



SPE 99979

Model-based Framework for Oil Production Forecasting and Optimization: A Case Study in Integrated Asset Management

Cong Zhang¹, Abdollah Orangi¹, Amol Bakshi¹, Will Da Sie², and Viktor K. Prasanna¹

¹University of Southern California, Los Angeles, CA 90089 USA

²Chevron Corporation, San Ramon, CA 94583 USA

Copyright 2006, Society of Petroleum Engineers

This paper was prepared for presentation at the 2006 SPE Intelligent Energy Conference and Exhibition held in Amsterdam, The Netherlands, 11–13 April 2006.

This paper was selected for presentation by an SPE Program Committee following review of information contained in an abstract submitted by the author(s). Contents of the paper, as presented, have not been reviewed by the Society of Petroleum Engineers and are subject to correction by the author(s). The material, as presented, does not necessarily reflect any position of the Society of Petroleum Engineers, its officers, or members. Papers presented at SPE meetings are subject to publication review by Editorial Committees of the Society of Petroleum Engineers. Electronic reproduction, distribution, or storage of any part of this paper for commercial purposes without the written consent of the Society of Petroleum Engineers is prohibited. Permission to reproduce in print is restricted to an abstract of not more than 300 words; illustrations may not be copied. The abstract must contain conspicuous acknowledgment of where and by whom the paper was presented. Write Librarian, SPE, P.O. Box 833836, Richardson, TX 75083-3836, U.S.A., fax 01-972-952-9435.

Abstract

This paper describes the design and implementation of a prototype toolkit that demonstrates Integrated Asset Management (IAM) functionality through an integrated production and forecasting workflow. A graphical modeling environment specially configured for this domain is used to instantiate the asset model. Automatic conversion of legacy data into structured model representations is facilitated through a model synthesis tool. The actual optimization is performed using a commercially available solver. For an oilfield with about 75 wells, the tool requires only a few seconds to read the model information and produce the forecast. The time required to generate the forecast output in the desired format depends on the duration of forecasting, the size of the field, and whether the output is to be produced as a text file or a Microsoft Excel spreadsheet.

1. Introduction

The push towards “digital oilfields” has highlighted the need for efficient decision support systems that enable the integration of myriad software tools for modeling, simulation, and prediction of reservoir performance. Integrated asset management (IAM) frameworks promise to enable systematic management of oil field assets in order to facilitate high-level optimization and decision support.

IAM presents an intensive operational environment involving continuous series of decisions based on multiple criteria including safety, environmental policy, component reliability,

efficient capital, operating expenditures, and revenue. Asset management decisions require interactions among multiple domain experts, each capable of running detailed technical analysis on highly specialized and often compute-intensive applications. Technical analysis executed in parallel domains over extended periods can result in divergence of assumptions regarding boundary conditions between domains. A good example of this is pre-development facilities design while reservoir modeling and performance forecasting evaluations progress. Alternatively, many established proxy (or reduced form engineering) models are incorporated to meet demands of rapid decision making in an operational environment or when data is limited or unavailable.

The delivery of enriched information from technical analysis into real time operational domains is another challenge addressed by IAM. An IAM system should ensure proper coordination between data collection sources and data processing destinations. Ultimately, meeting these conditions increases the demand for rapid delivery of relevant data to applications at the desired frequency and/or density, and synchronized in time over multiple sources. Large volumes of data from multiple sources result from progressively improving new capabilities for well measurement, seismic data acquisition, and continuous data collection.

We advocate a model-based approach for designing IAM frameworks, based on a modeling paradigm that is representative of the generic oilfield domain. Some of the key objectives of our design approach are:

- **Generic and reusable architecture:** The IAM framework should be configurable for the needs of different types of asset and for a variety of workflows without the need for extensive redesign or reimplementations. For instance, moving from a small to a big asset, or moving from an onshore to an offshore asset should require only minimal changes that are performed through a well-defined procedure.
- **Single version of the “truth”:** Information about the asset is stored in a structured, canonical form in a common

model database that can be accessed by any tool in the framework. The access mechanism is location-transparent. The model database acts as a single version of the truth and avoids the myriad problems that are caused by multiple inconsistent copies of the same dataset distributed across the organization.

