Towards the Development of Large-scale Data warehouse Application Frameworks

Duong Thi Anh Hoang[1], Hieu Tran [1], Binh Thanh Nguyen [2] and A Min Tjoa[1]

[1] Institute of Software Technology and Interactive Systems, Vienna University of Technology
Favoritenstrasse 9-11/188, Vienna, Austria
{htaduong,tran.hieu, amin}@ifs.tuwien.ac.at
[2] International Institute for Applied Systems Analysis (IIASA)
Schlossplatz 1, A-2361 Laxenburg, Austria
nguyenb@iiasa.ac.at

Abstract. Facing with growing data volumes and deeper analysis requirements, current development of Business Intelligence (BI) and Data warehousing systems (DWHs) is a challenging and complicated task, which largely involves in ad-hoc integration and data re-engineering. This arises an increasing requirement for a scalable application framework which can be used for the implementation and administration of diverse BI applications in a straightforward and cost-efficient way. In this context, this paper presents a large-scale application framework for standardized BI applications, supporting the ability to define and construct data warehouse processes, new data analytics capabilities as well as to support the deployment requirements of multi scalable front-end applications. The core of the framework consists of defined metadata repositories with pre-built and function specific information templates as well as application definition. Moreover, the application framework is also based on workflow mechanisms for developing and running automatic data processing tasks. Hence, the framework is capable of offering an unified reference architecture to end users, which spans various aspects of development lifecycle and can be adapted or extended to better meet application-specific BI engineering process.

Keywords: Business intelligence, Data warehouses, Application framework, Large-scale development.

1 Introduction

Within the application scope of developing DWHs, it is necessary to implement the underlying data basis whereupon this task comprises selection and configuration of relevant software and hardware components for the DHWs architecture [5]. Based on the requirements definitions, the design specification develops suitable database schemas, selects adequate system components, determines partitions of data base tables and specifies ETL processes. The size and complexity of DWH systems are increasingly involved in enabling large data service to scale when needed to ensure consistent and reliable operations. Therefore, it is difficult to design and maintain large-scale application framework with its own characteristics (dynamicity, scalability, etc) [1] over heterogeneous environments.

The existing systems available in the market already offer pre-built data models, preconfigured applications and prearranged contents for common business domains, e.g. customer analytics, financial analytics, HR analytics and supply chain analytics. Moreover, the systems also enable various kinds of data integration and information delivery. Several BI suites also offer deliver ready-to-use templates and include features to build own ones. However, these approaches still tightly bound to the BI solution products. As the analysis of various multi-dimensional modeling methods shows, libraries comprising reusable elements of DWH reference models are mostly specialized on particular model element types [4].

Although these various emerging services have reduced the cost of data storage and delivery in DWHs, DWH application development still remains a non-automated, risk prone and costly process. Much of the cost and effort for developing a complete enterprise DWH stems from the continuous re-discovery and re-invention of core concepts and components across the BI industry. In this context, there arise the requirements for an generic application framework that can enable an asset base of reusable information models and a structured platform to develop and incorporate DW and BI applications, in a cost effective manner and in the available set of tools and technologies.

Within the scope of this paper, a large-scale DWH application framework lays the groundwork for a new paradigm to specify, design, host and deliver standardized BI applications, with common consistent procedures such as data processing defined as consistently reusable components. The basis of the framework is the metadata-based architecture, including generic data models libraries and pre-built application libraries along with the associated mappings. Integrated with the data modeling components, the application framework enables an integrated set of services that collaborate to provide a reusable architecture for a family of DWH eco-system formed of a large numbers of decentralized data analysis services. On this basis, the application model can then be executed and replicated up to the deployment model in a flexible way, enhancing reusability as well as enabling diverse data model preprocessing and analyzing, thus reduces the implementation time and improves the quality of the BI applications.

