

A Semantic Framework for Integrated Asset Management in Smart Oilfields

Ramakrishna Soma
Department of Computer Science
University Of Southern California
Los Angeles, USA
rsoma@usc.edu

Amol Bakshi, Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University Of Southern California
Los Angeles, USA
{amol, prasanna}@usc.edu

Abstract

Integrated Asset Management (IAM) is the vision of IT-enabled transformation of oilfield operations where information integration from a variety of tools for reservoir modeling, simulation, and performance prediction will lead to rapid decision making for continuous production optimization. This paper describes the design of a model-based IAM system for production forecasting. Domain knowledge is captured through a formal modeling language that forms the basis for an intuitive user interface to the system. An IAM metacatalog captures domain knowledge as well as metadata about computational resources and data sets in a single ontological framework, thereby providing a unified mechanism for application, data, and workflow integration. The framework is designed to be portable across oilfield assets, to allow different classes of end users to interact with the integrated system, and to accommodate new domain knowledge, software applications, data sets, and workflows for IAM.

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. The computational challenges of oilfield management - especially reservoir simulation - have been the subject of prior work in the grid and cluster computing community [12, 10, 11]. Such work has focused on automating the creation and execution of a large number of computation-intensive and data-intensive reservoir simulations and the subsequent collecting and processing of the simulation results. The scope of integrated asset management is much broader than reservoir simulation. IAM systems are expected to add value to the oilfield operation by providing on-demand access to information from a wide variety of sources, automate time-

consuming, repetitive tasks, enable what-if scenario analysis and design space exploration, and facilitate collaboration between groups and between applications for “whole field optimization”. The IAM system will essentially provide a layer of workflow orchestration and knowledge management. Computational applications and data sources will be abstracted as services, thereby allowing integration of heterogeneous functionality, including grid-enabled subsystems.

The work described in this paper is part of the Integrated Asset Management (IAM) project at the Chevron-funded Center for Interactive Smart Oilfield Technologies at the University of Southern California, Los Angeles [2]. The current focus of the IAM project is on enabling model-driven reservoir management. In model driven reservoir management, the reservoir engineer relies on simulations (and hence simulation models) to make key operational decisions pertaining to the reservoir on a day-to-day basis. Some of the problems that arise in this setting are

Simulation model management: A typical asset could have hundreds or thousands of alternate models for its reservoir(s), each representing a variation on the current understanding of the subsurface reality. The uncertainty in estimating the reservoir properties is manifested in the form of a large catalog of alternate model realizations, each with a different set of values of key uncertainty parameters. The IAM framework should provide services to manage these models including search, retrieval, providing audit trails etc.

Simulation model integration and what-if scenario analysis: Multiple simulation models capture different aspects of the oil-field. Decision making at the asset level necessitates that the data in the many models and the results of their simulation be merged in meaningful ways. Moreover one key goal of our Integrated Asset Management work, is to enable the evaluation of *what-if scenarios* where each scenario represents a certain realization of the asset and/or a certain operational strategy adopted. An *evaluation* of the scenario is a forecast of the production of the asset and an *optimization* of a scenario chooses the ideal operational

strategy to maximize oil production. An evaluation or an optimization may need to integrate data from multiple simulations. This process may involve variety of complex scientific workflows [18, 17].

Simulation model updation and refinement: When an oilfield starts producing, a large number of data points have to be continually processed, sometimes in real-time. These data points include pressure, temperature, and flow rate in the different producer and injector wells. Smart wells and smart fields of the future are expected to be heavily instrumented, producing millions of pieces of data that have to be handled rapidly. Recalibrating (or ‘history matching’) a reservoir model such that the simulated behavior matches the observed behavior is a complex problem that is an ongoing subject of research in the petroleum engineering community. When reservoir model parameters are updated, all the associated forecasts (which could be now obsolete) have to be rerun and operation strategies revisited. Also, close interaction between geophysicists, geologists, and reservoir engineers is necessary to maintain the consistency of assumptions that are made by each expert and ensure that the overall reservoir study is coherent. For instance, a reservoir engineer may infer the presence of a sealing fault at a particular location in the reservoir based on well production data. Before the reservoir model is altered to model the fault, there should be a mechanism for the alternation to be consistent with the geological scheme. This process of *continuous model-driven optimization and model refinement* must be supported efficiently by the IAM system.

