A Generic BI Application for Real-Time Monitoring of Care Processes

Shirley Awusi Baffoe

Directed By:
Prof. Liam Peyton

Thesis

Submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Science in Electronic Business Technologies

uOttawa

University of Ottawa

Ottawa, Ontario, Canada

May 2013

Abstract

Patient wait times and care service times are key performance measures for care processes in hospitals. Managing the quality of care delivered by these processes in real-time is challenging. A key challenge is to correlate source medical events to infer the care process states that define patient wait times and care service times. Commercially available complex event processing engines do not have built in support for the concept of care process state. This makes it unnecessarily complex to define and maintain rules for inferring states from source medical events in a care process. Another challenge is how to present the data in a real-time BI dashboard and the underlying data model to use to support this BI dashboard. Data representation architecture can potentially lead to delays in processing and presenting the data in the BI dashboard.

In this research, we have investigated the problem of real-time monitoring of care processes, performed a gap analysis of current information system support for it, researched and assessed available technologies, and shown how to most effectively leverage event driven and BI architectures when building information support for real-time monitoring of care processes. We introduce a state monitoring engine for inferring and managing states based on an application model for care process monitoring. A BI architecture is also leveraged for the data model to support the real-time data processing and reporting requirements of the application's portal. The research is validated with a case study to create a real-time care process monitoring application for an Acute Coronary Syndrome (ACS) clinical pathway in collaboration with IBM and Osler hospital. The research methodology is based on design-oriented research.

Acknowledgements

First and foremost I would like to thank the Almighty God for His guidance and protection throughout my studies. Next, my sincerest gratitude goes to my supervisor, Liam Peyton for his encouragement, support and guidance during my research.

I am also very grateful to IBM, IBM-Telfer Centre for Business Analytics and Performance, the Osler Hospital, Mitacs and NSERC (discovery grant and Business Intelligence Network) for helping me finance my education through their research grants.

I also wish to express my deep sense of appreciation for Mr. Randy Giffen, my manager during my Internship with IBM and also for collaborating with us on the project and offering assistance with the IBM tools.

Finally, I wish to express my sincere thanks to my husband for his support and encouraging words throughout my studies. I would not have made it this far without his support.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
ACS	Acute Coronary Syndrome
API	Application Programming Interface
B2B	Business-to-Business
BA	Business Analytics
BAM	Business Activity Monitoring
BI	Business Intelligence
BPM	Business Process Management
CCU	Cardiac Care Unit
CEM	Continuous Event Monitoring
CEP	Complex Event Processing
CTU	Capture Transform Update
CW	Cardiology Ward
DAO	Data Access Object
ED	Emergency Department
EDA	Event-driven Architecture
EHR	Electronic Health Record
ETL	Extract Transform Load
FM	Framework Manager
GPS	Global Positioning System
HIS	Hospital Information Systems

HTTP	Hypertext Transfer Protocol
IS	Information System
IT	Information Technology
KPI	Key Performance Indicator
OLAP	Online Analytical Processing
ORM	Object Relational Mapping
REST	Representational State Transfer
RFID	Radio Frequency Identification
RTLS	Real-time Location Systems
SIPS	Strategic Information and Performance Systems
SME	State Monitoring Engine
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Data Discovery Interface
WS	Web Service
WSDL	Web Service Description Language
XML	eXtensible Mark-up Language

Chapter 1. Introduction

1.1. Problem Statement

In Canadian hospitals, as in other countries, patient wait times and care service times are key performance measures for care processes in hospitals. Improving these times is difficult in part because data collected to measure processes and make decisions is often imprecise and delayed. In addition, improving these times is highly dependent on the way patient flow management for a care process is handled inside the hospital. The main difficulty associated with managing patient flow within a care process in hospitals is a lack of visibility, across hospital departments, on patients, procedures and staff information (Mouttham, Peyton, & Kuziemsky, 2011).

Proper management of patient wait times and service times may support smarter and proactive decisions in managing patient flow. Those decisions usually impact on resource allocation and hospital process management. For example, how can the hospital know where to provide more staff to reduce the increased wait times in the hospital? Given that additional resource allocation comes with increased cost and there are competing priorities, the hospital decision makers need to determine which course of action is the most appropriate one to take. Decisions made based on information within a departmental silo may not be the optimal one. Such decisions may only transfer the patient flow bottleneck to another department.

According to Fan et. al., to empower decision makers to make timely decisions, data needs to be collected, integrated, analyzed and presented in real-time (Fan & Bai,

2004). Real-time visibility into performance measures for care processes require a data processing architecture that can aggregate disparate low level events in operational care processes to provide high level views of the states defined by the care process (Fan & Bai, 2004). A key challenge in this context, is to correlate source medical events to infer the care process states that define patient wait times and care service times. Another key challenge is to define a generic data model that can integrate event data from disparate sources to support the real-time reporting of performance measures in a BI Dashboard.

Some hospitals attempt to integrate the data from the disconnected systems using ETL data warehouses. The data from the operational systems are loaded into the data warehouse in batches through an Extract Transform and Load (ETL) process. Due to the batch processing, only historical reports on performance measures are obtained from these data warehouses. It is usually too late to solve performance problems based on the performance reports from the ETL data warehouses. The ETL data warehouse, therefore, does not address the need for data integration and reporting in real-time. It is necessary to provide fine-grained analytics in real-time to support decision making where bottlenecks occur in order to improve performance management.

Some researchers have attempted to solve the problem of real-time data integration with the use of a Complex Events Processor (CEP) for processing events to infer medical events in real-time (Baarah, Mouttham, & Peyton, 2011), (Ku, Zhu, Hu, & Lv, 2008), and (Boubbeta-Puig, Ortiz, & Medina-Bulo, 2011). Both Business Process Management (BPM) technology (Fan & Bai, 2004) and Real Time Location Systems (RTLS) (Yao, Chu, & Li, 2011) have also been used as a source of events to monitor and

measure where patients are in a care process and how long they have been there. However, commercially available complex event processing engines do not have built in support for the concept of care process state. This makes it unnecessarily complex to define and maintain rules for inferring states from source medical events in a care process. BPM only and RTLS only solutions are also unable to infer all the required states in the care process.

In this thesis, we propose a state monitoring engine (SME) for inferring and managing states that is an improvement over traditional CEP approaches. Our proposed engine is based on an application meta-model for care process monitoring developed by (Baarah & Peyton, 2012). In addition, to inferring and managing states, the SME dynamically populates a real-time data warehouse for care process monitoring by mapping the events and inferred states into an Online Analytical Processing (OLAP) model. The OLAP model is optimized for the reporting requirements of the care processes. It allows easy access and presenting of the key performance indicators, and also enables one to drill down to an individual care process. The Care process OLAP model feeds a BI data mart, which is queried for real-time performance reports.

The SME, OLAP data model, and BI data mart have been integrated to develop a generic BI Portal for care process monitoring. The proposed generic BI Portal has been validated with a case study for patient flow monitoring of cardiac patients admitted to hospital through the emergency room. The case study was developed and the results of our approach were validated in collaboration with medical, nursing and IT staff at a large community hospital in Ontario, Canada.

1.2. Thesis Motivation and Contributions

The main motivation behind this study is to provide a better way of implementing a care process monitoring application. We improve on the web application for care process monitoring that was developed using CEP processing for real-time data integration (Baarah, Mouttham, & Peyton, 2012). We felt, that the scalability and efficiency of real-time data integration could be improved by replacing the CEP with an engine that provided direct support for inferring state from events by using a state handler architecture. Another motivation was to improve on the custom simplified performance reporting embedded in the web application by providing a more effective BI Portal for Care Process Monitoring driven directly off an OLAP data model. This BI portal will provide hospital managerial staff with the ability to drill up, drill down, slice and dice the analytics to identify the root cause of problems unlike the traditional web application with its object model and Object Relational Mapping (ORM) database. Empowering hospital managerial staff with this BI portal that supports fine-grained analytics may allow them to support smarter and proactive decisions at managing care processes. And since these decisions usually impact on resource allocation and hospital process management, improving decision making and management of care processes should help drive business sustainability and help improve hospital overall performance.

The main contributions of this thesis are:

1. A generic BI application for real-time monitoring of care processes that includes:
    a. A generic BI portal and Patient Tracking Dashboard

b. A generic OLAP data model and data mart
c. A generic state monitoring engine (SME) based on a state-based application model for care process monitoring which can process clinical events and maintain an OLAP data model of the care process using a state handler architecture.

2. The SME which has:
    a. A state handler architecture to support state-based inferencing and updates as it processes streamed clinical events. This state handler architecture is based on a care process management application model which is more convenient to use than the general purpose commercial Complex Event Processing (CEP) engines.
    b. An object model to OLAP model transformation. This transformation enables us to bridge the gap between the operational system and the performance analytics monitoring system. Thereby, allowing us to produce real-time BI reports.
3. A case study to validate our monitoring application that implements the complete clinical pathway for Acute Coronary Syndrome and validates the advantages of our generic care process monitoring application over traditional web application architectures that use complex event processing engines.

The following publications have been published related to the thesis:

1. **S. Baffoe**, A. Baarah, A. Mouttham, and L. Peyton "Inferring State for Real-Time Monitoring of Care Processes", *SEHC 2013*, San Francisco, USA, May, 2013.
2. A. Baarah, **S. Baffoe**, A. Mouttham, and L. Peyton "State Monitoring Service for Real-Time Patient Flow Management", *Poster: CASCON 2012*, Toronto, Canada, November, 2012.

The research was done in a team and the following are related works performed by the other team members:

- A. Baarah, "An Application Framework for Monitoring Care Processes", *Thesis: PhD in Software Engineering*. University of Ottawa, Ottawa, Ontario, (expected May 2013). The main contribution of his research is the design of a generic application meta-model for engineering care process monitoring applications.
- R. Bougueng, "Location-Aware Business Process Management for Real-time Monitoring of Patient Care Processes", *Thesis: MSc in Computer Science*. University of Ottawa, Ontario, (expected May 2013). The main contributions of his research is the design of an integrated business process management and real-time location service solution architecture that was the source of events processed by our care process monitoring application in our ACS clinical pathway case study.
- A. Mouttham, "A Framework for Real-Time Analytics and Decision Support in Patient Flow Management", *Thesis: PhD in Computer Science*. University of Ottawa, Ottawa, Ontario, (in progress). The contributions of that thesis include

- Design of framework for real-time measurement and analysis of patient wait times, based on patient tracking, without burdening Clinicians
- Design of framework for real-time decision support in the management of patient flows inside a hospital.
- Validation of the overall ACS cast study (domain expert).

1.3. Thesis Methodology and Organization

Since this research is still in its early stages, the methodology used is design-oriented research (Hevner, March, Park, & Pam, 2004). We perform a gap analysis of existing approaches proposed in literature, develop a prototype for a possible solution to address the gaps identified and evaluate it based on a set of criteria. Below are the steps we followed in the thesis methodology:

1. Problem identification based on existing literature and interviews with business experts:
    a. Comparative study of real-time data integration and performance reporting in existing literature (literature survey).
    b. Discussion with experts in Hospital Information Systems (HIS)
    c. Interaction with existing hospitals (case studies) on actual real-time care process monitoring problems
    d. Identify the critical evaluation criteria that any solution to the problem must address

e. Perform gap analyses between existing and proposed solutions to address the problem and evaluation criteria.

2. Model design and prototype development based on existing or a combination of existing enabling technologies.
3. Application of prototype to ACS care process case study (implementation)
4. Evaluation of prototype based on criteria drawn from the gap analysis based on the literature survey and discussion with experts (testing and analysis)
5. Comparative study with related works in care process management.
6. Conclusions and further work.

This thesis document is organized as follows: Chapter 1 presents an introduction, together with the context and the scope of the research. Chapter 2 provides a background to our research. We review literature from related works. The technologies and tools used in these related works are analysed and compared. Chapter 3 elaborates the research problem, our research model and evaluation criteria. In Chapter 4, we present our case study and provide details of our experiment. Based on the evaluation criteria, we evaluate the research in Chapter 5. Chapter 6 mentions conclusions drawn from our work.

Chapter 2. Background

In this chapter, we first introduce and define the key concepts that are relevant to the development of this thesis. Then, a more detailed background is given on the most important areas related to data integration and care process monitoring applications. Finally, in the related works section we identify and explain other approaches in the literature that have been taken to solve our problem. These are used for comparison in the evaluation of our thesis in Chapter 5.

2.1. Concepts

2.1.1 Business Process

A Business Process is a "coordinated chain of activities intended to produce a business result" (Bruce Silver Associates, 2006).

2.1.2 Business Process Management (BPM)

Business Process Management is defined as "the management of diverse and cross-organizational processes using methods and tools to support the design, execution, management, and analysis of business processes" (Pourshahid, et al., 2009). BPM requires the continuous monitoring of business activities in order to supply real-time information about the results of the business processes (key performance Indicators) (Kang & Han, 2008). According to (Pourshahid, et al., 2009), BPM is comprised of several iterative steps to enhance the business process (see Figure 1). This research focuses on the monitoring and performance management aspect of BPM.

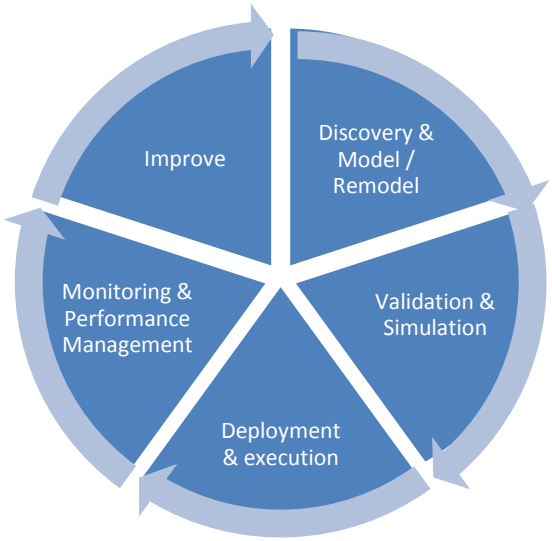


Figure 1: BPM Life Cycle

Source: (Pourshahid, et al., 2009)

2.1.3 Performance Management

Performance management is used to tie evaluation of the success and compliance of business processes directly to the data collected about them, in terms of metrics (Middleton, Peyton, Kuziemsky, & Eze, 2009).

2.1.4 Care Process Monitoring

This involves monitoring the metrics that measure performance in a care process, defining the care process in terms of states that a patient goes through including service and wait times. These key metrics measure the length of those wait and service states and are used to determine the quality of care offered to a patient.

2.1.5 Wait Time

The wait time refers to the time it takes the patient to wait for a service in the hospital. This usually starts from the time the service is requested till the time the service provision begins.

2.1.6 Service Time

Service time refers to the time it takes a service provider to offer a particular care service to a patient. For example, the service time for a physician consult usually takes from the time the physician enters the room with the patient till the time he leaves the room.

2.1.7 Delay

Delay can be defined as the time difference between the occurrence of an event and when the data transformed into business intelligence was made available to stakeholders.

2.1.8 Business Intelligence (BI)

Business Intelligence is defined as "the application of various advanced analytic techniques to data to answer questions or solve business problems" (Trkman, McCormack, Valadares de Oliveira, & Ladeira, 2010). Business Intelligence (BI) systems measure the outcomes of strategic initiatives and present them in the form of key performance indicators.

2.1.9 Metric

A metric can be defined as "a quantitative measure of the degree to which a system, component, or process possesses a given attribute" (Villar, 2010). Key Performance Indicators (KPIs) are the most critical high level metrics that measure how well the overall business is performing. Hospitals have KPIs which measure the quality of care and give insight into what is happening in a specific care process (Ferrand, Amyot, & Villar, 2010). The time a patient spends waiting for a service is one example of a KPI used to assess the quality of care.

2.1.10 Real-time Analytics

Real-time analytics measure that a particular outcome resulted from a particular action and involves delivering charts, metrics and alerts in real-time (Chieu & Zeng, 2008). The performance of an organisation can be assessed by quantifying the organisational objectives as real-time performance analytics and measuring the ability to achieve these desired outcomes (Mouttham, Peyton, & Kuziemsky, 2011).

2.1.11 Real-time Location Systems (RTLS)

Real-time Location Systems (RTLS) are systems which can track the location of people and resources in real-time (Zang & Wu, 2010). They employ technology like GPS or RFID. For example, inexpensive Radio Frequency Identification (RFID) tags can be worn by people or attached to resources and then RFID readers can identify the location of objects in real time (Violino, 2005). The technology used in RTLS is sufficiently mature and cost effective that it is starting to be used in hospitals. These devices make it possible to track the location of patients, nurses, and physicians in real-time. These

systems, however, simply deliver raw location data as events. These events must be transformed and correlated with other data from hospital information systems in order to understand the relationship with care processes.

2.1.12 Event

An event is an atomic occurrence of interest that happens in a point in time (Chakravarthy & Mishra, 1994). A correlation of events can be interpreted as a state transition (Li & Jacobsen, 2005).

2.1.13 Event-Driven Architecture

(Niblett & Graham, 2005) define an event-driven architecture as "a system architecture pattern whereby the behaviour of the systems is characterized by the production, notification and consumption of events". In this architecture, events are collected from diverse sources and processed (Middleton, Peyton, Kuziemsky, & Eze, 2009). Significant progress in Information Technology (IT) makes real-time processing and event delivery possible. A key technology used for real-time events gathering and processing is Complex Event Processing (CEP).

2.1.14 Complex Event Processing (CEP)

Complex event processing (CEP) is an emerging technology for recognizing and inferring high level business events (or complex events) in a real-time event driven enterprise (Ku, et al., 2008). The main goals of CEP are to process events from distributed sources in real-time and infer causal relationships discovered by correlating other events to produce other meaningful complex events (Kang & Han, 2008). The CEP is an event consumer that receives basic events from all the sources it is interested in and

produces complex events that it propagates to interested consumers for decision-making or actions to be taken (Baarah, et al., 2011).

2.1.15 Data Integration Architecture

Data Integration refers to the ability of combine data from different sources and providing the user with a unified view of these data (Lenzerini, 2002). Data integration architecture is an IT data model capable of receiving event data from heterogeneous and distributed sources thereby providing more detailed view of processes. The occurrence of one event may mean little, but when coupled with other events, it may indicate something of importance (Middleton, Peyton, Kuziemsky, & Eze, 2009).

2.1.16 Web Service

The W3C defines web services as "a software system designed to support interoperable, machine-to-machine interaction over a network" (Web Services Architecture, 2004). Web services allow interoperability of software components in a heterogeneous environment. They can be accessed over a network using HTTP and SOAP protocols using the pre-defined interfaces. Web Service Description Language (WSDL) is an XML based language used to describe web service interfaces. HyperText Transfer Protocol (HTTP) is the protocol used the transfer information on the Internet while Simple Object Access Protocol (SOAP) is an XML based protocol for message exchange (Curbera, Duftler, Khalaf, Nagy, Mukhi, & Weerawarana, 2002).

2.1.17 Data Access Object

The Data Access Object is a design pattern used in service-oriented architecture. The DAO implements the access mechanism required to work with any data source or

low-level sockets (Alur, Crupi, & Malks, 2003). The DAO provides an abstract API, which makes no reference to how the data source is implemented, for manipulating a data source (Singh, Stearns, Johnson, & Team, 2002).

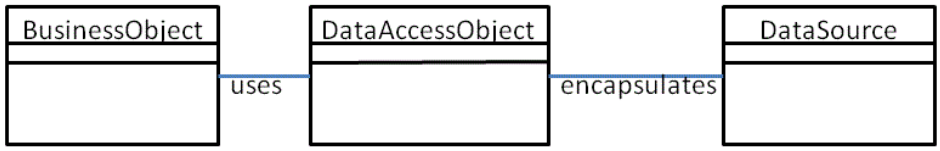


Figure 2 Data Access Object

The main advantage of using a DAO is that it decouples the user of the object from the programming mechanism that accesses the underlying data. Thus, changes to the schema or function specification of the database or database technology may not impact the DAO's user interface or the client system (Alur, Crupi, & Malks, 2003).

2.1.18 Data Warehouse

A data warehouse, as defined by Inmon, is "a subject-oriented, integrated, time varying, non-volatile collection of data that is used primarily in organizational decision making" (Inmon, 2005). The data from the operational systems are usually extracted and transferred to the data warehouse in batches. The data warehouse is maintained separately from an organization's operational databases because it's functional and performance requirements are different from that of operational databases (Chaudhuri & Dayal, 1997). While operational databases are optimized for transactional efficiency, data warehouses are optimized for reporting. The data in the data warehouse is, therefore, organised differently from that of the operational database systems. The tangible measurements or metrics resulting from a business process or measurement event are stored in fact tables,

while the descriptive/qualitative attributes or characteristics associated with the facts are stored in dimensional tables (Kimball & Ross, 2013)

2.1.19 Star Schema

A star schema is a dimensional model physically instantiated in a relational database (Kimball, Ross, Thornthwaite, Mundy, & Becker, 2008). In the star schema dimensional model, data stored in fact tables and joined to data in multiple dimensional tables. Hierarchical many-to-one relationships in operational tables are denormalized into single dimension tables in the star schema model thereby optimizing it for query performance in reporting (Kimball, Ross, Thornthwaite, Mundy, & Becker, 2008).