The rest of this paper is organized as follows: section 2 introduces some approaches related to our work; in section 3, framework architecture and its core component are presented, along with the its design principles. In section 4, the mechanisms of using the framework in developing DWH applications is analyzed. Section 5 illustrates our experimental design to highlight the feasibility of proposed framework. At last, section 6 gives a summary of what have been achieved and future works.

2 Related works

The concept of the application framework have been proposed to describe a set of reusable designs and code in the development of software applications [8]. However, currently, there is still a lack of concepts for designing and implementing a possible adaption of large-scale context to the DWH application frameworks. To sketch the background of the presented research, this section discusses approaches proposed in the domain of BI and DWHs, which could help with large-scale development.

The key to a BI application development, in the large-scale context, depends on scalable, configurable and well-maintained application models. In this context, the scalability can range from complete BI applications down to atomic functional blocks, from data models, ETL tools to deployment packages. Research efforts are very much aware of this trend, and there are no fewer than a dozen companies in recent years that build specialized analytical data management software (e.g., Netezza, Vertica, DATAllegro, Greenplum, etc.), along with the known systems available in the market such as Sybase PowerDesigner, Business Objects Foundation, Oracle BI, etc.

However, within the scope of our knowledge, reference application model of Data Warehouse projects mostly refers to an ad-hoc modification of existing information models [6]. A traditional approach proposed either by industry and academia for applying application and data framework is one of manually assembling and configuring a set of compatible elements as a reference application model [4]. Most of these frameworks still demand a vast amount of repetitive and tedious work to implement similar parts of a DWH application; these products are tied to their own extraction and information delivery tools and lack of pre-built data mappings. This is in part because in current reference DWH models, neither the data models nor the application models has any awareness of the other and these layers operate independently. Therefore, it is difficult to design and maintain DWH applications in large-scale context, especially as the data application models change heterogeneously.

Moreover, to solve the data management problems in large-scale development context, technologies have been developed to manage high volumes of data efficiently, e.g. Google Bigtable (http://labs.google.com/), Amazon Dynamo [3], and so on. Recent years have seen an enormous increase in MapReduce adoption, a Google's programming model to solve the problem of massive explosion in data. Due to its convenience and efficiency, MapReduce is used in various applications as a basis for data frameworks. For example, Apache (http://apache.org/) Hadoop framework is commonly used on a scale-our storage cloud to perform the data transformation necessary to create the analytics database. Hive is a data warehouse infrastructure built on top of Hadoop that provides tools to enable easy data summarization, ad-hoc querying of large datasets stored in Hadoop files.

In this context, this paper is aimed to develop a structured application framework that enables a comprehensive, hosted BI solution tailored to scale and reshape to meet the needs of changing business needs and requirements. Integrated with the data modeling components, the DWH application framework can support implementation and deployment of the DWH applications by leveraging pre-built modeling and application framework, providing a high degree of re-usability, and configurability.

3 Large-scale data warehouse application framework

The proposed application framework enables users to develop DWHs by enabling framework reuse across application designs and platforms. The design of DWH application, along with the tool suites, simplifies the ability to add and remove analytic features without taking into account the underlying framework logic. This section presents an integrated DWH application framework (DWHAF), providing a flexible, scalable solution to enable the rapid development of differentiated DWHs.

3.1 Component analysis of DWHAF

A typical method for implementation and maintenance of DWH application comprising the steps of: analyzing the critical subject area, existing data models and data needs; designing in sequence the logical data modeling and physical database; developing back end services and end user applications; deploying database implementation, initial load and validation of the data warehouse. In this context, it needs to be taken into account the fact that developing BI applications requires an efficient application framework of tightly interoperating components along with a scalable data modeling framework.

