ODBIS: Towards a Platform for On-Demand Business Intelligence Services

Moez Essaidi
LIPN - UMR CNRS 7030, Université Paris-Nord, 99 Avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
moez.essaidi@lipn.univ-paris13.fr

ABSTRACT
In recent years, Software-as-a-Service (SaaS) is gaining momentum with more and more successful adoptions. Several companies including some well known names have embraced this new model for software distribution. In this paper we present ODBIS, an open source infrastructure to build and deliver On-Demand Business Intelligence Services. The ODBIS platform constitutes a future perspective to provide SaaS deployment model for business intelligence applications. ODBIS offers a unified and innovative multi-layered architecture to support different business intelligence needs, this includes: (i) Core business intelligence services which cover essential data warehousing tasks as meta-data management, data integration, reporting and analysis; all built in an integrated manner using standard APIs. (ii) Services for data warehousing projects management and models design based on a model-driven approach using the 2 Track Unified Process (2TUP) and the Model Driven Architecture (MDA) standards. (iii) Services for a centralized administration which provide an enterprise-grade security including authorities, roles, users and groups management mainly based on spring security.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous; D.2.11 [Software Engineering]: Software Architectures

General Terms
Standardization

Keywords
Business Intelligence (BI), Software-as-a-Service (SaaS)

1. INTRODUCTION
The Data Warehouse (DW) has become the central element of current Decision Support Systems (DSS) because they provide crucial business information to improve strategic decision-making processes [3]. The data warehousing architecture is defined through several heterogeneous and interrelated layers. Moreover, each layer contains different components, using different modeling profiles, and depends on several technologies. DW projects are also exposed to several technical risks and require more knowledge about the underlying business domain. These aspects increase the costs and the time of DW development and make it a very difficult and challenging task. Several DW development approaches [7, 4, 6, 8, 5, 1] have been proposed during the last few years. However, they fail to provide a unified and standard method for DW engineering. In addition, there is no proposal for an integrated DW construction environment including: (i) an engineering process that addresses the whole development cycle of DW in an iterative and incremental manner while considering both the business and the technical requirements; (ii) a standard and integrated DW design framework; and (iii) support for many standard industry DW tools and APIs.

In the light of this situation, the first goal of our work is to propose a unified approach to develop DWs. So, in [2], we address this issue by introducing a unified method for developing DWs, which integrates the DW design framework and the DW engineering process. We use the Model Driven Architecture (MDA) to define the DW design framework and we adapt the 2 Track Unified Process (2TUP) for the DW development using MDA transformation process in order to define the DW engineering process. The DW design framework shows the projection of MDA viewpoints and meta-levels on the data warehousing architecture to tackle the design of every DW component in an integrated and standard manner. The DW engineering process shows the alignment of the 2TUP disciplines with the MDA transformation process in order (i) to ensure the coherence between the DW design framework and the DW engineering process and to make our approach more integrated, (ii) to enable an iterative and an incremental development of DW layers, and (iii) to consider, at the same time, the business and the technical aspects of the DW. The main advantages of the proposed approach are: (i) it addresses, at the same time, the DW design and engineering problems, indeed it integrates the DW design framework and the DW engineering process; (ii) it uses open industry standards (i.e. MDA and 2TUP), which facilitate its implementation and extension; (iii) it offers a mix-driven approach, in which the data-driven, the user-driven, and the goal-driven approaches are combined with the model-driven, the semantic-driven, and the risk-driven approaches.
Most current data warehousing infrastructures deployment follows the traditional software model. Traditional software applications are based on a model with large upfront licensing costs and annual greenware support costs. Important functions of the package are often optioned in a front-loaded licensing arrangement with annual renewal for upgrades and support. Increasing the number of users may raise the base cost of the package due to the need for additional hardware server deployments and IT support. Licensing costs are often based on metrics that are not directly aligned with usage (e.g., server type, number of CPUs, etc.). The customer is also responsible for providing the logical and physical security to the applications as well as offering end-user training and support. On the other hand, the Software-as-a-Service (SaaS) revolution allows companies to subscribe to software applications and outsource operating the backend infrastructure to the SaaS vendor. In most cases, the SaaS vendor can do this much more cost effective; providing overall cost savings for the company. As a result, companies can spread their IT budget across many more applications to support and grow their business operations which will in turn contribute to the bottom line. In addition, the Web 2.0 revolution makes now the web pages the ideal development environment for end-users.

