Architectural Framework for Operational Business Intelligence System

A. D. N. Sarma, *Member, IACSIT,* and R. Sivarama Prasad

***Abstract*—Business Intelligence (BI) becomes an essential element in decision making system. BI provides decision making information for strategic and tactical users. Operational Business Intelligence (Operational BI) is an extension of BI functionality into operational level of the business. Thus, Operational BI provides decision making information not only strategic and tactical users but also operational users. The characteristics of Operational BI system is different from traditional BI system in terms of low latency, reduced access time, real time alerts and notification, large number of users, process oriented and event driven. So there is a great need for enterprise architectural framework that supports the characteristics of Operational BI system. In this paper, we present an architectural framework for Operational BI system that uses Model View Controller (MVC) framework which is a well proven design pattern of Java 2 Platform, Enterprise Edition (J2EE). The proposed framework of the system is presented interms of multi-tiered architecture, MVC Model 2 architecture, generic objects flow and sequence diagram. Finally, the system architecture of the proposed system is envisaged. The proposed architectural framework is highly scalable and supports for enterprise Operational BI applications and systems.**

***Index Terms*—Architectural framework, business intelligence, J2EE, model view controller, model 2 architecture, operational business intelligence.**

I. Introduction

Business Intelligence (BI) becomes an essential element in decision making system. Decision making is a major part of the modern business and is essential in all most all business organizations. BI software aims [1] to enable business users to easily access and analyze relevant enterprise information for timely and fact-based decisions. The use of BI is gaining in all most all organizations irrespective of business size and functionality which includes small, medium and large in domestic, multi-national and even transnational organizations for strategic and tactical decision making purposes. Today, it is difficult to find a successful enterprise that has not leveraged BI technology for its business [1]. The advancement of technology and tools in today's word provide us to extend the functionality of BI into operational level decision making of the business in addition strategic and tactical decision making that evolves as new area of BI which is known as Operational Business Intelligence (Operational BI). Operational BI is one of the fastest growing areas of BI [2]. Operational BI works on near real time basis and provides decision making information in current time. Thus, Operational BI systems are also known as dynamic BI, real time BI, operational intelligence and operational analytics. Operational BI provides low level decision making information to front line managers of business for day to day business operations in addition to tactical and strategic decision making information as opposed to traditional BI.

Traditional BI systems are static, historic in nature, non-process oriented and highly data driven. In addition, they have very limited user access and limited view of decision making information, what is happening in current time of the business is not known. Moreover, traditional BI systems are monolithic, client server and non-web based architectures. The characteristics of Operational BI system are low latency, reduced access time, real time alerts and notification, large number of user access, event driven, process-orient and decision making information in current time those are different from traditional BI system. Thus, traditional BI systems suffer many drawbacks as compared with Operational BI systems interms of providing decision making information in current time, low scalability, non-monitoring of business performance measurements, dynamic configuration of business performance parameters, real time alerts and limited user access. So there is a great need to develop an architectural framework for Operational BI systems which is found to be an open research problem. This motivates us to develop an architectural framework that supports for operational business of an organization to provide decision making information not only in present time but also strategic and tactical users.

The goal of this paper is to propose an architectural framework for Operational BI system that supports multi-tier architecture, decision making information in current time, low response time, large number of user access, business process monitoring information and real time alert notification. The proposed architectural framework uses Model View Controller (MVC) Model 2 architecture of Java 2 Platform, Enterprise Edition (J2EE). MVC architecture supports multi-tier architecture, faster response and highly scalable. The proposed architectural framework of the system is based on the functional architecture of the system as described [3]. The proposed architectural framework is browser based and is highly suitable for real time or near real time applications on day to day business environment. The proposed architectural framework supports to meet the characteristics of Operational BI system as described [3]. Discussion on data warehouse system and extraction

Manuscript received May 26, 2014; revised July 28, 2014.

A. D. N. Sarma is with the Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur - 522 510, Andhra Pradesh, India (e-mail: adnsarma@yahoo.com).

R. Siva Rama Prasad is with the Department of Computer Science and Engineering, Coordinator of Department of International Business Studies, Acharya Nagarjuna University, Guntur - 522 510, Andhra Pradesh, India (e-mail: raminenisivaram@yahoo.co.in).

mechanism of data from source systems are out of the scope of the paper.

The rest of the paper is organized as follows. Section II, we discusses about relevant work. Section III covers architectural framework of Operational BI system. Section IV covers discussion. Section V covers conclusion and further work.