2.1.20 Data Mart

A data mart is a collection of facts related to a subject or process that need to be used together. They contain a single subject or fact tables. According to (Kimball, Ross, Thornthwaite, Mundy, & Becker, 2008) a data warehouse is a collection of data marts, each having a star schema design at its base. A data warehouse bus matrix is used to model different data marts and dimensions together.

2.1.21 Online Analytical Processing (OLAP)

On-Line Analytical Processing (OLAP) is an essential element of decision support (Chaudhuri & Dayal, 1997). OLAP, as defined by the OLAP council, "is a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user. OLAP functionality is

characterized by dynamic multidimensional analysis of consolidated enterprise data supporting end user analytical and navigational activities including calculations and modeling applied across dimensions, through hierarchies and/or across members, trend analysis over sequential time periods, slicing subsets for on-screen viewing, drill-down to deeper levels of consolidation, rotation to new dimensional comparisons in the viewing area " (Vassiliadis & Sellis, 1999). Dimensional models stored in an OLAP database are known as cubes (Kimball, Ross, Thornthwaite, Mundy, & Becker, 2008).

2.2. Data Integration and Care Process Monitoring Applications

A key component required for the development of health care analytics is the Information Technology (IT) required to support the clinical processes (Mouttham, Peyton, & Kuziemsky, 2011). There is a cognitive gap between the Hospital Information Systems (HISs) which support the clinical processes and cluster of technologies that provide the business analytics required for performance management (Barone, Jiang, Amyot, & Mylopoulos, 2011). This problem is exacerbated by the fact that existing HISs are complex and receive data from diverse sources. In this section we provide background information on existing technologies one might consider for a care process monitoring application.

2.2.1 ETL Data Warehouse Architecture

Some hospitals provide business analytics using traditional data warehouses that collect and integrate data from source HIS using ETL (Extract, Transform and Load) processes. ETL is the process by which operational data is extracted and transferred into

the data warehouse. ETL is the foundation of a typical data warehouse / business intelligence project and has 4 major components Extracting, Cleaning and Conforming, Delivering and Managing (Kimball, Ross, Thornthwaite, Mundy, & Becker, 2008).

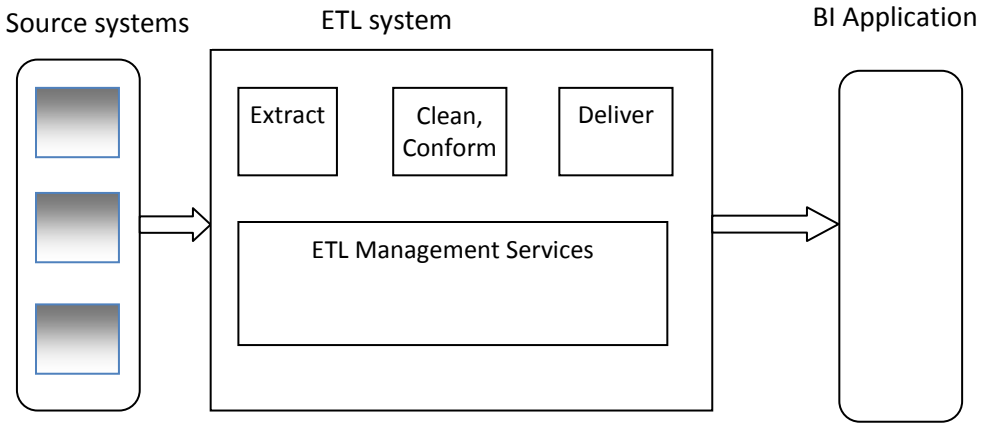


Figure 3: ETL Data Warehouse Architecture

This process of extracting, cleaning, and conforming process data from the diverse sources is usually performed in batches and can take days or even weeks to produce the desired healthcare analytics. This batch processing of the information from the source systems introduces undesirable delays in obtaining the analytics and evaluating performance (Chieu & Zeng, 2008). Another problem is that, the traditional ETL system usually does not provide enough detail to be able to determine the cause and effect relationship of a particular event happening. Real-time analytics is required to help the hospitals tie outcomes to actions and help them know what event resulted in what action.

2.2.2 Web Application

A Web Application is a three tier client/server application consisting of a browser-based HTML front end, a service side application model and a database

(Gellersen & Gaedke, 1999). In many web applications, the mapping from server-side application model to database is done using Object Relational Mapping (ORM) technology. Web Applications benefit from the pervasive distribution of Web browsers for cross-platform access and are usually organised in such a way as to facilitate navigation and searching through the application.

2.2.3 BI Portal Application

A BI portal is defined as a gateway to information and services from multiple sources and is organised in such a way as to facilitate accessing resources to meet users needs (Palavitsinis, Protonotarios, & Manouselis, 2011). They provide a single integrated user interface for accessing the reporting, monitoring, and analysis functionality of BI applications (Peyton, Zhan, & Stepien, 2008). The BI portal usually interacts with the database through a business model. It differs from traditional web applications because it does not have an application model.

2.2.4 Object Relational Mapping (ORM)

Object oriented applications use object hierarchies to hold application data in memory (Tegegne & Peyton, 2011). It is difficult to map these object hierarchies to relational database tables directly in order to persist the data. Object Relational Mapping (ORM) is a technique used to map the objects to database tables (Ambler, 1997). ORMs use a mapping definition to determine the class-to-table or property-to-column mapping in the object relational database (Fowler, 2003). Hibernate is an example of ORM commonly used in Online Grails or Java Applications. However, using ORM usually produces to poorly designed databases that require multiple joins between tables when

queried. ORM designed tables are therefore not optimized for either reporting or persisting transactional data. In the first stage of our work, we used an ORM Web application to persist the patient care process state information and also to present the real-time analytics to the users. The BI application in this thesis will be compared with the ORM Web application.

2.2.5 Service Oriented Architecture

SOA provides a way of reorganizing software applications and infrastructure into a set of interacting and re-useable services with well-defined interfaces (Papazoglou, 2003). As illustrated in Figure 4 below, SOA is composed of three main parts: a service provider, a consumer or service requestor and a service discovery agency or registry (Huhns & Singh, 2005). The services are published on the registry by the service provider and consumers search these registries to find and invoke these services. Web services form a basis for SOA and provide a suitable technical foundation for making business processes accessible within an enterprise or across different enterprises operating in a heterogeneous environment (Leymann, Roller, & Schmidt, 2002).

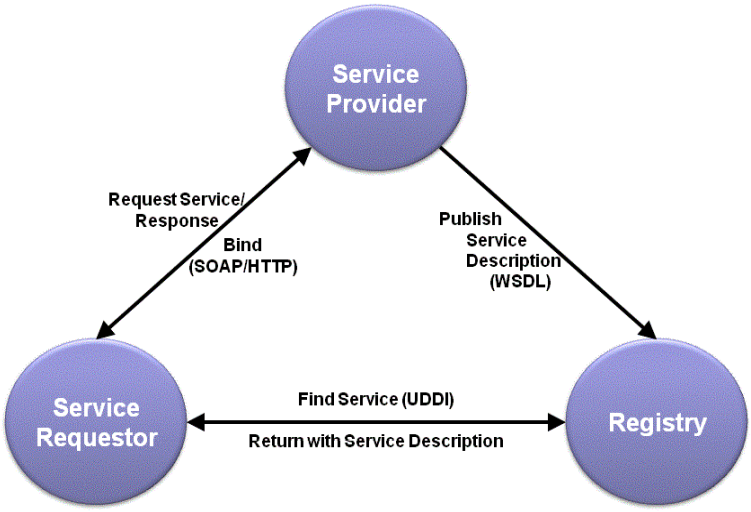


Figure 4 Service Oriented Architecture

Web service technologies (such as SOAP or REST) allow applications to exchange information regardless of the programming language and platform being used by the individual applications (Curbera, Duftler, Khalaf, Nagy, Mukhi, & Weerawarana, 2002).

2.2.6 Message Broker

One of the new and emerging technologies which make it possible to deliver event data from the myriad of data sources and information systems in the Hospital in real-time is the Message broker solution. The message broker is a communication component in charge of transforming events from different sources in a variety of formats into a single stream of events in a standardized format (Baarah, Mouttham, & Peyton, 2011). Examples of message broker products include IBM Websphere Message Broker and BizTalk. However, the message broker delivers raw event data and not business level events. In order to better manage the performance, business level events in the form of (events, state and alerts) need to be provided so the hospitals can react in real-time to the occurrence of particular business events.

2.3. Related Works

Different approaches were proposed in the literature including the use of Complex Events Processing, Enterprise Message Buses, Near Real-time ETL data warehouses, and Business Activity Monitoring technologies (Jeng, Schiefer, & Chang, 2003), (Abo-

Hamad & Arisha, 2012) , (Abrahiem, 2007), (Naeem, Dobbie, & Webber, 2008), (Yao, Chu, & Li, 2011).

One other interesting approach to providing the decision support to improve patient wait times proposed in the literature was through the use of multi-criteria simulation framework (Abo-Hamad & Arisha, 2012). The framework integrates simulation modeling, balance scorecard and multi-criteria decision analysis to provide decision support to Emergency Department (ED) managers. The main contribution of this work was the Emergency Department simulation model which when run for multiple scenarios provided the optimal decisions for resource utilization in the ED. However, a key limitation of the approach is the work focused on only one department (ED) and did not consider that bottlenecks in other departments could lead to longer waiting times in the ED. Also, though the research results indicated that patient wait times in the ED were reduced, there was no indication that the waiting times in the other departments were monitored to ascertain patient flow bottlenecks were not simply transferred to other departments. According to (Mouttham, Peyton, & Kuziemsky, 2011), decisions made based on a departmental view are often not the optimal decision. This is because, they usually do not address the root cause of patient flow bottlenecks, but rather, it leads to the transfer of the bottleneck from one department to the other. In addition, the approach had an ad hoc process for the selection of key performance indicators, Due to these limitations of the framework, we did not compare our work to the framework.

We selected the following three approaches as the most relevant related work to compare our approach to. The approach by (Middleton, Peyton, Kuziemsky, & Eze,

2009), developed a framework for health care performance management, (Chieu & Zeng, 2008), developed a framework for real-time data integration and performance reporting, while (Baarah, Mouttham, & Peyton, 2011), proposed a CEP based approach for real-time patient flow monitoring.

2.3.1 Agent Based Data Integration

(Middleton, 2009), developed an agent-based architecture for continuous compliance monitoring of quality of care policies for palliative care patients. He used his architecture to manage the performance in the business process application for these patients who usually receive care at home. In his architecture, event messages from patients and providers were collected, filtered, correlated and logged into an event repository. These events were then analysed to produce key performance indicators which were measured against the expected values. The main contribution of his framework was a comparison of different rule-based engines for correlating events to show the advantages of agent-based approaches. The main weakness of Middleton's architecture is that there was no model that defined the relationship between metrics and care processes or care processes and events or event metrics and events. There was a goal-based performance model that identified the metrics. For each metric, an agent was crafted to process rules that defined how the metric should be updated in response to events and how alerts should be flagged. As well, the BI reporting of events and metrics was not designed to be real-time (Middleton, Peyton, Kuziemsky, & Eze, 2009). This was partially, because the sources of events could not be relied upon to deliver their events in real time. Metric computations and events were simply logged into a data warehouse to support reporting and analysis after the fact (in order to verify compliance).

2.3.2 Complex Event Processing (CEP) Based Data Integration

(Yao, et al., 2011), used a CEP architecture for processing a continuous stream of business events from diverse sources into meaningful complex events. The framework employed pre-defined rules to generate the higher level complex events required for performance management in real-time. (Yao, et al., 2011) used only sensor-based events from RFID sensors and did not explicitly integrate a business process monitoring engine to manage business processes. As a result, it was difficult to compute business process metrics with Yao et al.'s approach.

2.3.3 Real-Time Capture, Transform and Update (CTU) Data Integration

(Chieu & Zeng, 2008), describe a novel CTU architecture that is capable of incrementally updating the performance data warehouse. They used the approach to enable on-demand performance monitoring of electronic contracts in an enterprise information management system (Chieu & Zeng, 2008). The main contribution of their work is the introduction of a novel data model and a database trigger mechanism, which allows the system to support real-time data processing and performance monitoring. Though their approach was suitable for enterprise information management, it lacked support for detailed models of business processes and integration of other event types (like location-based events) that are important for care process management.

Chapter 3. Generic BI Application for Care Process Monitoring

In this chapter, we present our approach for developing care process monitoring applications by defining a generic BI application that can serve as a template that can be configured and instantiated to monitor any care process. In Section 3.1, we provide a detailed description of the problem and a summary of our gap analysis. In section 3.2, we define the criteria that will be used to evaluate any solution to the problem. The design of the generic BI application used in our approach is described in section 3.3. In section 3.4, we introduce a state-based State Monitoring Engine (SME) which uses state-specific handlers to infer and update states from events. This is a significant innovation for real-time data integration to support care process monitoring. In section 3.5, we describe in detail the state handler architecture used in the SME. In section 3.6, we introduce the object model to OLAP model transformation which enables us to bridge the gap between the operational and performance systems. In Section 3.7, we define a generic care process monitoring OLAP model as the foundation for performance reporting and in section 3.8 we discuss our generic BI portal for care process monitoring that leverages the OLAP model.

3.1. Problem Description

Most hospitals today have a traditional HIS data warehouse architecture in which data is extracted from disparate and disconnected data sources, transformed and loaded into a data warehouse view over a period of days or weeks to support historical reporting

(Baarah, Mouttham, & Peyton, 2012). These systems typically do not integrate location-based information or other forms of real-time data feeds. As a result, it is difficult to provide detailed monitoring of care processes in real-time in order to measure and display service and wait times for patients at different steps in the care process. The problem we wish to address is how to integrate data from the disconnected source systems, infer the care process states that define patient wait times and care service times and present these analytics in real-time in a BI portal to support decision making.

Figure 5 below, shows an example of the traditional HIS data architecture.

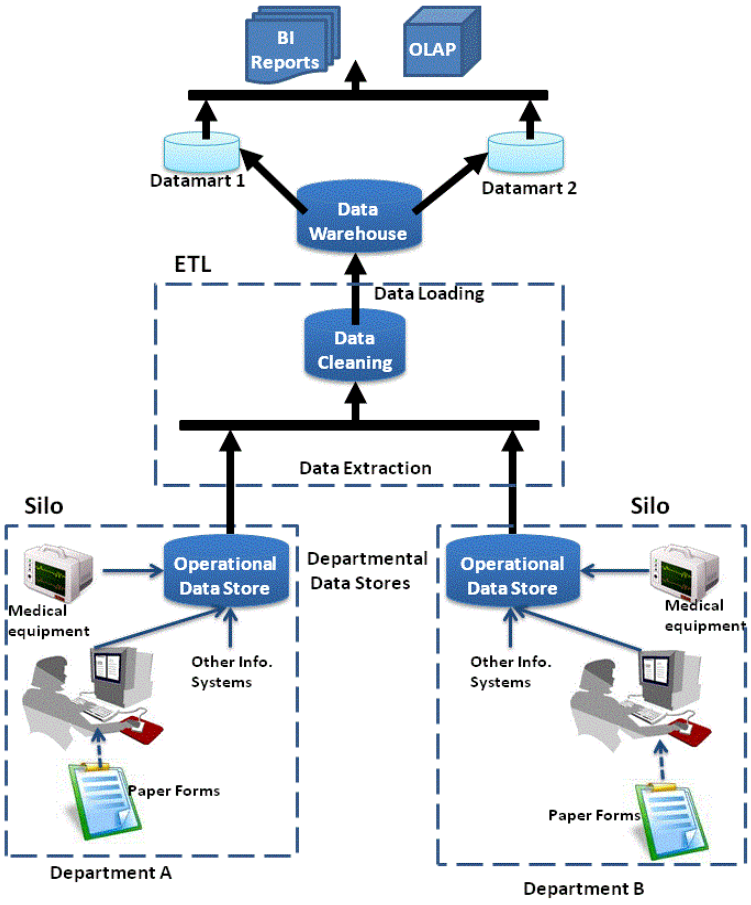


Figure 5: Traditional HIS Data Architecture

In this traditional architecture, data collection at point of care is predominantly paper-based. This data is later transcribed into the operational data stores in the department. Data may also be obtained from other information systems e.g. medical and diagnostic equipment, in the department. Different departments often have different data stores, which hold information related to the processes in that department. There is usually very little data exchange between the various departmental data stores leading to the phenomenon of data silos. A department may, therefore, have a detailed knowledge of their current bottlenecks, and may make decisions to resolve these problems without having a hospital-wide view. Though their decisions may be optimal for the department, it may not necessarily be the optimal one for the entire hospital. For example, if a large number of patients are in the ED, the patient flow managers may decide to bring in more ED physicians to attend to these patients. However, if the root cause of the bottleneck was the unavailability of beds in the Cardiology Ward (CW), then bringing in more ED physicians will only make to problem worse by increasing the number of patients waiting for beds in CW. In such a scenario, sending more Cardiologists to the CW, to expedite the discharge of patients, may be a more optimal solution for the entire hospital. Making decisions based on a narrow departmental view, may only transfer the bottleneck from one department to the other.

The hospitals also typically extract information from these operational data stores into ETL data warehouses for historical reporting. Typically, an electronic health record (EHR) is assembled for the patient as well in this manner. These data warehouses are usually separate from the operational systems. Data are extracted from the operational stores in batches, transformed, and loaded into the data warehouse. Due to the batch

processing, reports on performance measures obtained from the data warehouse is usually delayed by hours, days or weeks making it too late to solve performance problems while care is being delivered. In addition, the information assembled into the data warehouse or EHR is not detailed enough to pinpoint precisely where and at what step in the care process bottlenecks are occurring.

The problem we are addressing is how to integrate care process data in real-time to measure patient wait times and service times as they receive care. Another aspect of the problem is how to present and display these analytics on a dashboard in real-time to support decision making related to resource staffing at different steps in the care process. To solve the problem, we should be able to:

1. Collect process event data from the disparate source systems in real-time.
2. Correlate this information to infer new states or update existing states real-time.
3. Store care process data and compute performance metrics in real-time
4. Display the analytics in real-time at the point of care

Current research in this area seeks to improve on the Historical ETL data warehouse approach, through the use of a Complex Event Processing (CEP) and a dynamic Web Application.

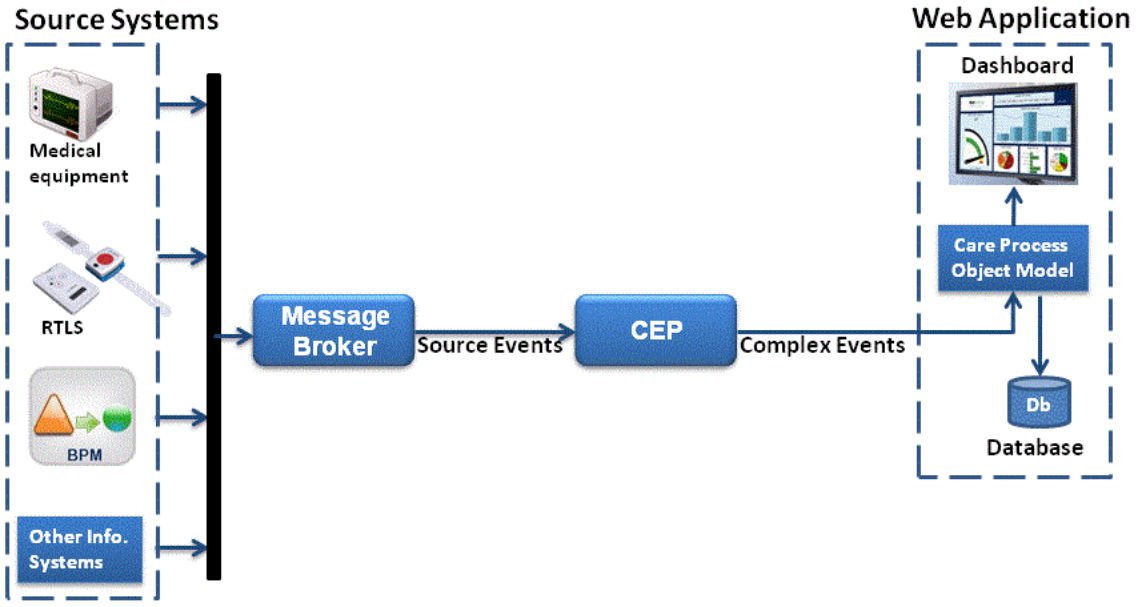


Figure 6 Architecture of CEP Based Approach

In the CEP based approach, low level source events are correlated from the various sources systems including hospital equipment, BPM and RTLS systems. The CEP then infers higher level business events based on pre-defined rules. These high level business events are forwarded to the Web Application which updates the patient states based on the events received and displays the analytics in real-time through its dashboard. The care process states information is persisted in database, which is accessed by only the Web application. Often the database is generated automatically using Object-Relational Mapping (ORM) technology. This approach provides real-time views into the care process. However, the CEP has no inbuilt understanding of care process states. As a result, it can be complex to write the CEP rules and complex event handlers to update the Care Process object model. Also ORM based databases are typically not optimised for reporting.