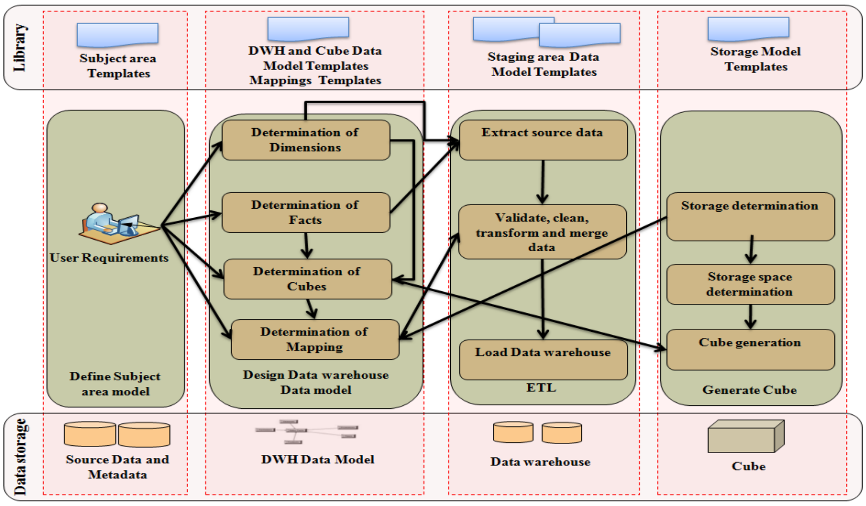

Fig. 1. The required DWH application framework

Figure 1 shows the required application framework with typical development processes resulted from a comprehensive literature review we have conducted. The DWH application framework comprises the phases of Subject area model Definition, DWH Data model Design, ETL and Cube Generation. Phases are subdivided into activities. Within the Subject area model Definition phase, basic questions concerning user requirements and scope of the DWH are answered, in comparison with existing environment templates. To model Data warehouse data model, logical data model and database as well as mappings from source to target are designed. This way an information model of the required DW is created. This information model can be analyzed and the results can be used as a basis for ETL phase. Finally, subject to the development phase is a continuous adaptation and maintenance of the cube models.

The data modeling framework enables end users to define the logical data model using a series of graphical user interfaces (GUI) or application-programming interfaces (API), then dynamically translates the logical data model into a corresponding physical data model. Because the application framework is integrated with the data modeling framework, the application framework features are automatically available to end users, enables them to configure various application features and data management operations also by terms of UI and API. Through the API, BI applications can interact with the application framework, which in turn interacts with the logical data model.

It is clear that the common business domains require common application processing functionality, e.g. data delivery provides users with processing results tailored to their specific needs, for example what data format they accept, what are the relevant parameters they want to retrieve, etc. The required application framework also supports the description and execution of application-specific data processing workflow as well as predefined workflow patterns between distributed components. When considering the whole application processing from input to delivery it appears that instead of a single workflow there could be a variety of workflow definitions for a single application according to different user needs.

In this context, the DWH application framework, including the data modeling functions and application features, is aimed to be a scalable architecture to serve as the warehouse’s technical and application foundation, identifying and selecting the middleware components to implement it. New business requirements or changed requirements may result in the addition of new components to the framework and a modification of the existing framework components. Therefore, it takes a series of iterations to get the framework right for the applications that are built on top of it.

3.2 Constructing the relationships between data and application models

The model component provides the access to pre-packaged data and application models, enabling end users to reconfigure and customize as well as provide aids to dimensional modeling in the DW and BI context. The data models can be distinguished as the model that stores fact and its related dimensions and the model that defines cube structure or relational schema [5].

The data model component of the system offers the means to capture all the possible dimensions and their respective attributes/properties as well as establish mappings between the analytic facts and dimensions. Meanwhile, built-in dimensional models representing the best practices for comprehensive analysis relating to particular business functions. Moreover, data model can be generated based on definitions of dimensions, facts and analytical needs. End users can thus reconfigure these data models via user interfaces.

Meanwhile, metadata associated with a application model used to describe of a template of the interface and functionality of the application. These templates consist of ETL and warehouse design, mappings, data storage, deployment templates or application specifications. An example of application metadata model can include defined options to personalize the DWH applications, which are used in the runtime reconfiguration process when it is instanced in a specific application. This metadata can be executed at run time to provide the design interface for the designer, and to deploy the model into the output data storage structure in DWH applications.

