

# An architecture of a workflow system for Integrated Asset Management in the smart oil field domain

Ramakrishna Soma  
Dept. of Computer Science  
University of Southern California  
Los Angeles, CA 90089  
Email: rsoma@usc.edu

Amol Bakshi, Viktor K Prasanna  
Ming Hsieh Dept of EE  
University of Southern California  
Los Angeles, CA 90089  
Email: {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 discusses the similarities and differences of our applications to typical e-Science applications. We then propose an architecture for the system based on the three key goals of the system: support for integrating legacy tools and analysis modules, support for audit trails and data quality indicators for data objects and usability. Our architecture builds upon the rich research in the scientific workflow area and applies many of its learnings to address the unique requirements of our domain. We discuss the implementation strategies and technologies to achieve it and identify some of the key research challenges in realizing our architectural vision.

## I. 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 [25], [23]. 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 application and data integration, workflow orchestration and knowledge management.

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 [6]. The current focus of the IAM project is on enabling *model-driven reservoir management*. In model driven reservoir management, the production

engineer relies on simulations (and hence simulation models) to make key operational decisions pertaining to the reservoir on a day-to-day basis.

The IAM application domain exhibits many of the characteristics observed in many e-Science applications. Some of these characteristics of the IAM application domain are:

**Pervasiveness of legacy tools and datasets:** A typical model-driven reservoir operation setting involves the use of many legacy tools which were not designed to interact with each other. Most important of these tools are the simulators for the reservoirs, surface facilities etc. Typically, decision making involves workflows invoking and processing data from more than one of these tools. Legacy data sets have various data formats including unstructured (ASCII), structured data in XML format and in databases, data only accessible through specialized API invocations etc. We note that the amounts of data we currently work with is much smaller than usually observed in e-Science applications [17]. In such an environment, the most challenging problem to be solved is that of **Integration**. Integration in the IAM system involves both application integration and data integration. Service Oriented Architectures is a style of architecting systems where services are used to provide a simplified, (generally) open standards based access to applications and/or the data they produce, thereby abstracting many of the problems commonly observed in application/component integration [15]. Typically, legacy applications are *wrapped* by service facades- and the creation of these facades is a manual and mostly tedious process. The IAM system must address the integration problem that arises here.

**Multiple organizations and classes of users:** Typical oilfield operations involves multiple classes of users and stake holders, with different specializations and roles across departmental boundaries [33]. Much like in e-Science, each class of stakeholders shares data with the other stakeholders and performs certain analyses on the data products, which is typically not conveyed across the departmental boundaries.

**Complex data objects:** We refer to a logical data objects that contains multiple data sets within it as a complex data object. For e.g. a typical simulation run is usually a single logical object which contains various useful information including production of the reservoir at different timesteps, assumptions

about the reservoir etc. The IAM system needs to support modification of these data objects and as well as extracting and retrieving some of the data embedded within the objects for use in analysis modules. Similar characteristics have been seen and addressed in [5], [24].

**Real time data and operations:** The IAM system needs to support the production engineer to make decisions in real time. Data is constantly produced as the oil-field is operated and needs to be processed and decisions made in a timely manner.

Some of the key requirements for the IAM system are summarized below:

**Support for workflows:** The IAM system must provide support for workflows that process data produced in the system. These workflows can be used to automate repetitive tasks and enable what-if scenario analysis. They typically involve invoking and processing simulation data, their results and real time data. In IAM, we typically observe workflows which involve humans as well as those that do not involve humans. This paper only addresses the workflows which do not have humans elements in them i.e. can be completely automated.

**Support for audit trails, provenance and quality indicators:** Audit trails and reproducibility of results are important when evaluating or revisiting a decision made- it is oftentimes important to understand the reasons, the data used and the context under which the decision was made. Hence, a very desirable feature of the IAM system is to enable audit trails of data objects.

**Usability:** 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.

**Extensibility:** The system must be extensible, in being able to integrate new (legacy) applications, analysis modules and workflows(which use these applications and analysis modules).

We note that most of the characteristics and goals of the IAM system are similar to the requirements of a typical workflow system for the eScience domain. In this paper, we present a system to address the IAM problems that is centered around a workflow system. Our contributions are:

- 1) We present an architecture for a workflow system for the Integrated Asset Modeling which addresses the three key issues outlined above.
- 2) We highlight some of the research and implementation challenges in realising the components of the architecture.