In this thesis, we will propose a State Monitoring Engine (SME) and OLAP Care Process data model which improves on the CEP based approach by leveraging a generic state-based application meta-model for care process monitoring, with built in support for inferring and updating states from events and delivering a BI portal to compute and present the performance measures in a more comprehensive and easy to use format in real-time.

3.1.1 Summary of Gap Analysis

In this section we summarize the available approaches and their limitations.

TABLE 1: PATIENT FLOW MONITORING APPROACHES AND LIMITATIONS

Approach	**Limitations**
Paper Based with Historical ETL Data warehouse	-Paper based data collection and transcribing is a laborious process which introduces delays and errors in data entry. - Due to batch processing in ETL data warehouse, only historical performance reports can be obtained. - Analytics not available to care providers at the point of care.
CEP + Web Application	- CEP not able to understand care process states making state inferring unnecessarily complex. - Event inferring dependent on the sequence of the events received. - ORM database used in web portal not optimized for reporting. - Unable to drill in to individual care process details. - Complex to present new analytics in web portal. - some amount of events processing takes place in web portal (no separation of concerns)

3.2. Evaluation Criteria

Based on our literature survey, gap analysis, and interaction with domain experts at IBM and our hospital collaborators, we identified 17 criteria by which we evaluate our

research. The criteria are based on complexity, scalability, usability and the features of the BI application.

Complexity, scalability, and usability are standard software engineering criteria mentioned in literature for evaluating the quality of software (Reddy & Rao, 2009), (Basseda, Alinaghi, & Ghoroghi, 2009), (Misra, Akman, & Colomo-Palacios, 2012), (Green & Pearson, 2006), (Kannampallil, Schauer, Cohen, & Patel, 2011), (Brataas & Hughes, 2004) .

The complexity is defined as "the degree of interrelatedness of the system components" (Kannampallil, Schauer, Cohen, & Patel, 2011). They explain that the complexity of a system increases with the number of components in the system, and the number of relations between these components. According to (Clancy, Effken, & Pesut, 2008), systems can become complex when there are many combinations of events at a point in time. Based on these definitions, we introduced 5 sub-criteria to evaluate the complexity of the care process monitoring application:

- Complexity of modeling care process elements (i.e. Combination of events required to model scenarios),
- complexity of data integration rules (i.e. the number of interactions between different rules),
- complexity of transformation to database (i.e. Number of interactions required between systems (or people) to transform the operational data and load into the database),

- Complexity of database (i.e. the number of interactions required between different database tables during insertion and querying). In other words does the database require lookup tables to insert data and does it require multiple cross joins between tables when queried?
- Complexity of building real-time dashboard (i.e. ease of building and expressing reports in dashboard).

ISO 9241, part 11 defines usability as 'the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use' (Green & Pearson, 2006). In other words, usability is concerned with the extent to which users of an application are able to work effectively, efficiently, and with satisfaction in their particular contexts (Stone & Stone, 2005). The usability criteria used to assess the care process monitoring application were obtained from our interactions with patient flow managers and domain experts. One domain expert, who was embedded in both the hospital research team and our research team, was instrumental in articulating the criteria during the weekly research meetings. In addition, after the development of these criteria, they were reviewed in 5 key review sessions with 4 different hospitals (3 in Ontario and 1 from Thailand). The 2 key sub-criteria used to evaluate the usability of the application are:

- The ability to drill down to a particular care process details, as well as, drill up to obtain a hospital wide view.
- Ability to access the application at the point of care.

The scalability of a system can be defined in terms of many specific qualities that are commonly used in the literature, e.g. architecture of system, response time, performance, system efficiency, etc (Duboc, Rosenblum, & Wick, 2007). According to (Brataas & Hughes, 2004) an "architecture is scalable if it has a linear increase in physical resource usage as capacity increases". (Smith & Williams, 2002) also define scalability as "the ability of a system to continue to meet its response time or throughput objectives as the demand for the software functions increases." In this research, we did not evaluate the scalability of the system based on its response time, or throughput objectives. It is expected that these tests will be done in future work. However, the architectural scalability of our application was evaluated. The following 4 sub-criteria were introduced to assess the scalability of our architecture.

- Technology used to define the application (hard coded, generated code from model, declarative business rules interpreted, etc.)
- Effort and skills required to build a first care process monitoring application.
- Effort and skills required to update an existing care process monitoring application with new or changed events and states (maintainability)
- Effort and skills required to build new but similar care process monitoring applications

Experiences with the system developed in the first iteration of this project and some of its limitations led us to introduce key features required for the real-time care process monitoring application as evaluation criteria. For instance, one limitation of the existing CEP- based system was its inability to infer states directly from source events. We introduced 6 sub-criteria for the features of the care process monitoring application to assess it completeness. These include:

- Ability to model explicitly care process to be monitored based on performance measures, resources, process states, critical events and the mappings between them.
- ability to infer care process state transitions (start of state, end of state) from source process events (and/or states and/or resources)
- Ability to update attributes of existing states and/or resources (entities) from source events
- Ability to compute performance metrics from source process events (and/or states and/or resources) in real-time.
- Ability to persist source process events, and persist continually updated care process states and resources to support both real-time and historical reporting.
- Ability to communicate individual and aggregate status of care process states and resources along several dimensions for flexible and focused real-time reporting

The list below shows the detailed evaluation criteria.

1. Complexity:
    1.1. Complexity of modeling care process elements.
    1.2. Complexity of data integration rules
    1.3. Complexity of transformation to database
    1.4. Complexity of database
    1.5. Complexity of building real-time dashboard

2. Usability
    2.1. Ease of aggregating, filtering, drilling up, drilling down and drilling through the analytics in the dashboard.
    2.2. Ability to access dashboard at the point of care.

3. Scalability:
    3.1. Technology used to define the application (hard coded, generated code from model, declarative business rules interpreted, etc.)
    3.2. Effort and skills required to build a first care process monitoring application.
    3.3. Effort and skills required to update an existing care process monitoring application with new or changed events and states
    3.4. Effort and skills required to build new but similar care process monitoring applications

4. Real-time Care Process Monitoring Features

    4.1. Ability to model explicitly care process to be monitored based on performance measures, resources, process states, critical events and the mappings between them.

    4.2. Ability to infer care process state transitions (start of state, end of state) from source process events (and/or states and/or resources)

    4.3. Ability to update attributes of existing states and/or resources (entities) from source events

    4.4. Ability to compute performance metrics from source process events (and/or states and/or resources) in real-time

    4.5. Ability to persist source process events, and persist continually updated care process states and resources to support both real-time and historical reporting.

    4.6. Ability to communicate individual and aggregate status of care process states and resources along several dimensions for flexible and focused real-time reporting.

These evaluation criteria are used in chapter 5, to compare our solution to related works in literature and other existing solutions.

3.3. Overview of Generic BI Application for Care Process monitoring

In this section we provide an overview of the design and architecture of our generic BI application for care process monitoring. Figure 7 below shows the architecture of the generic BI application. We use a State Monitoring Engine (SME), to integrate events from various care process sources in real-time to maintain an up to date care process model and populate an OLAP database to support performance reporting. The SME processes the source clinical events, updates the entities and states, and maintains an OLAP data model of the care process using a state handler architecture and state inference rules. The SME events processing is based on an application meta-model for care process monitoring developed by (Baarah, Mouttham, & Peyton, 2011).

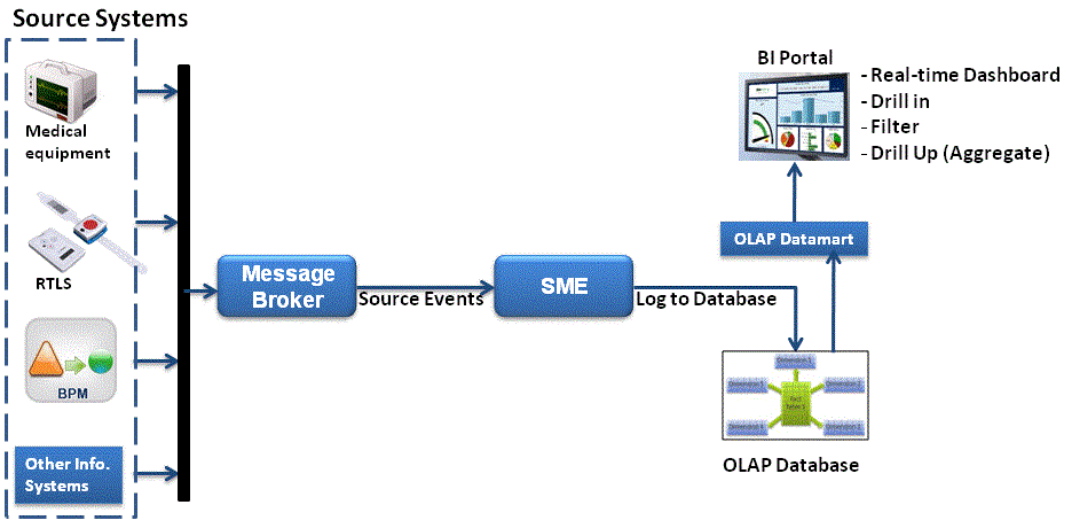


Figure 7: Architecture of Generic Care Process Monitoring BI Application

The application meta-model provides a linkage between the key care process states and the source events used to infer those states, on the one hand, and a linkage between the key care process states and the metrics used to measure the performance of

the care process (Baarah, Mouttham, & Peyton, 2011). Section 3.4 provides a detailed description of the SME architecture, while Section 3.5 describes the manner in which care process states are inferred and updated.

The SME persists data in a star schema reporting database by mapping the SME's object model to an OLAP model. This object to OLAP model transformation bridges the gap between the operational system and the reporting one. As a result, the delay introduced by batch processing of data, required in ETL data warehouses, is eliminated. This transformation enables us to produce real-time performance reports on care process states. We describe this object model to OLAP model transformation in Section 3.6.

The OLAP data model consists of a star schema reporting database and an OLAP data mart. It is optimized for reporting and also supports drill down, drill up, filtering and aggregating. The star schema database stores real-time and historical data of patients and key resources like beds. Section 3.7 discusses the care process OLAP data model in detail.

A BI portal is used to present the performance measures in real-time. The portal offers the generic ability to explore and navigate through real-time and historical performance reports. In addition, the portal incorporates a dashboard which displays the current state of the patients in the hospital. We discuss the BI portal architecture and its advantages over the Web Application in section 3.8.

3.4. State Monitoring Engine

Figure 8 below, shows the architecture of the State Monitoring Engine (SME) which provides data integration of care process events in real-time. The SME incorporates a Web Service (WS) interface, which enables communication with external and disparate source systems; an event queue, which queues all source events received for processing; a state handler architecture, which infers states and updates entities in the Care Process object model; and a Data Access Object (DAO) which is responsible for persisting data in the OLAP database.

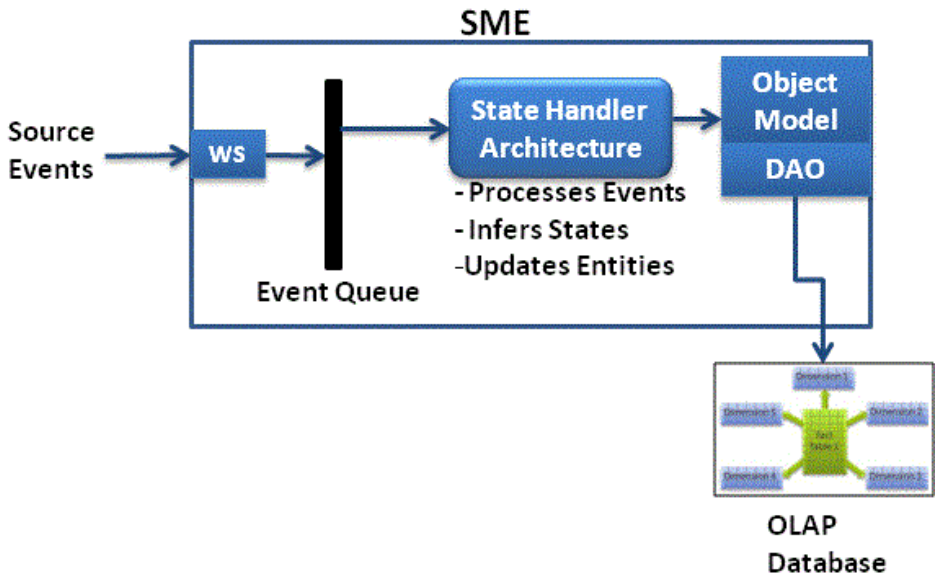


Figure 8 SME Architecture

Source events are received into the SME through a SOAP web service (WS) interface and are placed on the event queue. The SME state handler architecture retrieves events off the queue, sending each event to the relevant state handler for the current state of the care process for the patient to which the event applies. The state handler processes the source event, using rules to update the corresponding entities in the object model and

infer transitions to the next state in the process. The state handler calls the appropriate methods in the DAO map to the object model and persist data in the OLAP database.

The rule-based state handler architecture is a significant improvement over general complex event processing (CEP) engines. This ability to infer care process states directly from source events eliminates the need for producing higher level events required by the CEP. Its behaviour is explicitly based on an application meta-model for real-time care process monitoring. Section 3.5 provides more details of the state handler architecture.

3.5. State Handler Architecture

In this section, we describe the state handler architecture and rules. Figure 9 below depicts the state handler architecture.

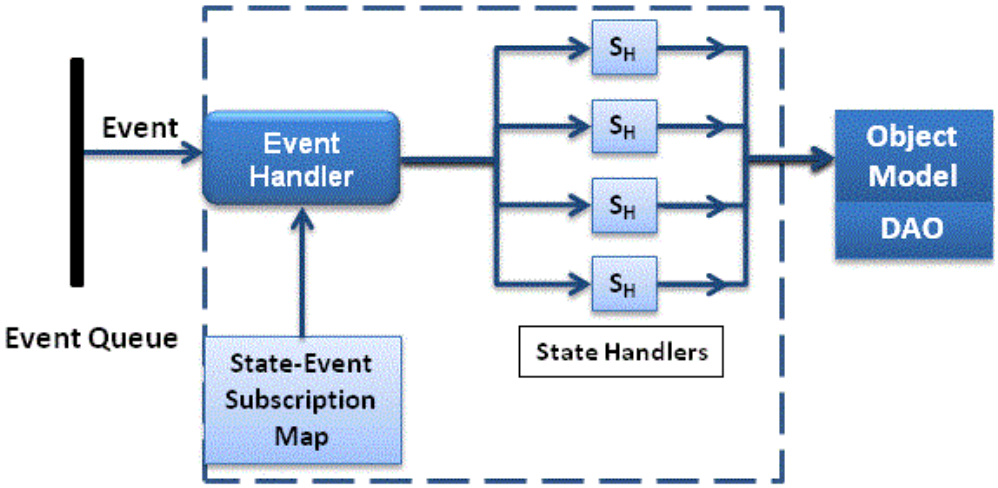


Figure 9: State Handler Architecture

The Event Handler receives events from the Event Queue of the SME and forwards them to the appropriate state handlers. There is a state-event subscription map,

which is used by the event handler to determine which state handlers to forward the events to. Figure 10 below is a generic example of a state handler. Each State handler has state inference rules, based on the current state of the patient (state monitored) and the correlation of source events, to define state transitions from the current patient state. The state handler uses the *getCurrentPatient( )* and *getCurrentState( )* methods to retrieve the current patient and patient state.

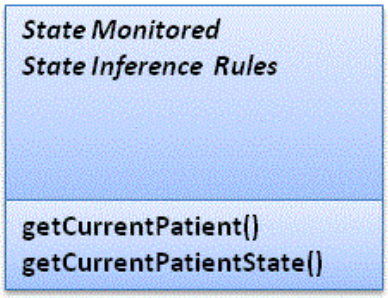


Figure 10: Generic State Handler

3.6. Object Model to OLAP Model transformation

In order for the SME to dynamically populate a real-time data warehouse for care process monitoring, we map the events and inferred states in the object model into an Online Analytical Processing (OLAP) model. We replace the object relational database mapping used in ORM based web applications with our object to OLAP mapping for state-based care process monitoring.

We use a DAO in the SME to persist data in the database. The DAO encapsulates all access to the database thereby hiding all the database implementation details from the SME. The DAO is also responsible for managing the connection with the database. An advantage of using this DAO design pattern is it reduces the coupling between the

persistence logic and the business logic. In other words, changes in the database or database technology will not affect the SME and vice versa.

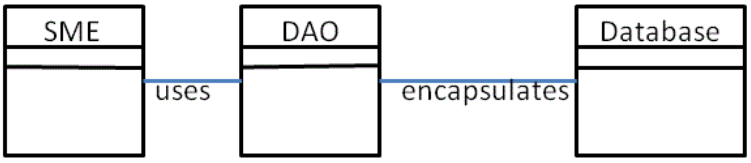


Figure 11: DAO for SME

The OLAP model is optimized for the reporting requirements of the care processes. It allows easy access and presenting of the key performance indicators. The Care process OLAP model feeds dimensionally modeled BI data marts which are queried for real-time performance reports. These dimensionally modeled data marts enable one to drill down to an individual care process, drill up (aggregate), slice and dice.

3.7. Care Process OLAP Data Model

We use a star schema data architecture, where measures are in a fact table and descriptive attributes about the measures are in the dimensional tables. Using this data representation architecture helps us to eliminate delays in processing and presenting the data in the BI Portal. Figure 12 represents a sample star schema database model for patient states.

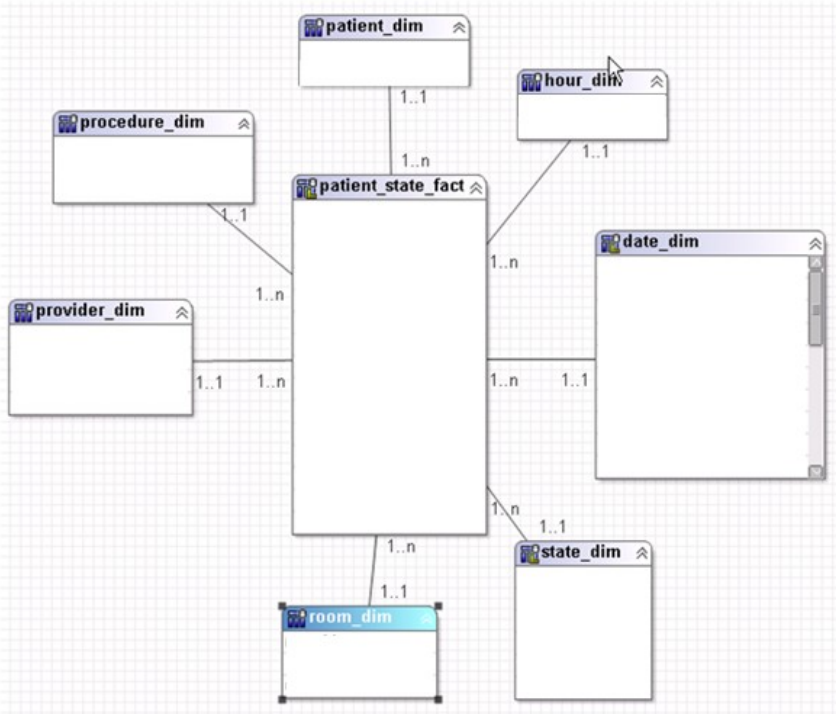


Figure 12: Star Schema Example

This star schema data architecture is more efficient as compared with the ORM modeled database because it is used to log information related to the care process states being monitored while the ORM database only logs information related to the objects in the application.

Also the star schema data architecture is based on the Kimball's core concepts of dimensional modeling (Kimball & Ross, 2013). It is optimized for query performance in reporting as compared to the ORM modeled databases and operational databases. This is because ORM and operational databases usually use normalized / hierarchical many-to-one relationships, whereas, the star schema modeled database has denormalized dimension tables.

Table 2 below represents the data warehouse bus matrix that contains the data marts we created and the dimensional tables associated with each data mart. The Patient

States we wish to monitor and the Patient Events we received, are the two patient related subject areas we created data marts for.

TABLE 2: CARE PROCESS MONITORING APPLICATION BUS MATRIX

	Dimensions							
Data marts	**Date**	**Hour**	**Patient**	**State**	**Event**	**Procedure**	**Provider**	**Room**
Patient State	X	X	X	X		X	X	X
Patient Events	X	X	X		X	X	X	X

3.8. Care Process Monitoring Portal

The Care Process Monitoring Portal provides the interface for viewing both real-time and historical performance reports (See Figure 13 below).