In accordance to the requirements of integrated framework, the variability of the application configuration should be based on the consistent variability of application/business models, i.e. data models, application models or domain models. By merging the application constraints with the data model constraints, we obtain a unified set of constraints that is used by the configuration tool to ensure that the template model generated is consistent and up to the business domain.

The first step towards the construction of a relationship is to capture the variability of a given domain by means of a data modeling service. Similarly, this data model should be represented by configurable data exchange and data store services model, respectively. Application specification should be identified with the variants of the applications and constraints should be defined to ensure the correctness. Once data model, application specification and their constraints have been identified, it is possible to define the mapping by means of the impact analysis:

- from data model to application: given a data model specification, we need to estimate what are the implications to the setting in the application model;
- from application to data model: given a change in application specification, we need to consider which data model are impacted.

A valid mapping is constructed when a valid application specification is defined with respect to the data model. The specification space is obtained by the intersection of the two specification spaces (data and application) via the mapping. By representing the data model variability in a separate model, we can avoid capturing the interdependencies of the data model in the configurable application model, as these interdependencies are propagated to the application model via the valid mapping. Once each customized data model has been configured, a set of service specification, attached to the application specification that captures the selected data model, are performed on the application model to define a configuration.

3.3 Architecture of proposed DWH application framework

The system architecture consists of abstract services that are the core components of DWHAF applications namely Application, Client, DWH and Platform services. Specifically, the framework also contains the main components: the metadata services, the management services and the libraries. The Client layer contains top-level component such as Application interfaces, Service APIs, as well as specific services related to Application service framework.

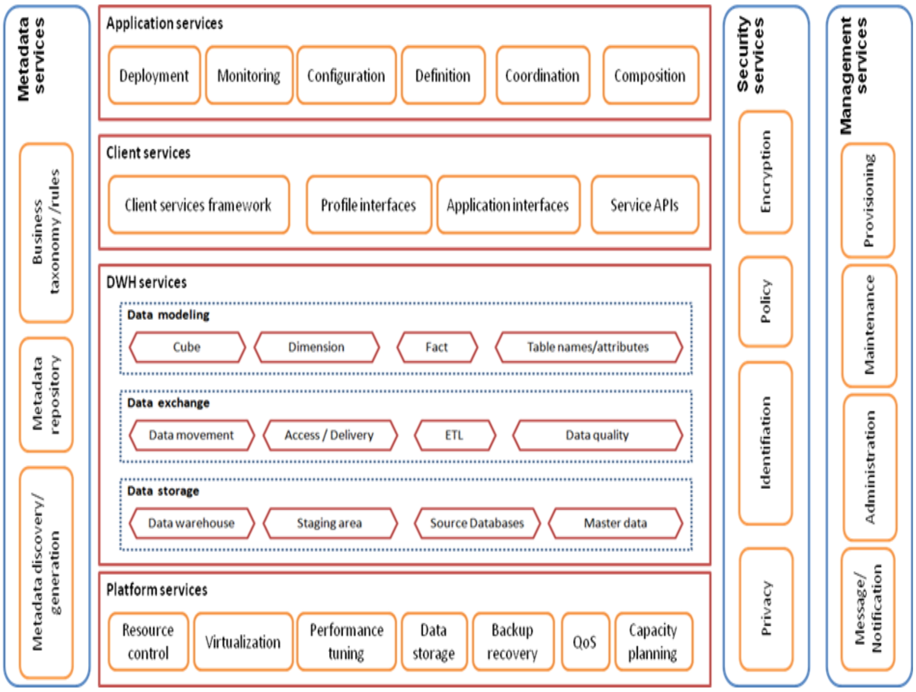

Fig. 2. DWHAF conceptual architecture

3.3.1 Application services

DWH reference models are the deliverable end result of an application developed in DWHAF. This layer is where context-aware DWH configuration are implemented. The DWH application designer can extend the services by means of the API libraries. All instantiated application models, both native and third party, and other supplementary components, reside in this layer. The application layer runs within the system run time, using the templates and services available from underlying layers.