The second goal of our work is to provide an integrated business intelligence as service architecture based on standard open source technologies. Thus, we present ODBIS, an open source infrastructure to build and deliver on-demand business intelligence services. The ODBIS platform constitutes a future perspective to provide SaaS deployment model for business intelligence applications. The main advantages of the proposed SaaS business intelligence solution are: (i) it covers at the same time the on-demand DW design, the DW project management, the on-demand applications and accounts administration, and the on-demand business intelligence services delivery; (ii) it is based on standard and multi-layered Java enterprise architecture with a comprehensive security (based on spring security); (iii) it uses the most known open source tools and APIs; and (iv) an out-of-the-box integration is provided by the use of Spring framework which allows flexible configuration and personalization.

The rest of this paper is organized as follows: in Section 2 we give an overview of the SaaS deployment model and present some solutions that use it. Section 3 describes the multi-layered architecture of ODBIS. Finally, in Section 4, we present our conclusion and future work.

2. SOFTWARE-AS-A-SERVICE OVERVIEW
Software as a Service (SaaS) is a model for software delivery where a software company publishes one copy of their software on the Internet. It allows individuals and companies (multi-tenant architecture) to “rent” it through a subscription model (pay as you go model). The software company centrally operates, maintains and supports all its customers using this centralized service. On-Demand and pay as you go models mean that in a SaaS model, costs are directly aligned with usage. The cost may increase as the usage of the application increases. Multi-tenant architecture model means that the physical backend hardware infrastructure is shared among many different customers but logically is unique for each customer.

Lower Total Cost of Ownership (TCO) and better Return On Investment (ROI) constitute the major benefits of SaaS. Indeed, several factors contribute to making it considerably less expensive to implement a SaaS application than a traditional on-premises application. These factors include: (i) lower IT costs; since when you subscribe to a SaaS application, you avoid the overhead associated with implementing conventional software (installing and maintaining servers, etc.). (ii) Economies of scale; subscription costs for SaaS applications reflect the economies of scale achieved by multi-tenancy; as example on database is used to store all customers data, so, this makes the overall system scalable at a far lower cost. (iii) Pay as you go; companies who subscribe to a SaaS application pay a monthly or annual subscription fee, sometimes depending also on the number of users or transactions. Others key variables such as simplicity, flexibility and accessibility constitute the advantages of SaaS deployment model.

In recent years, SaaS is gaining momentum with more and more successful adoptions. Several companies including some well known names have embraced this new model for software distribution. As SaaS providers, we cite Salesforce.com for on-demand Customer Relationship Management (CRM) services; PeopleSoft On-Demand from Oracle provides SaaS infrastructure for enterprise applications; ShareMinds that offers an on-demand Enterprise Content Management (ECM), Google maps and apps (mail, docs, sites, etc.); and Microsoft, that announces Office Web Apps early 2010. Several business intelligence software solutions are exposed in a SaaS model. SAP BusinessObjects, the world’s leading business intelligence software company launched a hosted on-demand platform (available at http://www.ondemand.com) to deliver analytic and reporting functionality. PivotLink is one of the major actors of the on-demand business intelligence (http://www.pivotlink.com). PivotLink offers a SaaS business intelligence solution covering data analysis, reporting and dashboards. LogiXML (http://www.logixml.com) provides web-based ad-hoc reporting, analysis, dashboard and data integration (ETL) applications. LogiXML solution is one of the few solutions that offer a fully web-based data integration environment. Talend (http://www.talend.com) one of the most known open-source data integration solutions, starts recently the Talend on-demand project. Talend on-demand is a centralized and shared repository hosted by Talend in order to consolidate Talend Open Studio metadata and project information in online and to facilitate collaboration, object and code reuse. However, it consists of a partial SaaS solution since the design of ETL package is made with Talend Open Studio, the on-demand service covers only uploading, sharing and collaboration functions. Finally, OpenReports (http://oreports.com) and OpenPl (http://openi.org) are open-source solutions for web-based business intelligence deployment that allow build and publish reports, analyses, and dashboards.