- **Single view of information:** The asset model forms the conceptual basis for the end user to navigate through the wide variety of information relevant to that asset. The framework hides the disparity in data formats, distribution of asset data among various databases, etc., and provides a single logic space for retrieving the desired data.
- **Tool integration through loose coupling:** The model database also enables a loosely coupled tool integration architecture because all the tools (simulators, optimizers, real-time data collectors, etc.) now read from and write to this common database. Indirect coupling of disparate applications through the model database leads to a modular and highly extensible framework.
- **Standards-based implementation:** The design and implementation of the framework will be heavily based on open, platform-independent standards and protocols such as XML for data storage, HTTP and SOAP (web services) for remote application access, etc. This allows us to leverage the data schemas, APIs, etc., being defined by standards bodies such as the Petrotechnical Open Standards Consortium (POSC).

We implemented a prototype IAM framework for an integrated forecasting workflow to demonstrate the feasibility and usefulness of a model-based approach for IAM.

Section 2 provides a brief introduction to the integrated forecasting use case and summarizes the main inputs, outputs, and functional requirements from this workflow. In Section 3, we describe the three main components of our software architecture: the graphical modeling front end which is the user interface to the IAM framework, the model database which is the key to enabling rapid tool integration which maintaining scalability and extensibility, and the software tools themselves. Section 4 describes the end-user experience of using our prototype framework and walks the reader through a typical workflow for this use case. We conclude in Section 5 with a discussion of the strengths and weaknesses of this prototype and future work.

2. The Integrated Forecasting Workflow

Our prototype framework is designed to demonstrate an integrated production forecasting and optimization workflow as a proof of concept implementation of IAM functionality. 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 pertaining to the number, names, and properties of reservoir volume elements, location and capabilities of wells, capability of surface facilities for gas compression, water handling, separator system, etc. The data set also includes fine- or coarse-grained characterization of reservoir behavior in terms of, say, fractional recovery curves for oil, gas, and water, etc. Controls that include production targets, well or block events of significance, etc are passed to an optimization core. The main (default) objective function is to maximize oil production. Secondary objectives at the well, block, or reservoir level could also be specified depending on the particular workflow requirements and the capabilities of the optimization engine.

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

For this workflow, our IAM framework 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. The architecture of the framework is also required to fulfill the key design objectives discussed in the earlier section.

3 Design of the model-based IAM framework

3.1. Overview of the architecture

Our framework is based on the Generic Modeling Environment [1], a graphical toolsuite developed by the Institute for Software Integrated Systems (ISIS), Vanderbilt University. GME provides a visual language for describing the composition rules for models in a particular domain, and then automatically generates a visual modeling environment for that domain. GME supports model-based system design and is a cornerstone of our prototype implementation. We refer to our GME-based forecasting toolkit as **GIFT** (GME-Integrated Forecasting Toolkit).

The GIFT framework has been designed to seamlessly transition in the long term into a completely service-oriented architecture where all components – including the visual modeling front-end – will interact with other components through well-defined service interfaces using open, platform-independent standards and protocols. Implementation of individual modules will be completely hidden from its service requestors. This enables an unprecedented level of reuse of legacy tools and data sources through the use of wrappers that mediate between incoming service requests and the actual tool implementation. Web services and service-oriented architectures are emerging as a widely adopted and useful means of leveraging Internet technologies for improving business processes in the oil and gas industry [2].