The rest of the paper is organized as follows: We present an overview the architecture of the system in section II, and in the following sections III IV V VI we examine the individual components of the system in more detail and present some of the key research and implementation issues in each of them.

In section VII we review some related work, and finally we provide some conclusions and future work.

## II. OVERVIEW OF THE ARCHITECTURE

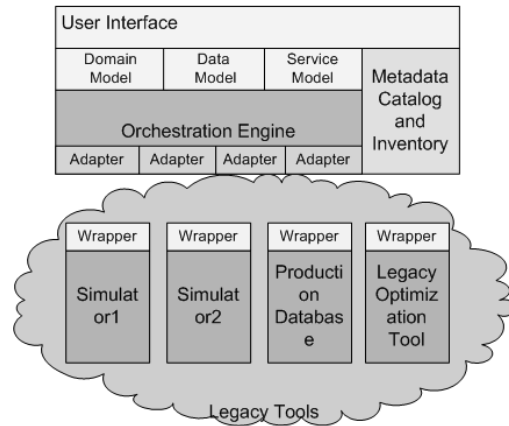


Fig. 1. Architecture of IAM workflow system

Figure 1 shows the architecture of the IAM workflow system. The workflow system consists of four major components: the user interface, the orchestration engine, a metadata catalog and inventory component and the legacy tools and associated wrappers. The *User Interface* for the system provides a very high-level, domain oriented view to the domain engineer to build workflows. The User Interface is built on three formal models (ontologies) of the system, the domain model which captures the entities in the oil-field and their attributes, the data model, which is a catalog of predefined data types in the system and a service model, which is a high level model of the underlying services. The details of each of these models is presented below. The *orchestration engine* is the component that can execute/orchestrate, and manage workflows. A recent study [10] argues that many of the commercially available orchestration engines (MS-Biztalk, Oracle workflow, Windows-WF etc) are suitable for e-Science workflows and is consistent with our current understanding. We have used Windows-WF as the orchestration engine in our architecture. The *Metadata Catalog(MDC)* [13] is a component that indexes the data objects in the system and hence provides an addressing mechanism for the data objects in the system. This alleviates the need to transfer large data objects many times in a workflow. The metadata catalog also holds provenance and quality information of the data objects, which are typically published as the result of workflows. Finally *wrappers* and *adapters* are important components in a integration system like IAM. Wrappers provided standardized data and application interface for legacy programs. Adapters are components that enable components with incompatible I/O's to interact with each other by converting the output of one component to be compatible with the input syntax and semantics of the second component. The components are discussed in greater detail below.