II. Related Work

Many researchers [4]-[6] reported work on architecture of BI system. There is very limited work has been attempted [2], [3], [7]-[9] on Operational BI system. As described [3], Service Oriented Architecture (SOA) for business intelligence makes possible a seamless integration of technologies into a coherent BI environment which enables simplified data delivery and low-latency analytics. In addition, SOA based approach not only reduces the total development and maintenance cost but also minimize the risk and impact across an entire enterprise when introducing business intelligence solutions.

The business process intelligence system (BPI) architecture for process oriented decision support was described [4] that links Operational Data Store (ODS) and Process Data Store (PDS) to provide real time BI and integrates Data Warehouse (DWH) and Process Warehouse (PWH) to provide strategic BI. The architecture of BPI consists of four major constructs namely data, operation, information and knowledge. The data construct consists of internal, external data sources and process audit log databases. The operational construct includes operational data in ODS and PDS. The information construct includes DWH and PWH. The knowledge construct includes real time business intelligence and strategic business intelligence. In addition, extract transform load (ETL) application collects data and loads into ODS, PDS, DWH and PWH.

A low cost BI system was proposed [6] using self-organized multi-agent system (MAS) technology that consists of four layers which are application, business, data cleansing and data source. Each layer is dedicated to execute one major task of the system. The upper layer interacts with the lower layer through Web services. Building of BI systems using self-organized MAS technology would minimize the cost mainly due to execution of various tasks by agents locally at various stages such as data integration, data cleansing, metadata extraction, querying, analyzing and data mining. Executing the various tasks of BI with agents would reduce the data transfer and storage which minimizes the cost.

A layered methodology for designing ETL processes in Operational BI systems was proposed [7] that follow in successive, stepwise refinements from high level business requirements, through several levels of more concrete specifications and down to execution models. The key feature of this layered methodology is a unified formalism for modeling the operational business processes of the enterprise as well as the processes for generating the end-to-end information views required by operational decision-making. This layered methodology starts with a conceptual specification from which the logical definition and physical implementation are systematically derived. Included in the conceptual model is the specification of quality objects (or QoX objectives) which drive the design and optimization at the logical and physical levels.

An Operational BI system essentially consists of two architectural entities namely Corporate Information Factory (CIF) and ODS that was described [7] which is based on the concepts of W.H. Inmon. These two components CIF and ODS will play a dual role interms of supporting both decision support and operational transaction processing. ODS is used as an intermediate layer between operational systems and a data warehouse that has three properties such as volatile, detailed and current valued. The concept of integrated ODS was proposed for large scale architecture system.

The impact while shifting from strategic BI to Operational BI was discussed [9] in terms of increased number of users, the data volumes needed to support operational BI for handling of a mixed workload including operational response time, short tactical queries, massive analytical queries, thousands of concurrent users and large volume of data.

The different levels of Operational BI were described [8] that are analyse, monitor, facilitate and execute. In the first level of operational BI, users analyze operational process using traditional reports. The next level occurs when user monitor process on a just-in-time basis using graphical key performance indicator. In the next level, IT developers facilitate processes by embedding BI into operational applications using SOA to merge operational and analytical processes into a single application. Finally, the culmination of Operational BI is when organization executes the process using event driven analytic engines, predictive models and other techniques that monitor events and trigger rules to automate or guide actions.

The key features of Operational BI system were presented [10] which are namely low latency and reduced action time, access to lowest granularity data, real-time alerts, faster query response time, more ad hoc querying capability, support for Streaming SQL, flexible to integrate the existing business processes and workflows, performance measurements of configurable parameters and timely information to the users of the system. Moreover, mapping between the key features of the system were presented with their equivalent functional modules. A holistic view of Operational BI system was presented [10] that consists of set of abstract layers namely data sources, data services, operational BI engines and support services, service delivery and delivery channels.

The functional architecture of Operational BI system was presented [3] which are based on the key features [10]. The various functional layers of the system are envisaged as data sources, data services, analytics engine, reporting engine and portal. In addition, alert engine, business services, metadata management and user and security. The data sources consist of set of operational databases which acts as an input to the system. The functionality of data services layers includes data integration, storage, compression, OLAP, querying and streaming. The functionality of analytic engine is not only to provide decision making information in current time but also form historical systems. The business services layer covers the functionality of various modules such as workflow, key

performance indicators (KPIs) and service level agreements (SLAs), business rules definition, logging and monitoring. Alert engine generates alerts on real time basis to the user for timely information. Reporting engine will provide various reporting tools that include data visualization, dash boards and linking between operational and strategic reports. The portal will acts a single point of contact for information dissemination to the various users of the system. The technology architecture interms of conceptual, logical, physical views and deployment architectures of the enterprise architecture of operational business intelligence system [11] were presented.