3.3.2 Client services

This layer is the place where the context information is processed and modeled. This context information acquired by the components at this layer helps services to respond better to varying user activities and environmental changes. Additionally it serves as an "inference engine" that deduces and generalizes facts during the process of updating context models as a result of information coming from Profile interfaces.

3.3.3 DWH services

This layer provides the services used to develop DWH applications as well as a generic abstraction for platform access and manages the user interface and application resources. Built around API connectivity, the proposed framework also includes pre-integrated DWH solutions, validated on multiple platforms, cutting down the time cost to launch DWH model. Moreover, customers can plug in components of their choice, with minimal impact on system integrity.

3.3.4 Platform services

To ensure efficient development, customization and administration of DWH applications, the platform layer offers application delivery. Specifically, DWH development relies on a variety of core services, i.e. security, performance tuning, data storage, visualization packages, data analysis tools, and basic environments. Along with the libraries, this layer forms the deployment basis for the application framework. For example, the Management Services offer application configuration and reporting capabilities; Virtualization delivers abstraction between physical and functional elements in service management.

3.3.5 Metadata services

As presented in the component analysis process, there’s a clear separation of different kinds of metadata, i.e. metadata that describes the data model, the application model, the base functionality of an application, and the customizations. The Metadata repository provides a central, shared source of metadata including pre-built metadata, enabling reduction in implementation and maintenance costs. The metadata repository also serves as a key repository of all mappings between application models and function specific data models housed in the data model component.

Along with the data model component, the system enabling user interfaces to reconfigure the reference models, preventing the risks associated with being locked into a specific configuration. Meanwhile, the library layer contains repositories of pre-built templates for DWH application development. This layer will grow as development of the framework continues and common themes are identified. Specifically, the architecture provides common practice solutions to reduce the effort of data modeling, in which they can be used as a starting point for the construction of application-specific models. Meanwhile, a metadata repository supports application implementation, i.e. definitions of data objects, application components, runtime customizations, etc.

4 Applying DWHAF in application development

Built on top of the framework, DWH application service provides the design for specific applications and the query interface required by the DWHs solution. The application service provides both the tools and applications that enable predefined and ad hoc collection, analysis, and presentation of end user's data by means of user interfaces and service API. The result of this design service includes an application prototype, recommended design templates and development tools to support implementation, and deployment plans for the required applications.

Fig. 3 provides an overview of our proposed framework for generating DWH application architecture. DWH designers interact with the service composition environment through a composition user interface, navigating through the generated DWH reference models, edit the process, and deploy DWH applications. The back-end includes major components that process the contextual information, generate reference processes and analyze historic usage data to recommend DWH templates.