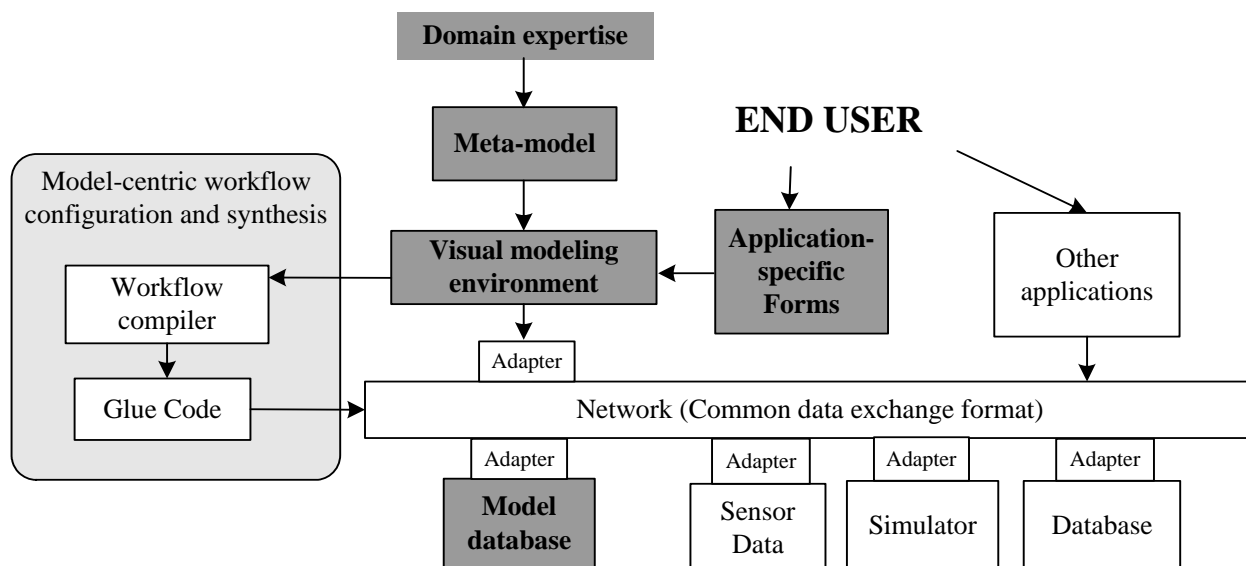


Figure 1: The GIFT framework is designed to be a collection of loosely coupled, integrated tools working on a common, structured model database. The visualization of the model data, definition and synthesis of specific workflows, and the invocation of integrated tools, is all managed through a domain-specific visual modeling environment. The grammar of the visual modeling language is itself specified through a UML-like metamodeling language in the GME toolkit. The metamodel is defined in consultation with domain experts. Shaded areas in this figure indicate the modules that have been prototyped in the current version of GIFT, and described in this paper.

The GIFT installation (shaded portions of Figure 1) consists of three distinct components: (i) the GME front-end which works as the visual modeling environment and the primary user interface, (ii) the model database that stores the information about the asset model and enables interoperability between various tools, and (iii) the forecasting program itself.

The GME front-end consists of the generic GME toolsuite configured for our specific domain – oilfield asset modeling. It provides an easy-to-use visual paradigm for specifying and manipulating information about various entities such as wells, reservoir volume elements, surface facilities, etc. This front-end is also the launching point for different programs for scenario comparison, oil production forecasting, etc.

The model database consists of a set of files that holds the actual information specified by the user through the GME front-end. Note that creating the model in the GME toolkit does not automatically add or update information in the model database. In order to update the model database, the user has to explicitly *commit* the changes to the model database. The process of committing changes from the modeling front-end to the database is termed as “exporting” the model. The model database can (and typically will) also be modified by the integrated tools, and the process of “importing” an updated model into the modeling environment could also be required.

The forecasting program can be automatically invoked for a particular model through GME. Although the program is launched by a GME add-on, it is a standalone application that reads the required input data from the on-disk model database. The application is not coupled with the GME environment.

The forecasting code also allows the user to inspect the model data through a set of custom-designed MS Windows Forms and also allows manipulation of data through the forms interface. The forms provide the end user with a different summarization (visualization) of the same input data that is specified through the modeling environment. Similar to the process of exporting a GME model to disk, any changes made to the model via the Forms interface offered by the forecasting tool have to be “imported” back into the modeling environment.