III. Operational Business Intelligence Framework

In this section, Operational BI architectural framework is presented which covers multi-tiered architecture and various components associated in each tier, MVC framework, MVC Model 2 architecture, generic objects flow. Moreover, the sequence diagram and deployment diagrams of the proposed framework of the system are presented.

A. Multi-Tiered Architecture

The proposed Operational BI is considered as stack of tiers. A tier is a logical representation of concerns in the system. Each tier is responsible for set of tasks that are performed by the components associated with in the tier. Each tier is logically separated from another and is loosely coupled with the adjacent tier. The multi-tiered architecture of Operational BI system framework is shown in Fig. 1.

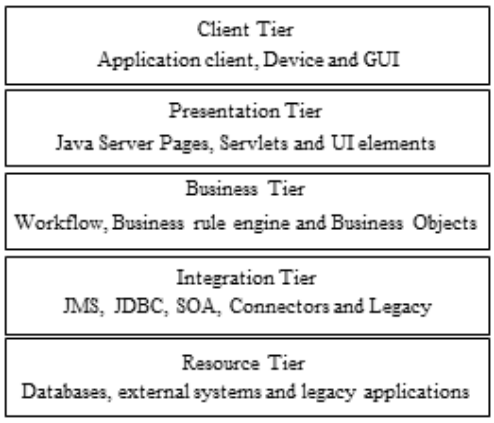

Fig. 1. Multi-tiered architecture of operational business intelligence.

Resource tier consists of differ data sources that includes operational data sources, external systems and legacy applications, email and web repositories. In addition, this may include other transaction applications, ftp servers, directory servers, back-office systems like enterprise resource planning (ERP), supply chain management (SCM) front-office systems like customer relationship management (CRM) and custom business applications. Resource tier will acts as input to the proposed system. In addition, resource tier contains the business data and external data or application sources, legacy systems and data warehouse system. The design of data warehouse system is out of scope of this paper and considered as an external system to the proposed architecture.

Integration tier consists of components like Java Data Base Connectivity (JDBC), connectors, SOA, Java Messaging Service (JMS), other legacy applications and real time integration. Integration tier is responsible for communicating with external resources and systems. The business tier is coupled with the integration tier whenever the business objects require data or services that reside in the resource tier. In addition, this tier consist of real time ETL, enterprise information integration (EII) and enterprise application integration (EAI) frameworks.

Business tier provides the business services required by the client. This tier contains the business data and business logic. Typically, most business processing for the application is centralized in this tier as described [12]. Mostly workflow, Java bean components or Enterprise beans are used as business objects. As described [13] Java bean components are Java classes that can be easily reused and composed together into applications. JSP technology directly supports using Java bean components with JSP language elements. Java beans are easy to create and initialize parameter's values using get and set methods. This layer consists of a set of components. A component is physical and replaceable part of a system [14] that confirms to and provides the realization of a set of interfaces. These components provide services to other components. The various business services of Operational BI includes login, logging, master data management, analytical, Metadata, key performance indicators (KPIs), Service level agreements (SLAs) monitoring, workflow, business rule engine, monitoring engine, alert engine, reports and dash boards as described in [3].

Presentation tier encapsulates all presentation logic required to service the clients that access the system. The presentation tier intercepts the client's requests; controls access to business services, construct the responses and finally deliver the response to the clients. The presentation layer contains Servlets and Java Server Pages (JSPs) that produce User Interface (UI) elements.

Client tier represents all devices or system clients accessing the system resources. A client can be a Web browser, a Java application, or a device and Graphical User Interface (GUI).

B. Model View Controller (MVC) Architectural Framework

Patterns are an important vehicle for constructing high-quality software architectures [15] which a description of the subsystems and components of a software system and the relationships between them. MVC (Model-View-Controller) is software architectural design pattern which is a well proven, standard and industry accepted architecture. The MVC architecture has its roots in Smalltalk, where it was originally applied to map the traditional input, processing, and output tasks to the graphical user interaction model as mentioned [16]. Many enterprise applications use the design principles of MVC architecture that were described [16]-[20]. The MVC architecture is a way of decomposing an application into three parts: the model, the view, and the controller. It was originally applied in the graphical user interaction model of input, processing, and

output [12]. The very advantage of MVC architecture is separations of layers which supports multi-layer architecture. This provides change in one layer can be accomplished without altering the other layers of the system.