User focus: It is also important to note that the ultimate purpose of the IAM system is to assist a petroleum engineer (not a computer scientist) in the rapid execution of day to day operations. In fact, our experience has shown that ‘soft’ factors such as the design of *user interfaces, novel visualization tools, guided workflow wizards*, etc., are as important from the end users’ perspective as the underlying technological solutions.

These problems addressed by our framework, touch upon some of the core issues of the “grid problem” i.e. *flexible, secure, co-ordinated resource sharing among dynamic collections of individuals, institutions and resources*, defined in [3]. A very important aspect of our ongoing effort is to provide a SOA or a similar infrastructure like OGSA that will enable us to access the computational resources and applications transparently. However, in this paper, we focus on the problem of providing the user with a unified interface to the underlying data and services to enable definition of what-if scenarios and their evaluations and optimizations. Our framework employs semantically enhanced models and meta-models as the core design principle. The meta-model for our IAM system, and the ontology derived from the metamodel, act as the key component of our system to enable integration, and is also the basis for the intuitive

graphical user interfaces that allow petroleum engineers to interact with our framework. *Our prototype model-based IAM system for production forecasting has been deployed and tested in a producing oilfield asset and the system is being extended to address more challenging use cases.* The rest of the paper is organized as follows: In Section 2 we review some of the related work. Section 3 discusses the basic principles of model-based system design. Section 4 provides a high level overview of the application of these principles for creating an IAM system for a production forecasting use case. In Section 5, we discuss an extension of the relatively simple system of Section 4 and the design of a metacatalog that uses the OWL ontology language to define a common vocabulary and semantics across a distributed IAM system. More aspects of our research – e.g., design of a modeling paradigm for smart oilfields, model-based workflow definition and compilation, application development in a semantically enhanced IAM system, etc. – are discussed in [16, 17].

2. Related work

The convergence of ideas from grid computing, service-oriented architectures, web services, and the semantic web holds great potential for enabling IT solutions to these challenges. The basic infrastructure of a computational or data grid can provide a uniform abstraction layer for resource marshaling and management. This aspect has already been demonstrated in the use of the grid for computation and data-intensive reservoir simulations [11, 12]. Further, the use of web services to provide access to computational resources on the grid is especially attractive for IAM because of the heterogeneity in technical and business applications that must interact in an IAM system.

The areas of work most pertinent to this paper are those which focus on data management on the grid including work on metadata catalogs [15], provenance capture, tracking and management [14] and use of semantically rich models and semantic web technologies in particular to enable the above services [13, 9]. In this work we discuss how our system has evolved to incorporate many key ideas from these works to suit our application needs. Our use of GME and the model based approach is novel, and has allowed us to build a domain model/ontology with a huge degree of active participation from the domain expert. The tailored GME environment also doubles as a tool for definition of what-if scenarios, which acts as a portal for the underlying services in the system.

Although integrated asset management is of great interest to the petroleum engineering community, we believe that our project is the only research effort in the computer science community that is focusing on the challenges of high level workflow orchestration and technical knowledge man-

agement for IAM. There are, however, comparable efforts in other domains that are focused on exploiting distributed computing (including grid computing) and semantic web technologies. For instance, the GEODISE project [5] aims to “bring together and further the technologies of Design Optimisation, CFD, GRID computation, Knowledge Management and Ontology in a demonstration of solutions to a challenging industrial problem” of optimization and design search for engineering.

The grid-based system for product design optimization in [7] aims to improve end user productivity by “automating the process of organizing, maintaining, submitting and monitoring computational workflows comprising hundreds of jobs in a heterogeneous distributed environment.” Similar to our GIFT prototype where the user defines scenarios in a graphical, tool-independent language without worrying about the resultant configuration and execution of the forecasting tool and other applications, the system in [7] also provides a graphical interface for the engineer to define a finite element analysis problem. Once the problem is defined, the optimization process runs asynchronously and autonomously in the system. The user interface to the system in [7] is through an augmented commercial tool that is familiar to the target end user.