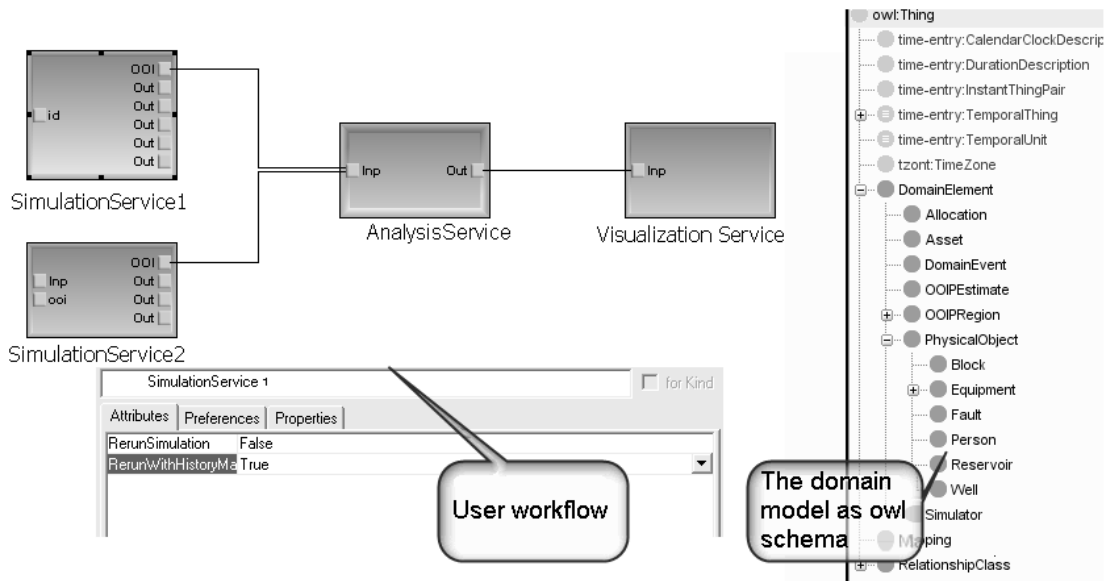


Fig. 2. A workflow modeling environment for the user

### III. USER INTERFACE

The user interface is used by a knowledgeable domain engineer to create workflows. Figure 2 shows the user interface for creating the workflows. The figure shows a simplified workflow that is created by the user, in which two simulations *SimulationService1* and *SimulationService2* are invoked and some part of the results are sent to a analysis service. The output of the analysis service is then sent off to a visualization service. To aid the user to create such services, a list of available services and workflows are made available to the user as a palette from which the user chooses the services that he wants to use in the workflow. The “vocabulary” of elements of this user interface are derived from three formal models: the domain model, the data model and the service model. These are explained in more detail below.

#### A. Domain Model

The domain model is a model that captures the elements of the oilfield asset their attributes and their relationships. [31]. Some elements of domain model are showed in the right hand side of Figure 2. The domain elements include objects such as Wells, blocks<sup>1</sup> etc, as well as domain specific concepts like Original Oil In Place Estimate(OOIPEstimate), which indicates what is the amount of oil contained in a reservoir volume. The goal of using a domain model is to enable the user to create queries and/or service requests which request for data corresponding to the entities in the oil-field. For example, *get me the OOIPEstimate for the Block called "BLOCK007"*. We think that most of the IAM services the domain user wants to use, provide data which is related to the domain element and by capturing and formalizing the domain model, we can create interfaces for new IAM services which use the

same semantic model for querying and requesting for data from the service. We have noticed a similar practice with in industry-wide standards like ProdML [3], where efforts are on to capture the domain model using a standardized vocabulary.

#### B. Data Model

The data model is essentially a set of schemas which provides the user with a model of the data handled by the IAM system. It consists of two parts: *a)*a schema which identifies the legacy data objects in the system along with the metadata associated with them and *b)*a catalog of data types that are used in the IAM workflows and their schemas. The former is modeled using the OWL formalism and forms the basis of the Metadata Catalog, a component explained in detail in Section V. The definition of standard agreed upon schemas, similar to the catalog of data types in our framework, is important for tool integration and has been the focus of committees like ProdML [3]. We have tried to leverage these existing standard schemas wherever possible but we have provisioned for adding newer schemas (data objects) to the system. We have used XML as the standardized way for interchanging data between components. However, most of the legacy tools use data in different formats including plain text files, custom query based interfaces etc. To accomodate these data sources, in our data model it is necessary to also allow non-XML schemas for the data objects.

#### C. Service Model

The service model is a simplified view of the underlying services and workflows in the system, presented to the domain engineer. For example the semantics of a *simulator service* in the service model represents the invocation of simulation service and the data objects that can be extracted from the simulation results. A simulation service might have a flag called a “perform history matching”. This flag switched

<sup>1</sup>Blocks are parts of an oilfield which contain a set of wells

on indicates a “history matched workflow” which maps to a workflow which pulls the current production data from the database, filters the data to remove noise and outliers, formats the data as necessary for consumption to the simulator, runs the simulator and records the provenance and other metadata of the results. Thus, complex workflows that are commonly performed around simulation services are provided in a simplified and intuitive way to the user. A palette of such high-level services are presented to the user which can be tied together into a workflow. These high-level services or *user-level workflows* are translated into *executable workflows* that can be performed by an orchestration engine.

**Implementation status:** We have defined the domain model as part of our previous work. The data model which defines the schema for the metadata catalog has been defined previously using OWL ontologies [31]. An environment for defining data types belonging to the data model as well as a service model will be a part of our future work. In our previous work [37] we have used the GME [1] to define such domain specific environments for the user. GME enables the rapid creation of such specialized domain specific environments (Figure 2 shown above is from a GME based DSE). However, the disadvantage of using GME is that it uses own proprietary format to store the model information, making it hard to integrate it into a larger framework. The process of considering alternatives for the environment as well as building a palette of domain specific services and data objects is a part of our future work.

**Comparison with other systems:** The approach used in Taverna to provide the user an interface which is easy to use for the domain worker is quite similar to ours, but is built for the bio-informatics domain. Taverna does not use a formal model of the domain as a part of its user interface. Teuta [29], a part of the Askalon workflow system, defines a domain specific language using the UML modeling language, for creating grid based workflows. This language extends the UML activity diagram by providing control flow elements and a *GridAction*, which encapsulates units of grid workflow. Our user model is quite unlike Teuta and much more specific to the oil-field domain.