These solutions present many advantages derived in general from the advantages of the SaaS concept. But, not much information on the architecture of these on-demand solutions is provided. In addition, few solutions offer integrated platforms that cover all functional and technical aspects of the data warehousing architecture. Indeed, these tools focus only on the problem of the business intelligence services.
deployment and they provide a partial on-demand solution to the problem of data warehousing design. Furthermore, none of these solutions offer an integrated model driven DW development approach and a web-based tool that support this approach.

3. ARCHITECTURE FOR SOFTWARE-AS-A-SERVICE BUSINESS INTELLIGENCE

This section is divided on three parts. In the first part we propose a general architecture for on-demand business intelligence. It presents also our main goals to provide a complete SaaS platform to deliver business intelligence services. The second section presents our implementation for on-demand model driven data warehouse. The third section describes the technical architecture of ODBIS and present the frameworks used to build it.

3.1 The General Architecture

The ODBIS Software as a Service (SaaS) architecture is defined through five main layers (figure 1): (i) the technical resources layer (deployment layer); (ii) the DW design and management layer; (iii) the infrastructure administration and configuration layer; (iv) the core business intelligence services layer (green bricks); and (v) the end-users access tools layer.

The technical resources layer contains the data warehousing tools (e.g., database, ETL engine, analysis server, etc.) used to deploy and to execute the designed DW models. This layer integrates also many Business Intelligence APIs for reporting (we use BIRT), and data mining (RapidMiner for example). Interoperability between all of these tools and APIs can be ensured using an Enterprise Service Bus (ESB) like framework (we plan to use spring integration module).

The design and management layer contains services to design DW models (ETL jobs, multidimensional models, reports, etc.) and services to manage DW development projects. MDDWS constitutes a web-based environment to design and manage DW projects using our model driven development approach for DW development presented in [2]. This layer offers an on-demand DW design in order to ensure platform integrity (same technologies for all layers). It also simplifies the deployment and the access to the development environment for developers whom want to subscribe to this service. This reduces the installation time of development infrastructure and its costs.

The core business intelligence services layer represents the basic applications used by business users. We identify five essential business intelligence services: (i) the meta-data service (MDS), which allows meta-data and business information definition to facilitate information sharing and exchange between all services. (ii) the integration service (IS), which offers an ad-hoc way to define data integration jobs, jobs scheduling, etc. (iii) the analysis service (AS), which allows definition of analysis data models (OLAP data cube), data cube visualization and navigation. (iv) the reporting service (RS) can be defined using existing reporting APIs like BIRT, or it can present same ad-hoc reporting functionalities; the current version of the RS implementation supports BIRT reporting and ad-hoc reporting. (v) the information delivery service (IDS) is an abstraction level to support many client Interfaces and technologies (e.g., web browser, mobile, office tools). It can be also presented as a web services for more flexibility to access the platform.

The infrastructure administration and configuration layer offers a web-based tool for administrators to manage users accounts, to customize services configuration and to report same information on platform usage and performance.