3.2. Asset modeling and scenario modeling

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,

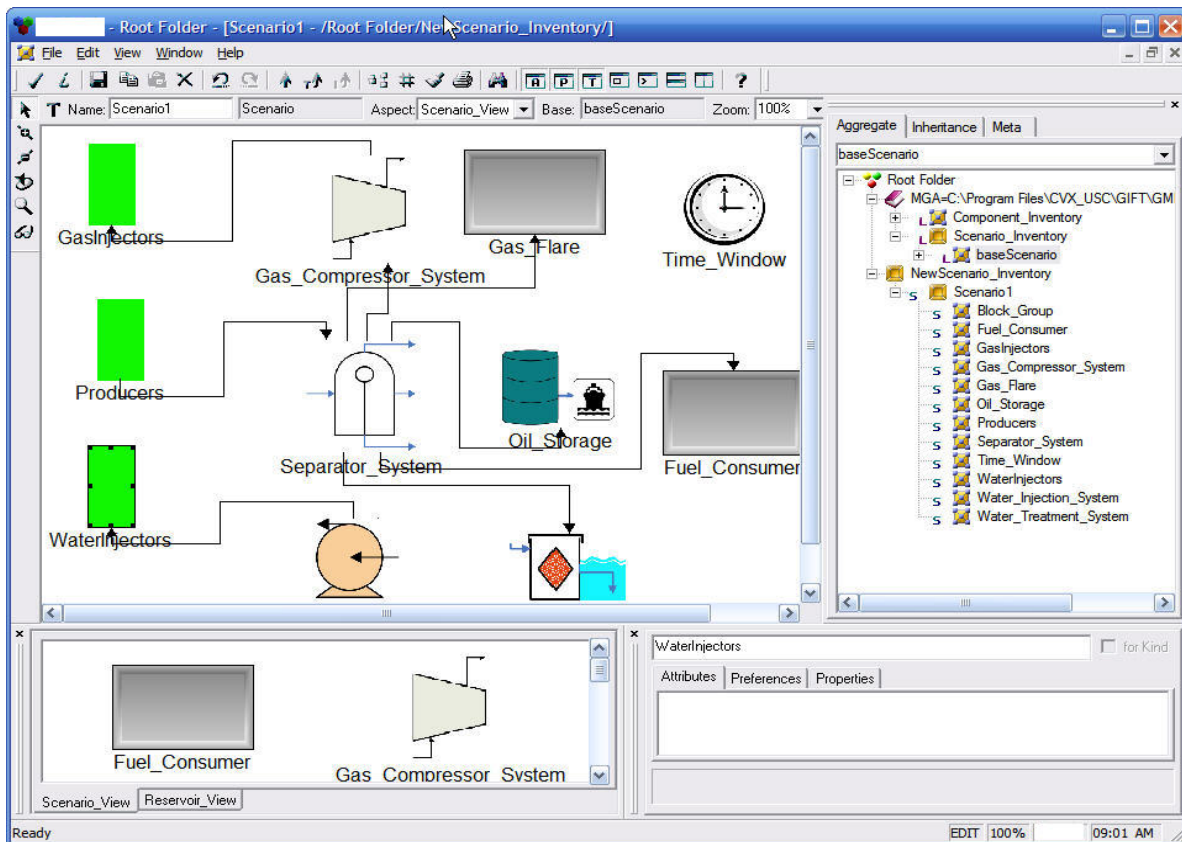


Figure 2: The graphical interface to the forecasting and optimization module. Software components called “model interpreters” automatically configure and launch this tool from the GME environment.

reliability models, among others. The details of how each component is modeled in this language can be found in [5] and are omitted here. Instead, we now focus on two concepts – *inventory* and *scenarios* – that are central to the rapid specification and execution of this workflow.

With inventory and scenario concepts, we have separated the concerns of asset modeling from scenario definition and analysis. An inventory is essentially a library of building blocks that are used to compose a particular scenario. The purpose of creating an inventory of such “immutable” model elements and attributes is to be able to define these elements only once and include them by reference in each scenario. Once the components of an asset are described or modeled within the inventory, the end-user can focus on the analysis of different scenarios for the asset.