#### IV. ORCHESTRATION ENGINE

The high level user workflows are translated into lower level, *executable* workflows. Various workflow languages/paradigms like BPEL [7], XScufl [26], Windows-WF [27], Ptolemy [21] have been proposed and successfully used in e-Science workflows. We note that all of these workflow languages have quite similar semantics and characteristics with the exception of Kepler/Ptolemy which has the interesting characteristic of being able to employ a variety of data and control flow mechanisms in the framework by the use of different directors. We have chosen the Windows Workflow Foundation (WF) as the workflow orchestration engine for our system but we feel that it can be replaced by a different framework like a BPEL system or a Biz Talk server. Below we

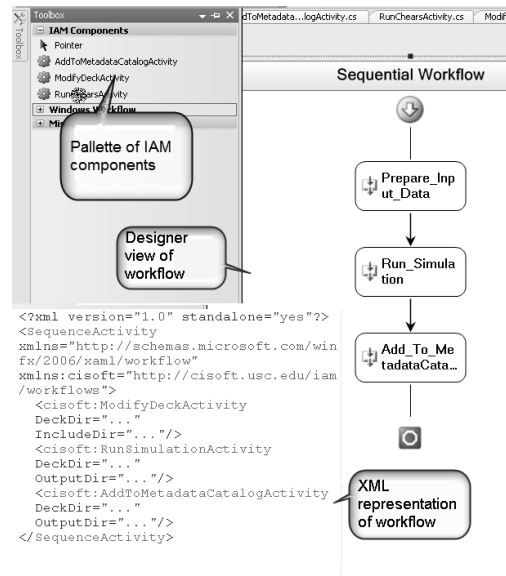


Fig. 3. A simple workflow in IAM that a user can define and the domain model

briefly discuss the WF framework along with the motivations for choosing it.

##### A. Windows Workflow Foundation

The Windows Workflow Foundation is generally described as consisting of three layers: the *workflow model layer*, the *runtime layer* and the *hosting layer*. The workflow model layer provides the ability to use various workflow models including sequential workflows, state-transition based workflows etc. The framework is flexible enough to allow the definition of new workflow models like BPEL. The runtime layer of the framework provides services required to execute the workflows, manage state of the workflow, tracking etc. Finally, the hosting layer provides services to facilitate a host application embedding and invoking the workflow to communicate and coordinate with the workflow itself. It provides interfaces to such core framework functionalities such as persistence, workflow tracking and workflow-host application communications.

Many characteristics of the WF make it an attractive choice to be used as the orchestration engine in our framework. These include the ability to embed workflow engine/runtime in a host applications and to control and monitor the workflow status makes it possible to write advanced workflow management environments which builds on the basic functionality framework. For example, it will be possible to embed a workflow host within the environment used to define the high level user workflows. This will enable the user to track the progress of the execution at both levels of the detail. The ability to deploy the defined as web-services, makes it possible for the workflows to be invoked from any client application. Finally, WF provides a declarative representation of workflows through a XML based representation (XAML). This feature makes it possible to generate workflows corresponding to the high-level workflows descriptions specified by the user.

Figure 3 shows a simple workflow in IAM, which consists of the three steps, of preparing the data, executing the simulation using the data and finally extracting the provenance and other data for storage in the Metadata Catalog. The figure also shows a set of expandable list of IAM activities that are defined to be used in the workflows. Finally the XML representation of the same workflow is also shown in the figure.

One challenge of using the workflow foundation is the translation of the user defined workflows into workflows that can be executed by WF. To simplify the problem, we assume a one-to-one mapping between a workflow element in the user model and a WF workflow- which simplifies the translation problem to mere concatenation of such workflows (in WF, a workflow itself can be invoked like a normal activity<sup>2</sup>, as a part of a larger workflow). However, even a fairly sophisticated strategy should take care of issues such as parallelizing the workflows where possible, fault handling etc. Moreover, although the data produced by one activity of a (WF) workflow is likely to be compatible to the data required by the following activity within a workflow, the problem of incompatible types may be an issue. The possibility of applying shims or adapters dynamically in such scenario is possible and is a challenge. Finally, publishing new workflows to the user (through service models) when executable workflows are created is an interesting challenge.

**Implementation status:** We have investigated and prototyped some typical workflows observed in the domain using this framework. Based on our experience, we feel WF will provide an ideal framework for our target applications.