The key difference between our effort and the ones mentioned above is that the IAM framework is not targeted for a single problem or a single class of end users. In fact, in the prototype IAM system, different domain specialists contribute information to a common asset inventory, reservoir engineers can define and simulate a variety of operational scenarios, and asset managers can rapidly compare scenarios and view the associated forecasts as an aid to decision making. The common model database enables the sharing of information in a common modeling language that is defined by and hence is easily understandable to the domain experts. The modeling paradigm as well as the ontology can be easily modified to generate new graphical modeling environments and new types of metadata as more data sets, software applications, and workflows are integrated into the system.

Finally, our work is inspired by a wealth of work on model/ontology based information and tool integration [19, 1]. Our work uses a *single ontology* approach to data integration [19]. One important difference between the techniques presented in other work and in our work is that in [19], the author assumes that all the data sources are databases. Our problem is more complex as the information stored in a simulation model (or simulation results) is neither as well structured as in databases nor as easily accessible. In that respect, [1] is closer to our work because they used the approach to integrate embedded system simulators. However, the simulation models, their results, their inter-relationships etc in the petroleum domain tend to be

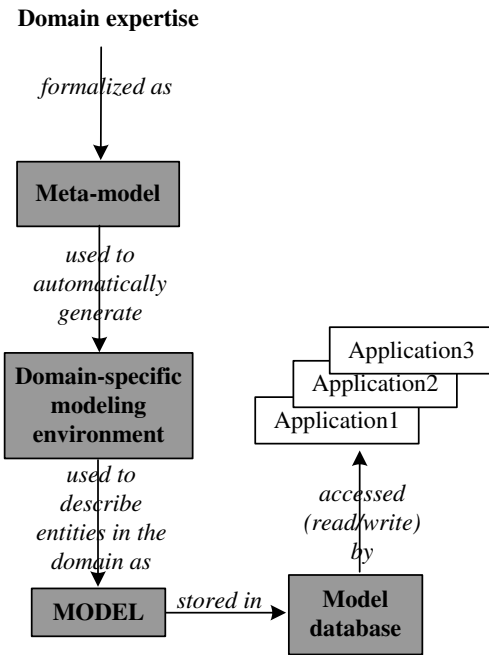


Figure 1. Model-based system design

much more complex and hence makes our problem harder.

3. Model-based System Design

Our IAM system adopts model-based system design as the core design philosophy. Model-based system design (or model-integrated computing) is especially valuable for the design of complex computer-based systems that involve interactions among applications that are not specifically designed to interact with each other. Figure 1 illustrates the general process of model-based system design. As shown in the figure, the first step is to formalize relevant domain knowledge as a modeling paradigm or meta-model. This step typically requires the participation of a domain expert. The meta-model represents the set of conceptual building blocks (entities) and the relationships between the entities. It effectively forms a language for the end user to describe a real-world system using the pre-defined building blocks. The meta-model forms the basis for a domain-specific modeling environment that provides the end user with a (graphical) palette for model definition. The model information is stored in a model database in a canonical form that can be manipulated programmatically through a well-defined API.

For our prototype IAM system, we used the Generic Modeling Environment (GME), which is a configurable graphical tool suite supporting model-based system design [6]. In GME, the configuration of the environment to support domain-specific modeling is done in a formal manner through the use of metamodels. The metamodeling lan-