The MVC pattern can be implemented with programming languages such as Smalltalk, Java, C, C++, and Microsoft .NET. The MVC Model 1 architecture is shown in Fig. 2.

Inorder to develop web applications Servlets and JSPs technologies are commonly used in J2EE. The advantages of servlets over Common Gateway Interface (CGI) are easier to develop web applications, faster to run, platform independent, handle multiple request concurrently and synchronize requests, improved performance, scalability, reusability, safe and secure.

Servlets create threads to handle requests instead of creating process this results no separate memory area is required. Thus many subsequent requests can be easily handled by the servlets. However, there are limitations of Servlet interms of recompilation of code as and when there is change. Moreover, Servlets does not provide separation of business logic from presentation logic.

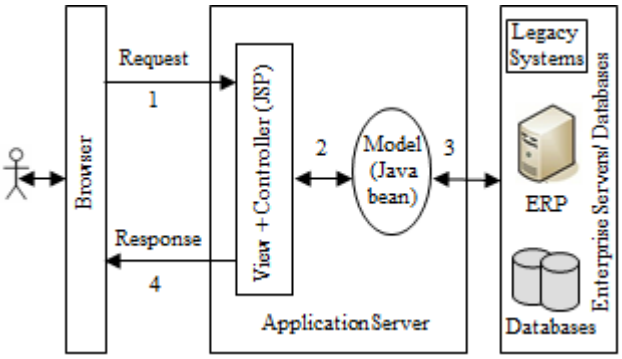

Fig. 2. MVC Model 1 architecture.

JSP overcomes the limitations of Servlets which provides separation between presentation and business logic. There is no need to redeploy the application if JSP code is modified unlike Servlets. Java beans are used in JSPs for developing applications. Custom tags and Java Standard Tag Library (JSTL) will provide reuse of components in JSP pages which results changes in system can be handled more easily.

The logical flow of request and response in Model 1 architecture is summed up as follows:

1) Browser sends request for the JSP page.
2) JSP page access Java bean and invokes business logic.
3) Java bean connects to the database and gets or saves data into database.
4) Response sends to the browser which is generated by JSP.

In MVC Model 1 architecture, JSP alone acts as view and controller this result no separation of presentation logic from business logic, decentralized navigation control and support for small and medium size applications.

Fig. 3 shows MVC Model 2 architecture. The client will access the application through browser. All user requests are handled via controller and responses through JSP. Internally, model (Java bean) will connect to the enterprise servers/ databases inorder to access the required data.

Model encapsulates the core data and functionality. The model represents enterprise data and the business rules that govern access to and updates of the data.

View encapsulates the presentation of the data. The view renders the contents of a model. It accesses enterprise data through the model and specifies how that data should be presented. It is the view's responsibility to maintain consistency in its presentation when the model changes. This can be achieved either by using a push model or pull model. In push model the view registers itself with the model for change notifications whereas in pull model the view is responsible for calling the model when it needs to retrieve the most current data.

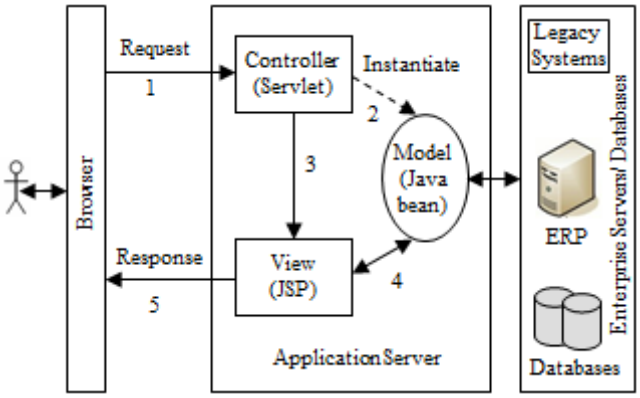

Fig. 3. MVC Model 2 architecture.

Controller accepts inputs from the user and makes request from the model for the data to provide a new view. The controller translates interactions with the view into actions to be performed by the model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a web application, they appear as GET and POST HTTP requests. The actions performed by the model include activating business processes or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view.

The general structure of a web application request and response flow using the JSP Model 2 architecture is envisaged [21] as follows:

1) User requests are directed to the controller servlet.
2) The controller servlet accesses required data and builds the model, possibly delegating the processing to helper classes.
3) The controller servlet (or the appropriate sub-ordinate task) selects and passes control to the appropriate JSP responsible for presenting the view.
4) The view page is presented to the requesting user.
5) The user interacts with the controller servlet (via the view) to enter/modify data, traverse through results.