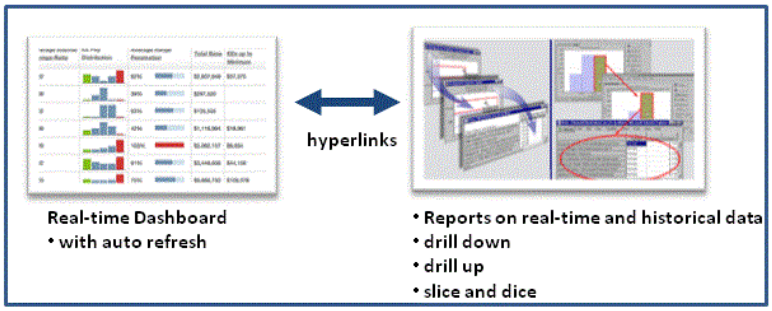


Figure 13: Care Process Monitoring Portal

The Care Process Monitoring Portal includes a Patient Tracking Dashboard which allows care providers to view real-time analytics for current patients in the care process at a glance. The dashboard shows what their current state is in the care process, how long they have been in that state, and whether they have exceeded hospital guidelines for the duration of that state. This dashboard is auto-refreshed and does not require any intervention from the user to update the analytics. This portal provides hospital

managerial staff with the ability to drill up, drill down, and drill through the analytics to identify the root cause of patient flow bottlenecks.

Chapter 4. The Case Study

We have conducted an extensive case study, for patient flow monitoring of cardiac patients admitted to the hospital through the emergency room, to evaluate our approach. Based on the clinical pathway guideline for Acute Coronary Syndrome (ACS) followed by the hospital, we developed a Care Process Monitoring application that tracks patients through the clinical care process and monitors for bottlenecks where patients were waiting or services were taking longer than expected. We also evaluated the Web Application implementation used in phase 1 of the project (Baarah & Peyton, 2012) to serve as a basis for comparison with our Care Process Monitoring BI Application. In section 4.1, we briefly describe the ACS Clinical Pathway monitored in our case study. In section 4.2, we describe our experiment. In section 4.3, we describe the Web Application implementation of the scenario, its data integration architecture, some implementation constraints and its ability to produce the real-time BI reports. In section 4.4, we describe in detail the implementation of our Care Process Monitoring BI Application, its data integration architecture, the data model and real-time Portal.

4.1. ACS Clinical Pathway

The work for this research was conducted in partnership with IBM, and the William Osler Hospital Strategic Information and Performance Systems (SIPS) Research Program. The ACS clinical pathway was chosen because ACS patients require care within a short window of time. Monitoring and reducing the wait times for these patients is particularly important because the patient may require an urgent cardiac re-

vascularisation which must be completed within 90 minutes of the event to be effective. It is important that care-givers complete the required service (e.g. consultation, lab tests, etc.) quickly so the patient can be diagnosed and receive the required procedure in time.

A key performance indicator for the hospital is the overall duration of the process from triage until procedure completion. This duration is usually referred to as the time from door-to-balloon. As part of its implementation of the ACS clinical pathway, the hospital has target duration times for each step in the process from when the patient enters the emergency room to when the procedure is performed until when the patient is discharged. It is critical that each state's duration is accurately measured to be able to compute the time the patient spent in that state to confirm it is below the threshold. A key objective of the monitoring application is to measure the durations of these low level process states and identify where bottlenecks are occurring that might impede the timely delivery of care for cardiac patients.

Currently, the hospital uses data warehouses to measure performance metrics for its processes, like the overall length door-to-balloon time, in historical reports days or even weeks after the fact. They do not have real-time analytics available to them during the delivery of care in order to inform their reactions to perceived bottlenecks. Also these data warehouses are unable to provide the fine-grained metrics required to determine the root cause of a problem. For instance, if high patient volumes were experienced in the ED was it as a result of an inadequate number of care providers or was it as a result of beds not being available in the CW? Knowing the root cause may help in taking the correct decision to solve the problem. The managers need real-time fine-grained metrics to be

able to make the right decisions while care is being delivered. In order to provide these wait times and service times in real-time, the information system should be capable of collecting and integrating data from the disparate sources, computing the analytics and displaying them as the events are taking place.

4.2. The Experiment

The flow of cardiac patients through the Emergency department (ED) and the Cardiac Care Unit (CCU) was modeled as a care process with states in order to create an application to monitor both service and wait times in real-time. In order to measure the duration of each state, events are correlated from BPM and RTLS sources to precisely determine start and end of the key process states for the ACS clinical pathway.

Figure 14 illustrates the correlation of the main wait and service states for the ACS clinical pathway with events from either BPM or RTLS. The figure depicts a subset of the application model for monitoring the ACS clinical pathway that was developed in collaboration with hospital experts to track cardiac care patients' progress from the emergency department (ED) till they are discharged. The RTLS server communicates events that identify the location of patients and care providers based on RFID readers in each room and RFID tags worn by the patients and care providers. The BPM server communicates events that indicate when key administrative steps in the process have been performed (patient triaged, tests ordered, bed ordered etc.). The states after *In Procedure* are in the model, because how long one waits for a bed is affected by how

long patients stay in their beds, so we model the states right up to where the patient is discharged.

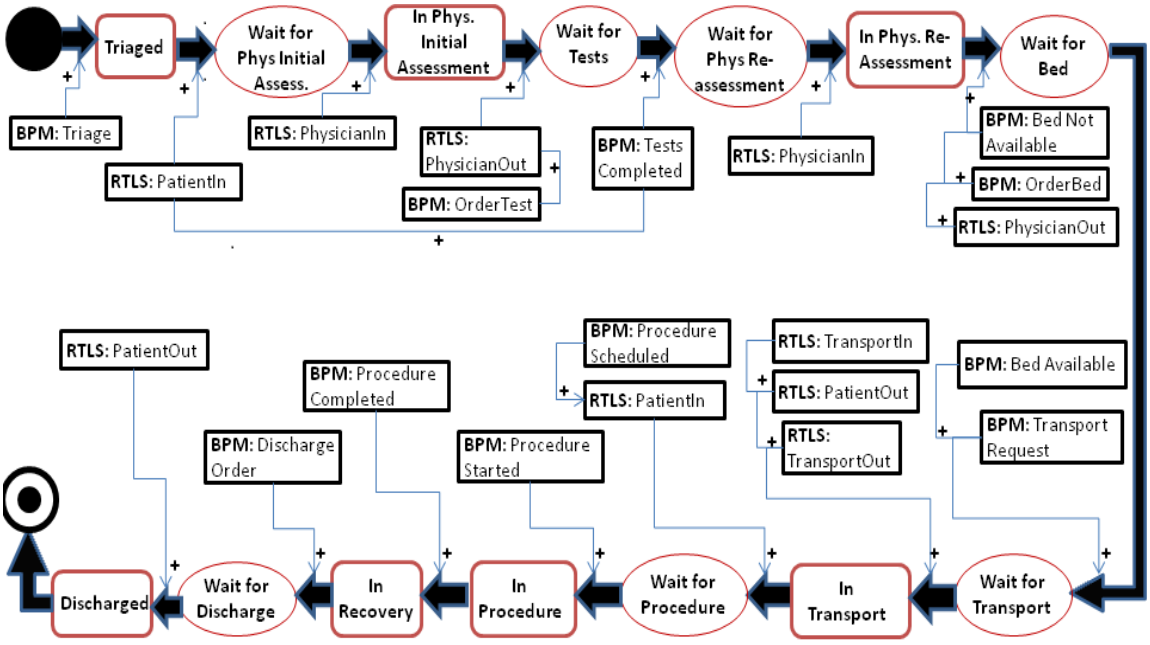


Figure 14: Care Process States for Cardiac Patient

The process begins when the new cardiac patient arrives in the ED for triage. The patient undergoes several wait states (*Wait for Physician Initial Assessment*, *Wait for Tests*, *Wait for Physician Re-Assessment*, *Wait for Bed*, *Wait for Procedure*, *Wait for Discharge*) and service states (*In Physician Initial Assessment*, *In Physician Re-Assessment*, *In Procedure*) as they are receiving care. These wait and service states affect the amount of time a patient spends at the hospital and can be used to pinpoint exactly where bottlenecks are occurring that affect the overall performance of the hospital.

A key requirement is to obtain the analytics in real-time to assist in patient flow management decisions. However, the analytics obtained for the existing manual ETL approach is often incomplete and received after the fact. A Care Process Monitoring

application is needed which receives these source events, computes the analytics and displays the result in real-time. We describe in this case study how we developed a generic Care Processing Monitoring application that was a significant improvement over the initial web application used in the case study.

4.3. Initial Web Application

Originally, in the first phase of the case study, a Grails web application was developed for monitoring the ACS clinical pathway. Figure 15 is a sample real-time report from this web application. The report shows the states and durations along with the defined target durations for each state the patient has been through. The report is in reverse chronological order so the most recent state is shown first. It shows the progress of the patient through the care process from being TRIAGED on arrival to the ED through to the current state of IN_BED_CW in the cardiac ward. The patient has not yet reached the state of a cardiac procedure being performed or the state of being discharged.

The green-coloured rows represent states where the measured duration is within the acceptable range (i.e. actual duration < 2/3 * target duration). The rows are flagged yellow where the actual duration is near the limit (i.e. 2/3 * target < actual duration < target) and red where the actual duration exceeds the target (i.e. actual duration > target). Rows without color-coding have not had a 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 15: Patient State Duration versus Target Duration in Web Portal

4.3.1 System Architecture

Figure 16 illustrates the system architecture that provided data integration in support of the care process monitoring application. This application received start and end state events generated by an IBM Business Events 8 CEP server. The IBM Business Events 8 CEP server correlates events that originated from an IBM BPM server, and an Ekahau RTLS server, through an IBM Message Broker, and based on rules to generate start and end events for each state. These CEP generated events were forwarded to the Grails Web Application for further processing. Inside the Web Application, there is a Web Service (WS) that uses Active Message queue to cache all events received and dispatch them to corresponding specific hard-coded event handler for each type of event to infer states, update entities and compute metrics related to wait times and care service times. Alerts are raised when wait times approach or exceed specified targets. The Web Application logs information related to the objects into a MySQL database via an ORM

engine (Hibernate). The states and wait time metrics are displayed in real-time in the Web Application to provide a real-time view of the care process and all current patients.

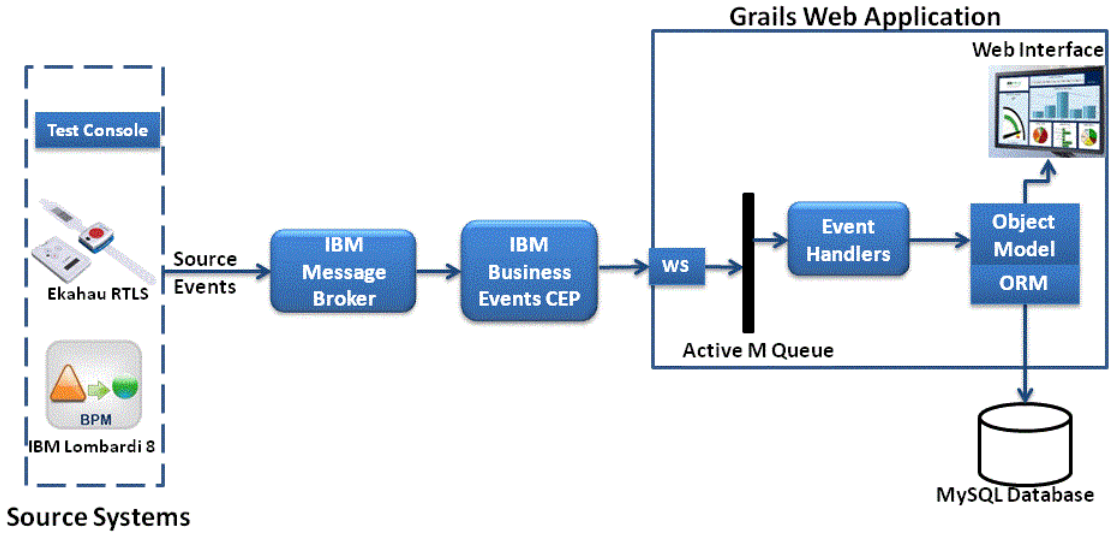


Figure 16: Care Processing Monitoring Web Application Implementation Architecture

4.3.2 Event Processing

Event processing was done in both the CEP and the Grails web application. The CEP used its event inference rules to produce higher level start and end events. However, these event rules had no built-in support for state. The web portal further processed these CEP generated events using its event-based handlers to infer the patient states and interact with the Object Model. In the web portal application each event had a corresponding event handler, thus, there was a one-to-one correlation between the types of events and the number of event handlers. A total of 32 event handlers were required to handle the entire ACS clinical pathway. Every time a state was added, modified, or deleted as the model of the ACS clinical pathway was refined, all CEP rules and all 32 event handlers had to be reviewed to see if they were affected by the change. Further,

every time an event was added, modified, or deleted all CEP rules had to be reviewed for possible side effects in all possible states of the ACS clinical pathway.

4.3.3 Event Inference Rules

Below is a sample event inference rule for generating the event *Wait for Transport Started* in the CEP.

Interaction Set: Patient Wait for Transport Related by Patient.Patient_ID
In response to: TransportRequest From BPM
Immediately
If This Event Follows WaitForBedSync
AND
If This Event Follows PatientAdmittedWithBed
Then: Immediately WaitForTransportStarted

There are actually several rules in combination that respond to the set of events that correlate to the start of the *Wait for Transport* state. The rules are sensitive to the order events occur in, so the rule requires that the events occur in a particular order. E.g. both the events *WaitForBed* and *PatientAdmittedWithBed* must be received before the event *TransportRequest* for the rule to be triggered. If a network delay causes the *TransportRequest* event to be received before the *PatientAdmittedWithBed* event, then the *WaitForTransport* event will not be generated.

In order for the CEP to handle these anomalies, the rules have to be significantly more elaborate, so intermediate events like *WaitForBedSync* are used to coordinate. Basically, one needs different rules to handle all possible combinations of event orderings. As well, since a state is not actually inferred or managed, the Web Application had to process these start and end events and maintain its own representation of the care process as a sequence of states.

4.3.4 The Data Model

The Web Application used an ORM generated MySQL database to log information related to the objects in the application. This database is not particularly optimized for query performance or reporting and was only accessible through the web application.

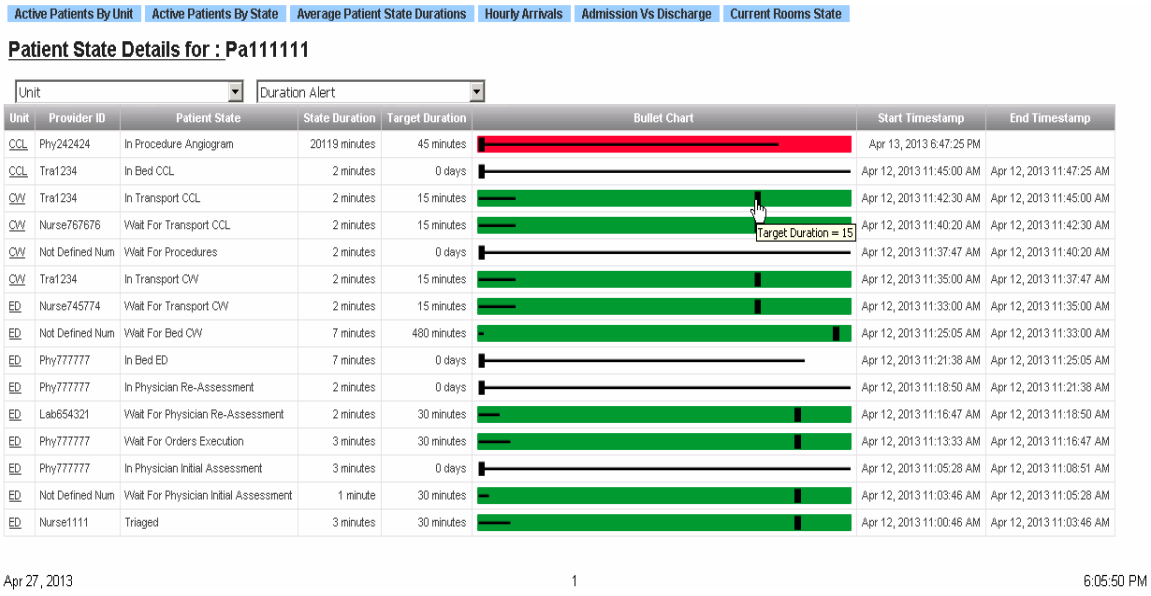
4.4. Care Process Monitoring BI Application

In phase 2 of our case study, we replaced the web application with a generic Care Process Monitoring BI application. Amongst other things, the new generic BI application used IBM Cognos 10 Business Intelligence suite to create a real-time BI Portal. Figure 17 is a sample real-time report from the BI Portal. The report improves on the similar report from the Web Application shown in Figure 15. It shows the same states and durations along with the defined target durations. In addition, this report shows the unit of the patient and offers the ability to drill down from the unit to individual rooms. The report is also in reverse chronological order with the most recent state shown first so the current state is *In Procedure Angiogram*.

The report also shows a bullet chart that indicates the target duration and the measured duration. The black rectangular vertical bar represents the target duration, while the black line is the measured duration. This bullet chart is coloured green, yellow or red depending on whether the measured states duration is within the acceptable range, is near the limit or has exceeded the target. Rows without color-coding have not had a time limit specified. The real-time flagging together with the estimation of the target and measured

durations on the bullet chart, enables care providers to see where the target durations are near the limit or exceed the limit. This enables care providers to quickly identify in real-time situations where the quality of care for an individual patient may be compromised. For example, in this case there was an unusual long wait in the procedure angiogram. Care providers can drill down from unit to room to obtain see the room the patient is in and quickly resolve the problem if this patient was waiting for a service.

A detailed analysis of the reporting features available in the BI Portal is given in section 4.4.6. The database schema of the OLAP database used by the BI Portal is given in Appendix A.



Unit	Provider ID	Patient State	State Duration	Target Duration	Bullet Chart	Start Timestamp	End Timestamp
CCL	Phy242424	In Procedure Angiogram	20119 minutes	45 minutes		Apr 13, 2013 6:47:25 PM	
CCL	Tra1234	In Bed CCL	2 minutes	0 days		Apr 12, 2013 11:45:00 AM	Apr 12, 2013 11:47:25 AM
CW	Tra1234	In Transport CCL	2 minutes	15 minutes		Apr 12, 2013 11:42:30 AM	Apr 12, 2013 11:45:00 AM
CW	Nurse767676	Wait For Transport CCL	2 minutes	15 minutes		Apr 12, 2013 11:40:20 AM	Apr 12, 2013 11:42:30 AM
CW	Not Defined Num	Wait For Procedures	2 minutes	0 days		Apr 12, 2013 11:37:47 AM	Apr 12, 2013 11:40:20 AM
CW	Tra1234	In Transport CW	2 minutes	15 minutes		Apr 12, 2013 11:35:00 AM	Apr 12, 2013 11:37:47 AM
ED	Nurse745774	Wait For Transport CW	2 minutes	15 minutes		Apr 12, 2013 11:33:00 AM	Apr 12, 2013 11:35:00 AM
ED	Not Defined Num	Wait For Bed CW	7 minutes	480 minutes		Apr 12, 2013 11:25:05 AM	Apr 12, 2013 11:33:00 AM
ED	Phy777777	In Bed ED	7 minutes	0 days		Apr 12, 2013 11:21:38 AM	Apr 12, 2013 11:25:05 AM
ED	Phy777777	In Physician Re-Assessment	2 minutes	0 days		Apr 12, 2013 11:18:50 AM	Apr 12, 2013 11:21:38 AM
ED	Lab654321	Wait For Physician Re-Assessment	2 minutes	30 minutes		Apr 12, 2013 11:16:47 AM	Apr 12, 2013 11:18:50 AM
ED	Phy777777	Wait For Orders Execution	3 minutes	30 minutes		Apr 12, 2013 11:13:33 AM	Apr 12, 2013 11:16:47 AM
ED	Phy777777	In Physician Initial Assessment	3 minutes	0 days		Apr 12, 2013 11:05:28 AM	Apr 12, 2013 11:08:51 AM
ED	Not Defined Num	Wait For Physician Initial Assessment	1 minute	30 minutes		Apr 12, 2013 11:03:46 AM	Apr 12, 2013 11:05:28 AM
ED	Nurse1111	Triaged	3 minutes	30 minutes		Apr 12, 2013 11:00:46 AM	Apr 12, 2013 11:03:46 AM

Figure 17 Patient State Durations versus Target Durations in BI Portal

4.4.1 System Architecture

Figure 18 below details the implementation architecture for the system. We replaced the Grails application web application with a Cognos BI Portal which is driven off a Framework Manager (FM) OLAP data mart accessing a star schema MS SQL

database. We replaced the external CEP and internal event handlers of the Web Application with an integrated State Monitoring Engine (SME) and state handler architecture that we developed for inferring and updating care process states directly based on an application model.

