patientID, providerID, unitID, startTime

-Event: BedCleanUpRequest
Description: "This event is generated by BPM when the manager request the housekeeping to come and clean a bed in CW"
EventAttributes: housekeepingID, locationID. startTime

SOURCES

-Source: RTLS
Description: "This source generates location based events for any resource in the hospital wearing active RFID tag"
Events: PatientInED, PatientOutED, PatientInED, PatientOutED, PatientInCCL, PatientOutCCL, PhysicianInED, PhysicianOutED, PhysicianInCW, PhysicianOutCW, TransporterInED, TransporterOutED, TransporterInCW, TransporterOutCW, TransporterInCCL, TransporterOutCCL, HouseKeepingInCW, HouseKeepingOutCW

-Source: BPM
Description: "This source generates events once a physician, a nurse or a manager fill out a form when performing their tasks"
Events: TriageScore, OrderRequest, OrderRequestCompleted, BedRequest, PatientAdmittedWithNoBed, ProceduresScheduled, PatientAdmittedWithBed, PatientTransportRequest, ProcedureStarted, ProcedureCompleted, DischargeRequest, BedCleanUpRequest

Appendix B: Community Care CPMA Application Model

CARE PROCESS

-CareProcess: Palliative Care Process
Description: "This care process shows the steps in which palliative care patients have to follow in a particular region, once they are referred to the program, in order to get care at their home in a regular basis through scheduled appointments with the clinicians who provide consultations"
Goals: Care, Education
Resources: Patient, Physician, Resident, Facility
States: WAIT_TRIAGE, WAIT_SCHEDULED, WAIT_FOLLOWUP, UNSCHEDULED, DISCHARGED, DECEASED

GOALS

-Goal: Care
Description: "This goal is related to quality and coverage of care provided to the palliative patients in the community"
Metrics:
SubGoals: Maximize Diagnosis Coverage, Minimize Wait Time, Minimize Unwanted Outcomes
-Goal: Education
Description: "This goal measure the effectiveness of the Palliative care program in promoting palliative care from the physicians, facilities and residence perspective"
Metrics:
SubGoals: Maximize Physician Participation, Maximize Resident Education, Maximize Facility Participation
-Goal: Maximize Diagnosis Coverage
Description: "This goal ensures the coverage of patients' (Cancer patients, non cancer patients, male and female patients) population in the community with"
Metrics: Number of Patients Cancer, Number of Patients Non Cancer
SubGoals:
-Goal: Minimize Wait Time
Description: "This goal ensures that the average wait times from referral to schedule an appointment and average time to respond to alerts triggered by the palliative patients are minimized"
Metrics: Average Wait_Triage Time, Average Wait_FollowUp Time
SubGoals:
-Goal: Minimize Unwanted Outcomes
Description: "This goal aims to measure the outcomes by minimizing the alerts when abnormal occurred and by minimizing number of patients visit to hospital and number of patient who have Chemo at the end of their lives"
Metrics: Number of Alerts, Number of Patients ER Last 2 Weeks, Number of Patients Chemo Last 2 Weeks
SubGoals:
-Goal: Maximize Physician Participation
Description: "This goal aims to maximize the participation of physicians in the palliative care program across the community"
Metrics: Number of Patients by Referring Physician
Subgoals:
-Goal: Maximize Resident Education
Description: "This goal aims to maximize the participation of residents in the palliative care program across community through their presence in the consultation with the physicians"
Metrics: Number of Consults with Resident Present
SubGoals:
-Goal: Maximize Facility Participation
Description: "This goal aims to maximize the participation of facilities in palliative care program across the community"
Metrics: Number of Patients by Referring Facility
SubGoals:

METRICS

- Metric: Number of Patients Cancer
Description: "This metric counts the number of patients referred to the palliative care program whose primary diagnosis is cancer over a period of time"
Computation: COUNT EVENT: Referral. primaryDiagnosis = Cancer Over a Period of Time
State:
Events: Referral
Target: N/A
Alert:
- Metric: Number of Patients Non Cancer

Description: "This metric counts the number of patients referred to the palliative care program whose primary diagnosis is not cancer over a period of time"
Computation: COUNT EVENT: Referral. primaryDiagnosis = NonCancer Over a Period of Time
State:
Events: Referral
Target: N/A
Alert:

- Metric: Average Wait_Triage Time
Description: "This metric measure the average time the palliative patients wait once they are referred to the palliative care program until a scheduled appointment is booked"
Computation: AVERAGE STATE: WAIT_TRIAGE. duration Over Period of Time
State: WAIT_TRIAGE
Events:
Target: <7 days
Alert: WAIT_TRIAGE_WARNING, WAIT_TRIAGE_UNACCEPTABLE

- Metric: Average Wait_FollowUp Time
Description: "This metric measure the average time the palliative patients wait when abnormal condition occurred until a consult occurs"
Computation: AVERAGE STATE: WAIT_FOLLOWUP. duration Over Period of Time
State: WAIT_FOLLOWUP
Events:
Target: <=4hours
Alert: WAIT_FOLLOWUP_WARNING, WAIT_FOLLOWUP_UNACCEPTABLE