MVC Model 2 architecture has the following advantages over MVC Model 1. In MVC Model 2 Servlets acts as controller, JSP as view and Java bean as model. Hence, there is a clear separation between all the three layers as compared with MVC Model 1 architecture, centralized navigation control, easy to maintain, easy to test and highly scalable for enterprise applications.

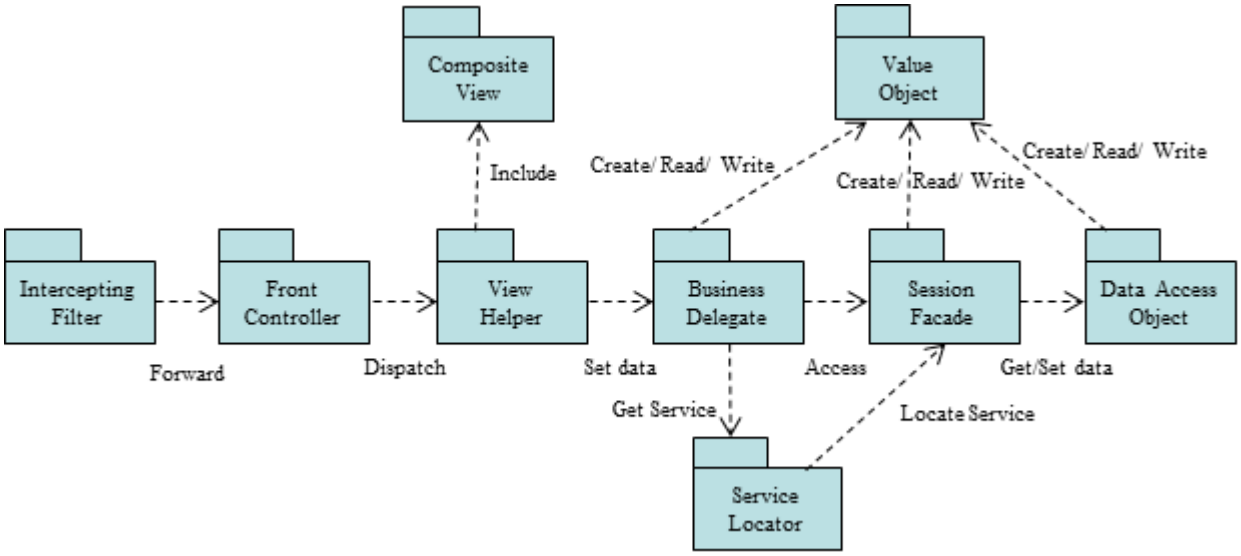

Fig. 4. Generic object flow of operational business intelligence system.

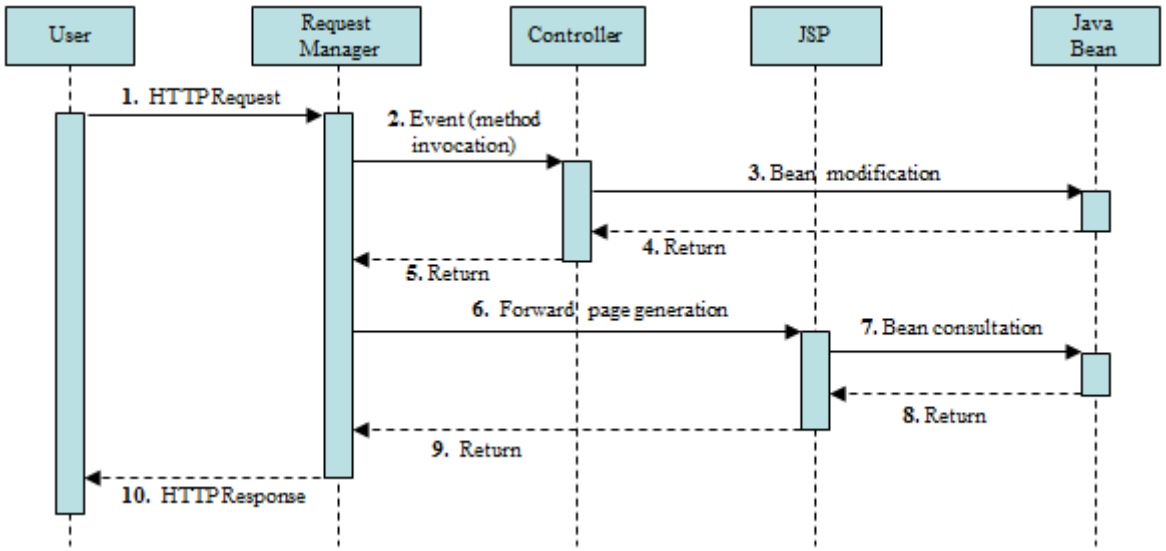

Fig. 5. Sequence diagram for request manager.