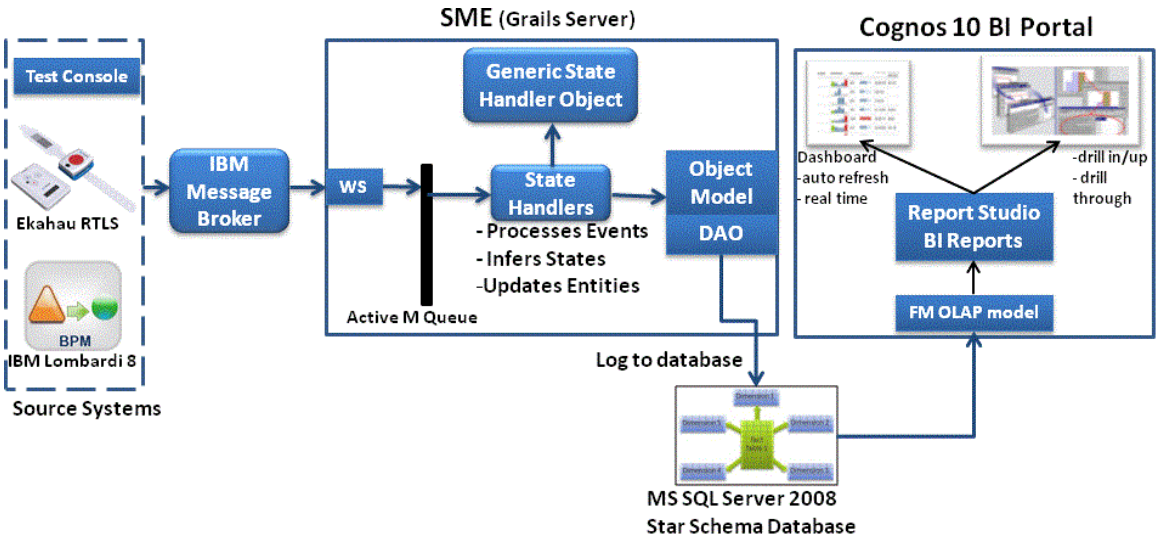


Figure 18: Implementation Architecture for ACS Case Study

The SME logs state and event related data in the OLAP modelled MS SQL Server 2008 database through a DAO. Communication between the SME and all source systems is via its SOAP web service interface.

4.4.2 State Monitoring Engine

The SME processes source events and directly infers patient states using its state-based handlers. After the Web Service caches incoming events into the Active Message Queue, a message service object forwards the messages to the appropriate state handler based on the current state of the patient. Figure 19 below shows code snippet from the Message Service.

```
class MessageService {
    // MessageService is listening to the JMS queue
    def jmsService
    static transactional = true
    static expose = ['jms']
    def Patient patient
        ...
    @Queue(name='SME_Event')
    def SME_EventArrive(msg){
        def props = processMsg(msg)
        def patientId = props['Patient_ID']
        patient = Patient.findByPatientID(patientId)
        String state = patient.getCurrentState()
        switch (state) {
                case 'TRIAGED':
                    EventHandler h = ctx.getBean("triagedstateHandler")
                    h.handle(props)
                    break
                case 'WAIT_FOR_PHYS_INIT_ASSESS':
                    EventHandler h = ctx.getBean("waitforconsultation1stateHandler")
                    h.handle (props)
                    break
                ...
            }
        }
    }
```

Figure 19: Code Snippet from the Message Service

The *getCurrentState()* method retrieves the current state of the patient while the switch statement is used to dispatch the event to the state handler responsible for handling events that occur in that state. Each state handler subscribes to the set of events which are relevant to it. Table 3 below lists all the state handlers and the list of events they subscribe to. A total of 19 state-based handlers were required to handle the entire ACS clinical pathway.

TABLE 3: STATE HANDLER - EVENT SUBSCRIPTION MAP

State Handler	Events Subscribed to
NoStateHandler	TriageScore
TriagedStateHandler	PatientInED
WaitForPhysInitAssessStateHandler	PhysicianInED
InPhysInitAssessStateHandler	PhysicianOutED
InBedEDStateHandler	OrderRequest, PatientAdmittedWithNoBed, PatientAdmittedWithBed, ProceduresScheduled, PatientTransportRequest
WaitForTestsExecutionStateHandler	OrderRequest,OrderRequestCompleted,PhysicianOutED
WaitForPhysReAssessStateHandler	PhysicianInED
InPhysReAssessStateHandler	PatientAdmittedWitthNoBed, PatientAdmittedWithBed, ProceduresScheduled, PhysicianOutED,
WaitForBedCWStateHandler	PatientAdmittedWithBed, ProceduresScheduled
WaitForTransportCWStateHandler	PatientOutED, TransportInED, TransportOutED, PatientOutCCL, TransportInCCL, TransportOutCCL, ProceduresScheduled
InTransportCWStateHandler	PatientInCW, TransportInCW, TransportOutCW, ProceduresScheduled
InBedCWStateHandler	ProceduresScheduled, PatientTransportRequest
WaitForProceduresStateHandler	PatientTransportRequest
WaitForTransportCCLStateHandler	PatientOutCW, TransportInCW, TransportOutCW
InTransportCCLStateHandler	PatientInCCL, TransportInCCL, TransportOutCCL
InProcedureStateHandler	ProcedureCompleted
InBedCCLStateHandler	ProcedureStarted, PatientTransportRequest
InConsulation3StateHandler	PhysicianOutCW, DischargeRequest
WaitForDischargeStateHandler	PatientOutCW, PhysicianOutCW
WaitForProceduresStateHandler	PatientTransportRequest

4.4.3 State Event Processing and State Inferencing

When the state handler receives an event it subscribes to, it processes it and directly infer states and update entities. The state handler also ignores events it does not subscribe to. The code snippet below is shows the event processing mechanism in the TriagedStateHandler.

```
class TriagedStateHandler extends EventHandler{
    def process(Map props){
        evnt = props['event']
        patientId = props['Patient_ID']
        locationId = props['Location_ID']
        timestamp = props['timestamp']

        if(evnt == 'PatientInED'){
            event.eventName = EventName.PatientInED
            patient.roomID = props['Location_ID']

            def patientState = new PatientState()
            patientState.stateName = PatientStateName.WAIT_FOR_PHYS_INIT_ASSESS
            patientState.target = 30
            updatePatientState(patientState)

            return null;
        }
    }
}
```

Figure 20: Code Snippet of Event Processing Mechanism in TriagedStateHandler

The *eventName* attribute of the event object and the r*oomID* attribute of the patient object are first updated. Then an instance of the *patientState* Object is created for the newly inferred state. Key attributes stored by this object are the *patientStateName*, the state's target duration. Table 4 shows the state inference rules used by the State Monitoring Engine to define state transitions from the current patient state based on correlating source events. The State Monitoring Engine rules are state-based and not event-based. In the state-based engine, the current state of the patient as well as the event received determines the inferred state.

TABLE 4: 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
Wait for Tests	Tests Completed	BPM	Wait for Phys. Re-assessment
Wait for Phys. Re-assessment	PhysicianIn	RTLS	In Phys. Re-Assessment
In Phys. Re-Assessment	PhysicianOut OrderBed Bed Not Available	RTLS BPM BPM	Wait for Bed CW
Wait for Bed CW	Bed Available Transport Request	BPM BPM	Wait for Transport CW
Wait for Transport CW	TransportIn TransportOut PatientOut	RTLS RTLS RTLS	In Transport CW
In Transport CW	TransportIn TransportOut PatientIn	RTLS RTLS RTLS	In Bed CW
In Bed CW	ProceduresScheduled PhysicianIn Discharge Request	BPM RTLS BPM	Wait for Procedures In Consultation 3 Wait for Discharge
Wait for Procedures	Transport Request	BPM	Wait for Transport CCL
Wait for Transport CCL	TransportIn TransportOut PatientOut	RTLS RTLS RTLS	In Transport CCL
In Transport CCL	TransportIn TransportOut PatientIn Procedure Started	RTLS RTLS RTLS BPM	In Procedure
In Procedure	Procedure Completed	BPM	In Bed CCL
In Bed CCL	Transport Request	BPM	Wait for Transport CW
In Consultation 3	PhysicianOut	RTLS	In Bed CW
Wait for Discharge	PatientOut	RTLS	Discharged

In our state handler architecture, each state handler has a set of rules which are used to infer a new state from a set of received events. For instance, if the patient is in the *Triaged* state and the event *PatientIn* is received from the RTLS, the patient is transitioned into *Wait for Phys Initial Assessment* state. Below is a sample rule for transitioning patient to the state *Wait for Transport*.

If CURRENT_STATE is *Wait for Bed*
AND EVENT *Bed Available* received
AND EVENT *TransportRequest* received
Then CURRENT_STATE is *Wait for Transport*

One advantage of having a state-based model is regardless of the order in which events are received the patient states are transitioned accurately. It is not dependent on the order in which events are received so long as all the required events are received.

4.4.4 Object Model to OLAP Model transformation

As represented in Figure 18, as the state handlers process events, they update the object model. At the same time, the state handlers make the appropriate calls to the DAO to log events and state updates to the FM OLAP model stored in the star schema database created with MS SQL Server 2008. The state handler knows the attributes which define the current state. This function call to the DAO to log events in the patient_event_fact and the patient_state_fact is made as events are processed by the state handler. The code snippet below shows calls to the DAO to log the relevant attributes from the *Triaged* state as a patient_state_fact in the OLAP database.

```
Class TriagedStateHandler extends EventHandler{
...
/*** Create an array string called data hold the information to be logged **/

String[] data = [patientId, patientStateId, providerId, procedureId,
locationId, startTimestamp, endTimestamp, duration, currentStateFlag]

/****** Calling DAO to log the state in the Patient State Fact **********/
       QueryBuilder qb1=new QueryBuilder();
       int ii=qb1.buildQueryPatientStateFact(data);

return null;
}
```

4.4.5 Care Process OLAP Data Model

The OLAP Data Model contains 12 tables: 2 fact tables and 10 dimensional tables. The particular dimensions, attributes chosen and measures recorded in each data mart were validated with domain experts and hospital staff who collaborated on the scenario. Table 5 below lists all the database tables, the table type and their function.

TABLE 5 DATABASE TABLES AND THEIR FUNCTION

Table Name	Type	Purpose
active_patient_dim	Dimension	Contains flags to indicate whether the patient is active (not discharged) or inactive (discharged). Enables dimensional modeling and filtering based on active or inactive patients.
current_flag_dim	Dimension	Contains flags to indicate which the current state of the patient or room. It is set to 1 for the current state, and 0 for all previous states. It enables dimensional modeling and filtering based on current or previous states
date_dim	Dimension	Created using Kimball's Date Dimension script **(Kimball Group, 2008)**. Contains date descriptive attributes for 6 years (from 2011 to 2016). It enables dimensional modeling and filtering based on date
hour_dim	Dimension	Contains the hours of a day. Used to enable dimensional modeling based on hour of the day.
overdue_dim	Dimension	Contains flags to which indicate the patient state duration as compared with the acceptable threshold (target). Has 3 flags overdue (duration>target), near overdue (2/3target<duration<target) and within target (duration<2/3target). Supports dimensional modeling, filtering and alerts based on the time spent in the state.
patient_dim	Dimension	Contains all the descriptive information about each Patient. Enables dimensional modeling and filtering patients
procedure_dim	Dimension	Contains descriptive information regarding all possible procedures for ACS Cardiac Patients. Enables dimensional modeling and filtering based on the patients' procedures.
provider_dim	Dimension	Contains descriptive attributes about the care providers for each service. Allows filtering and dimensional modeling based on the care providers
room_dim	Dimension	Contains descriptive information about each room and their corresponding units. Enables dimensional modeling and filtering based on unit and room
state_dim	Dimension	Contains descriptive information about state transitions for rooms and patients. Supports dimensional modeling and filtering based on each state.
patient_event_fact	Fact	Contains measures related to all the source events received for a patient and the foreign keys which link these measures to the dimensional tables.
patient_state_fact	Fact	Contains the measures related to the state transitions of each patient and the foreign keys which link these measures to the dimensional tables

The database has 2 data marts: each one has a fact table and is modeled in a star schema. Each data mart is modeled and described below. Figure 21 below shows the star schema database model for patient states. It provides detailed analytics about all patient states (past and current), the duration and target duration of each state, and whether they

have exceeded the limit or not. This is useful for both historical and real-time analysis of wait time and service time quality of care metrics.

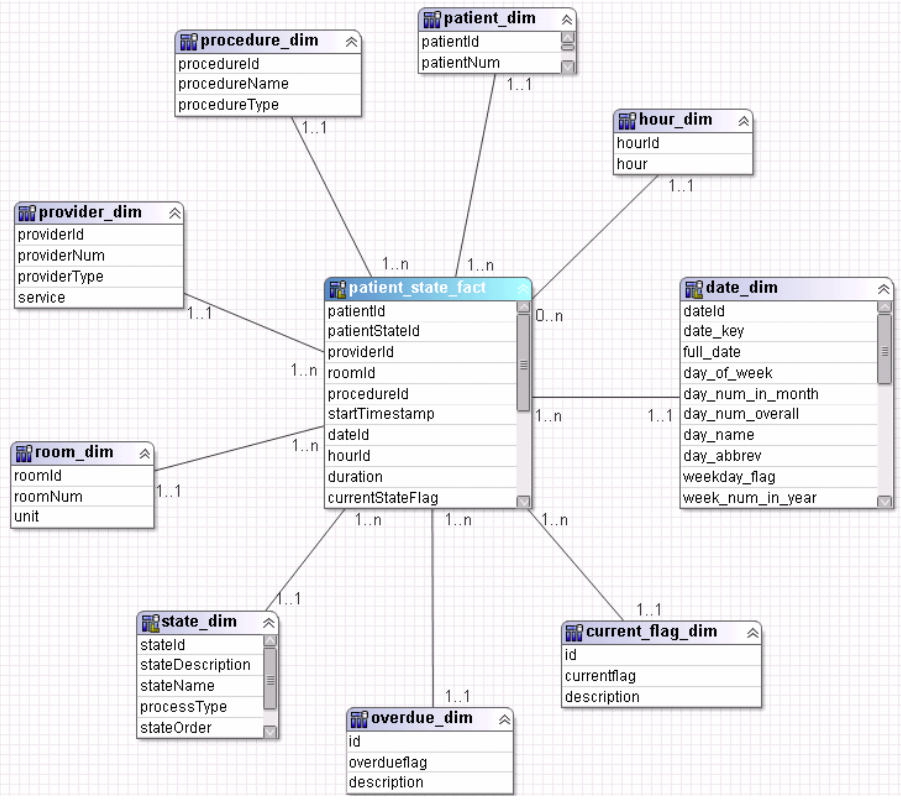


Figure 21 Patient States Star Schema Data Model

In our test database, the patient states data mart had a total of 266 rows in the fact table corresponding to 14 patients and 19 patient states. Each row in this data mart had a total of 41 descriptive attributes in the dimensional tables and 3 measures in the patient_state_fact table.

Figure 22 below shows the star schema database model for patient events and it provides complete history of the event trail for the process for historical analysis and troubleshooting of the event delivery architecture

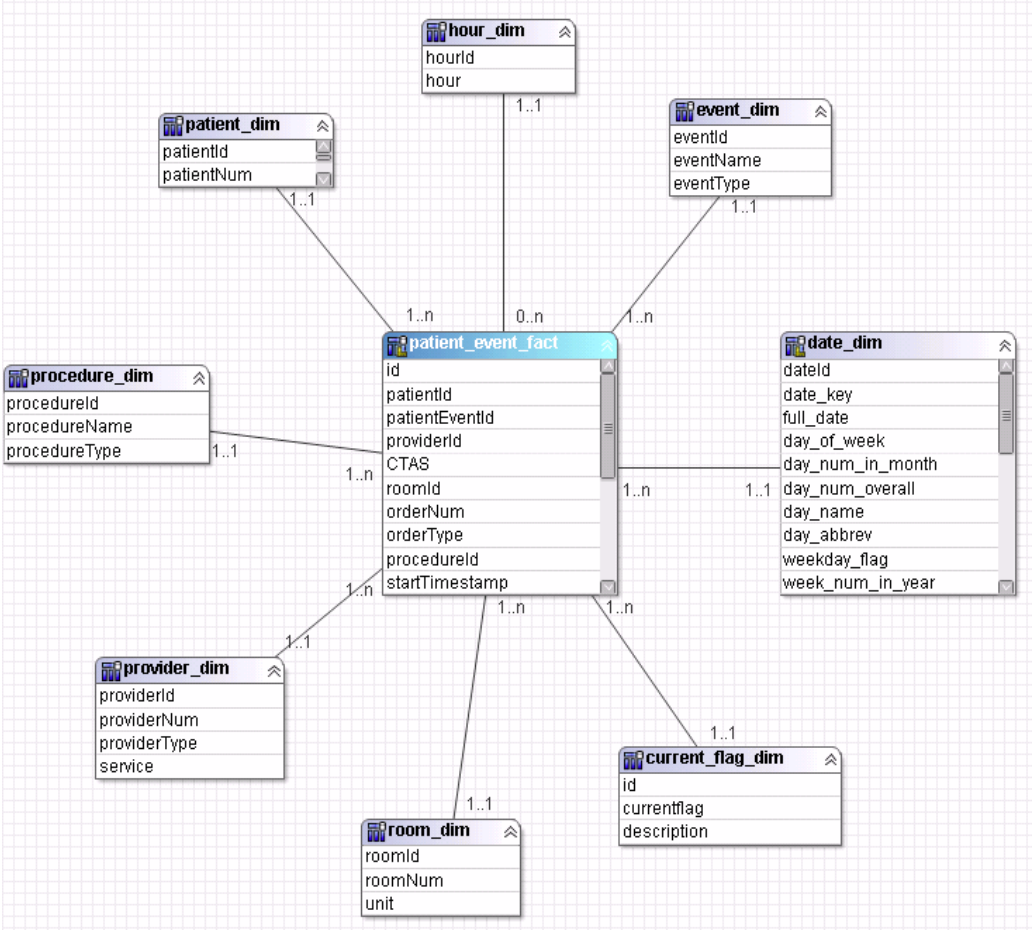


Figure 22: Patient Event Fact Table

The patient events data mart had a total of 448 rows in the fact table corresponding to 14 patients and 32 patient events. Each row in this data mart had a total of 37 descriptive attributes in the dimensional tables and 4 measures in the patient_event_fact table.

We use a Cognos Framework Manager (FM) model to create our dimensional hierarchies to support drill down and drill up. We had 3 levels of modeling: SME Source, SME Logical and SME Dimensional. Figure 23 is a screenshot of framework manager project viewer showing the 3 levels of modeling.

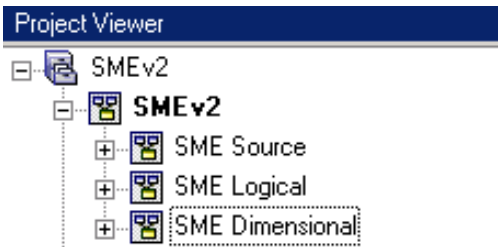


Figure 23: Screenshot of Cognos Framework Manager Project Viewer

The SME Source view contains the data model imported from the MS SQL server 2008 database without any modifications. In the SME logical view, some relationships are modified, calculated measures are created. For example, MS SQL Server 2008 database has no data type to represent time intervals, so all the state durations are computed in the logical view. Creating these measures in the Framework manager model ensures they are available to be used in all reports, rather than creating them in each report. Also in this view, we replace the cryptic field names with more human readable names to ensure reports created are user-friendly. Table 6 summarizes the major modifications made in the logical view.

TABLE 6 SME LOGICAL VIEW MODIFICATIONS

Modification	Reason
Changing relationship between hour_dim and fact tables from 1..n to 0…n	This is required to ensure that all the 24 hours are displayed even when there are no measures for a particular hour.
Calculation of patient state durations	This measure is the difference between the endTimestamp and startTimestamp. For current states which have no endTimestamp, we used the difference between the currentTimestamp on the server and the startTimestamp. The data type for this is timeInterval.
Modification of query item type	This was needed to correctly state which measures are facts, identifiers, or attributes.
Computation of OverDueFlag measures in patient_state_fact table	This measure computes compares the state duration with the threshold (target value) and sets the overdueflag for that state to either overdue (when duration>target), near overdue (when 2/3target<duration<target) or within target (when duration<2/3target)

In the dimensional view, we created our dimensional models and hierarchies to support drill up and down along the dimensions. Figure 24 is a screenshot of the

hierarchy created for the state_dim. The hierarchy created in this dimension enables us to drill down from a particular process or pathway (e.g. ACS, or room process, etc.) to a state in the process (e.g. triaged state). It is also possible to drill up from a particular state in the process. This functionality enables care providers to drill in to fine-grained metrics that show the root causes of problems.