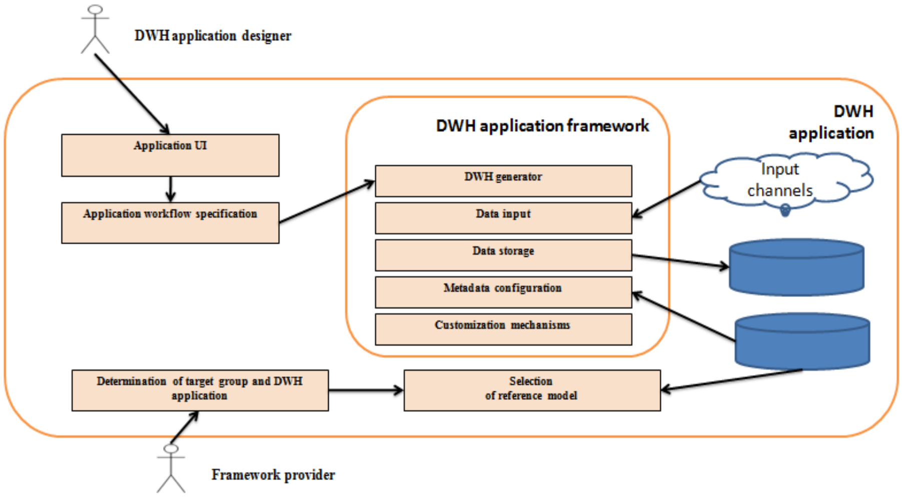

Fig. 3. DWH application framework and application service

From the modeling viewpoint, the ability to use history data is an important feature. An intuitive view to application-specific DWH design would be to satisfy the need for (non-existing) DWH services by bringing together existing ones. We aim to reuse development knowledge and provide a starting template for the users. This knowledge can be based on past successful deployment, and can be seeded by domain-specific knowledge and task-specific knowledge about the core types of information processing activities.

In this context, the presented application framework enables designers to develop DWHs by enabling pre-built service libraries reused across application designs and platforms. The design of DWH applications, along with the tool suites, simplifies the ability to add and remove analytic features without taking into account the underlying framework logic. The result of the architecture design is the Reference Architecture, which is an architectural framework along with a set of middleware elements facilitating the integration of services, and context modeling. An example of adapting the generic component for the implementation of an ETL specific component is presented in Figure 4. The example shows the implementation of the selective data storage service, and the other components can be implemented in a similar manner.

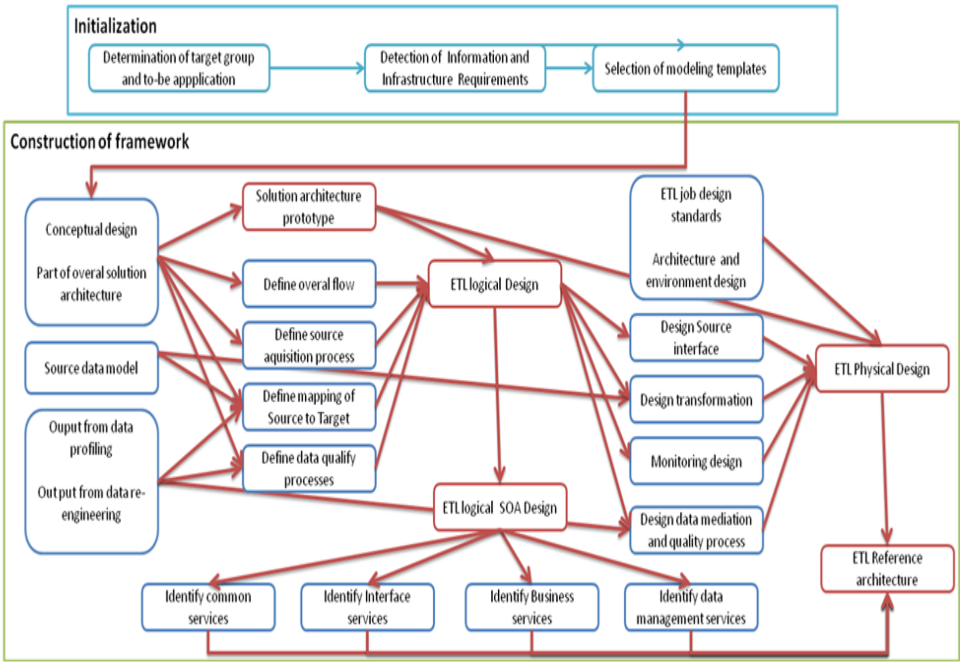
Initialization
Determination of target group and to-be application
Detection of Information and Infrastructure Requirements
Selection of modeling templates
Construction of framework
Conceptual design
Part of overal solution architecture
Source data model
Ouput from data profiling
Output from data re-engineering
Solution architecture prototype
Define overal flow
Define source aquisition process
Define mapping of Source to Target
Define data qualify processes
ETL logical Design
ETL logical SOA Design
ETL job design standards
Architecture and environment design
Design Source interface
Design transformation
Monitoring design
Design data mediation and quality process
ETL Physical Design
ETL Reference architecture
Identify common services
Identify Interface services
Identify Business services
Identify data management services

Fig. 4. Example process of defining ETL reference architecture

The application framework aims at providing a reference template configured to enable end users to leverage prebuilt practices and to avoid designing and developing DWH applications from scratch [9]. Moreover, we focus on the stage of dynamic template instantiation, in which we need to identify specific service instantiation for each of the generic service in the template. The focus of supporting end users means that, we aim to automate only the main aspects of the process: selecting suitable services for each task and working out compatibilities between services in terms of data flow, pre- and post-conditions. For example, for querying services, the related pattern can be defined as Warehouse GUI to en-queue queries, check status and retrieve results.