Inclusion of a component by reference to the inventory entity is extremely valuable because any change made to some component of the model inventory can be instantly reflected in each scenario that contains that component. Suppose a given asset has 5 reservoir volume elements (blocks), and 20 wells in each block. Now, each scenario might assign a different functionality to a different subset of the wells – e.g., a well which is configured for water injection in one scenario might be modeled as a producer in another. Also, one scenario might model only four of the five blocks for forecasting purposes, and another might model all five. Regardless of how the

individual scenario is configured, some basic properties of the asset such as the location and name of each well, the well-to-block association, the fluid region properties of each block, etc., remain fundamentally unchanged.

The other alternative that does not include an inventory model is to offer a template of possible model elements to the end user and require him/her to manually construct each scenario by instantiating the desired number and relationship of model elements, setting the attributes of each element to reflect the reality of the asset, and then running the forecasting toolkit on the scenario thus configured.

There are two main advantages of the former (with-inventory) approach over the latter (without-inventory). The first is the *cost of scenario definition*. If each scenario has to be constructed from scratch, the number of scenarios that can be defined and analyzed in a given time becomes significantly reduced compared to the former approach where a bulk of the definition already exists in the inventory. The second advantage is the *cost of change propagation*. Suppose that a hundred scenarios are defined for each approach, and each scenario includes a well, say W1, with a production capacity of 1000 m³/day. Also assume that this number plays an important role in influencing the output of production forecasting. Sometime after these hundred scenarios are defined, the capacity of W1 changes to 1500 m³/day. All scenarios have now to be rerun and more important, the

