

Towards a Model-based Application Integration Framework for Smart Oilfields

Cong Zhang, Amol Bakshi, Viktor Prasanna*, Will Da Sie†

Abstract

The increasing demand for cost-effective oil and gas production has led to an industry-wide push to develop smart oilfields for the future. Applications for smart oilfields are characterized with heterogeneous data and resources, complicated business processes, and changing business requirements from users. Existing software development process and techniques have become increasingly incapable of managing such complex software systems. Model-based integration frameworks are based on a domain-specific modeling language and a common model database. They offer the benefits of extensibility, modularity, and resuability of both code and design to the applications. In this paper, we describe a prototype integration framework for a class of oilfield applications. To demonstrate the advantages of the integration framework, we will show how applications are developed and integrated in the framework in a systematic manner.

1 Introduction

A smart oilfield is one that uses technical innovations to measure, analyze, and optimize the cost-effective extraction of oil and gas production [2]. In the past few decades, the major effort has been invested in the use of engineering technologies and computer technology to model, simulate, and predict the behavior of reservoirs. Such software applications have become very sophisticated, but they generally do not work together. The key to developing smart oilfields is building an integration environment (framework) which allows a variety of independent simulators, databases that store historical data, and real-time production data to communicate with each other. With an integration framework, completely new synergies, such as new optimizations, collaborative problem solving and decision making, can be envisioned and achieved.

*EE-Systems, USC, Los Angeles, California 90089. Email: {congzhzhan, amol, prasanna}@usc.edu

†Chevron, San Ramon, California 94583. Email: Will.DaSie@chevron.com

In software development, a framework is a defined support structure in which another software project can be organized and developed [3]. Generally, a framework consists of support programs, code libraries, or scripting language to facilitate application development in the framework. For example, Microsoft .NET Framework [8] provides a large amount of pre-coded solutions to common program requirements, and manages the execution of programs written specially for the framework. The pre-coded solutions make up the framework's class library which addresses a wide range of programming needs, such as user interface, data access, cryptography, numeric algorithms, and network communications.

Generally, application integration supported by an integration framework can be classified into two kinds. The first is integration of legacy applications or services. These applications may be developed by different people using different languages and technologies, run on different hardware platform, use different operating systems, and provide very different functionalities. A framework that supports this kind of application integration should have: (i) a common interface for applications to communicate; (ii) data transformation logic; (iii) abstract interfaces to insulate the framework from existing technologies.

The second type of integration is of applications that are consciously designed with the express purpose of integration into the larger framework. When implementing these applications, services and APIs provided by the framework will be used to facilitate the development process. A framework that supports this kind of application integration, which is also the focus of this paper, should provide a common set of services so that applications can be designed and composed in the framework.

In this paper, we describe our experience with the design and implementation of an integration framework for a class of oilfield applications. This work is part of a broader effort towards developing a software framework for Integrated Asset Management in smart oilfields [5]. Our framework is based on the concept of Model-Integrated Computing [7], which is a system software development approach that promotes the use of domain-specific models to represent relevant aspects of a system. It consists of three major components: a *domain-specific modeling language*,

schema definition library, and a *common model database*. The domain-specific modeling language in the framework provides a common vocabulary for domain-experts to define and “understand” domain concepts, and forms the basis for various tools to navigate the model database and retrieve and update specific model parameters. Schema definition library contains the descriptions of the structure and format of data. Each schema describes property information pertaining to the records and fields within the structure of data so that data can be shared and exchanged across different applications. The model database stores real data itself or metadata for accessing the real data that is used by the applications. The applications that are integrated through the framework read and write directly from the model database. By allowing multiple applications to work on the data in a coordinated manner, the model database eliminates the need to write adapters for tools to communicate directly with each other. With the three components, model-based integration frameworks offer the benefits of extensibility, modularity, and resuability of both code and design to the applications.

Section 2 gives background of smart oilfields. In Section 3, we examine current practices of application integration. Section 4 outlines the architecture of our model-based integration framework. Details about application development using our integration framework is described in Section 5. We conclude the paper in Section 6.

2 Smart Oilfields

The objectives of smart oilfields are reducing cost and increasing efficiency of oilfield operations by deployment of permanent sensors and controllers to monitor an oilfield and alter production schedules continuously or on demand. Such an oilfield management strategy can allow decision making with integration of static and dynamic data, and visualization of risk, uncertainty ranges and various financial metrics, using experts in remote collaborative environments.