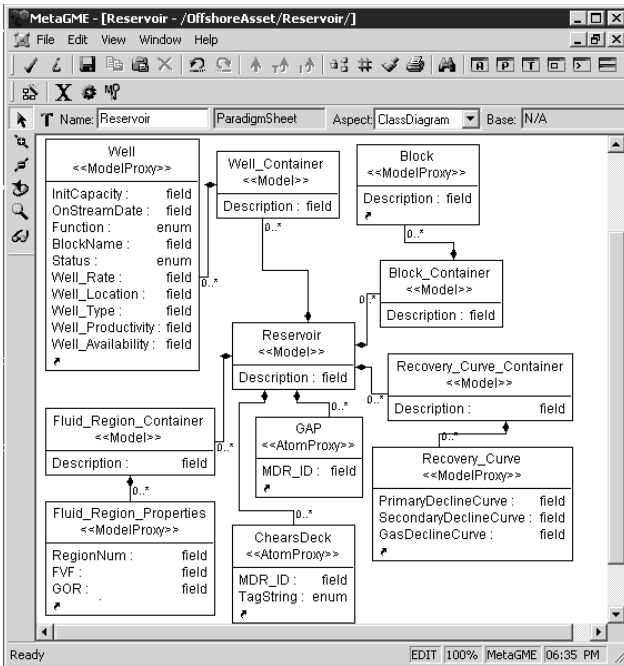


Figure 2. Meta-modeling with GME

guage is the UML class diagram notation. Well-formedness rules that are also part of the metamodels are specified using the Object Constraint Language (OCL). These constraints, along with the syntactical rules of the domain language, are enforced by the automatically generated target environment. When a meta-model is specified in GME, the domain-specific environment for that meta-model is automatically generated. As the meta-model continues to evolve in the early stages of system design, this feature is especially convenient because changes in the modeling language can be instantly reflected in the domain-specific modeling environment.

4. Model-based IAM framework for Integrated Production Forecasting

4.1. Use Case Description

In the integrated production forecasting use case, the end user wished to analyze future production of the particular oilfield asset by configuring various "what-if" scenarios - each corresponding to a different decision point related to well scheduling, change in capacity of processing facilities, etc. The input data set was divided into two main categories: model information, and system and production controls. The model information consisted of data pertaining to the reservoir volume elements, wells, surface facilities. Controls that include production targets were passed to

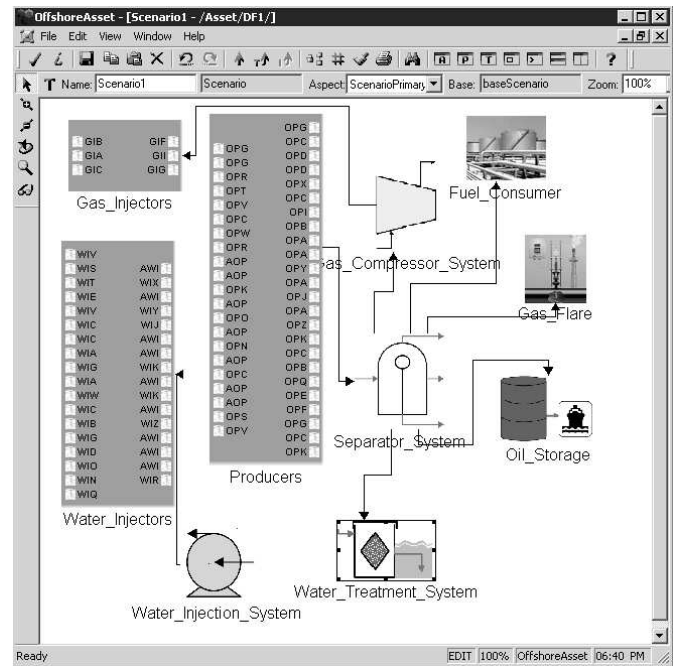


Figure 3. Modeling a forecasting scenario

an optimization core and the default objective function was to maximize cumulative oil production for the whole field. Secondary objectives at the well, block, or reservoir level could also be specified. The output of forecasting was the simulated production data at the desired level of granularity for the whole field.

State-of-the-art implementation for this workflow was a Microsoft Excel file with Visual Basic macros. Despite the significant disadvantages of using spreadsheets for data processing (such as the need to reformat data, proliferation of multiple copies of the same dataset, etc.), the ease of use and powerful graphing capabilities of Excel have contributed to its widespread adoption for data analysis, optimization, and visualization in the oil industry. For this particular use case, the input data was in specific cells and worksheets of the Excel file. A VB macro implemented the optimization and forecasting logic, and the production forecast was output in yet another worksheet in the same Excel file. To create a new forecasting scenario, users typically made multiple copies of the file, used their own file naming convention to record the scenario metadata, and modified the dataset in the copy to suit the new scenario. One of the most significant drawback to this approach was that if an input data value was to be corrected or updated, the updation had to be performed in every single file. Similarly, if an update was required to the forecasting macro, there was no easy way of proliferating the change to all copies of the file. Finally, it was impossible to quickly compare a set of scenarios to understand the key differences between the operational strate-

gies that resulted in the difference in their corresponding production forecasts.