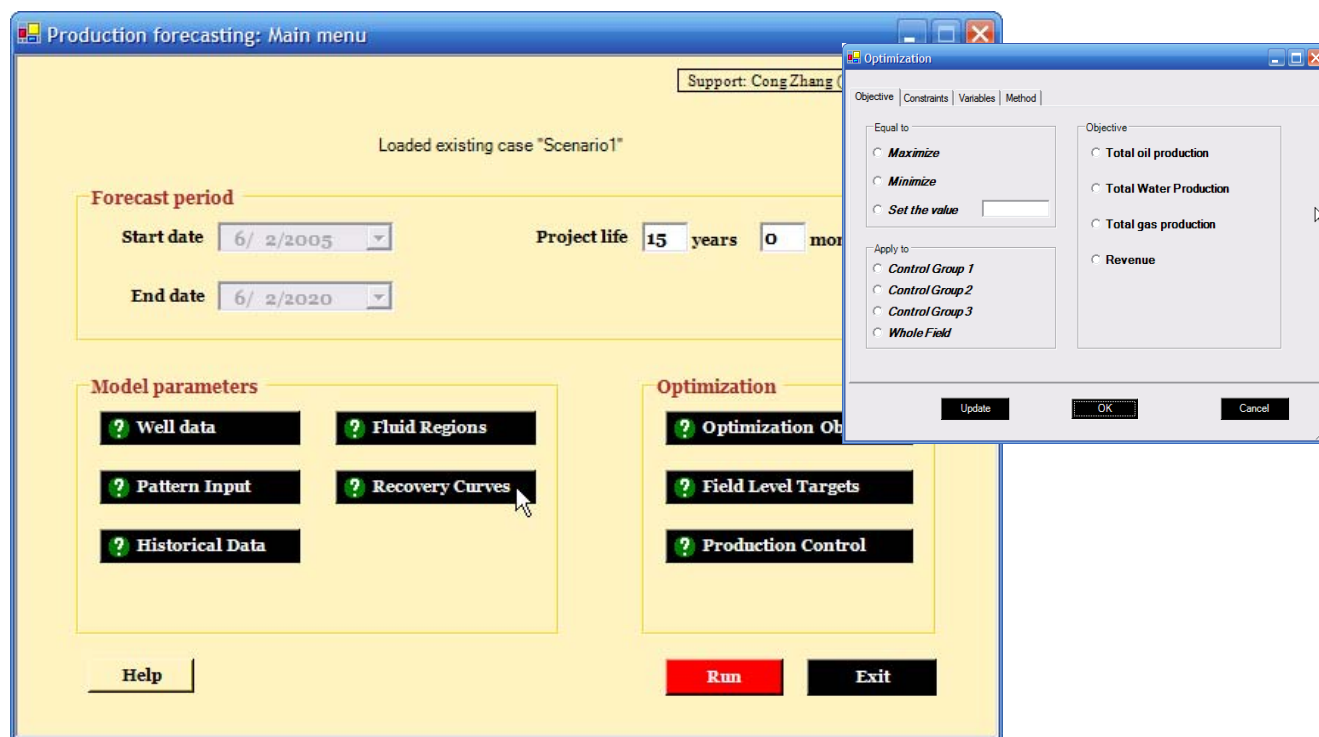


Figure 3: The graphical interface to the forecasting and optimization module. Software components called “model interpreters” automatically configure and launch this tool from the GME environment.

capacity attribute of W1 has to be updated in each scenario. An inventory based approach requires a single update to the W1 entity in the inventory, and this change is implicitly reflected in each scenario that contains a pointer (reference) to W1. In the other approach, the W1 entity in each of the hundred scenarios has to be updated manually.

3.3. The model database

The GME front end provides a graphical user interface that is used to instantiate, inspect, and modify inventory and scenario models. GME 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 store the model data outside the GME environment as a set of XML-formatted text files. This set of files forms the **model database**.

The tools that are integrated into our framework now read and write directly from this model database. Allowing multiple tools to work on the same model in a coordinated manner eliminates the need to write adapters for tools to communicate directly with each other. Also, the GME modeling environment itself now becomes yet another tool in the framework, at the same level as the forecasting tool. If a different model visualization and manipulation interface is implemented, it can be plugged into the framework to replace or complement GME without requiring any modification to the existing infrastructure.

Our prototype IAM framework uses XML as the data storage format for the model database. There are two reasons for adopting XML format. First, XML is a structured format and is readable by humans. Unlike with a relational database, the end user can easily manipulate the data stored in the XML files by simply opening the file in a text editor. Second, our long term vision is for the framework to be compatible with XML-based data storage and transfer standards such as the ones being defined by the Petrotechnical Open Standards Consortium (POSC) [3].

Eventually, the model database will be abstracted through a web-service interface that returns the desired data set to the requesting tool. Whether the actual data is stored as unstructured text files, structured XML files, a relational database such as Oracle or SQL Server, or even a combination of the above, will be hidden from the requester. In fact, data could even be stored across multiple databases and aggregated on-demand when the request arrives at the service interface. By cleanly separating the model database from the modeling environment and from the integrated tools, our architecture eases such a transition.

4. The User Experience

The use of GIFT for integrated forecasting can be divided into the following broad phases: asset modeling, scenario modeling, forecasting, and inspection of the output for possible scenario refinement and/or decision making.

The asset modeling phase involves instantiation of model elements that represent the structure and properties of the asset. For small assets – with, say, tens of wells – this model

The user experience is driven almost entirely through a point-and-click graphical interface that guides the user through the workflow. The details of how and where the model data is