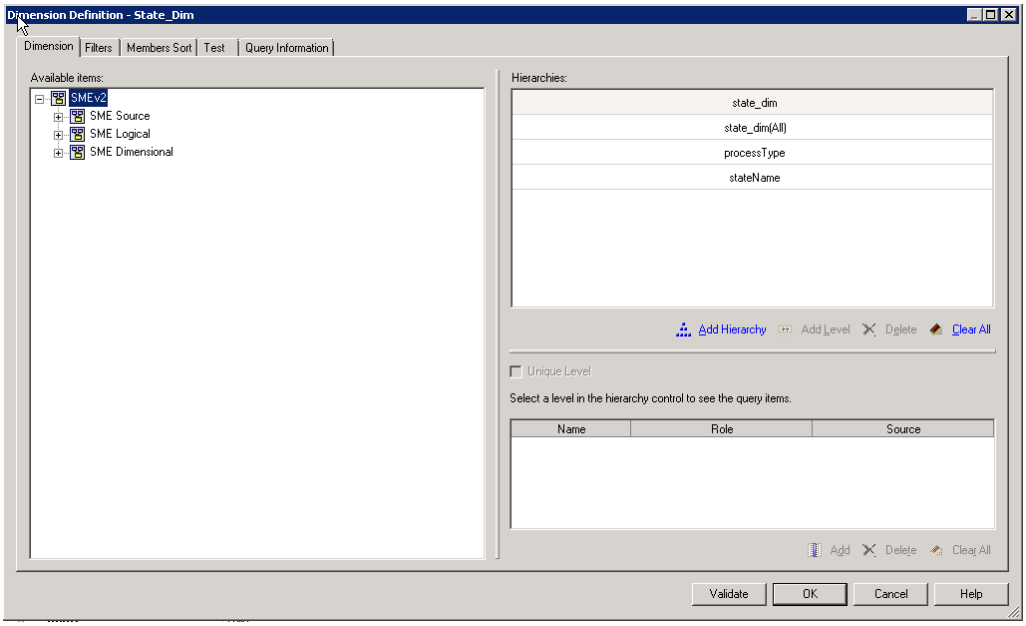
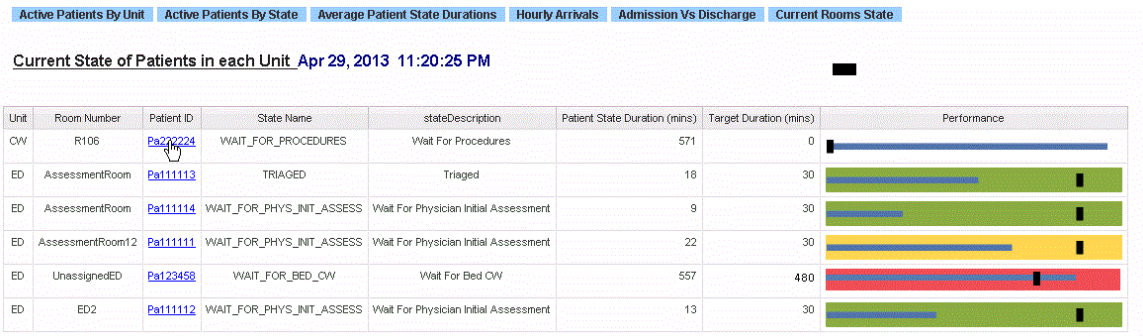


Figure 24: Screenshot of Dimension Definition for State_Dim in Framework Manager

4.4.6 Care Process Monitoring Portal

We use Cognos Report Studio to create the Care Process Monitoring Portal. Figure 25 below is a screenshot of the Patient Tracking Dashboard in the portal. The Patient Tracking Dashboard is auto-refreshed every 30secs and shows "at a glance" the current state of the ACS clinical pathway.

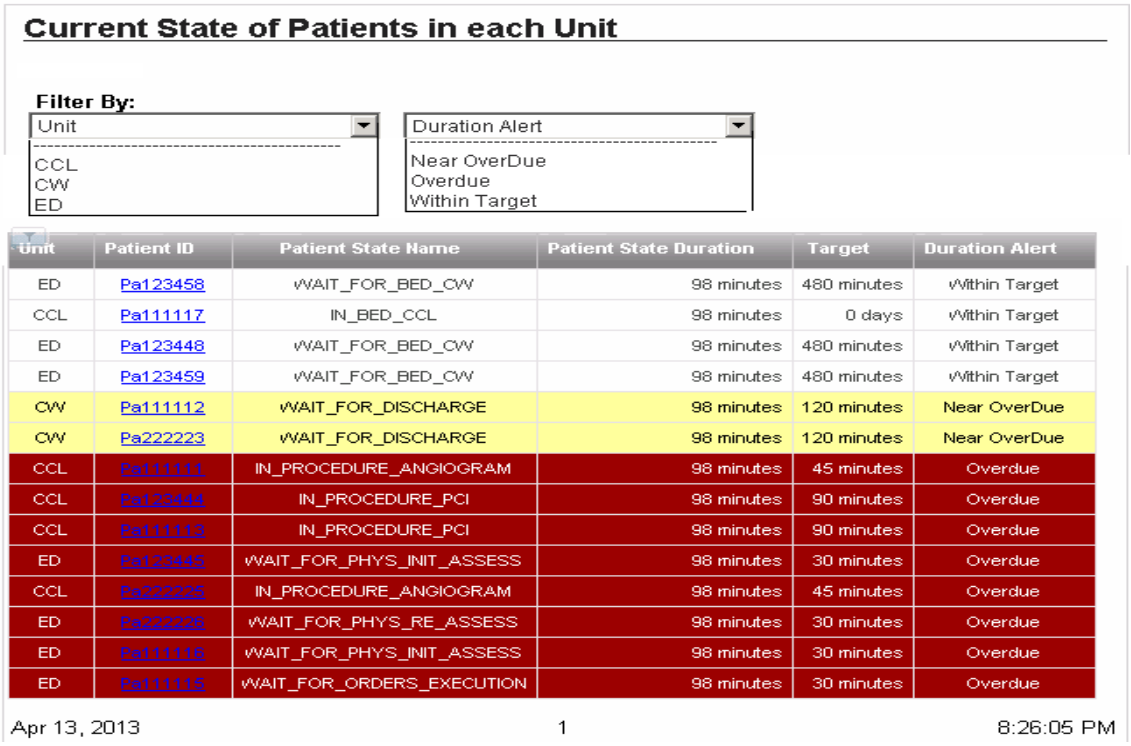


Unit	Room Number	Patient ID	State Name	stateDescription	Patient State Duration (mins)	Target Duration (mins)	Performance
CW	R106	Pa222224	WAIT_FOR_PROCEDURES	Wait For Procedures	571	0	
ED	AssessmentRoom	Pa111113	TRIAGED	Triaged	18	30	
ED	AssessmentRoom	Pa111114	WAIT_FOR_PHYS_INIT_ASSESS	Wait For Physician 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 25: Patient Tracking Dashboard

The report shows all the active patients in the hospital, the current state they are in, the measured and target durations for the state. The report also shows the patient's unit, room number, and a performance bullet chart based on a comparison of the measured and target durations.

Figure 26 shows the current state of patients in each Unit in the fully functional BI Portal. The report allows filtering based on Unit (ED, CW, CCL) and Duration alert (Overdue, Near Overdue or Within Target), while staying in the same page. It allows drill through based on patientId to open the detailed patient states report shown in Figure 28 and drill down (and drill up) based on Unit to identify the current room of the patient.



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	Pa222226	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	Pa111116	WAIT_FOR_ORDERS_EXECUTION	98 minutes	30 minutes	Overdue

Apr 13, 2013 1 8:26:05 PM

Figure 26: Sample Report in BI Portal

A similar report in Figure 27 shows the number of active patients in each state grouped by the measured duration alert. For instance, there are a total of 3 patients in *Wait for Physician Initial Assessment* state; however, 2 are within target while 1 is near overdue. This report offers care providers detailed analytics at a quick glance. They are able to tell if the long wait times are due to a bottleneck which is affecting all patients or if there is only a few isolated incidents.

Hyperlinks are used to link multiple reports together to offer an excellent user experience in navigating reports, similar to the user experience in a web application. The BI portal is also accessible through the Cognos BI native application installed on iPads. This ensures that hospital administrators can access these real-time reports from anywhere and at anytime on these tablets.

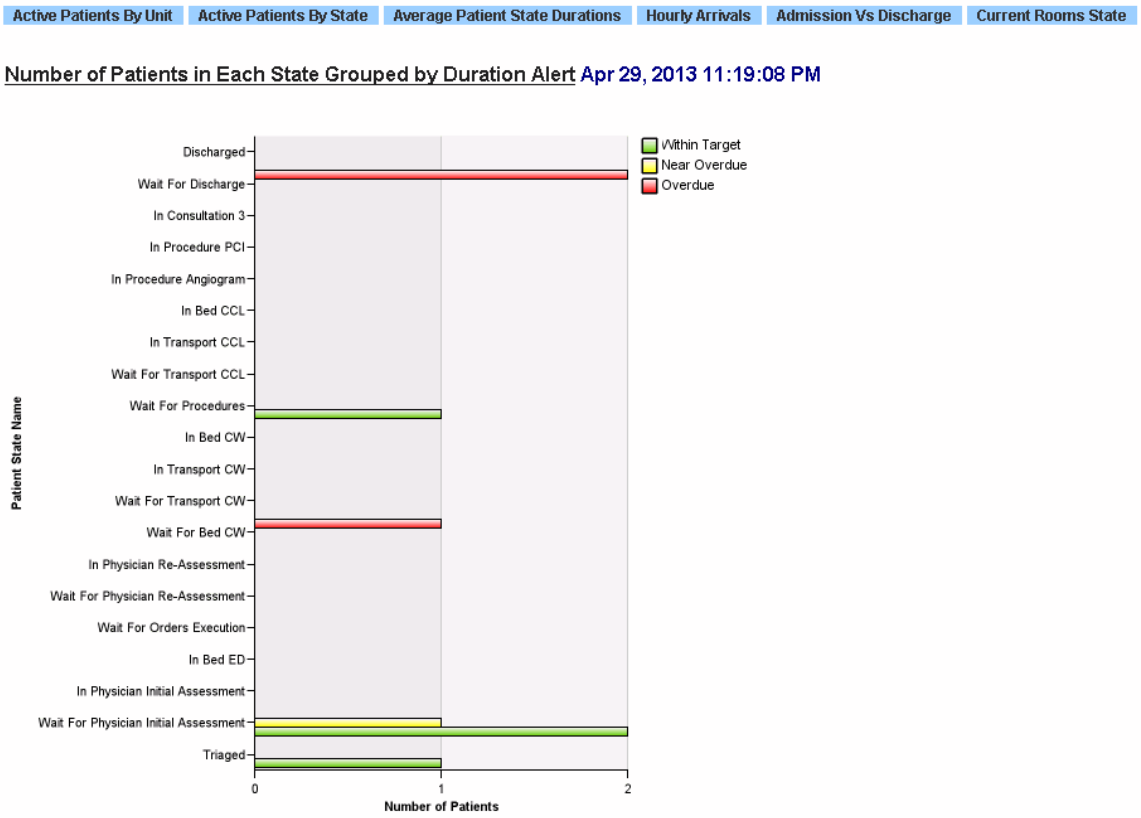
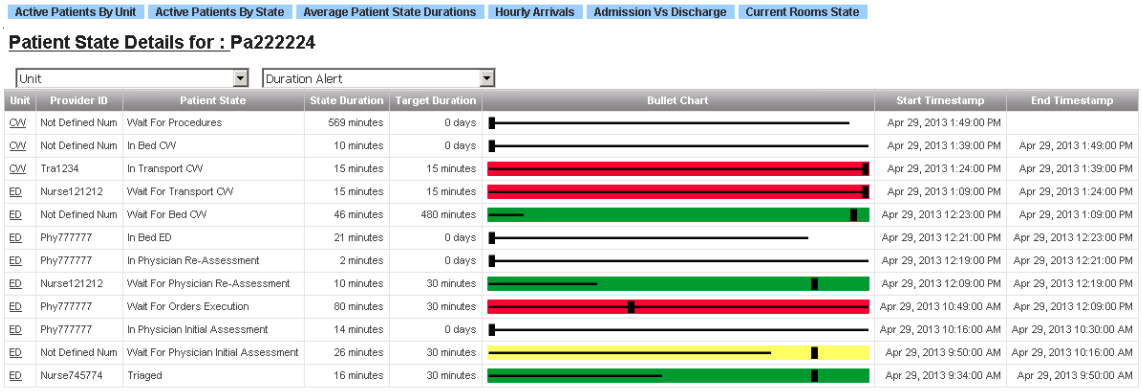


Figure 27: Number of Active Patients in Each State

The Care Process Monitoring Portal allows users to drill in, filter, drill up (aggregate) and drill through. One can click on a particular patient to drill through to that patient's state report. For instance, clicking on the patient Pa222224 in the report shown in Figure 25 produces the detailed patient state report shown in Figure 28. This patient report allows filtering by unit and measured duration alerts (overdue, near overdue, within target).



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 28 Detailed Patient State Report

This ability to drill through, filter, drill up and drill down offers care providers the fine-grained analytics in discovering and solving patient flow bottlenecks. Figure 29 and Figure 30, are additional reports from the BI Portal, which show the hourly distribution of patient arrivals and admissions versus discharges for a particular day.

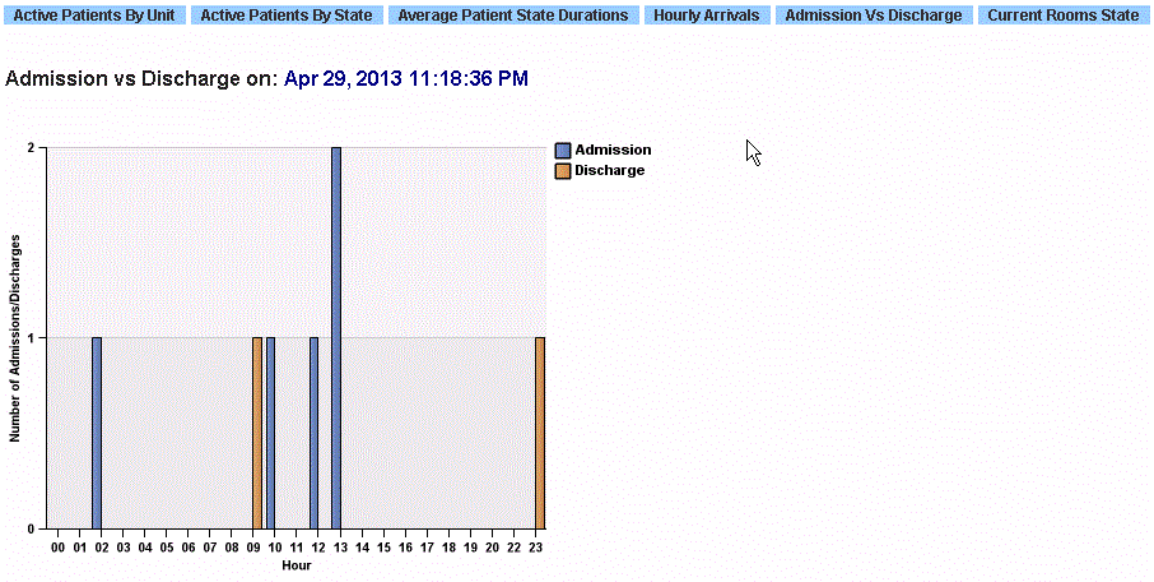


Figure 29: Number of Patient Admissions vs. Discharges by Hour

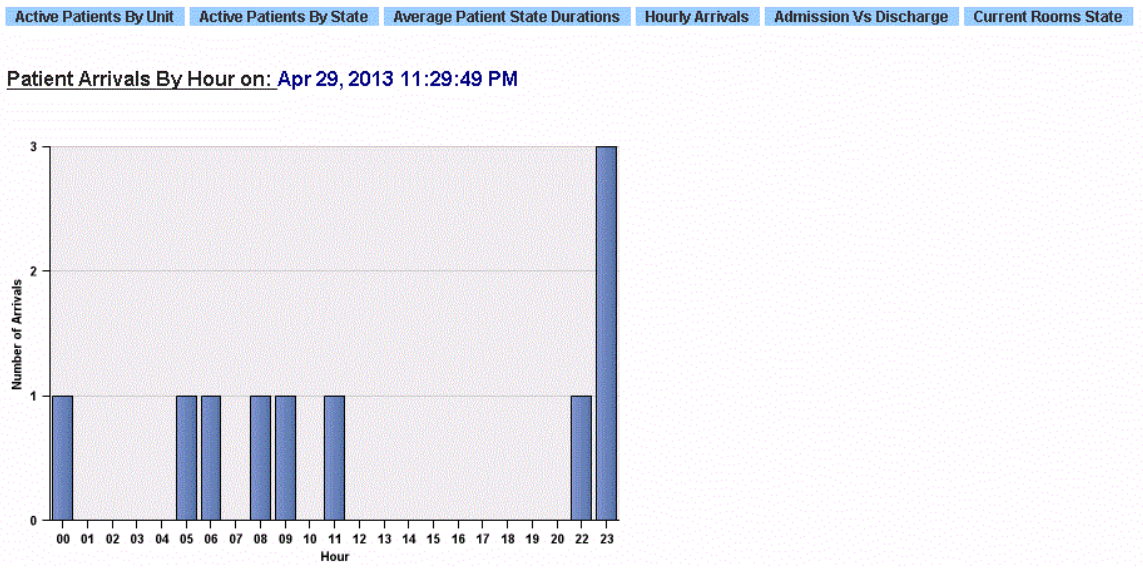


Figure 30: Number of Patient Arrivals by Hour

Chapter 5. Evaluation

In this chapter, we use the results of our case study demonstrations in the ACS clinical pathway to evaluate our approach to providing a generic BI application for Care Process Monitoring. In sections 5.1 to 5.4, we use the evaluation criteria set out in section 3.2 to compare our SME-enabled BI Portal with the CEP-enabled Web Application used in Phase 1 of this project and ETL-enabled Data Warehouse used in the existing Hospital Information System. In section 5.5, we compare our approach to the related works identified in Chapter 2: Agent-based, CEP-based, and Real-Time Capture, Transform and Update data integration. Finally, in section 5.6, we discuss the limitations and assumptions of our approach.

5.1. Complexity

Table 7 summarizes our evaluation for the SME-enabled BI Application as compared with the CEP-enabled Web Application and ETL-enabled Data Warehouse based on complexity.

TABLE 7 COMPLEXITY OF CARE PROCESS MONITORING PORTAL

Feature	ETL-enabled Data Warehouse	CEP-enabled Web Application	SME-enabled BI Application
Complexity of modeling care process elements	No model support.	Verbose and difficult to model situations in rules. 27 event handlers, 27 CEP pre-processing rules, and combinatorial explosion of rule interactions.	Straight forward. Defined by a clear state-based application model. 19 state handlers.
Complexity of data integration rules	ETL processing.	Complex Event Processing have unexpected side effects and event interactions. Combinatorial explosion of interactions between all rules	State handler architecture direct and straight forward. No interactions between rules
Complexity of transformation to database	ETL batch processing with no specific process model in database.	ORM mapping automatic but provides poor aggregation and analytics support.	Object model to OLAP model mapped via DAO.
Complexity of database	Straight forward Star schema database optimized for reporting but no process support.	ORM database not optimize for reporting and requires multiple cross joins between tables when queried	Star-schema database optimized for reporting with process specific data marts.
Complexity of building real-time dashboard	Does not have real-time dashboard	Quite complex Requires interpreting events, custom coding of states and metrics as well as building, expressing and presenting reports.	Standard BI applications

5.1.1 Complexity of modeling care process elements

The ETL-enabled Data Warehouse has no care process model support. On the other hand, the CEP server in the CEP-enabled Web Application and SME in the SME-enabled BI Application performed exactly the same task of correlating events to infer and define process states. However, the CEP does not have built in support for the concept of

care process state. This makes it unnecessarily complex to define and maintain rules for inferring states from source medical events in a care process. In the CEP–enabled approach, there were complex interactions between both the CEP and the Web Application in order to infer a new state. This is because the CEP could not infer states directly from the source events. In particular, the behaviour of our SME is explicitly based on our application model for real-time care process monitoring. As a result, the SME infers states directly from the source events and eliminates the complex interactions required in the CEP-enabled web application. It is, therefore, less complex to model care process elements in the SME-enabled BI Application as compared with the CEP-enabled Web Application.