## V. METADATA CATALOG AND INVENTORY SERVICES

Many systems in the e-Science community have recognized the metadata catalog (MDC) as an important component in their architecture. A MDC contains entries for all the data objects in the system and metadata related to each of these including, attributes on its provenance, quality indicators, access parameters, relationships to other objects etc. One role of the metadata catalog in the workflows is to provide a sort of universal addressing for the data objects in the system. This ensures that rather than passing the data objects themselves, pointers to the data objects can be passed, thereby reducing the amount of data being transferred. Another role of the metadata catalog in this architecture is to also enable the user to describe data objects used in the workflows in more flexible ways. This is enabled by using metadata attributes to *bind* data objects to workflows, as opposed to binding specific data objects. For example, the user can specify that the simulation case to be bound to the workflow is the one which has been *most recently history matched*. The attribute specifying that the case *most recently history matched* is generated as a result of a different workflow which performs history matched simulations (described in section III-C). Thus the data used in the workflow changes dynamically, using the most up-to-date results as they become available.

<sup>2</sup>an *activity* is an abstraction for an unit of work in a workflow

As in [30], our MDC contains three kinds of metadata for each data object. These are, *a)* access information or properties that specify how the object can be accessed (e.g. file name on a shared drive), *b)* provenance information which specifies information about how it was created, and *c)* data type specific metadata which usually contains some summary information of the data contained in the data object. This summary information for simulation models generally includes information such as which element in the oil-field it models, and some key assumptions about them. Such information about the elements of the oil-fields is captured through the domain model and thus, the schema for the MDC uses the domain model.

An important problem that is observed in typical oil-field assets (and indeed in many integration projects) is the problem of aliasing- the same entity in the asset may be addressed by many different names. To address this problem we have introduced the notion of an *inventory*. An inventory is a component which contains the *master data* about the entities in the system and their aliasing information. Thus it becomes possible to interoperate between data from different sources even though they may use different naming schemes by using the inventory service to lookup and resolve aliases. The notion of using an inventory in the system introduces the interesting and non-trivial problems of creating and maintaining it, which is one of the research problems we are currently addressing.

**Implementation status:** The use of OWL as the data schema has enabled us to quickly prototype our MDC and capture the rich relationships between data objects in the system and hence implement rich querying capabilities very quickly. We have used the Sparql querying language [4] and Jena API [2] for interfacing with the OWL data-store. We have implementations that use a file backed data store as well as a database backed (currently MySQL) data-store and we are currently in the process of evaluating the performance, and scalability of these data stores. Prior studies have shown that RDF stores are scalable to fairly large data sizes [16]. We currently envisage a typical MDC to index a few thousand data objects, as opposed to larger data sets typically seen in e-Science literature and thus an OWL based implementation is expected to scale to our application needs. Our implementation of the MDC has a generic web-service interface which will enable a completely different implementation (e.g RDBMS based implementation) to be plugged in with-out losing any of the capabilities of the MDC service. We have built a number of utilities as clients of the MDC which enable the user to search through and utilize the data objects in useful ways.

## VI. WRAPPERS AND ADAPTERS

Wrappers are used to enable the legacy applications and data sets to be accessed in workflows. Most wrappers in our architecture provide a web-service interface for the legacy tools and XML based data for the data-sets. An interesting problem in our system is mechanisms to efficiently define wrappers from complex data objects. XDTM [24] and DF DL [5] are two other frameworks that have been proposed to address this problem.

We find the idea of generating wrappers and extracting data from such data-sets by describing the contents of the data-sets in a declarative manner to be extremely appealing. However, we believe that both these notations are in an extremely early form and a lot of work in terms of validation of the idea, design and implementation guidelines and best-practices for different types of datasets (binary data, ASCII/semi-structured data) and limits of the approach are not well understood and we plan to address these issues in our research.

Adapters are components that convert enable the integration of one legacy system to another. Adapters can be classified as data and protocol adapters [8]. Data adapters perform the task of converting the output data of one workflow component to be suitable to be the input of another workflow component. Two problems are generally addressed during the data adaptation process: syntactic adaptation and semantic adaptation. Syntactic adaptation deals with the conversion of data in one format to data in another format (for e.g data in XML to data in csv format, one data in one xml format to another xml format etc). Semantic adaptation solves a harder problem of converting data represented using one set of assumptions to data using another set of assumptions. For e.g. it is common that production volumes are be encoded using SI units in one data source and as stock tank barrels (stb) in another data source. The use of xml schemas to explicitly describe and record the syntactic format of the data enables the syntactic adaptation of the data. To enable the semantic adaptation of the data, we need means to capture the semantics of the data as well as primitives that can re-code the data to follow a different set of assumptions. Similar problems have been encountered in the data integration literature and the notion of a *context* has been used to describe the assumptions under which the data was represented, a set of commonly occurring semantic heterogeneities have been articulated in [12], [19].