Our prototype IAM system for this use case was tasked with providing scenario management capabilities including efficient propagation of changes to the input data set or the processing logic. Another objective was to automate 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.

4.2. Modeling language and the model database

Using the GME tool suite, we first developed a domain-specific modeling language for describing a generic oilfield asset [?]. The metamodel definition was performed mainly by a reservoir engineer through the GME user interface. 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 oilfield assets. Figures 2 and 3 show a portion of the meta-model and a screenshot of the resultant graphical modeling environment that was automatically created based on the meta-model. In our modeling language, the objects in an oilfield asset model are classified into physical (wells, separators, compressors, etc.) and non-physical components (production controls, field constraints, drilling schedules, etc.). As discussed in Section 5, the modeling language has subsequently been extended to encompass computational services and data sets through the creation of a ‘metacatalog’. A detailed description of the modeling language and the resultant workflow appears in [21]. Here, we highlight the scenario management capabilities enabled by the model-based approach for IAM system design for this use case.

The concept of an *inventory* that is a set of building blocks representing oilfield components, and a *scenario* that represents a particular configuration of these building blocks together accomplish a separation of concerns between asset modeling from scenario definition and analysis. In an inventory, the elements and their attributes need to be defined only once, e.g., an element could represent a particular oil well and its attributes could indicate its onstream date, location, production capacity, etc. The inventory elements are included by reference in the scenario definition. In our graphical interface, the end user merely drags elements from the inventory folder and drops them into the scenario definition palette. Inclusion of an inventory component by reference means that any change in the inventory is instantly and implicitly reflected in each derived scenario. Another significant advantage is the reduction in cost of sce-

nario 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 lesser compared to our approach where a bulk of the definition already exists in the inventory. The inventory also acts as a single version of the ‘truth’ and prevents the errors caused by data duplication and inconsistent modification.

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 open, platform-independent access to the model database, we stored the data in an XML format and wrote software agents that synchronize between the GME environment and the XML storage.

4.3. The User Experience

The use of the GME-integrated Forecasting Toolkit (GIFT) can be divided into multiple phases. The first phase is the creation of the asset inventory, which involves instantiation of the generic classes and relationships defined in the meta-model. For small assets, this one-time inventory building could be done manually through the GME interface. For larger assets, the inventory can be built up programmatically by parsing the information from legacy data and automatically creating the entries in the model database. After the inventory is created, the next step is for the end user to define scenarios and launch the forecasting tool. As mentioned above, all data in the inventory and scenarios is synchronized with an XML store and can be accessed by any software application from outside the GME environment. This loose coupling between the modeling front-end, the model data, and the model consumers results in a highly scalable and extensible design.

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 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. In fact, the enhancement of the framework’s capabilities through the creation of a semantically-enhanced metacatalog (described in the next section) has not at all affected the end users’ experience with this earlier prototype.

5. The IAM Metacatalog: Augmenting Metadata with Domain Knowledge

The biggest strength of GME is that it is a very intuitive and interactive tool for capturing the domain knowledge from experts that have little or no computer science background. GME allowed formal models of the domain to be built almost entirely by reservoir engineers, with minimum involvement of a computer scientist. This has saved us the additional time and effort that would have been needed for the computer scientist to go through the steep learning curve to understand the domain. Apart from this role of meta-model definition, GME serves the dual role of the primary user interface to the IAM system - in our case, for enabling the definition of what-if scenarios and as a launching pad for the forecasting tool.