In an ideal digital oilfield operation that is governed by an intelligent supervisory management system, continuous data are obtained from individual wells, geophysical recorders and surface operation facilities. The data is analyzed in real time and can serve as the basis for progressively more accurate oilfield definition and characterization, management planning and control of the wells. Such an idealized system will enable unmanned actuation of smart devices to control flow by plugging, checking or choking production intervals without intervention and production halt. Real time interpretation of a heterogeneous data system and instant consequential analysis of alternative will open the possibilities for intelligent asset protection, recovery optimization, human safety in the oilfield and environmental

protection leading to substantial economic gains over conventional operation.

There are many challenges in realizing smart oilfields, some technical and some organizational. One of the key technical challenges is building an integration environment which allows a variety of independent simulators, databases, real-time production data, and business components to communicate with each other. Complex architectural issues can arise in such an environment, such as data heterogeneity, tool interoperability, etc.

3 State of the Art

Over the past few years, there have been large amount of research and industry effort spent on smart oilfields. For example, Intelligent Energy 2006 [6] is a groundbreaking conference that focuses on new ways of improving performance of oil and gas fields in the future. Many key challenges in developing a smart oilfield have been addressed, such as real-time asset management [11], data exchange format [12]. In terms of tool interoperability, Integrated Production Modeling toolkit (IPM) developed by Petroleum Experts [10] integrates applications from reservoir to the surface network and performs predictions and accurate optimization over an entire oil field. In [9], experience in implementing smart oilfield concept on a real oilfield asset is reported.

Existing practices of application integration can be classified into two major types, depending on where integration effort is spent. Data compatibility and tool interoperability are the two major issues in integration application. The first type is process-centric integration. It deals with the automation of business processes by integrating functionality from different applications. For instance, Aspen Framework [1] is an integration infrastructure for enterprise-wide business process solutions. It provides management of security and roles, supports automation and execution of business processes, provides enterprise wide data-sharing via object and file repositories, and a full set of capabilities for solution builders. The Aspen Framework architecture is a multi-tier distributed application component architecture. Aspen Framework is based on the Microsoft Distributed Internet Architecture (Microsoft DNA) and is fully Web-enabled.

The second type is data-centric integration. This kind of integraton facilitates data exchange and manipulation among disparate systems by creating mappings between data exchanged among them. Based on this type of integration, organizations can increase their functionality and performance because their information system is capable to process and combine data from various applications. Techniques that support data-centric integration include Electronic Data Interchange (EDI), data adapters, etc.

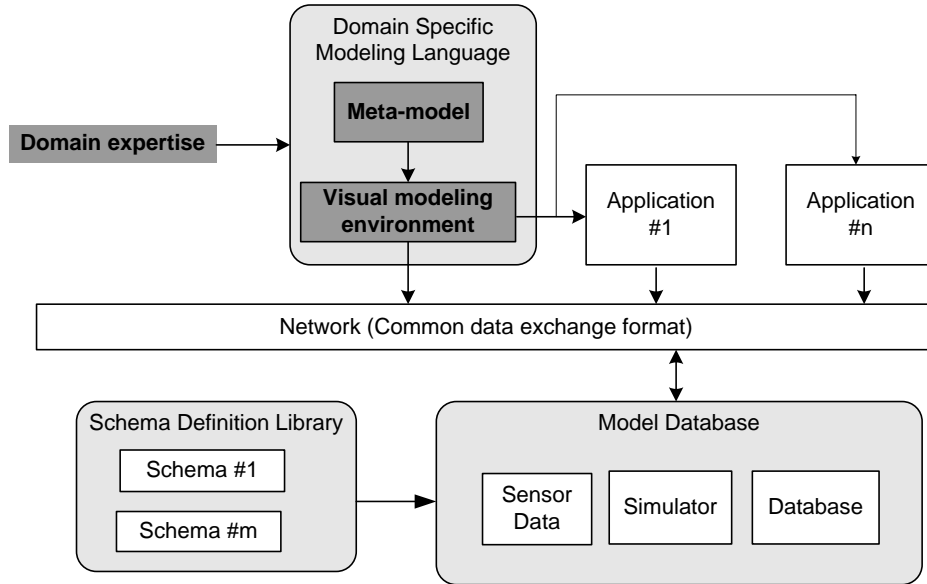


Figure 1. Architecture of our framework

Our integration framework can be categorized as data-centric. The model database in our framework stores real data or metadata for accessing the real data. The applications that are integrated using our framework read and write directly from the model database. A uniform interface is provided for them to perform data access with the model database. By allowing multiple applications to work on the data in a coordinated manner, the model database eliminates the need to write adapters for tools to communicate directly with each other. To the best of our knowledge, our work is the first to use domain models to address application integration for smart oilfield.