In the scientific workflow literature, only Kepler [22] and taverna [18] address the issue of adapters in the system. A classification of the kind of *shims* found in the Taverna system provides some insight into the kinds of incompatibilities between the components of the workflow. However, we feel that the examination of incompatibilities between tools and services that need to be composed into workflows is an important problem and an interesting area of research and validation for our system. Another interesting problem is how these adapters will be applied transparently in the workflows-especially when third party workflow systems are used. WS-BPEL uses an *assign* operator to converts the xml data of one component (web-service) to be suitable to the input of another. However we feel that an assign operator and use of xslt is a limiting and an insufficient mechanism to bridge the incompatibilities between the services [32]. Apart from a richer mechanism for applying transforms at runtime, another interesting problem to be investigate is the possibility of using a (possibly expandable) set of data transformation primitives at runtime- and hence automatically or semi-automatically choosing the transformation to be applied as data passes form one context to another. Finally an interesting problem is to

create templates for capturing the context of the components and using it along with the data transformation primitives in the adapters.

**Implementation status:** No implementation of this component currently exists and this will be addressed in our future work.

## VII. RELATED WORK

The area of scientific workflows has received much attention in the recent years in the research community and many systems have been proposed [36]. Based on the most important problem that the system is designed to address, we categorize the existing workflow systems as being *computation resource centric* or *data centric* systems. Computational resource centric systems focus on the problem of allocating and managing large computational resources to a given workflow description. Examples of systems in this category include Pegasus [9], Askalon [11], [35]. Data centric systems on the other hand, typically assume that the sources of computation and data services are fixed and address the problems involved with extracting, processing and adapting data from different data services. Examples of these data oriented workflow systems include Taverna [26], Kepler [21] and VDS/Chimera [38]. Note that the Chimera system address both these issues and hence appears in both categories. According to this categorization, our system for the IAM application falls into the data centric category.

The main technical challenge and hence the focus of our work is to facilitate data and application integration. Many of these integration issues are addressed extensively in the Taverna, Kepler and Chimera systems, which we believe are the closest to our system in terms of architecture. Below we compare our system to each of these systems.

The Taverna system, also has a similar goal of providing a simplified interface to the user while still being able to translate it into executable workflows. Our system is similar to Taverna in terms of providing a separation between the high-level/ user centric workflows. Also like in Taverna we profess the use of *shims* or adapters to enable integration of incompatible services. In our we articulate user-centric workflow model, which is sophisticated yet easy for the domain engineer to use, since it maps well to his view of the resources. Some of the key differences between Taverna and our system occurs due to the different domain and hence characteristics of the data-sets and applications. Our system necessarily addresses the need to handle *complex data objects* and the use of wrappers to extract useful data from them. We also note the use of a MDC as a central architectural component in our workflow system to be different from the Taverna approach.

Kepler is a workflow definition and execution system based on the Ptolemy framework. The Ptolemy approach allows different components (actors) to be connected into a workflow, which is co-ordinated by a director. Kepler uses a notion of a *semantic type system* to describe the underlying data and as a means to automatically adapt the data to be suitable for data

and application integration. Our system uses a more traditional approach to executing workflows by the use of an existing off-the-shelf workflow engine based on BPEL like constructs. Prior work has validated the suitability of using such workflow engines for scientific workflows [10], [28]. One of the key drivers for us to use this approach is because we believe that our work will find greater acceptability with our customers if it is based on standard off-the-shelf product rather than a research system.

Chimera/VDS, introduces the notion of virtual data [14] and present a simple executable workflow language, which is executed by the Pegasus/DAGMan workflow engine [9]. VDS addresses the problem of dealing with complex data objects by using the XDTM to create wrappers from descriptions of data in physical data objects. Our approach is similar to Chimera in terms of the use of metadata driven approach to extracting data from complex objects. However, our approach uses the notion of adapters to bridge incompatibilities between services, which is not addressed in VDS. Chimera/VDS also supports specifying workflows at various levels of abstraction, a highlevel Virtual Data Language(VDL) which is translated to an abstract workflow for Pegasus which then translates that into a concrete workflow for DAGMan. Our approach to multilevel modeling of workflows relies heavily on the domain and data models to simplify service definition for the domain experts.