However, using shared GME models as the basis for model-based integration in a distributed system is not simple. This is because GME models are only accessible through a proprietary API which is tightly coupled to its visual interface. An alternative to representing the meta-models (modeling languages) for the domain is to use semantic web technologies. Semantic web technologies such as ontologies are particularly attractive because they are built on open standards, provide the requisite level of expressiveness to capture the rich relationships in the domain, have precisely defined semantics, and are accessible through more flexible APIs. Therefore, we have created a parallel representation of the GME modeling paradigm in the form of an ontology definition in the OWL language, to enable a more powerful framework for model based integration and orchestration.

In addition to the use of ontologies to enable integration of software applications and orchestration of the corresponding web service wrappers, the use of semantic web technologies can be extended to manage other kinds of information in the IAM system. This includes annotations about the data objects (similar to [15]) and about the operational decisions made in the oil field – forming the basis of an *organizational memory* [20]. This idea is similar in spirit with the work on semantic grid and knowledge grid [13, 22].

We refer to the component that stores and manages such information as the *metacatalog*. The metacatalog is an extension of the traditional metadata catalog [15]. The metadata catalog is typically used to manage the metadata for the data objects in the system while our metacatalog deals with more diverse information. In this paper, we discuss two main components of our metacatalog, viz. the domain catalog and the metadata catalog.

Domain Catalog. The domain catalog is analogous to inventories and formally defines the entities in the asset. The schema for the domain catalog is identical to the GME meta-model-it describes the elements of the oil-field and

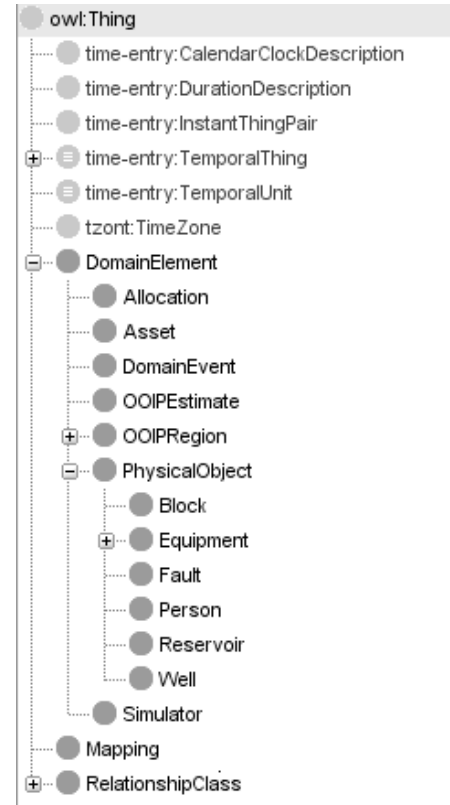


Figure 4. Domain modeling with OWL

their relationships. The inventory is used in the metadata catalog in the next sub-section. The class hierarchy of the domain catalog schema is shown in the figure below. In general, the entities/classes in domain model schema are the entities found in the oil-field reservoir e.g. Wells, Equipment etc. Apart from these physical objects, the domain schema also contains entities denoting objects which are frequently found across different types of simulation cases. For e.g. in the figure OOIPRegion may or may not refer to a physical entity in the oil-field. However, some notion of a region with its own "pool" of oil is common across models and is thus represented as another entity in the schema. Note that the domain catalog schema uses the OWL-Time ontology [8] to capture the time aspect, which is required to describe events occurring in the oil-field.

Metadata Catalog. A common problem in the data grid is the management- tracking, search, and retrieval, of data objects. Prior work [15] has identified a component called the Metadata catalog to manage the data objects in the system. Our system also uses a metadata catalog to store various information about a data object including (a) provenance of the object (creator, date of creation, version history, etc.) (b) metadata required to locate and retrieve the dataset, (c) special annotations that are relevant for the type of entity