4 Architecture of Model-based Integration Framework

Our framework is a model-based simulation environment which enables the integration of various simulators and tools into a unified environment through its common model database. Using our framework, the end user formally models the data set required by a workflow through a graphical interface provided by our framework. The models are stored in a model database. The model information is translated through model interpreters into suitable input formats required by the integrated simulators and tools. Model interpreters are the software components that translate the information captured in the models based on the input format required by the integrated tools and simulators.

As depicted by Figure 1, there are three major components in the architecture. In the following subsections, we

describe each component in detail.

Domain specific modeling language

Our framework adopts Model Integrated Computing (MIC) as the core design technology. The visualization of the data, definition and synthesis of specific workflows, and the invocation of integrated applications, is all managed through a domain-specific visual modeling environment. The grammar of the visual modeling language is itself specified through a metamodel which is a UML-like metamodeling language. The metamodel is defined in consultation with domain experts.

We use Generic Modeling Environment (GME) [4] to create our domain-specific visual modeling environment. GME is a configurable graphical tool suite supporting MIC. It allows the designer to create domain-specific models. A metamodel (modeling paradigm) is a formal description of model construction semantics. Once the metamodel is specified by the user, it can be used to configure GME itself to present a modeling environment specific to the problem domain.

In Section 5, we will show details about a modeling language for our specific domain - oilfield asset modeling.

Schema definition library

A schema definition library is the key to application integration at the data level. It tells the application developers what data is available and how to access it. Besides, the schema definition library can be a basis for code generation, thereby reducing the time and effort in application development.

Schemas describe the structure and format of data. Each schema contains property information pertaining to the records and fields within the structure of data. This information is vital to ensuring that the data is human readable. Schema definitions are presented in a number of different forms in different applications. These forms include well-formed XML, document type definitions (DTDs), EDI, and structured document formats. The schema definition language used in our framework is XML Schema Definition (XSD) [13].

Schema definitions are the foundation of a number of data transformation tools provided by our framework. In the form of standalone programs or XSLT scripts, these tools render the content of input data elements to the corresponding elements of the output data as specified by the schema definitions. One of the most important tasks of these tools is data aggregation/disaggregation. Data is combined from multiple sources or from a single source over time in data aggregation. In data disaggregation, data is broken up into pieces of output data.

Model database

The visual modeling environment provides a graphical user interface that is used to instantiate, inspect, and modify various models. It stores the model information in a proprietary format that is programmatically accessible. In the interest of standardization and open, platform-independent access to the model database, we choose to decouple the storage of the model data from the modeling environment. Thus changes to the storage technology do not lead to major changes to the whole architecture.

One benefit of having a platform-independent model database is that we have much freedom to choose the appropriate storage technology, depending on user and application requirements. For instance, both traditional SQL database and flat or structured files can be used to implement the model database.

To facilitate data access to the model database, our framework provides classes and interfaces for persistent data management with the model database. They allow users to set up the connection string to the model database and support *read* and *write* operations. The classes and interfaces provided by our framework ensure a uniform data access to the model database across different applications.

Alongside the data access classes, our framework supports a number of programs to let user manipulate data in the model database through the modeling environment. Specifically, the process of committing changes from the modeling environment to the database is termed as “exporting” the model and the process of creating and updating models in the modeling environment with the data from model database is called “importing”. Although the model database can (and typically will) be modified by all the integrated applications, these prepackaged applications are con-

sidered as an integral part of our framework and provided to the user.

It is not trivial to design and implement the model database. In [14], we describe a simplified version of the model database that we have implemented. It consists of various data files in the form of ASCII or XML format. Work is in progress to improve the model database. For example, the next version of our model database will extract and manage metadata of models, such as relationships of between models, model-specific information, etc. The metadata will serve as the basis of providing better data access to the model database.