5.1.2 Complexity of data integration rules

The ETL-enabled Data Warehouse uses ad hoc and complex ETL data integration rules. In the CEP-enabled Web Application approach, Complex Events Processing data integration rules are required. These CEP rules have unexpected side effects and event interactions. For example, the CEP rules are sensitive to the order events occur in, so the data integration rules require that events occur in a particular order. In order to ensure the correct events are generated even when anomalies occur and events arrive in the wrong order, the CEP rules need to be more elaborate and complex. Basically, one needs different rules to handle all possible combinations of event orderings. Also, there were complex interactions between the various event handlers and CEP rules, since more than one event was sometimes required to infer a state, For example in order to infer a Wait for Bed state, interactions were required between the ConsultationCompleted, OrderBed, and BedNotAvailable rules, as well as, the OrderBed, ConsultationCompleted, and

WaitforBed event handlers. A total of 27 CEP event rules were required in the CEP-enabled Web Application, and even then we could not be certain that this application would be robust in the face of all possible event sequencing that might occur.

In contrast, the state monitoring engine in the SME-enabled BI Application is able to manage all aspects of state and metrics using state handlers that infer and define states using the inference rules. The inference rules for the State Monitoring engine were much easier to articulate and manage and we were much more confident that the SME-enabled BI Application would be robust in the face of all possible event sequencing. Since only one state handler was required to infer a new state, no interactions were required between the various state handlers. Also, only 19 state handler rules were needed (proportional to the number of state transitions in the care process).

5.1.3 Complexity of transformation to database

The transformation to the database in the CEP-enabled Web Application is less complex and automatically done by the application's ORM, however, ORM mapping provides poor aggregation and analytics support. In addition, in order to store state information about a particular patient, the ORM–based database required inserting data into multiple interacting tables. The ETL-enabled Data Warehouse uses complex ETL batch processing with no specific process model in database. The SME-enabled BI Application uses object model to OLAP model transformations which are mapped via a DAO to persist data in the database. Though these transformations are not automatic, as in the case of the CEP-enabled Web Application, they are straight forward and provide

good analytics support. Inserting state information about an existing patient involved inserting into only the PatientStateFact table.

5.1.4 Complexity of database

The CEP-enabled Web Application uses a complex ORM database. This database had multiple many-to-many relationships between tables and was not optimized for reporting. As a result, the database requires multiple cross joins between tables when queried. The ETL-enabled Data Warehouse, on the other hand, uses a straight forward star schema database, which various literature on data warehousing validate as optimized being for reporting (Kimball & Ross, 2013). However, it has no support for care processes. The SME-enabled BI Application also uses a straight forward star-schema database, but its database is optimized for reporting on care process states with process specific data marts.

5.1.5 Complexity of building real-time dashboard

The ETL-enabled Data Warehouse does not have a real-time dashboard. The CEP-enabled Web Application requires interpreting events, custom coding of states and metrics as well as expressing and presenting reports to build its real-time dashboard. In comparison, the SME-enabled BI Application requires less complex and standard industry benchmarked Cognos BI tools to build its dashboard.

5.2. Usability

Table 8 below summarizes our evaluation of the SME-enabled BI Application as compared with the CEP-enabled Web Application and ETL-enabled Data Warehouse based on usability.

TABLE 8 USABILITY OF CARE PROCESS MONITORING PORTAL

Feature	ETL-enabled Data Warehouse	CEP-enabled Web Application	SME-enabled BI Application
Ease of aggregating, filtering, drilling up, drilling down and drilling through in dashboard	Yes - Offers ability to drill up (aggregate) and down, drill through, filter but no specific process support.	No - Aggregating and filtering done through custom coding user has little flexibility to customize reports. 9 reports were produced.	Yes - Offers ability to drill up (aggregate) and down, drill through, filter with a process specific OLAP data model. 15 reports were produced.
Access to dashboard at the point of care.	Yes but provides only historical reports	Yes through the web browser on portable mobile devices	Yes - through native app and browser

5.2.1 Ease of aggregating, filtering analytics in the dashboard

In order to evaluate the usability of the application, we used the 'cognitive walkthrough method' proposed by (Jaspers, 2009). This approach involved having about 10 domain experts walking through the interfaces step-by-step and carrying out tasks.

Based on our usability tests, the domain experts unanimously agreed that the SME-enabled BI Application improves upon reporting in the CEP-enabled Web Application, by providing hospital managerial staff and patient flow managers with the ability to drill up, drill down, and drill through the analytics to identify the perceived cause of patient flow bottlenecks. The CEP-enabled Web Application does not offer this flexibility since reports are custom coded and the data is not dimensionally modelled.

The BI tools used in the ETL-enabled Data Warehouse and SME-enabled BI Application, also offer users the flexibility to customize reports by aggregating and filtering, the analytics in the dashboard. The SME-enabled BI Application complements this flexibility by providing a process specific OLAP data model.

5.2.2 Ability to access dashboard at the point of care

The dashboards of all the 3 systems can be accessed at the point of care through web browsers on portable mobile devices. However, the Cognos BI tool used in the SME-enabled BI Application also has a native iPad application which makes it easier to interact with the reports in the dashboard on these mobile devices. In comparison, the ETL-enabled Data Warehouse does not provide real-time reports, so access to the dashboards at the point of care may not help in locating the root cause of current patient flow bottlenecks. The domain experts also validated that the native application on the mobile devices offered a better user experience than the web browser version used in the CEP-enabled web application.

5.3. Scalability

Table 9 summarizes our evaluation for the SME-enabled BI Application as compared with the CEP-enabled Web Application and ETL-enabled Data Warehouse based on scalability.

TABLE 9 SCALABILITY OF CARE PROCESS MONITORING PORTAL

Criteria	ETL-enabled Data Warehouse	CEP-enabled Web Application	SME-enabled BI Application
Technology used to define application	Ad Hoc use of ETL tools, Data Warehouse	Complex Event Processing, Web Application	State Monitoring Engine, Data Warehouse with application specific OLAP Data Model
Effort and Skills to build first application	Expert's knowledge and intensive manual effort to build ETL processes	Web Application coding, Application Model, Expert knowledge and intensive manual effort for CEP rules due to verbose CEP rules. (27 verbose CEP event handler rules required in our scenario). Each report generation requires running multiple queries against the database	Application model (Object and OLAP), SME required less effort since there were fewer rules with no complex interactions. Also BI OLAP data model eliminates the need to run multiple queries against multiple tables to generate a report.
Effort and skills to update an existing application with new states and events	Expert knowledge and intensive manual effort for ETL processes.	Web application coding, expert knowledge and intensive manual effort for CEP rules.	Straight forward SME rules and BI application. 19 state handler rules with no complex interactions
Effort and skills required to build new but similar care process monitoring applications	A lot of effort required to configure batch ETL processing. Expert's skills level needed	A lot of effort required because there are more rules (27) which are also verbose and complex. Expert's skill level needed.	Very little effort since if application similar both the Object model and OLAP model are largely reusable. Also fewer number of rules (19) to implement.

5.3.1 Technology used to define the application

The ETL-enabled Data Warehouse uses ad hoc ETL tools and data warehouse technologies to define the application, while the CEP-enabled Web Application uses a Complex Events Processor and a Web Application. The SME-enabled BI Application, on the other hand, uses a State Monitoring Engine, and Data Warehouse with application specific OLAP Data Model. Due to the use of ad hoc ETL tools in the ETL-enabled Data Warehouse and the Web Application in the CEP-enabled Web Application, these approaches may not be as scalable as the SME-enabled BI application.

5.3.2 Effort and skills needed to build first application

Building the first application using the ETL-enabled Data Warehouse approach requires the skill level of an ETL data cleaner. The engineering effort involves determining the manual processes required to feed data to the operational databases, automating the ETL process to extract data from multiple operational databases, clean and transform the data as well as automatically loading this data into the ETL data warehouse. This ETL approach also requires the creation of a user interface with a BI tool, where the analytics computed can be displayed. Though the skill level required is moderate, an intensive manual effort is required for data entry and data cleaning in the ETL process.

The CEP-enabled Web Application requires the skills of CEP expert, a care process domain expert, as well as a web developer. Expert's knowledge and intensive manual effort is required to set up the CEP server. Also more time and effort need to be spent on understanding and developing robust business rules in CEP. This is because, the CEP rules have to be more elaborate in order to handle anomalies which may occur due to events arriving in the wrong order or incorrect events being sent and also the complex interactions between the various rules. In our scenario, 27 event handler rules were required, and even then we were not confident all anomalies were handled. Furthermore, since the CEP can only produce start and end events, the Web Application developer is required maintain a representation of the care process in the Web Application to process these start and end events in addition to presenting the analytics.

The SME-enabled BI portal, on the other hand, requires the skills of a data warehouse expert, a care process domain expert and a Web Developer. However, the Application model (Object and OLAP), SME rules and BI application are straight forward and do not require intensive manual effort. Also, because the SME has an inbuilt understanding of care process states and is state based, the fewer rules are required to handle the entire scenario (19) as compared with the CEP based approach.

Though all the three systems require different skills, and it is difficult to determine which skill level is higher. However, the SME-enabled BI Application may require less intensive manual effort and time to set up the entire first application.

5.3.3 Updating application with new events and states

After the system has been developed, however, the SME-enabled BI portal is also easier to maintain. In the ETL-enabled Data Warehouse, adding new events may require providing a new system for manual data entry. Additional effort is also required to incorporate these new events into the ETL process.

In the CEP-enabled Web Application, more time and effort is required to update the care process application with new or changed events and state. This is because the number of event rules is proportional to the number of state transitions multiplied the number of possible event sequencings for each state transition. Also since the CEP and event processing rules in this application are order dependent, the entire 27 rules have to be re-assessed with the introduction of more events.

In contrast, in the SME-enabled BI application, adding new patient events only requires updating the rules in the relevant state handler where the event is relevant. Therefore, an increase in the number of events does not correspond to more rules and handlers. Also adding a new event or state may not require any change to the model or BI portal of this application.

5.3.4 Effort and skills required to build similar applications

In general, the experience gained from building the first application makes it easier to build subsequent applications. However, the data cleaning in the ETL-enabled Data Warehouse and the development of the business rules in the CEP-enabled Web Application still require intensive manual effort as compared with the SME-enabled BI Application. This is because the SME-enabled BI Application has fewer rules which do not interact with each other. The SME requires only 1 rule for a state but the CEP requires a dozen interacting rules for each state. Also, if both applications are similar both the object model and OLAP model in the SME-enabled application are largely reusable, while, the CEP requires a reimplementation of code and database model for a new application.

5.4. Real-time Care Process Monitoring Features

Based on our criteria outlined in Section 3.2, Table 10 summarizes our evaluation of BI Portal Features for care process monitoring in each of the three approaches: the existing ETL-enabled Data Warehouse, CEP-enabled Web Application, and SME-enabled BI Application. Features of Care Process Monitoring BI portal as compared with

the Features of Care Processing Monitoring Portal and ETL approaches used in existing HIS.

TABLE 10 CARE PROCESS MONITORING BI PORTAL FEATURES

Feature	ETL-enabled Data Warehouse	CEP-enabled Web Application	SME-enabled BI Application
Application Model	No	Yes- event based. Have to anticipate all combination of event orderings and interactions.	Yes state-based. Not dependent on event orderings or interactions.
Inferring New States from source events	No	Yes –complex event processing. Required pre-processing of source events	Yes – state handler architecture
Update attributes and entities from source events	No	Yes –complex event processing	Yes – state handler architecture
Persist process events and states to support real-time and historical reporting	Persists data only to support historical reporting	Yes, ORM technology	Yes - DAO to OLAP Data model mapping
Compute metrics from source events in real time	No – able to compute metrics but not in real time	Yes – but ORM makes analytics complex	Yes – DAO to OLAP Data model mapping.
Aggregation of states along dimensions	Yes but only for historical reporting.	No - ORM generated data model is not OLAP.	Yes flexible and focused real-time and historical reporting

5.4.1 Application Model

The ETL-enabled Data Warehouse does not have an explication application model of the care process being monitored, thus it has no mapping between the performance measures, process states, critical events. Both the CEP-enabled Web Application and SME-enabled BI application are able to explicitly model the care process to be monitored based on performance measures. However, in the CEP-enabled Web Application approach there is an application object model but neither the CEP nor the Real-time dashboard has an explicit representation of the care process model. In the BI

application, there is both an application object model and an OLAP data model mapping performance measures, process states and critical events.

5.4.2 Inferring care process state transitions

The ETL-enabled Data Warehouse has no clear mapping of care process state transitions and source events. It is unable to infer care process state transitions from these source events. In the CEP-enabled Web Application, the CEP first needs to pre-process the source events to generate start and end state events. These are then further processed by event handlers in the Web Application to define and manage states. In contrast, in the BI application, the SME directly infers care process state transitions from the source process events using its state handlers without requiring any pre-processing or higher level events. Also, the complexity of analyzing event interactions made the CEP rules difficult to maintain. In the SME-enabled BI Application, the state handlers were not affected by event orderings as they could track what events had occurred regardless of ordering. In total, 27 CEP event handler rules were required for the entire scenario, while only 19 state handler rules were required for the SME-enabled application.

5.4.3 Updating Attributes of States and Entities in real-time

Since the ETL-enabled Data Warehouse has no clear mapping between care process states and events; we are unable to update attributes of states and resources from events. The CEP-enabled Web Application required source event pre-processing, so it is only able to update the attributes of existing states and entities from the CEP generated high level events. However, in the SME-enabled BI application, the SME directly updates attributes of existing states and entities from sources events using its state handlers.

5.4.4 Persisting events and states in real-time

In the ETL-enabled Data Warehouse, there are delays in persisting data due to the batch processing in the ETL process. The analytics are usually obtained after the fact and are therefore unable to support real-time reporting. Both the CEP-enabled Web Application and the SME-enabled BI application are able to persist source process events and care process states. In the CEP-enabled Web Application, events and states related to the objects in the application are persisted using an Object Relational Mapping (ORM). In the SME-enabled BI application, events and care process state updates are persisted in an OLAP modeled database.

5.4.5 Computing performance metrics in real-time

The ETL-enabled Data Warehouse requires batch processing of operational data and is therefore unable to support real-time processing of events to compute performance metrics. Both the CEP-enabled Web Application and the SME-enabled BI application support real-time computation of performance metrics.

5.4.6 Communicating care process states along dimensions

The ETL-enabled Data Warehouse offers the ability to communicate information and resources along several dimensions. However, it has no in-built understanding of care process states, so it cannot communicate care process states along dimensions. In addition, since ETL process is not done in real-time, this approach cannot support real-time reporting.

The CEP-enabled Web Application uses an ORM database which is not dimensionally modeled; as a result, it is unable to support the communication of the

status of care process states along dimensions. The user, therefore, has very little flexibility to customize reports displayed in the web application approach.

The SME-enabled BI application, however, leverages the architecture of the ETL-enabled Data Warehouse and uses the application model to provide the details of the care process states. The SME-enabled BI application, therefore, has dimensionally modeled care process states, events and resources. Also since data is persisted in the OLAP modeled database in real-time, this application is able to communicate status of the care process states and resources along several dimensions for flexible and focused real-time reporting.

5.5. Care Process Monitoring Application Approaches

In this section we compare our SME-enabled BI Application with other technological approaches proposed in literature. Table 11 below compares our work with other systems proposed in literature.

TABLE 11 COMPARISONS BASED ON CARE PROCESS MONITORING APPLICATION APPROACHES

Criteria	***Agent Based***	***CTU***	***BI Application***
Real time care process monitoring features	No real-time data integration No application model	Real-time data integration Not based on care process application model	Real-time data integration Based on application model
Scalability	No	No	Partially
Complexity	Yes	Yes	No
Ease of use	More effort required to display reports	Yes - Offers ability to drill up (aggregate) and down, drill through, filter	Yes - Offers ability to drill up (aggregate) and down, drill through, filter

The main advantages of the agent based system include automated event correlation and logging. However, the system relies on manual paper-based data collection processes. This is not only labour intensive, but makes it impossible to provide real-time analytics and also makes the system unscalable. In addition, this system does not have a care process model for mapping the performance metrics to the care process states we monitor analytics. Also, more effort is required to display the reports in this system because a custom coded interface is used. Finally, this interface also does not allow users flexibility to customise the reports in the dashboard.

The main advantage of the CTU approach is its novel real-time data integration approach. It has a database trigger mechanism that offers support for real-time data processing and performance reporting. However, this system was complex since it required 2 data warehouse – one to hold daily real-time and the other to hold the historical data. The data in both data warehouses were joined together logically using database views before OLAP queries were run. Though this approach provided the real-time performance monitoring, it is not scalable because as the number of records increase creating the database view will not be feasible. Also, this system was used for Enterprise Information Management and may not be suitable for Hospital information Systems since it has no underlying application model to support care process states. In addition, it may not support location-based information required for updating patient states.

The BI portal proposed has the real-time care process monitoring features we require. Though we have not tested the scalability of the systems, architectural choices made lead us to assume that the system is scalable. We used industry benchmarked and

scalable software like IBM Message Broker, and IBM Cognos BI Suite, which are used by Fortune 100 companies. In addition, we implemented DAO pattern, Active Message queue, Star schema database design, which are validated in literature as being scalable. The hard-coded rules in the SME may, however, negatively affect the scalability of our system.

5.6. Limitations and Assumptions

Our research is early design-oriented research that demonstrates the potential of our approach to address problems with existing approaches. In order to conclusively validate our approach, we would need to conduct comprehensive clinical trials at several hospitals and support a wide variety of care processes. This of course has not been done yet. Such trials might also uncover technology adoption barriers like cost, culture and other circumstances which have not been considered in this thesis.

Our research has shown that our approach was effective for monitoring the Acute Coronary Syndrome (ACS) Clinical Pathway. In theory, our approach should work for any care process that fits our application model. The care process must have a performance management goal model with metrics and a state-based event transition model. Based on our interactions with domain experts in other hospitals, most hospitals have clinical pathway guidelines for key processes that fit this model.

Another limitation is, currently almost all hospitals may not have the IT infrastructure to supply the detailed care process events needed at all, let alone in real-time. In order to access the feasibility and practicability of our approach, the hospitals

need to be able to implement the required IT infrastructure to provide these real-time source events.

Currently, the BI reports in our care process monitoring BI application have a response time of about 2secs; however, comprehensive performance testing to determine the number of records which will cause the system to reach its breaking point where the response time exceeds 10secs has not been done.

Chapter 6. Conclusions and Future work

6.1. Conclusions

Managing care processes in real-time is challenging. In this thesis, we introduced a generic care process monitoring BI application that leverages a care process monitoring application model, a state-monitoring engine for care processes, and the BI architecture of ETL data warehouses to provide enterprise reporting in real-time. In particular, our research contributions are:

- A generic care process monitoring BI application: This application can flexibly monitor any care process modeled using the application meta-model for care processes. It is simply a matter of implementing for each state in the care process its own state handler to process events to infer and update states in both the application object model and the care process OLAP data model. Our generic care process monitoring BI application consists of a generic BI portal and Patient Tracking Dashboard, which displays performance measures and reports in real-time; a generic OLAP data model and data mart, which offer us flexible and focused real-time reports by allowing us to communicate the status of the care process states and resources along several dimensions; and a generic state monitoring engine, which is based on a state-based application model for care process monitoring. We compared our BI application with other approaches proposed in literature and our results showed that our BI application was better at closing the gap for providing real-time performance analytics on care process states.

- A generic state monitoring engine (SME): We proposed a State Monitoring Engine for inferring and managing states based on an application model for care process monitoring. The SME introduces a state handler architecture, which can process source clinical events and directly update care process states based on defined state inference rules; and an object model to OLAP model transformation which allows us to produce real-time BI reports by bridging the gap between the operational hospital systems and the performance analytics monitoring system. We performed a detailed comparison of our State Monitoring Engine approach with the CEP based approach used in phase 1 of our case study. The results showed the proposed state-based approach is more robust and accurate since the application model does not depend on the order in which events are received. Fewer rules were required to handle the entire scenario and an increase in the number of events did not translate into an increase in the number of rules and handlers. This makes the new approach more reliable, and easier to maintain as compared with the CEP based approach. In addition, there is a clean separation between the application model and the real-time dashboard.

- We illustrated the potential of our approach using a case study that implements the complete clinical pathway for Acute Coronary Syndrome. Our results show that we are able to obtain better analytics and functionality with our approach than with traditional web applications using complex event processing engines or the other approaches proposed in literature.

6.2. Future Work

More clinical trials are required to ascertain that our approach is feasible and practical. This is the next logical step of the research. Two more Ontario hospitals have expressed an interest in participating. These trials will involve new care processes besides the Acute Cardiac Syndrome clinical pathway that we address in our clinical study.

Rigorous performance testing needs to be performed to ensure our database model is able to offer an acceptable response time (below 10secs) in support of the real-time reporting requirements of our BI application.

References

Abo-Hamad, W., & Arisha, A. (2012, December). Multi-criteria framework for emergency department in Irish hospital. *Proceedings of the 2012 Winter Simulation Conference* , 1-12.