- Metric: Number of Alerts per patient
Description: "This metric counts number of alerts triggered by the patients when the value of ESAS or PPS changes unexpectedly"
Computation: FOR EACH Patient COUNT EVENT: Alert Over a Period of Time
State:
Events: Alert
Target: <=1 per Patient
Alert: NUMBER_ALERTS_UNACCEPTABLE

- Metric: Number of Patients ER Last 2 Weeks
Description: "This metric counts number of deceased patients who visit ER in the last two weeks of their live over a period of time"
Computation: FOR EACH Patient COUNT EVENT Decease. chemoLast2Weeks = True Over a Period of Time
State:
Events: Decease
Target: 0
Alert: NUMBER_ER_VISIT_ UNACCEPTABLE

- Metric: Number of Patients Chemo Last 2 Weeks
Description: "This metric counts number of deceased patients who had chemo for the last two weeks of their live over a period of time"
Computation: FOR EACH Patient COUNT EVENT Decease. ERLast2Weeks = True Over a Period of Time
State:
Events: Decease
Target: 0
Alert: NUMBER_CHEMO_UNACCEPTABLE

- Metric: Number of Patients by Referring Physician
Description: "This metric counts number of patient referring by each physician in the community"
Computation: FOR EACH Physician COUNT EVENT: Referral. referralSource = Physician Over a Period of Time
State:
Events: Referral
Target: Minimum 1 per physician
Alert: NUMBER_REFER_PHY_UNACCEPTABLE

- Metric: Number of Consults with Resident Present per Resident
Description: "This metric counts the number of consults given with a resident present"
Computation: FOR EACH Resident COUNT EVENT: Consult. residentPresent = True Over a Period of Time
State:
Events: Consult
Target: Minimum 1 per resident
Alert: NUMBER_CONSULT_UNACCEPTABLE

- Metric: Number of Patients by Referring Facility
Description: "This metric counts number of patients referring by each facility in the community"
Computation: FOR EACH Facility COUNT EVENT: Referral. referralSource = Facility Over a Period of Time
State:
Events: Referral
Target: Minimum 1 per facility
Alert: NUMBER_REFER_FAC_UNACCEPTABLE

ALERTS

-Alert: WAIT_TRIAGE_WARNING
Description: "This message is displayed when the wait time for a specific patient is more than 66% of target"
Message: "WARNING: PATIENT P has been waiting for D days and the target is 7 days**"**
-Alert: WAIT_ TRIAGE_UNACCEPTABLE
Description: "This message is displayed when the wait time is over 7 days"
Message: "UNACCEPTABLE. PATIENT P has been waiting for M minutes and the target is 4 hours**"**
-Alert: WAIT_FOLLOWUP_WARNING
Description: "This message is displayed when the wait time for a specific patient is more than 66% of target"
Message: "WARNING: PATIENT P has been waiting for M minutes and the target is 4 hours**"**
-Alert: WAIT_FOLLOWUP_UNACCEPTABLE
Description: "This message is displayed when the wait time is over 4 hours"
Message: "UNACCEPTABLE. PATIENT P has been waiting for M minutes and the target is 4 hours**"**
Alert: NUMBER_ALERTS_UNACCEPTABLE
Description: "This message is displayed when there are more than ONE alert per patient"
Message: "UNACCEPTABLE. PATIENT P has A alerts and the target is 0 or 1 alert"
Alert: NUMBER_ER_VISIT_ UNACCEPTABLE
Description: "This message is displayed when the number of visits, per deceased patient, to ER within the last 2 weeks of his/her life is over target"
Message: "UNACCEPTABLE. PATIENT P had V visits to ER within the last 2 weeks of his/her life and the target is no visit to ER"
Alert: NUMBER_CHEMO_UNACCEPTABLE
Description: "This message is displayed when the number of chemo per deceased patients had within the last 2 weeks of his/her life is over target"
Message: "UNACCEPTABLE. PATIENT P had C chemo within the last 2 weeks of his/her life and the target is no chemo should been taken"
Alert: NUMBER_REFER_PHY_UNACCEPTABLE
Description: "This message is displayed when there are no referrals by specific physician"
Message: "UNACCEPTABLE. PHYSICIAN P has ZERO referrals and the target is 1 or more referrals per physician"
Alert: NUMBER_CONSULT_UNACCEPTABLE
Description: "This message is displayed when the resident is not presenting during the consultations"
Message: "UNACCEPTABLE. RESIDENT R has ZERO Consults and the target is 1 or more consult participation"
Alert: NUMBER_REFER_FAC_UNACCEPTABLE
Description: "This message is displayed when there are no referrals by specific facility"
Message: "UNACCEPTABLE. FACILITY F has ZERO referrals and the target is 1 or more referrals per facility"