SODIUM [34] is a scientific workflow system that accepts high-level domain specific specification of workflows with enough semantic and qos information and automatically translates them executable workflows, which may include web-services, grid services or P2P services. The semantic information for the abstract workflows is defined based on a domain ontology. Although we do not address the problem of using multiple kinds of services in our framework, the use of WF enables us to quickly create generic activities which wrap other kinds services. The concept of an abstract workflow is quite similar to the high level workflows but rather than defining the service required in a query like form in SODIUM, we provide a pre-configured palette to the user. We think that this approach is simpler to use and still practical for our application because we do not envisage thousands or even hundreds of services available to all the users. Even when many (hundreds of) services are available, the choices presented to the user can be filtered based on the sort of applications that he wants to build. We think that the use of formal ontologies for modeling the domain, data and service elements, enables the ability to build a search system for service discovery similar to SODIUM or GEODISE [20], if needed.

## VIII. DISCUSSION AND CONCLUSIONS

We have highlighted the key characteristics and problems addressed in a IAM system and propose an architecture for the system. The architecture consists of four main components, the user interface component, the workflow orchestration component, the Metadata Catalog component and wrappers and

adapters. The user interface provides a high-level, simplified workflow model to the user which is then mapped into executable workflows. The high-level workflow model is based on a service model, data model and domain model which forms vocabulary for the user to create workflows. The workflow execution component is based on the Windows WF system and is responsible for actually invoking the services that constitute the workflow. Wrappers and adapters are responsible for extracting data from legacy applications and to convert the data and protocol formats of one service to be compatible with the other. The Metadata Catalog provides a shared data id mechanism to alleviate the need to pass data objects multiple times within a workflow. An inventory service is provided to provide aliasing information required for the integration activities. We have highlighted the main ideas, research and implementation challenges and the current implementation status for each of these components of the architecture.

The four key requirements of the IAM system that we identified were *a)* Support for workflows *b)* Capture of provenance, audit trails and data quality indicators for data objects *c)* Extensibility and *d)* Usability. In this paper, we looked at the components to support workflows, including components to capture, execute and monitor workflows. The Metadata Catalog component captures the provenance, data quality indicators for the data objects and information required for audit trails. The user interface layer makes the system *usable* by the domain engineer by providing a high level view of the services and defined workflows while abstracting the lower level implementation details of the workflows including how the metadata of the data objects produced or modified by the workflow are captured, fault handling etc. *Extensibility* is achieved due to the modular architecture of the system. Legacy tools and analysis modules can be added to the system to be consumed in workflows by deploying them as web-services or providing Windows WF-activity wrappers for them.

## ACKNOWLEDGMENT

This research was 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 and providing feedback on our prototypes, thus keeping our research honest. We also acknowledge the intellectual contributions of Cong Zhang (USC) to the larger IAM architecture.

## REFERENCES

- [1] "Gme: Generic modeling environment." [Online]. Available: <http://www.isis.vanderbilt.edu/projects/gme/>
- [2] "Jena- a semantic web framework for java." [Online]. Available: <http://jena.sourceforge.net/>
- [3] "Prodml." [Online]. Available: <http://www.prodml.org/>
- [4] "Sparql, w3c working draft." [Online]. Available: <http://www.w3.org/TR/rdf-sparql-query/>
- [5] M. Beckerle and M. Westhead, "Ggf dfdl primer." *Technical report, Global Grid Forum*, 2004.
- [6] "Center for interactive smart oilfield technologies, <http://cisoft.usc.edu/>"