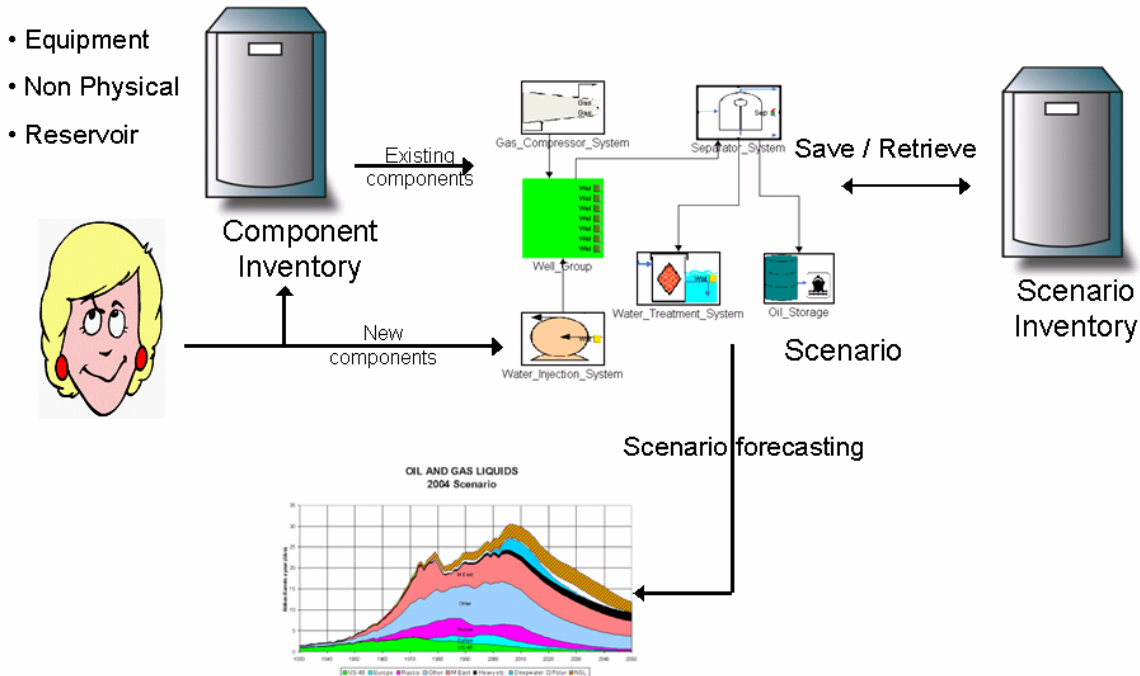


Figure 4: A typical integrated forecasting workflow using the GIFT framework. The end user creates a scenario model using existing components from the inventory. Additional components can also be instantiated directly in the scenario model if desired. The scenarios are stored in the model database through the export/import mechanism. Outputs from the forecasting tool for a particular scenario can be visualized as text files or as Excel spreadsheet that can be used for graphing.

instantiation can be performed manually, although the effort involved is not insignificant. For larger assets where manual modeling is not realistic or desirable, GIFT provides an automatic model synthesis mechanism that reads legacy data (in a specific format) and automatically creates the suitable entities in the modeling environment. Automating this routine work of model entry frees up valuable time of the end user that can be now spent on scenario planning and analysis.

Once the inventory model is finished, scenarios are defined. Next, the scenarios are committed to disk, and the forecasting toolkit is launched. The user interface of the forecasting toolkit is shown in Figure 3.

When the forecasting is finished, the user has the option of selecting the format of the output. Currently, the tool can output its results as unstructured ASCII text or as a Microsoft Excel spreadsheet. Note that the GIFT framework itself does not know or care about the output details which are entirely within the control of the tool developer. In future, the asset model will contain model elements that will capture not just a scenario configuration but also contain points to the forecasting output. In this enhanced model, the forecasting tool will feed the output back into the model database, where it can be accessed by any other component of the IAM framework.

stored and how the integrated tools are configured and invoked are completely hidden from the user. This encapsulation allows us to provide the end user with a consistent user experience while the implementation of the framework undergoes possibly radical changes as part of its evolution and adaptation to the needs of a particular asset and a particular workflow.

5. Discussion

5.1. Revisiting the design objectives

The key design considerations for an IAM framework, as outlined in Section 1 are: generic and reusable architecture, single version of truth, single view of information, tool integration through loose coupling, and a standards-based implementation. We now briefly discuss how our prototype framework accomplishes these objectives in the context of the integrated forecasting workflow.