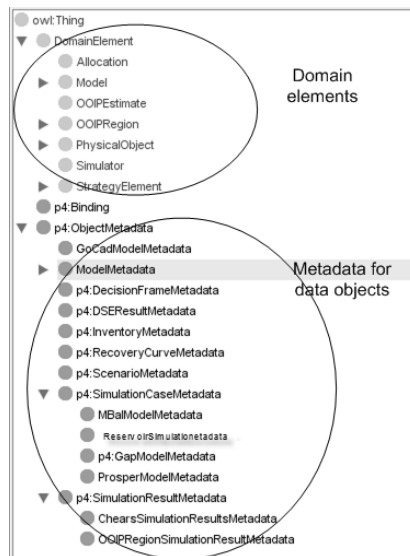


Figure 5. The IAM Metacatalog

represented by the data object, and (d) the relationships between data objects of different types and of the same type. The schema for the metadata catalog is shown in Figure 5. The entities of this schema correspond to various kinds of data objects in the system. Key data objects include the simulation models and the simulation results. The simulation cases and results are typically large data objects that also contain a lot of detailed model information. This makes them ideal candidates for summarization in a metadata catalog. As shown in the figure, the metadata catalog schema builds on the domain catalog schema because the metadata for the contents of the data objects are drawn from this schema. For example, consider a GAP model data object representing the GAP simulation case¹. The GAP model may describe a realization of the oil-field with a specific set of wells. This is captured in the metadata catalog by having a class `GAPModelMetadata`, used to store the metadata for a single GAP simulation model. It has a property called `describesWells` and the range for this property is the `Wells` class of the domain schema.

Different simulation cases model different elements of an oil-field, e.g. reservoir, wells, pipe-networks etc. In general, the different kinds of models are created by different processes and people. Therefore, although the models may refer to the same entity, there may be subtle differences in the names of the entities, assumptions about them etc. This forms a major barrier in obtaining an integrated view of the asset as well as in comparing the information in and the results of different simulation runs (a task that is frequently

¹GAP is a commercial modeling, simulation, and optimization package for surface pipe networks. GAP links with well and reservoir simulators to model entire reservoir and productions systems.

performed). This problem is alleviated by the metadata catalog, because the metadata entries for these data objects all refer to the same instance of the entity in the domain catalog. Thus the domain catalog acts as the "single version of truth" that forms the frame of reference among various data objects in the system.

An important problem that needs to be addressed is the acquisition of the metadata from data objects [22]. This problem is non-trivial because there is no generic method to extract metadata from these models. Some models are represented as ASCII text and metadata can be extracted by simply parsing the files. Others have proprietary or specialized interfaces and special parsers are required to read from the model data. The metadata acquisition is a semi-automated task because, many pieces of the metadata cannot be derived from the files itself and user input is required to fill in such information. The results of the simulation runs are cataloged by creating adapters for the simulator. The adapters accept a simulation case as an input, run the simulation using that case and extract the required metadata from the results.

Currently the metacatalog is being used to enable the following applications:

Search: Metadata based search is used to find the appropriate simulation case. Since hundreds models of the asset may be created, this is an extremely useful service for the reservoir engineer.

Audit trails: The metadata catalog enables audit trails, especially in the case of forecasts, where it is important to know the process used to create the forecast. This includes information such as the person who specified the what-if scenario, which simulation case was used for simulation, which tool was used for simulation.

Tracking information across models: An useful service we have enabled is the ability to track information across different kinds of models in the system. Since the metadata catalog uses the domain catalog, as the single version of the truth, it is possible to make meaningful comparisons between models in the system.

6. Concluding remarks

The Integrated Asset Management project is part of the Chevron-funded Center for Interactive Smart Oilfield Technologies at the University of Southern California. The mission of the IAM effort is to bring fundamental technologies from computer science and engineering to bear upon the very real problems of technical knowledge management that are being faced by the oil industry. The twin challenges of marshaling a distributed set of resources for technical computing, and abstracting the functionality in terms of semantics that are understandable to the petroleum engineer lead to a significant role for grid technologies, especially the

Semantic Grid, in this area. The work described in this paper illustrated the use of fundamental principles of model-based system design to create a semantic framework for integrated asset management. The principles of model-based system design, in the form of Model-Integrated Computing (MIC), widely adopted in the area of software synthesis for complex systems, have been employed in our system. Our work represents an extension of the MIC principles to the area of defining a shared ontology to capture domain knowledge as well as metadata about computational and data resources in a distributed software system. Our first deployed prototype for integrated forecasting did not involve the use of Grid toolkits due to IT policy considerations within the oilfield asset. We are however transitioning to an IAM system that uses the OGSA support in the Globus toolkit [4] for integrating simulators through semantically-enhanced web service interfaces. Eventually, the IAM system is expected to transition to a Service Oriented Architecture with the modeling paradigm and the ontology described in this paper providing the semantic framework for application integration and knowledge management.