5 Application Development in the Framework

Using the framework described in Section 4, we have developed a software system, which is a part of Integrated Asset Management (IAM) system for ‘smart oilfields’. More information on IAM can be found in [14, 15]. The software system is called Integrated Forecasting Workflow.

5.1 Integrated Forecasting Workflow

In this workflow, the end user wishes to analyze future production of the particular oilfield asset by configuring various “what-if” scenarios. Each scenario could correspond to a different decision point related to, say, investment in surface facilities.

The input data set for this workflow can be divided into two main categories: model information, and system and production controls. The model information consists of data about reservoir, wells, and other oilfield equipment. Controls that include production targets, well events of significance, etc are passed to an optimization core. The main (default) objective function is to maximize oil production.

The output of the workflow for a given scenario includes production data at the desired level of granularity for the whole field. Graphs are plotted based on the output data, as per the users requirements.

For this workflow, the domain specific modeling environment serves as the primary user interface. It is tasked with automating the routine work involved in setting up the data that is input to the forecasting tool, configuring various model parameters, invoking the tool, analyzing the output, and generating the desired reports and graphs.

5.2 Steps

There are two major steps to implement the software application.

Step 1: Domain specific modeling

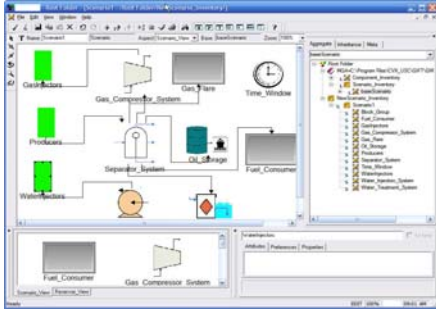


Figure 2. The domain-specific modeling environment

In collaboration with experts from Chevron and Petroleum Engineering department at USC, we have defined a domain specific modeling language [14, 15]. Figure 2 is a screenshot of the model environment based on GME.

With GME, we develop a domain-specific modeling language for describing a generic oilfield asset. The language is based on the Unified Modeling Language (UML), provides a common vocabulary for domain-experts to define and “understand” an asset model, and forms the basis for various tools (such as the forecasting tool) to navigate the model database and retrieve and update specific model parameters. The current version of the modeling language is capable enough to describe all the physical and non-physical model information that acts as input to the integrated forecasting workflow. We expect to continuously refine this modeling language based on experience with other workflows and other types of assets.

The objects in an oilfield asset model are classified into physical and non-physical components. Physical components include wells, reservoir volume elements, separators, compressors, etc. Non-physical components include production controls, field constraints, drilling schedules, reliability models, among others. The details of how each component is modeled in this language can be found in [15].

Step 2: Application development

The Integrated Forecasting Workflow that we have developed consists of two applications: an oil production forecasting tool and a well drilling schedule program. The oil production forecasting tool is the primary application in the workflow. It basically implements the logic described in Section 5.1. The well drilling schedule program takes as inputs a preferred drilling order, trajectory information of the wells, and rig allocation etc. The onstream date of each well in the drilling schedule is calculated by the program. In our current workflow, the well drilling schedule program is called whenever the onstream date of a well in a scenario is not known and is required by the forecasting tool. The cal-

culated results are propagated to the scenario which drives the oil production tool. Figure 3 shows the relationship between the two applications.

Different approaches were used in the development of the two applications. The design and implementation methodology for the oil production forecasting tool followed a ‘traditional’ approach. It included receiving the functional specifications from the end user in various forms. The requirements were translated into code and the final product was provided to the end user for evaluation and feedback.

After we developed the oil production forecasting tool, a domain specific modeling language, and some support programs and classes are available from our framework. Therefore, development of drilling schedule program is representative of application development in our framework.

We started with the design of data required by the drilling schedule program. Most of data was available in the existing model database. For new data types required for the drilling schedule program, we did the following:

1. New schemas are created and added to the schema definition library.
2. XSLT scripts are also written to perform necessary data transformation.
3. For the new schemas, data access classes are coded.

We then updated the domain specific modeling language to accommodate the new workflow which includes the drilling schedule program. Since large amount of information can be shared by the two applications, the update to the modeling language was actually very minor.