A. *Generic Objects Flow*

The general structure of a web application using MVC Model 2 architecture 2 is Sun's attempt to wrap JSP within the MVC paradigm [16] which has front controller, data access and application logic, Service-To-Worker and Dispatcher View, Intercepting Filter, Value List Handler and Data Access Objects (DAOs). The generic object flow of the proposed architectural framework is envisaged in figure 4.

Front Controller is a Servlet that acts as the centralized entry point in a web application. This performs managing request processing, authentication, authorization services and ultimately selecting the appropriate view.

Data access and application logic contain entirely within the controller servlet and its helper classes [21]. The controller servlet (or the helper class) should select the appropriate JSP page and transfer control to that page object based on the request parameters, state and session information. One of the major advances that come with JSP Model 2 is Sun's specification of the JSTL which specifies the standard set of tags for iteration, conditional processing, database access and many other formatting functions.

The guidelines associated with JSP Model 2, Sun also provided a set of blueprints for building application using the MVC paradigm and these blueprints renamed the J2EE Core Patterns.

Service-To-Worker and Dispatcher View strategies for MVC application where the front controller module defers processing to a dispatcher that is selected based on the request context. In the Dispatcher View pattern, the dispatcher performs static processing to select the ultimate presentation view.

In the Service-To-Worker pattern, the dispatcher's processing is more dynamic, translating logical task names into concrete task module references and allowing tasks to perform complex processing that determines the ultimate presentation view.

Intercepting Filter allows for pluggable filters to be inserted in the "request pipeline" to perform pre and post processing of incoming requests and outgoing responses.

B. *Sequence Diagram*

A sequence diagram [14] is an interaction diagram that emphasizes the time ordering of messages that send and receive which helps to model dynamic aspects of a system. The sequence diagram of MVC with request dispatcher (or

manager) is shown in Fig. 5. This shows the interaction between various objects such as user, request manager, controller, JSP and Java bean.

C. System Architecture

The system architecture of the proposed Operational BI is shown in Fig. 6. Data resource layer contains all the legacy application data, data sources, data warehouse and Metadata. Operational BI engines with admin services, user management and other resources code are deployed on application server. Operational BI engines have various sub-engines that include ETL, Real Time ETL, business rules, Online analytical processing (OLAP), data compression, reporting, dashboards, alert notification and monitoring, analytics and SQL streaming as mentioned in [10].

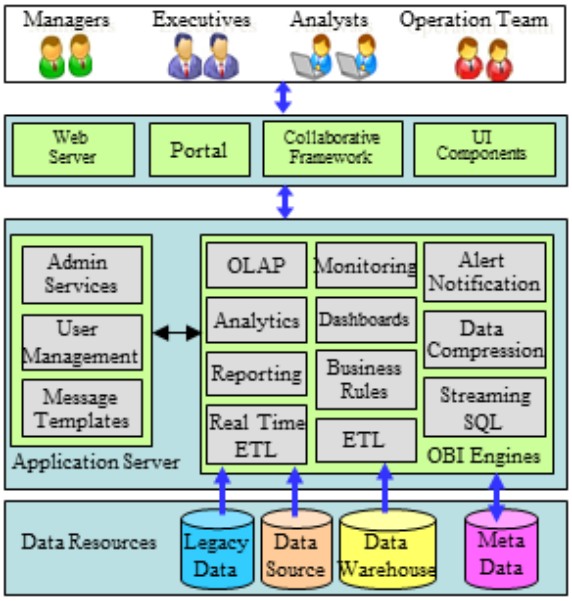

Fig. 6. System architecture of operational business intelligence.

In addition to this admin services include managing various business services, configuration of parameters to be measured and user management. Web server contains various UI components and as well as portal. All the users of the system will access the resources through the browser. The portal acts as singe entry point to the users to access the applications resources that are available in the system. This also acts information dissemination and collaboration tool.