7. 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. We are thankful to Will Da Sie (Chevron Corp) for sharing a wealth of domain knowledge, significantly contributing to the definition of the modeling paradigm for the production forecasting use case, and providing feedback on our early prototypes. We also acknowledge the intellectual contributions of Cong Zhang (USC) to the larger IAM architecture.

References

[1] A. Bakshi, V. Mathur, S. Mohanty, V. K. Prasanna, C. S. Raghavendra, M. Singh, A. Agrawal, J. Davis, B. Eames, A. Ledeczi, S. Neema, and G. Nordstrom. MILAN: A model based integrated simulation framework for design of embedded systems. *ACM SIGPLAN Notices*, 36(8):82–87, 2001.

[2] Center for interactive smart oilfield technologies, <http://cisoft.usc.edu/>.

[3] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.

[4] I. T. Foster. Globus toolkit version 4: Software for service-oriented systems. In *NPC*, pages 2–13, 2005.

[5] GEODISE: Grid-enabled optimisation and design search for engineering, <http://www.geodise.org/>.

[6] Generic modeling environment, <http://www.isis.vanderbilt.edu/projects/gme/>.

[7] T. A. Haupt, A. Voruganti, A. Kalyanasundaram, and I. Zhuk. Grid-based system for product design optimization.

In *2nd IEEE International conference on e-science and grid computing*, December 2006.

[8] J. R. Hobbs and F. Pan. An ontology of time for semantic web. *ACM Transactions on Asian Language Processing (TALIP)*, 3(1), 2004.

[9] J. Kim, Y. Gil, and V. Ratnakar. Semantic metadata generation for large scientific workflows. In *International Semantic Web Conference*, pages 357–370, 2006.

[10] J. L. Landa, R. K. Kalia, A. Nakano, K. Nomura, and P. Vashishta. History match and associated forecast uncertainty analysis - practical approaches using cluster computing. In *International Petroleum Technology Conference*, November 2005.

[11] V. Matossian, V. Bhat, M. Parashar, M. Peszynska, M. Sen, P. Stoffa, and M. F. Wheeler. Autonomic oil reservoir optimization on the grid. *Concurrency and Computation: Practice and Experience*, 17(1):1–26, 2005.

[12] K. Nomura, R. K. Kalia, A. Nakano, P. Vashishta, and J. L. Landa. Parallel history matching and associated forecast at the center for interactive smart oilfield technologies. In *International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 369–372, June 2005.

[13] D. D. Roure, N. R. Jennings, and N. R. Shadbolt. The semantic grid: Past, present and future. *Proceedings of the IEEE*, 93(3):669–681, 2005.

[14] Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science, 2005.

[15] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman. A metadata catalog service for data intensive applications. In *ACM/IEEE conference on Supercomputing*, 2003.

[16] R. Soma, A. Bakshi, A. Orangi, V. K. Prasanna, and W. D. Sie. A service-oriented data composition architecture for integrated asset management. In *SPE Intelligent Energy Conference and Exhibition (IECE)*, April 2006.

[17] R. Soma, A. Bakshi, V. K. Prasanna, and W. D. Sie. Model-based framework for developing and deploying data aggregation services. In *4th International Conference on Service Oriented Computing (ICSOC)*, December 2006.

[18] S. Venugopal, R. Buyya, and K. Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1), 2006.

[19] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information — a survey of existing approaches. In H. Stuckenschmidt, editor, *IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117, 2001.

[20] J. P. Walsh and G. R. Ungson. Organizational memory. *Academy of Management Review*, 16(1):57–91, 1991.

[21] 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 (IECE)*, April 2006.

[22] H. Zhuge. *The Knowledge Grid*. World Scientific Publishing Co., Singapore, 2004.