Abrahiem, R. (2007, May). A new generation of middleware solutions for a near-real-time data warehousing architecture. *2007 IEEE International Conference on Electro/Information Technology* , 192-197.

Alur, D., Crupi, J., & Malks, D. (2003). *Core J2EE™ Patterns* (2nd ed.). Sun Microsystems Press and Prentice Hall.

Ambler, S. W. (1997). *Mapping Objects To Relational Databases.* Retrieved 04 10, 2010, from AmbySoft: www.AmbySoft.com/mappingObjects.pdf

Baarah, A., & Peyton, L. (2012). Engineering a State Monitoring Service For Real-time Patient Flow Monitoring. *Proceedings of the 9th Middleware Doctoral Symposium of the 13th ACM/IFIP/USENIX International Middleware Conference* , pp. 1-6.

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, December 6-8). Improving Cardiac Patient flow based on Complex Event Processing. *Applied Electrical Engineering and Computing Technologies (AEECT)* , pp. 1-6.

Barone, D., Jiang, L., Amyot, D., & Mylopoulos, J. (2011). Strategic models for business intelligence. In L. D. Manfred AJeusfeld (Ed.), *The 30th international conference on Conceptual modeling (ER'11)* (pp. 429-439). Berlin, Heidelberg: Springer-Verlag.

Basseda, R., Alinaghi, T., & Ghoroghi, C. (2009, June 3). A dependency based framework for the evaluation of agent oriented methodologies. *IEEE International Conference on System of Systems Engineering, 2009. SoSE 2009.* , 1-9.

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.

Brataas, G., & Hughes, P. (2004). Exploring architectural scalability . *SIGSOFT Softw. Eng. Notes , 29* (1), 125-129.

Bruce Silver Associates. (2006). *The 2006 BPMS report: Understanding and evaluating BPM suite.* BPMInstitute.org.

Chakravarthy, S., & Mishra, D. (1994). Snoop: An Expressive Event Specification Language for Active Databases. *Data & Knowledge Engineering , 14* (1), 1-26.

Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *SIGMOD , 26* (1), 65-74 .

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). IEEE Computer Society.

Clancy, T. R., Effken, J. A., & Pesut, D. (2008). Applications of complex systems theory in nursing education, research, and practice. *Nursing outlook , 56* (5), 248.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002). Unraveling the Web Services Web. *IEEE* , vol. 6, no. 2, pp. 86–93.

Duboc, L., Rosenblum, D., & Wick, T. (2007). A framework for characterization and analysis of software system scalability. *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC-FSE '07)* , 375-384.

Fan, Y., & Bai, X. (2004). Real-Time Business Process Performance Management. *Proc. IEEE International Conference on E-Commerce Technology for Dynamic E-Business* (pp. 341 - 344). Beijing, China: IEEE Computer Society.

Ferrand, D., Amyot, D., & Villar, C. C. (2010). Towards a Business Intelligence Framework for Healthcare Safety. *Journal of Internet Banking and Commerce , 15* (3), 71-80.

Fowler, M. (2003). *Patterns of Enterprise Application Architecture.* Addison Wesley.

Gellersen, H., & Gaedke, M. (1999). Object-oriented Web application development . *Internet Computing , 3* (1), 60-68.

Green, D. T., & Pearson, M. J. (2011, March-April). Integrating website usability with the electronic commerce aceptance model. *Behaviour & Information Technology , 30* (2), pp. 181-199.

Green, D., & Pearson, J. M. (2006). Development of a website usability instrument based on ISO 9241-11. *Journal of Computer Information Systems , 47* (1), pp. 66-72.

Hevner, A. R., March, S. T., Park, J., & Pam, S. (2004). Design Science in Information Systems Research. *MIS Quarterly , 28 (1)* , 75-105.

Huhns, M. N., & Singh, M. P. (2005). Service-Oriented Computing: Key Concepts and Principles. *EEE Internet Computing* , vol. 9, no. 1, pp. 75-81.

Inmon, W. H. (2005). *Building the data warehouse* (4th ed.). Indianapolis: Wiley Publishers.

Jaspers, M. W. (2009). A comparison of usability methods for testing interactive health technologies: Methodological aspects and empirical evidence. *International journal of medical informatics , 78* (5), 340-353.

Jeng, J.-J., Schiefer, J., & Chang, H. (2003). An Agent-based Architecture for Analyzing Business Processes of Real-Time Enterprises. *Proceedings of the 7th International Conference on Enterprise Distributed Object Computing (EDOC '03)* , 86.

Kang, J., & Han, K. (2008). A Business activity monitoring system supporting real-time business performance management. *Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology*, *1*, pp. 473-478.

Kannampallil, T. G., Schauer, G. F., Cohen, T., & Patel, V. L. (2011). Considering complexity in healthcare systems . *Journal of Biomedical Informatics , 44* (6), 943-947.

Kimball Group. (2008, December 01). *Date Dimension Generator - Kimball Group.* Retrieved February 03, 2013, from Kimball Group: www.kimballgroup.com/wp-content/uploads/2012/06/Ch10-DateDim.xls

Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling* (3rd ed.). John Wiley & Sons.

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. (2008). *The Data Warehouse Lifecycle Toolkit* (2 ed.). Indianapolis, Indiana, USA: Wiley Publishing Inc.

Ku, T., Zhu, Y. L., Hu, K. Y., & Lv, C. X. (2008). A Novel Pattern for Complex Event Processing in RFID Applications. In *Enterprise Interoperability III* (pp. 595-607). London: Springer.

Lenzerini, M. (2002). Data integration: a theoretical perspective. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* , pp. 233--246.

Leymann, F., Roller, D., & Schmidt, M.-T. (2002). Web services and business process management. *IBM Systems Journal* , vol 41. no 2, p.198-211.

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

Middleton, G. (2009). A framework for continuous compliance monitoring of B2B Processes. *Theses : Msc in Electronic Business Technologies* . Ottawa, Ontario, Canada.

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.

Misra, S., Akman, I., & Colomo-Palacios, R. (2012). Framework for evaluation and validation of software complexity measures. *IET Software , 6* (4), 323-334.

Mouttham, A., Peyton, L., & Kuziemsky, C. (2011). Leveraging Performance Analytics to Improve Integration of Care. *SEHC,11* (pp. 56-62). Waikiki, Honolulu: ACM.

Naeem, A. M., Dobbie, G., & Webber, G. (2008). An Event-Based Near Real-Time Data Integration Architecture. *Proceedings of the 2008 12th Enterprise Distributed Object Computing Conference Workshops (EDOCW '08)* , 401-404.

Niblett, P., & Graham, S. (2005). Events and service-oriented architecture: The OASIS Web Services Notification Specifications. *IBM Systems Journal , 44* (4), 869-886.

Palavitsinis, N., Protonotarios, V., & Manouselis, N. (2011). Applying analytics for a learning portal: the Organic.Edunet case study. *1st International Conference on Learning Analytics and Knowledge* , pp. 140-146.

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

Peyton, L., Zhan, B., & Stepien, B. (2008). A Case Study in Integrated Quality Assurance for Performance Management Systems. *MSVVEIS* , pp. 129-138.

Pourshahid, A., Amyot, D. L., Peyton, L., Ghanayati, S. P., Chen, P., Weiss, M., et al. (2009). Business Process Management with the User Requirements Notation. *Electronic Commerce Reserach , 9* (4), 269-316.

Reddy, K. N., & Rao, A. A. (2009, December 16-18). A Quantitative Evaluation of Software Quality Enhancement by Refactoring Using Dependency Oriented Complexity Metrics. *Second International Conference on Emerging Trends in Engineering and Technology (ICETET-09)* , 1011-1018.

Rogers Y. (2004). New Theoritical Approaches For HCI. *ARIST: Annual Review of Information Science and Technology* , p. 38.

Singh, I., Stearns, B., Johnson, M., & Team, T. E. (2002). *Designing Enterprise Applications with the J2EE Platform* (2nd ed.). Sun Microsystems.

Smith, C. U., & Williams, L. G. (2002). *Performance solutions: a practical guide to creating responsive, scalable software* (Vol. 1). Addison-Wesley.

Stone, D. L., & Stone, D. (2005). *User Interface Design and Evaluation.* San Francisco: Morgan Kauffman.

Tegegne, A., & Peyton, L. (2011). Model-Based Engineering of a Managed Process Application Framework. In G. Babin, K. Stanoevska-Slabeva, & P. Kropf (Eds.), *E-Technologies: Transformation in a Connected World - 5th International Conference, MCETECH 2011, Les Diablerets, Switzerland, January 23-26, 2011, Revised Selected Papers* (Vol. 78, pp. 173-188). Springer Berlin Heidelberg.

Trkman, P., McCormack, K., Valadares de Oliveira, M. P., & Ladeira, M. B. (2010). The impact of business analytics on supply chain performance . *Decision Support Systems , 49* (3), 318-327.

Vassiliadis, P., & Sellis, T. (1999). A Survey of Logical Models for OLAP Databases. *SIGMOD , 28* (4), 64-69.

Villar, C. (2010). *A Goal-Driven Methodology for Developing Health Care Quality Metrics.* Masters Thesis in Electronic Business Technologies, University of Ottawa, Telfer School Of Management, Ottawa, Ontario.

Violino, B. (2005, January 16). *What is RFID*. Retrieved May 04, 2013, from RFID Journal: http://www.rfidjournal.com/articles/view?1339

Web Services Architecture. (2004, February 11). Retrieved May 20, 2009, from World Wide Web Consortium(W3C): http://www.w3.org/TR/ws-arch/

Yao, W., Chu, C., & Li, Z. (2011). Leveraging complex event processing for smart hospitals using RFID. *Journal of Network and Computer Applications , 34* (3), 799-810.

Zang, Y., & Wu, L. (2010, September 25). Application of RFID and RTLS Technology in Supply Chain Enterprise. *Wireless Communications Networking and Mobile Computing (WiCOM)* , 1-4.

Appendix A

ACS Case Study Database Script

```
CREATE DATABASE [PFMv4]

USE [PFMv4]

GO

/****** Object:  Table [dbo].[patient_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[patient_dim](

        [patientId] [bigint] IDENTITY(1,1) NOT NULL,

        [patientNum] [nvarchar](50) NOT NULL,

        [active flag] [smallint] NULL,

 CONSTRAINT [PK_patient_dim] PRIMARY KEY CLUSTERED

(       [patientId] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[overdue_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[overdue_dim](

        [id] [smallint] IDENTITY(1,1) NOT NULL,
```

```
	[overdueflag] [nvarchar](50) NOT NULL,

	[description] [nvarchar](50) NOT NULL,

 CONSTRAINT [PK_Overdue_dim] PRIMARY KEY CLUSTERED

(	[id] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[hour_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[hour_dim](

	[hourId] [bigint] IDENTITY(1,1) NOT NULL,

	[hour] [nvarchar](50) NOT NULL,

 CONSTRAINT [PK_hour_dim] PRIMARY KEY CLUSTERED

(	[hourId] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[event_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[event_dim](

	[eventId] [bigint] NOT NULL,
```

```
	[eventName] [nvarchar](50) NOT NULL,

	[eventType] [nvarchar](50) NULL,

 CONSTRAINT [PK_patient_event_dim] PRIMARY KEY CLUSTERED

(	[eventId] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[current_flag_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[current_flag_dim](

	[id] [smallint] IDENTITY(1,1) NOT NULL,

	[currentflag] [nvarchar](50) NOT NULL,

	[description] [nvarchar](50) NOT NULL,

 CONSTRAINT [PK_current_flag_dim] PRIMARY KEY CLUSTERED

(	[id] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[active_patient_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[active_patient_dim](
```

```
	[ID] [smallint] IDENTITY(1,1) NOT NULL,

	[activeId] [smallint] NOT NULL,

	[description] [nvarchar](50) NOT NULL,

 CONSTRAINT [PK_active_patient_dim] PRIMARY KEY CLUSTERED

(	[ID] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[state_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[state_dim](

	[stateId] [bigint] NOT NULL,

	[stateName] [nvarchar](50) NOT NULL,

	[processType] [nvarchar](50) NOT NULL,

	[stateOrder] [bigint] NOT NULL,

	[target] [bigint] NOT NULL,

	[stateDescription] [nvarchar](50) NULL,

 CONSTRAINT [PK_patient_state_dim] PRIMARY KEY CLUSTERED

(	[stateId] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[room_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON
```

```
GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[room_dim](

	[roomId] [bigint] IDENTITY(1,1) NOT NULL,

	[roomNum] [nvarchar](50) NOT NULL,

	[unit] [nvarchar](50) NOT NULL,

 CONSTRAINT [PK_room_dim] PRIMARY KEY CLUSTERED

(	[roomId] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[provider_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[provider_dim](

	[providerId] [bigint] NOT NULL,

	[providerNum] [nvarchar](50) NOT NULL,

	[providerType] [nvarchar](50) NULL,

	[service] [nvarchar](50) NULL,

 CONSTRAINT [PK_provider_dim] PRIMARY KEY CLUSTERED

(	[providerId] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO
```

```
/****** Object:  Table [dbo].[procedure_dim]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[procedure_dim](

        [procedureId] [bigint] NOT NULL,

        [procedureName] [nvarchar](50) NOT NULL,

        [procedureType] [nvarchar](50) NOT NULL,

 CONSTRAINT [PK_procedure_dim] PRIMARY KEY CLUSTERED

(       [procedureId] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[patient_state_fact]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[patient_state_fact](

        [patientId] [bigint] NOT NULL,

        [patientStateId] [bigint] NOT NULL,

        [providerId] [bigint] NOT NULL,

        [roomId] [bigint] NOT NULL,

        [procedureId] [bigint] NOT NULL,

        [startTimestamp] [datetime] NOT NULL,

        [dateId] [bigint] NOT NULL,
```

```
	[hourId] [bigint] NOT NULL,

	[duration] [float] NULL,

	[currentStateFlag] [numeric](1, 0) NULL,

	[endTimestamp] [datetime] NULL,

	[id] [bigint] IDENTITY(1,1) NOT NULL,

 CONSTRAINT [PK_patient_state_fact] PRIMARY KEY CLUSTERED

(	[id] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[patient_event_fact]    Script Date: 06/08/2013 09:39:38 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

SET ANSI_PADDING ON

GO

CREATE TABLE [dbo].[patient_event_fact](

	[id] [bigint] IDENTITY(1,1) NOT NULL,

	[patientId] [bigint] NOT NULL,

	[patientEventId] [bigint] NOT NULL,

	[providerId] [bigint] NOT NULL,

	[CTAS] [bigint] NULL,

	[roomId] [bigint] NOT NULL,

	[orderNum] [varchar](50) NULL,

	[orderType] [nvarchar](50) NULL,

	[procedureId] [bigint] NOT NULL,
```

```
	[startTimestamp] [datetime] NOT NULL,

	[dateId] [bigint] NOT NULL,

	[hourId] [bigint] NOT NULL,

	[duration] [float] NULL,

	[currentStateFlag] [numeric](1, 0) NOT NULL,

	[endTimestamp] [datetime] NULL,

 CONSTRAINT [PK_patient_event_fact] PRIMARY KEY CLUSTERED

(	[id] ASC

)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI_PADDING OFF

GO

/****** Object:  ForeignKey [FK_patient_event_fact_date_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_event_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_event_fact_date_dim] FOREIGN KEY([dateId])

REFERENCES [dbo].[date_dim] ([dateId])

GO

ALTER TABLE [dbo].[patient_event_fact] CHECK CONSTRAINT [FK_patient_event_fact_date_dim]

GO

/****** Object:  ForeignKey [FK_patient_event_fact_event_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_event_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_event_fact_event_dim] FOREIGN KEY([patientEventId])

REFERENCES [dbo].[event_dim] ([eventId])

GO

ALTER TABLE [dbo].[patient_event_fact] CHECK CONSTRAINT [FK_patient_event_fact_event_dim]

GO
```

```
/****** Object:  ForeignKey [FK_patient_event_fact_hour_dim]    Script Date: 06/08/2013 09:39:38 ******/
ALTER TABLE [dbo].[patient_event_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_event_fact_hour_dim] FOREIGN KEY([hourId])
REFERENCES [dbo].[hour_dim] ([hourId])
GO
ALTER TABLE [dbo].[patient_event_fact] CHECK CONSTRAINT [FK_patient_event_fact_hour_dim]
GO
/****** Object:  ForeignKey [FK_patient_event_fact_patient_dim]    Script Date: 06/08/2013 09:39:38 ******/
ALTER TABLE [dbo].[patient_event_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_event_fact_patient_dim] FOREIGN KEY([patientId])
REFERENCES [dbo].[patient_dim] ([patientId])
GO
ALTER TABLE [dbo].[patient_event_fact] CHECK CONSTRAINT [FK_patient_event_fact_patient_dim]
GO
/****** Object:  ForeignKey [FK_patient_event_fact_procedure_dim]    Script Date: 06/08/2013 09:39:38 ******/
ALTER TABLE [dbo].[patient_event_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_event_fact_procedure_dim] FOREIGN KEY([procedureId])
REFERENCES [dbo].[procedure_dim] ([procedureId])
GO
ALTER TABLE [dbo].[patient_event_fact] CHECK CONSTRAINT [FK_patient_event_fact_procedure_dim]
GO
/****** Object:  ForeignKey [FK_patient_event_fact_provider_dim]    Script Date: 06/08/2013 09:39:38 ******/
ALTER TABLE [dbo].[patient_event_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_event_fact_provider_dim] FOREIGN KEY([providerId])
REFERENCES [dbo].[provider_dim] ([providerId])
GO
```

```
ALTER TABLE [dbo].[patient_event_fact] CHECK CONSTRAINT [FK_patient_event_fact_provider_dim]

GO

/****** Object:  ForeignKey [FK_patient_event_fact_room_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_event_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_event_fact_room_dim] FOREIGN KEY([roomId])

REFERENCES [dbo].[room_dim] ([roomId])

GO

ALTER TABLE [dbo].[patient_event_fact] CHECK CONSTRAINT [FK_patient_event_fact_room_dim]

GO

/****** Object:  ForeignKey [FK_patient_state_fact_date_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_state_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_state_fact_date_dim] FOREIGN KEY([dateId])

REFERENCES [dbo].[date_dim] ([dateId])

GO

ALTER TABLE [dbo].[patient_state_fact] CHECK CONSTRAINT [FK_patient_state_fact_date_dim]

GO

/****** Object:  ForeignKey [FK_patient_state_fact_hour_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_state_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_state_fact_hour_dim] FOREIGN KEY([hourId])

REFERENCES [dbo].[hour_dim] ([hourId])

GO

ALTER TABLE [dbo].[patient_state_fact] CHECK CONSTRAINT [FK_patient_state_fact_hour_dim]

GO

/****** Object:  ForeignKey [FK_patient_state_fact_patient_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_state_fact]  WITH CHECK ADD  CONSTRAINT [FK_patient_state_fact_patient_dim] FOREIGN KEY([patientId])

REFERENCES [dbo].[patient_dim] ([patientId])
```

```
GO

ALTER TABLE [dbo].[patient_state_fact] CHECK CONSTRAINT [FK_patient_state_fact_patient_dim]

GO

/****** Object:  ForeignKey [FK_patient_state_fact_procedure_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_state_fact] WITH CHECK ADD CONSTRAINT [FK_patient_state_fact_procedure_dim] FOREIGN KEY([procedureId])

REFERENCES [dbo].[procedure_dim] ([procedureId])

GO

ALTER TABLE [dbo].[patient_state_fact] CHECK CONSTRAINT [FK_patient_state_fact_procedure_dim]

GO

/****** Object:  ForeignKey [FK_patient_state_fact_provider_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_state_fact] WITH CHECK ADD CONSTRAINT [FK_patient_state_fact_provider_dim] FOREIGN KEY([providerId])

REFERENCES [dbo].[provider_dim] ([providerId])

GO

ALTER TABLE [dbo].[patient_state_fact] CHECK CONSTRAINT [FK_patient_state_fact_provider_dim]

GO

/****** Object:  ForeignKey [FK_patient_state_fact_room_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_state_fact] WITH CHECK ADD CONSTRAINT [FK_patient_state_fact_room_dim] FOREIGN KEY([roomId])

REFERENCES [dbo].[room_dim] ([roomId])

GO

ALTER TABLE [dbo].[patient_state_fact] CHECK CONSTRAINT [FK_patient_state_fact_room_dim]

GO

/****** Object:  ForeignKey [FK_patient_state_fact_state_dim]    Script Date: 06/08/2013 09:39:38 ******/

ALTER TABLE [dbo].[patient_state_fact] WITH CHECK ADD CONSTRAINT [FK_patient_state_fact_state_dim] FOREIGN KEY([patientStateId])
```

```
REFERENCES [dbo].[state_dim] ([stateId])

GO

ALTER TABLE [dbo].[patient_state_fact] CHECK CONSTRAINT [FK_patient_state_fact_state_dim]

GO
```