Finally, the end-users access tools layer contains client applications used to access the platform and use its services. Only the web browser is supported as access tool by current ODBIS release. In the future, we plan to support mobile technologies, web services, and desktop tools.
3.2 The MDDWS Environment

The model-driven DW represents an approach that aligns the development of data warehousing systems with a general model-driven development paradigm. Recently, we have contributed to improve current model-driven DW approaches by providing a complete and integrated method based on the Model Driven Architecture (MDA) and the 2 Track Unified Process (2TUP) standards [2]. The ODBIS DW management and design module called MDDWS for Model Drivel Data Warehouse Service represents an implementation of the proposed metamodeling architecture. The MDDWS environment is defined through three main layers (figure 2): (i) the methodology layer, which contains the data warehouse engineering process based on the integration of 2TUP and the transformation process of MDA; (ii) the design layer, which contains the data warehouse design framework based on metamodeling architecture of MDA; and (iii) the deployment layer, which contains the data warehousing tools and APIs used to deploy and to execute the designed models, of course it corresponds to the shared technical resources layer. In the following paragraphs, we give a more detail of the two first layers.

The methodology layer defines features for DW projects management. It is based on the alignment of the 2TUP discipline with the model-driven activities (including the MDA transformation process and viewpoints). The integration of the model-driven development approach in a more general process like 2TUP is important. Indeed, the only use of the transformation process of MDA such as works presented in [6, 8] is not sufficient to provide a complete model driven data warehouse. For example, the MDA transformation process does not include all engineering disciplines such as preliminary study, tests and deployment. Moreover, result of a MDA process is a semi-complete system code. Thus, it is necessary to define a code completion activity in the global development process. Finally, transformation process of MDA is not an iterative and incremental process. To overcome these limits, we propose to use the 2TUP process in order to develop DW components while keeping the MDA approach in order to ensure coherence between the data warehouse design framework and the data warehouse engineering process. Therefore, in our global DW engineering process, the MDA transformation process is a sub-process. The resulting DW engineering process corresponds to the 2TUP activities mapped to the MDD activities, which contains the transformation process of MDA using Query View Transformation (QVT) to derive each viewpoint. Figure 3 shows the disciplines and the iterations applied to develop the components of a one layer of the data warehousing architecture. It depicts also the mapping between the 2TUP activities and the MDA transformation process. The DW engineering process starts by the preliminary study activity. The preliminary study contains a study of the enterprise business process to collect business informations, identifying preliminary requirements, and a study of market platforms to prepare the technical requirements modeling. Each DW layer is developed using a MDA transformation process starting by the definition of the layer BCIM (Business CIM), using the common TCIM (Technical CIM) and ends with components code generation. Consequently, the MDA process is repeated for the construction of each DW layer. If several components are defined in a layer, then several 2TUP iterations may be applied: for example, runs iteration per component. The final DW engineering process showing the development of all DW layers using QVT transformations (T) is given in [2].

The design layer represents the DW design framework; it shows the projection of MDA meta-levels on the data warehousing architecture. We use an MDA compliant metamodels (M2 meta-level) to design the data warehouse layers. The Common Warehouse Metamodel (CWM) and its extension (CWMX) are used as a specification for modeling metadata for the most objects found in a data warehousing environment. The Ontology Definition Metamodel (ODM)
is proposed to design some model presented as ontology, used to solve the semantic schemas integration and the semantic data integration problems. The Meta-Object Facility (MOF) is proposed as meta-metamodel (M3 meta-level) to implement the CWM, CWMX, ODM, and other profiles (some supplier’s platforms metamodels for example) using the metamodeling techniques. Note that the M1 meta-level (models) corresponds to the designed viewpoints (CIM, PIM, PDM, and PSM) during development. The ODBIS domain model itself is based on this metamodeling architecture to allow a semantic mapping between standards concepts provided by CWM and business concepts and to support standard meta-data serialization format. Please refer to [2] for more details on which packages of CWM, CWMX, and ODM are used to design DW components.

3.3 The Technical Architecture
The ODBIS implementation is based on Java Enterprise Edition (JEE) technologies using Spring framework. A typical JEE application architecture is given by figure 4. This architecture shares some layers (UI, services and data layers) with the global ODBIS architecture (figure 1). However, the functional architecture in figure 4 gives more detail by adding the data access, the domain model layers and communication links between them.

The data access layer allows a simplified way to access data and offers an abstraction level for the services layers to manipulate much heterogeneous persistent storage. The Object-Relational Mapping (ORM) tools, called also persistence APIs such as Hibernate or iBatis, are in general used to support the implementation of this layer.