5 Illustrative example

To establish the practical feasibility of our framework, we have designed a toolset that provides end-to-end support for DWH application model configuration. In this case study, we take advantage of Hadoop environment, which based on MapReduce programming model, in optimizing data warehouse performance. Specifically, experimental ETL application acts as a transformation engine, taking the extracted data from multiple data sources and processing this large-scale data into a common format for integration into the data warehouse. As a proof of concept, a data warehouse of Climate subject area will be deployed, in which we use Pentaho BI suite which extends their ETL (Pentaho Data Integration – PDI) to support processes that exploit Hadoop structures. Figure 5 provides an overview of the toolset's architecture.

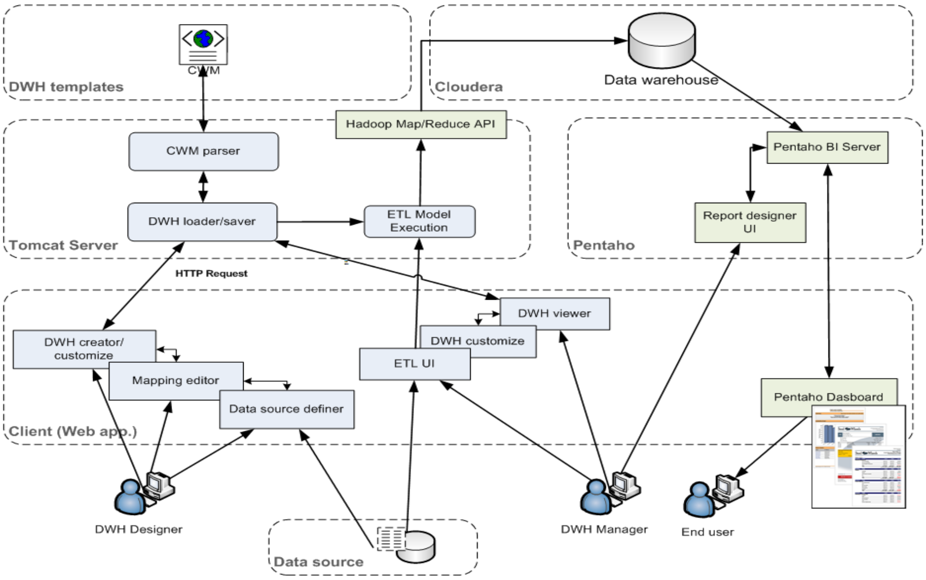

Fig. 5. Prototype implementation architecture

The U.S. National Climatic Data Center (NCDC, http://www.ncdc.noaa.gov/) provides free data on a set of meteorological elements, relating to *Daily Normals of Temperature, Precipitation, and Heating and Cooling Days* for the period 1971-2000 for stations of the U.S. Climate Reference Network (USCRN). These systems were considered as data sources for implementing the core of data warehouse, in which the dimensions can be:

- Time: time key, day number in month, month, season of the year
- Division: division key, region, state
- Weather station: station information (Station Name, State, Division, Call) , location (Latitude, Longitude, Elevation, Elements)

- Climatic variable: variable key, name (Minimum Temperature, Maximum Temperature, Average Temperature, Heating Degree Days, Cooling Degree Days), description, measurement units (deg F, Hundreths of an Inch, none), type (Temperature, Precipitation, Degree Days)

The fact tables can be defined from various subject areas, such as the value of a variable measured at one station an on a date given station key, time key, variable key, and measured value, monthly sum.