- [7] F. Curbera, Y. Golland, J. Klein, F. Leymann, D. Rollerand, and S. Weerawarana, "Business process execution language for web services, version 1.1. specification," May 2003.
- [8] J. de Bruijn and etal, "Web service modeling ontology (wsmo)," 2005. [Online]. Available: <http://www.w3.org/Submission/WSMO/>
- [9] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, "Pegasus: a framework for mapping complex scientific workflows onto distributed systems." *Scientific Programming*, vol. 13, no. 3, 2005.
- [10] S. Eswaran, D. D. Vecchio, G. Wasson, and M. Humphrey, "Adapting and evaluating commercial workflow engines for e-science," in *IEEE International Conference on e-Science and Grid Computing*, Dec 4-6, 2006.
- [11] T. F. et al. Truong., "Askalon: a tool set for cluster and grid computing." *Concurrency and Computation: Practice and Experience*, vol. 17, 2005.
- [12] A. Firat, "Information integration using contextual knowledge and ontology merging," Ph.D. dissertation, Sloan School of Management, 2003.
- [13] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
- [14] I. Foster, J. Voekler, M. Wilde, and Y. Zhao, "Chimera: A virtual data system for representing, querying, and automating data derivation," in *14th International Conference on Scientific and Statistical Database Management (SSDBM)*, 2002.
- [15] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural mismatch or why it's hard to build systems out of existing parts," in *17th international conference on Software engineering*, 1995, pp. 179–185.
- [16] Y. Guo, Z. Pan, and J. Hefflin, "Lubm: A benchmark for owl knowledge base systems," *Journal of Web Semantics*, 2005.
- [17] A. J. G. Hey and A. E. Trefethen, *The Data Deluge: An e-Science Perspective*. Wiley and Sons.
- [18] D. Hull, "Description and classification of shims in mygrid." [Online]. Available: <http://www.cs.man.ac.uk/hull/shims.html>
- [19] V. Kashyap and A. Sheth, "Semantic and schematic similarities between database objects: a context-based approach." *The VLDB Journal The International Journal on Very Large Data Bases*, vol. 5, no. 4, 2004.
- [20] C. L., S. N., T. F., and C. Goble, "Engineering grid resources metadata for resource and knowledge sharing." *International Journal of Web Service Research (JWSR) on Bridging Communities: Semantically Augmented Metadata for Services, Grids, and Software Engineering*, 2005.
- [21] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, 2006.
- [22] B. Ludascher, K. Lin, S. Bowers, E. Jaeger-Frank, B. Brodaric, and C. Baru, "Managing scientific data: From data integration to scientific workflows," *GSA Today, Special Issue on Geoinformatics*, 2005.
- [23] 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*, vol. 17, no. 1, pp. 1–26, 2005.
- [24] L. Moreau, Y. Zhao, J. V. I. Foster, and M. Wilde, "Xdtm: Xml dataset typing and mapping for specifying datasets," in *European Grid Conference*, 2005.
- [25] 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*, June 2005, pp. 369–372.
- [26] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, 2006.
- [27] e. a. P. Andrew, *Presenting Windows Workflow Foundation Beta Edition*. SAMS Publishing.
- [28] A. Paventhan, K. Takeda, S. J. Cox, and D. A. Nicole, "Leveraging windows workflow foundation for scientific workflows in wind tunnel applications." in *IEEE Workshop on Workflow and Data Flow for Scientific Applications, (SciFlow)*, 2006.
- [29] S. Pillana, J. Qin, and T. Fahringer, "Teuta: A tool for uml based composition of scientific grid workflows." in *1st Austrian Grid Symposium. Schloss Hagenberg, Austria*. Springer Verlag.
- [30] 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.
- [31] R. Soma, A. Bakshi, and V. Prasanna, "A semantic framework for integrated asset management," in *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, 2007.
- [32] R. Soma, A. Bakshi, V. Prasanna, and W. DaSie, "A model-based framework for developing and deploying data aggregation services," in *4th International Conference on Service-Oriented Computing (ICSOC)*, 2006.
- [33] G. C. Thakur and A. Satter, *Integrated Waterflood Asset Management*. PennWell Books, 1998.
- [34] A. Tsalgaidou, G. Athanasopoulos, M. Pantazoglou, C. Pautasso, T. Heinis, R. Grmo, H. Hoff, A.-J. Berre, M. Glittum, and S. Topouzidou, "Developing scientific workflows from heterogeneous services," *ACM SIGMOD Record*, vol. 35, 2006.
- [35] G. von Laszewski., "Java cog kit workflow concepts for scientific experiments." *Technical Report, Argonne National Laboratory, Argonne, IL, USA.*, 2005.
- [36] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," 2005. [Online]. Available: [citeseer.ist.psu.edu/05taxonomy.html](http://citeseer.ist.psu.edu/05taxonomy.html)
- [37] 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.
- [38] Y. Zhao, J. Dobson, I. Foster, L. Moreau, and M. Wilde, "A notation and system for expressing and executing cleanly typed workflows on messy scientific data." *SIGMOD Record*, Sep 2005.