Generic: The GIFT asset modeling language is expressive enough to allow for the representation of a variety of assets using the same modeling “paradigm”. Tools for importing or exporting models from and to the model database, and even

the forecasting tool, can be used unmodified for any other asset represented in this paradigm.

Reusable: The forecasting tool is an independently developed executable that is fully capable of running outside the context of the GIFT framework. The only configuration information that is passed to this standalone software application is the name of the particular scenario that is to be forecast. The retrieval of the scenario data from the model database is performed independently by the tool based on the scenario name, without the mediation of the modeling environment. Any number of forecasting engines or other applications can be plugged into the GIFT framework without the need for substantial modification.

Single version of truth: A common copy of the model database shared among the GIFT components ensures that updates made by one component are immediately visible to throughout the framework. Currently, an explicit export/import step is required to enforce consistency but the eventual goal is to implement a publish/subscribe event-triggered mechanism that can be used by components to register their interest in a particular subset of the model data and provide callback functions that are automatically invoked when some change occurs in the data set.

Single view of information: The structure of the model database is hidden from the user of GIFT. The generic oilfield asset modeling language provides the conceptual framework to define a model and navigate through an existing model. Reorganization of the database has no effect on this conceptual framework, assuming, of course, that information is not lost.

Tool integration through loose coupling: This prototype workflow required the integration of only one tool and does not really demonstrate integration of multiple tools. The technique for integrating the sole forecasting tool does however illustrate the idea of loose coupling. As mentioned above, the forecasting tool is a standalone, independent software application and does not require any GIFT-specific customization that renders it inoperable outside the framework context.

Standards-based implementation: The model database uses the XML standard to store the data sets. Data in the XML format can be internally translated into different data structures and formats depending on the individual tool. However, data transfer between tools and from an integrated tool to the model database will require XML-formatted data. In future work, we plan to define web-service interfaces for the model data and for the integrated tools to ensure platform-independent, universal interoperability between disparate applications.

5.2. Future directions

Integrated asset management poses significant challenges from the modeling perspective as well as the software architecture and deployment perspective. Some of the topics

of future work for our model-based IAM framework are as follows.

Scenario management. GIFT provides the user with a template to create a forecasting scenario, and a mechanism to automatically configure and invoke the forecasting tool for a particular scenario. In future work, we plan to support more sophisticated scenario management through mechanisms such as versioning, audit trails, checkpointing and rollback, etc.

Uncertainty modeling. The modeling language currently allows for single and discrete values for entity attributes. For instance, the reliability of a gas compression system has to be specified as a precise fraction between zero and one. In future work, the modeling language will be extended to capture uncertainty about such information. An associated challenge is to propagate such uncertainties into all the scenarios where such “uncertain” entities are instantiated, and allowing the user to analyze the design space that is now represented by each scenario.

Finally, we are in the process of implementing a service-oriented architecture to automatically orchestrate model-centric data composition workflows in the GIFT framework by extending it with additional data and workflow modeling paradigms. The current status of that work is summarized in [4].

Acknowledgments

This research was partly funded by CiSoft – a joint USC-Chevron Center of Excellence for Research and Academic Training on Interactive Smart Oilfield Technologies.

References

- [1] Generic Modeling Environment (GME), <http://www.isis.vanderbilt.edu/projects/gme/>
- [2] R. Gregovic, R. Foreman, D. Forrester, and J. Carroll. A common approach to accessing real-time operations data - Introducing service-oriented architecture to E&P, SPE ATCE 2005.
- [3] Petrotechnical Open Standards Consortium, <http://www.posc.org>
- [4] Cong Zhang, Abdollah Orangi, Amol Bakshi, Will Da Sie, and Viktor K. Prasanna. A service-oriented data composition architecture for integrated asset management, SPE Intelligent Energy Conference and Exhibition (IECE), April 2006.
- [5] Cong Zhang, Viktor Prasanna, Abdollah Orangi, Will Da Sie, Aditya Kwatra, Modeling methodology for application development in petroleum industry, IEEE International Conference on Information Reuse and Integration, Las Vegas, 2005.