IV. Discussion

MVC is an architectural design pattern of J2EE which is well proven and industry accepted architecture. The main advantage of the MVC design pattern is to separate the application object (model) from the way it is represented to the user (view) from the way in which the user controls it (controller). This pattern contains two models MVC 1 and MVC 2 for implementation. MVC 1 architecture is page centric and tightly coupled between page and model.

The proposed design of Operational BI system uses MVC model 2 (MVC 2) that removes page centric property of MVC 1 and separates presentation logic and application logic. The controller receives all requests for the application and is responsible for taking appropriate action for each request. Model contains the state (data), view displays model to user (presentation) and controller modifies model (business logic). The proposed design of the system will support the 9 Key features of the Operational BI as described [10] and functional architecture as described [3]. The proposed architecture framework supports the characteristics of Operational BI systems interms of multi-tier architecture, reduced data latency, supports real time applications, access for large number of users and can be used for enterprise Operational BI applications and systems.

The proposed architectural framework of Operational BI has the following advantages:

1) Clear separation between presentation logic and business logic: - Each object in MVC has distinct responsibilities. All objects and classes are independent of each other. So change in one class does not require alternation in other classes.
2) Multiple views using the same model: - The separation of model and view allows multiple views to use the same model. This is not only facilitates easier implementation of enterprise model but also easier to test, and maintaining of the enterprise application.
3) Efficient modularity: - This architecture highly supports modular development of the application either by the use of different controllers for each module or single controller with different action classes.
4) Easier support for new types of clients:-This model is easier to support for new types of clients. We need to a view and controller for each type of client and wire them into the existing enterprise model.
5) Support for web applications: This model is often seen in web applications.
6) High scalability: - Controllers and views can grow as the model grows; and older versions of the views and controllers can still be used as long as a common interface is maintained.
7) This model supports easier maintenance of the code and future improvements of the application.

V. Conclusion

Traditional BI is static, historic in nature and mainly used for strategic and tactical decision making whereas Operational BI is dynamic, current in time and provides decision making information not only in current time but also strategic and tactical levels. Operational BI works on near real time/ real time, event based and low data latency where as traditional BI based on data driven and high data latency. The characteristics of Operational BI system is different from traditional BI system in terms of low latency, reduced access time, real time alerts and notification, large number of users, process oriented and event driven.

In this paper, the architectural framework for Operational BI system is presented using MVC Model 2 architecture of J2EE which is well proven and industry accepted architectural pattern. The tiered (or layered) architecture of the proposed system is presented and explained the major components associated in each tier. Explained MVC Model 2 architecture works and how this is different form MVC Model 1. The flow of user request and system response is

explained. The interaction between various objects in MVC architecture such as user, request manager, controller, JSP and bean is explained with the help of sequence diagram. The structure of web application of the proposed system is explained. The generic object flow of the proposed Operational BI system is presented. The system architecture of the proposed system is presented and explained.

In the future, the proposed architectural framework of Operational BI system can be implemented as a prototype. The functionality of the proposed system will be tested for one of the business verticals. The proposed architecture can be further implemented using one of the programming languages such as Small talk, Java, Microsoft .NET. Further, the proposed architectural framework can be extended into hierarchical MVC architecture for multiple client tiers.

REFERENCES

[1] S. Chaudhuri, U. Dayal, and V. Narasayya, "An overview of business intelligence technology," *Communications of the ACM*, vol. 54, no. 8, pp. 88-98, August 2011.

[2] U. Christ, "An architecture for integrated operational business intelligence," in *Proc. Conference on Business, Technology, and Web (BTW), Lecture Notes in Informatics*, vol. P-144, pp. 460-468, 2009.

[3] A. D. N. Sarma and R. S. Ramprasad, "Functional architecture for operational business intelligence system," in *Proc. IEEE-International Conference on Advances in Engineering, Science and Management (ICAESM 2012)*, March 30-31, 2012, Nagapattinam, India, pp.213-218.

[4] L. Wu, G. Barash, and G. Bartolini, "A service oriented architecture for business intelligence," *Service-Oriented Computing and Applications (SOCA '07)*, June 2007, pp. 279 – 285.

[5] L. An, J. Yan, and L. Tong, "Business process intelligence system: architecture and data models," in *Proc. the Sixth Wuhan International Conference on e-Business*, Wuhan, China, 2007, pp. 6-13.

[6] M. Venkatadri, H. G. Satry, and G. Manjunath, "A novel business intelligence framework," *Universal Journal of Computer Science and Engineering Technology*, vol. 1, no. 2, pp. 112-116, November 2010.