There is also a version management used to relate the instances of the model back to the generation of application template. Every time a application is instanced from that template, the instanced application gets a copy of the version attribute that describes which version of the template was used. When a designer customizes an instance of the application, these customizations are stored with the instanced application, they are not propagated back to the template.

6 Conclusions and future works

The rising demand for BI and DWH applications is fostering the need to support flexible deployment models for all BI application services. With ever growing data volumes and deeper analysis requirements, it also follows that DWH application framework must be able to scale out to meet operational and technical requirements of hosting and delivering applications. In this paper, a metadata-driven application framework is presented as the core that enable the platform to deliver configurable and scalable DWH applications. The core of this framework is that the application framework is tightly integrated with data modeling components, thus supports an associated and centralized logic to ensure the consistency of the DWH applications.

Focusing on the potentials of semantic technologies [10, 11, 12], we are working on applying sound formal techniques to represent and automate the mapping between domain data model and application model. To establish the practical feasibility of our framework, we have designed a toolset that provides end-to-end support for DWH application model configuration and conducts DWH development experiments with various subject areas to empirically evaluate the applicability and impact of the proposed configuration approach on the deployment process.

7 Acknowledgments

This work is supported by a Technology Grant for South East Asia of the ***Austrian Council for Research and Technology Development*** and ***ASEA UNINET***.

References

1. Chuck, H., Jeff, W. and Karen, C. 2009. Navigating the Next-Generation Application Architecture. *IT Professional* 11, (2 2009), 18-22.

2. David, P., Awais, R., Alexandru, T. and Andreas, S. 2006. An architectural pattern for designing component-based application frameworks. *Software: Practice and Experience*, 36, (2 2006), 157-190.

3. Giuseppe, D., Deniz, H., Madan, J., Gunavardhan, K., Avinash, L., Alex, P., Swaminathan, S., Peter, V. and Werner, V. 2007. Dynamo: amazon's highly available key-value store. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles.* SOSP '07. ACM New York, 205-220.

4. Goeken, M. and Knackstedt, R. 2007. Multidimensional Reference Models for Data Warehouse Development. In *Proceedings of the International Conference on Enterprise Information Systems.* ICEIS. 347-354.

5. Kimball, R., Ross, M., Thornthwaite, W., Mundy, J. and Becker, B. *The Data Warehouse Lifecycle Toolkit*. Wiley Publishing, 2008.

6. Knackstedt, R. and Klose, K. 2005. Configurative reference model-based development of data warehouse systems. In M. Khosrow-Pour (Ed). *Proceedings of the 16th Information Resources Management Association International Conference. RMA 2005.* 32-39.

7. Luján-Mora, S. and Trujillo, J. 2003. Comprehensive Method for Data Warehouse Design. In *Proceedings of the 5th International Workshop on Design and Management of Data Warehouses (DMDW'03)*. 1.1-1.14.

8. Mohamed, F. and Douglas, C. S. 1997. Object-oriented application frameworks. *Commun. ACM* 40, (10 1997), 32-38.

9. Oracle. Enabling Pervasive BI Through a Practical Data Warehouse Reference Architecture. 2010

10. Oscar, R. and Alberto, A. 2007. Automating multidimensional design from ontologies. In *Proceedings of the The ACM tenth international workshop on Data warehousing and OLAP* (Lisbon, Portugal, 2007). ACM. 1-8.

11. Oscar, R., Diego, C., Alberto, A. and Mariano, R. M. 2009. Discovering functional dependencies for multidimensional design. In *Proceedings of the The ACM twelfth international workshop on Data warehousing and OLAP* (Hong Kong, China, 2009). ACM. 1-8.

12. Spahn, M., Kleb, J., Grimm, S. and Scheidl, S. 2008. Supporting business intelligence by providing ontology-based end-user information self-service. In *Proceedings of the The first international workshop on Ontology-supported business intelligence*. ACM. 1-12.