After above steps, we implemented the logic of the drilling schedule, which is a traditional application development process.

As a summary, we can see that besides providing support programs and classes, our framework essentially defines an application development process, which helps applications be developed and integrated in a systematic manner.

6 Concluding Remarks

This paper has shown a prototype model-based application integration framework designed specially for petroleum domain. Details of the major components in the architecture of the framework are given. To demonstrate the use the framework, a concrete example is also described in the paper. This example software system consists of two applications, which are integrated through the framework. Development of one of the two applications also shows the basic steps for using our framework in developing an integrated application.

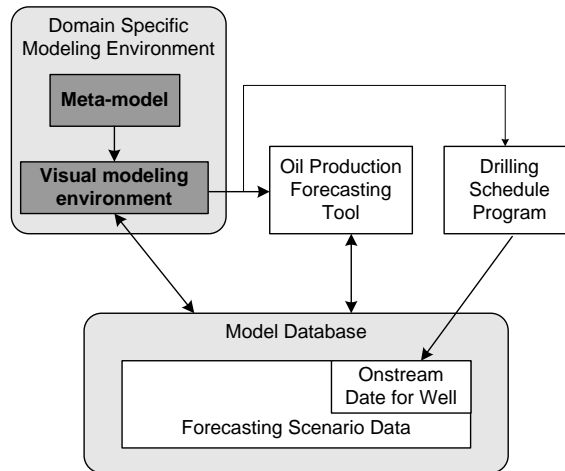


Figure 3. Integrated Forecasting Workflow

Since domain specific modeling raises the level of abstraction, domain models can be used to improve productivity of application development. Therefore, we plan to address the challenges in application development for such model-based integration frameworks and implement a new methodology for semi-automatic, assisted application synthesis that allows the domain-expert end users, software architects, and programmers to interact through a well-defined, integrated modeling environment.

Acknowledgment

This research was partly funded by CiSoft, (Center for Interactive Smart Oilfield Technologies), a Center of Research Excellence & Academic Training and a joint venture between the University of Southern California & Chevron. We are grateful to the management of CiSoft and Chevron for permission to present this work.

References

- [1] Aspen Technology. <http://www.aspentech.com>.
- [2] I. Ershaghi and Z. Omeregic. Continuing education needs for the digital oil fields of the future. In *2005 SPE Annual Technical Conference and Exhibition, SPE 97288*, Dallas, Texas, October 2005.
- [3] Framework Definition. <http://en.wikipedia.org/wiki/Framework>.
- [4] GME: Generic Modeling Environment. <http://www.isis.vanderbilt.edu/Projects/gme>.
- [5] Integrated Asset Management. <http://indus.usc.edu/cisoft-iam>.
- [6] Intelligent Energy 2006. <http://www.ie2006.com/>.
- [7] Model-Integrated Computing. <http://www.isis.vanderbilt.edu/research/mic.html>.
- [8] .NET Framework. <http://www.microsoft.com/netframework>.
- [9] J. Ouimette and K. Oran. Implementing chevrons ifield at the san ardo, california, asset. In *SPE Intelligent Energy Conference and Exhibition*, April 2006.
- [10] Petroleum Experts. <http://www.petex.com/>.
- [11] T. Unneland and M. Hauser. Real time asset management: From vision to engagement an operators experience. In *SPE Intelligent Energy Conference and Exhibition*, April 2006.
- [12] B. Weltevrede, R. Foreman, R. Morneau, B. Rugland, J. Foreman, S. D. Vries, T. Little, L. Ormerod, and A. Doniger. A multivendor data-exchange format to support digital oilfields. In *SPE Intelligent Energy Conference and Exhibition*, April 2006.
- [13] XML Schema. <http://www.w3.org/XML/Schema>.
- [14] C. Zhang, A. Orangi, A. Bakshi, W. D. Sie, and V. K. Prasanna. Model-based framework for oil production forecasting and optimization: A case study in integrated asset management. In *SPE Intelligent Energy Conference and Exhibition*, April 2006.
- [15] C. Zhang, V. K. Prasanna, A. Orangi, W. D. Sie, and A. Kwatra. Modeling methodology for application development in petroleum industry. In *IEEE International Conference on Information Reuse and Integration*, August 2005.