[7] U. Daya, K. Wilkinson, A. Simitsis, and M. Castellanos, "Business process meet operational business intelligence," *Data engineering*, Sep. 2009, vol. 32 no. 3, pp. 35-41.

[8] W. W. Eckerson, "Best practices in operational BI – converting analytical and operational process," *TDWI Best Practices Report*, 2007.

[9] C. Imhoff, "Operational business intelligence, It's time to expand the scope of business intelligences," *Teradata Magazine*, September 2006, pp.16-18.

[10] A. D. N. Sarma and S. R. Prasad, "9 key features of operational business intelligence," *Journal of Computer Engineering: An International Journal*, vol. 1. no. 1-2, pp. 11-18, 2012.

[11] A. D. N. Sarma and S. Prasad , "Enterprise architecture of operational BI system," *National Conference on Recent Advances in Soft Computing and Knowledge Discovery (SCKD-2012)*, January 19-21, 2012, Tirupati, India.

[12] N. Gulzar, *Practical J2EE Application Architecture*, McGraw-Hill: Osborne Media, March 2003, ch. 4, pp. 89-93.

[13] S. Bodoff *et al.*, *The J2EE Tutorial*, 2nd edition, Addison Wesley, 2002, ch. 12, pp. 269-277.

[14] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, 4th Indian reprint, Addison-Wesley, 1999, ch. 18. pp. 243-256, ch. 25, pp. 346-348.

[15] B. Uschmann *et al*., *Pattern-Oriented Software Architecture: A System of Patterns*, 1st edition, John Wiley and Sons, 1996, ch. 6, pp. 383-39.

[16] G. Krasner and S. Pope, "A description of the model view controller user interface paradigm in the smalltalk-80 system," *Journal of Object Oriented Programming*, vol. 1, no. 3, pp. 26-49, 1988.

[17] M. Gallego-Carrillo, I. Garc á-Alcaide, and S. Montalvo-Herranz, "Applying hierarchical MVC architecture to high interactive web applications," in *Proc. Third International Conference Information Research, Applications and Education*, June 27 - 30, 2005, Varna, Bulgaria, 2005, pp. 110-114.

[18] T. Dey, "A comparative analysis on modeling and implementing with MVC architecture," *International Journal of Computer Applications (IJCA)*, pp. 44-49, 2011.

[19] P. Gupta and M. C. Govil, "MVC design pattern for the multi framework distributed applications using XML, spring and struts framework," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 2, no. 4, pp. 1047-1051, 2010.

[20] Y. Ping , K. Kontogiannis, and T. C. Lau, "Transforming legacy web applications to the MVC architecture," presented at the Eleventh Annual International Workshop on Software Technology and Engineering Practice, 2004.

[21] L. Shklar and R. Rosen, *Web Application Architecture: Principles, Protocols and Practices*, John Wiley and Sons, 2003, ch. 9, pp. 256-264.

A. D. N. Sarma is from Vijayawada, Andhra Pradesh, India and was born in 1964. He earned B.Sc and M.Sc (electronics) degrees from Acharya Nagarjuna University in 1984 and 1998 respectively. He received M.Phil degree from Andhra University in 1993. He got M.S. (software systems) from B.I.T.S., Pilani in 1998 and received M.B.A from Pondicherry University in 2007. Currently the author is pursuing Ph.D in computer science and engineering from Acharya Nagarjuna University.

Sarma is currently working as an assistant vice president for Bartronics Limited, Hyderabad, Andhra Pradesh, India. Earlier he was associated with Goldstone Technologies Ltd and RAM Informatics Ltd at a capacity of general manager (Projects) and general manager (Software Systems) respectively.

Sarma has more than 24 years of experience that includes research, information technology and academic. His interesting areas include business operations management, information systems, information technology management, software engineering, software architecture, algorithms, data mining, business intelligence and e-commerce. He is a global member of Internet Society since 2009, member of International Association of Engineers (IAENG) and a member of the Computer Science Teachers Association (CSTA). He has published and presented more than 20 papers in various reputed International conferences and journals.

R. Siva Rama Prasad is working as a research director in computer science and engineering and a coordinator for international business studies, Acharya Nagarjuna University. He earned M.B.A and M.C.A degrees and Ph.D from Acharya Nagarjuna University.

Prasad has more than 20 years of experience that includes research, and academic. His interesting areas include e-commerce, information systems, e-governance, HRD and finance.

He has published more than 33 research papers, attended 80 national and international seminars, delivered 104 extension lectures and published 5 books.