The domain model layer contains domain classes that represent the business concepts of the information system. The domain objects are used by all layers can represent a large proportion of meta-data that are serialized into the data repository. Current ODBIS domain model implements the Common Warehouse Metamodel (CWM) and CWMX (a set of CWM extensions) metamodels. For the future, we plan to integrate other metamodels as the Ontology Definition Metamodel (ODM). The implementation of all these metamodels is based on the Java Metadata Interface (JMI) which offers an implementation of the Meta-Object Facility (MOF) specification.

The business rules plays an important role in a service oriented infrastructure and any business intelligence system (essentially performance management). Indeed, a SaaS platform is shared by several customers that have different business processes, the definition of a business rules engine is essential for the orchestration of services. The Business Process Management (BPM) defines the process logic while the Business Rules Management (BRM) implements the decision logic.

The technical architecture of ODBIS is given by figure 5. It is based on standards open-source Java enterprise libraries and tools. The all is configured using spring framework, a very popular framework for enterprise Java applications development and integration. At the data layer, we use PostgreSQL, one of the most mature and advanced open source databases. For persistence layer, we use the Java Persistence API (JPA) to define the object-relational mapping using Java metadata annotations and Hibernate is used as persistence provider for JPA. Drools (http://www.jboss.org/drools) is an open source business rules management tools that can be integrated with spring context. The domain model is based on a java implementation of CWM metamodel which is defined using Java Metadata Interface (JMI) specification. JMI is based on MOF specification. It defines the standard Java interfaces to build standard MOF-compliant metamodels. JMI allows also metamodel and metadata interchange via XML by using the industry standard XML Metadata Interchange (XMI) specification. There are many implementations of JMI, including the reference implementation from Unisys, Sun open-source implementation from the NetBeans...
group, and many other implementations that are part of various company’s products. We use the Sun open-source implementation called Metadata Repository (MDR) available on http://mdr.netbeans.org/. For the presentation layer we use Java Server Faces (JSF) technology. Sun Mojarra, Apache Trinidad, and Spring Faces are the JSF libraries used to define user interfaces. Finally, all services run under Apache Tomcat web server, a standard container for Java server technologies.

Current ODBIS release covers the administration, metadata and reporting services. The project is available at http://odbis-project.sourceforge.net/. The administration service provides a secure web-based application to manage authorities (privileges), roles, users, and groups. It offers also some search futures. The reporting services provides: (i) futures to manage report-groups and reports; (ii) a BIRT reporting module that allows upload and execute BIRT reports under the integrated BIRT Viewer; (iii) an ad-hoc reporting module which offers an easy way to defines chart reports, data-table reports and to build dashboards. A dashboard example using ODBIS ad-hoc reporting for Healthcare case is given by figure 6. The meta-data service provides futures to define data-sources and data-sets. DataSource objects provide a set of information (URL, User, Password, etc.) used to connect to database servers. DataSet objects are a SQL query abstraction used by charts, data-tables and dashboards.

4. CONCLUSION AND FUTURE WORK

In this paper, we have introduced the ODBIS project. The aim of this project is to provide an open-source platform to deliver On-Demand Business Intelligence Services. The main contributions of the work presented in this paper are: (i) a definition of current data warehousing and business intelligence problem statement including DWs components construction and BI applications deployment; (ii) a unified and standard multi-layered architecture to support on-demand business intelligence services; (iii) a DW development approach supported by an integrated environment for Model Driven Data Warehouse Services (MDDWS) allowing DW models design and derivation in order to reduce DWs development complexity (multi-layer and multi-component architecture, goals definition, etc.); (iv) a flexible technical architecture based on open-source tools and APIs with an out-of-the-box integration. For the future, we plan to improve current realized services (administration, reporting and meta-data). We start soon the development of the integration and analysis services. Finally, the MDDWS module is under incubation and a beta-version will be available at the end of this year.

5. REFERENCES