RESOURCES

-Resource: Patient
Description: "The patient for whom palliative care consults is given."
ResourceAttributes: patientID
States: WAIT_TRIAGE, WAIT_SCHEDULED, WAIT_FOLLOWUP, UNSCHEDULED, DISCHARGED, DECEASED
-Resource: Physician
Description: "The physician for the patient who refers the patient or who is present when a palliative specialist provides a consultation."
ResourceAttributes: physicianID
States:
-Resource: Resident
Description: "A doctor in training who is present during a consultation or education session"
ResourceAttributes: residentID
States:
-Resource: Facility
Description: "A healthcare facility (e.g. hospital, long-term care, clinic) that refers patients for palliative care consultation"
ResourceAttributes: facilityID
States:

STATES

-State: WAIT_TRIAGE
Description: "This state shows that the palliative patient is waiting a scheduled appointment to be booked for the first time"
StateAttributes: patientID, startTime, endTime
-State: WAIT_SCHEDULED
Description: "This state shows that the palliative patient is waiting for the scheduled consult to take place with the physician"
StateAttributes: patientID, startTime, endTime
-State: WAIT_FOLLOWUP

Description: "This state shows that the palliative patient is waiting for unscheduled followup consultation when an abnormal condition occurred"
StateAttributes: patientID, startTime, endTime

-State: UNSCHEDULED
Description: "This state shows that the palliative patients are not scheduled for an appointment for the next consultation"
StateAttributes: patientID, startTime, endTime

-State: DISCHARGED
Description: "This state shows that the palliative patient is discharged from the palliative care program and no longer available in the system"
StateAttributes: patientID, startTime, endTime

-State: DECEASED
Description: "This state shows that the palliative patient is deceased"
StateAttributes: patientID, startTime, endTime

RULES

-Rule: Patient Wait for Triage
Description: "This rule describes how the patient transit to the state WAIT_TRIAGE when the initial referral is received by the palliative care program"
Condition: IF EVENT: Referral
CurrentState:
NextState: WAIT_TRIAGE

-Rule: Patient Wait for Scheduled Consultation
Description: "This rule describes how the palliative patient transit to the state WAIT_SCHEDULED for the consultation when a scheduled appointment is booked for that patient"
Condition: IF EVENT: Appointment
CurrentState:
NextState: WAIT_SCHEDULED

-Rule: Patient Discharged from the Palliative Care Program
Description: "This rule describes how the patient transit to state DISCHARGED when the physician requests for that"
Condition: IF EVENT: Discharge
CurrentState:
NextState: DISHARGED

-Rule: Patient Deceased
Description: "This rule describes how the patient transit to the state DECEASED"
Condition: IF EVENT: Decease
CurrentState:
NextState: DECEASED

-Rule: Patient not Scheduled after Consult
Description: "This rule describes how the patient is transit to the state UNSCHEDULED after completing consult with the physician"
Condition: IF EVENT: Consult
CurrentState:
NextState: UNSCHEDULED

-Rule: Patient Wait for Unscheduled FollowUp Consultation
Description: "This rule describes how the patient transit to state WAIT_FOLLOWUP for unscheduled consultation due to an abnormal condition occurred"
Condition: IF EVENT: Alert
CurrentState:
NextState: WAIT_FOLLOWUP

EVENTS

-Event: Referral
Description: "This event is generated when the user fill out and submit the Referral online form for a particular palliative patient"
EventAttributes: patientID, referralSource, referralPhysician, primaryDiagnosis, referralDate, deliveryDate

-Event: Appointment
Description: "This event is generated when the clerk booked for appointment to the palliative patient, by filling out the Appointment form online, after he/she received a referral"
EventAttributes: patientID, appointmentDate, location, startTime

-Event: Discharge
Description: "This event is generated when the palliative patient gets discharged by the physician who fills out and submits the Discharge online form"
EventAttributes: patientID, providerID, startTime

-Event: Decease
Description: "This event is generated when the patient is deceased by filling out the Decease online form"
EventAttributes: patientID, ERLast2Weeks, chemoLast2Weeks, startTime

-Event: Consult

Description: "This event is generated when the physician conduct a consultation with the palliative patient by filling out the Consult online form"
EventAttributes: patientID, providerID, location, residentPresent, startTime

-Event: Alert
Description: "This event is generated when ESAS goes above a threshold, as well, when the PPS goes above threshold value by filling out the Alert online form"
EventAttributes: alertID, PPS, ESAS, startTime

SOURCES

-Source: Referral Form
Description: "This source generates an event after the user fills out and submits the Referral online form"
Events: Referral

-Source: Appointment Form
Description: "This source generates an event after the user fills out and submits the Appointment online form"
Events: Appointment

-Source: Discharge Form
Description: "This source generates an event after the user fills out and submits the Discharge online form"
Events: Discharge

-Source: Decease Form
Description: "This source generates an event after the user fills out and submits the Decease online form"
Type: HTML Form
Events: Decease

-Source: Consult Form
Description: "This source generates an event after the user fills out and submits the Consult online form"
Events: Consult

-Source: Alert Form
Description: "This source generates an event after the user fills out and submits the Alert online form"
Events: Alert