Using BPM as an Interoperability Platform

Dennis Taylor and Hisham Assal
CDM Technologies, Inc. and CADRC Center, Cal Poly, San Luis Obispo, California
djtaylor@cdmtech.com & hassal@cdmtech.com

Abstract

Data mediation is an essential component in the Modeling and Simulation field (M&S). Managing multiple data sources and exchanging data among multiple systems requires sophisticated tools and a powerful process management system.

Business Process Management (BPM) provides a framework for modeling and managing business activities, both manual and automated, in a consistent manner. Managing automated processes offers an opportunity to integrate external applications into the platform. By integrating automated data transformation tools into the business processes using graphical programming, we provide an approach to achieve operational interoperability among diverse applications without the need for any application to be aware of any other.

1. Introduction

Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged. The diversity of applications and the information on which they operate make interoperability a crucial issue in today's computing environments.

In modeling and simulation (M&S) the variety of data sources and data types and formats represent a challenge in integrating multiple systems or building a system of systems (SoS). Interoperability among systems has many levels, ranging from exchanging bits and bytes to sharing a common understanding of the information and the context, in which the systems interoperate. Tolk and Muguira [1] describe a model for levels of conceptual interoperability, detailing the conditions for the exchange and the type of information being exchanged in every level. Level 0 describes stand alone systems with no interoperability. Level 1 describes technical interoperability, where a communication protocol is established between the systems for the exchange of data at the bits and bytes level. Level 2 describes the syntactic interoperability, where data structures and data format are shared among systems. In level 3, which is called semantic interoperability, a common information exchange reference model exists, which defines meanings for information elements. Level 4, pragmatic interoperability, addresses the context, in which information is used. Systems at this level are aware of each other's context, i.e. how information is used and in what setting. In level 5, dynamic interoperability is achieved, whereby systems can comprehend the changes over time in their group of interoperating systems. In level 6, conceptual interoperability refers to the sharing of a conceptual model, which is built using engineering methods and can be interpreted and evaluated by other engineers.

Our focus in this paper is on the syntactic (level 2) and semantic (level 3) interoperability. Our approach to data exchange combines automated tools, which match data sources of compatible types and builds mappings for data elements, with user-driven mapping tools, which allow the user to connect data sources and provide mappings for them. The intelligent mapping tools can save any user-defined mappings along with information about the data sources, to which the mappings were applied. Any later requirement to connect similar data sources can make use of the stored mappings automatically.

Data mediation functions also include: importing of data from multiple sources, data cleansing and validation, and data source management. These functions are similarly supported by the automated and user-driven tools.

This paper presents an approach for syntactic interoperability using graphical programming within a BPM environment. This approach uses process design to identify the tasks that require data mapping and utilize external data mapping tools to perform the mapping. Graphical programming is used to define and implement data mappings between given tasks.

2. Business Process Management

Business Process Management (BPM) is the activity of defining, executing and monitoring business processes, defined independently of any single application. Business processes may be manual or automated. The complexity of a business process stems from the number of sub-processes that are included and the interactions among them. BPM provides a platform for managing this complexity and offering better visibility into the business process both in terms of its design and monitoring of its execution.

BPM is a good fit for implementing integrated systems, since each task in a business process can potentially be implemented by an individual system, which is offered within the given environment.

Existing BPM suites provide a platform for modeling business processes in a standardized graphical language. These process models are then translated by tools within the BPM suite into executable modules. Control mechanisms within BPM determine the paths through the executable modules and the conditions, under which a module is scheduled for execution, based on user interaction and information provided at execution time. If information is missing or insufficient to run the next module, the control is passed to a user, defined by their role in the process, to add more information or guide the process from this point onward. At each step of execution, going from one executable module to the next, the platform has the opportunity to examine information needs and utilize external tools to provide information for the next module.

The opportunity to use BPM suites as interoperability platforms comes from the structure of process control within these suites and adding data mapping as a standard process between processes that require data exchange. Formalizing the data mapping process and utilizing tools to automate some of the mappings, especially for data sources that have been standardized and whose structure is either known or can be inferred by the mapping tools, allows mapping to take place between processes and as the need arises.

3. Graphical Programming

Graphical programming provides a visual way to represent business data and processes at a business analysis level. It can be used in an execution environment to monitor and facilitate process execution.

Research into what became known as “graphical programming” came into vogue about twenty years ago. This coincided with the advent of relatively cheap semiconductor memory, which in turn made bitmapped displays and their connection to minicomputers possible at places like SRI, Bell Labs, and Xerox PARC. Once researchers had a viable graphical medium at their disposal, they began to dream in earnest of programming by drawing pictures instead of typing. There was widespread belief that a drawing metaphor would bring programming to the masses by making it more natural. Expert programmers too would benefit from a greater expressiveness.

Unfortunately, most efforts at graphical programming amounted to naïve transliterations of conventional programming constructs. Would-be graphical programmers metamorphosed into electricians of another color, laboriously wiring up boxes labeled “function”, “if”, and “+”. Had they the tenacity to create a complete program that way, the resulting mass of lines and boxes was difficult to read, let alone debug and maintain. The expressiveness of text proved hard to beat. [4]

It is important to note that graphical programming as we present it is not intended to model complex algorithms. Instead we look towards areas such as BPM systems where graphical programming for the purpose of modeling business activities has been successful [5].

Graphical Programming in the context of this paper encompasses a number of techniques which involve the modeling of software processes in a user friendly way. Usually such techniques use an actual graph including nodes and edges to model intended system behavior or symbolic information, and then the model can be translated by software tools which can execute the process. Graphical programming can be used in several ways to allow software architects and project managers to see at a high level how work will be orchestrated. An inherent advantage of all graphical programming techniques is that by using a graphical structure they can facilitate a commonly shared vision of the software process being modeled. The interaction between two systems as well as the need for external assistance by programmers can be exposed early in system design so that a statement of work can be clear and concise. This allows interoperability contract negotiation between project managers to take place in an organized and systematic way.

At CDM technologies software architects use graphical programming in several ways including UML object modeling, data mapping, and BPM workflow modeling. Workflow modeling allows the interplay

between a number of systems and human guided interaction to be expressed graphically. Data mapping allows the relationships between two data structures to also be expressed graphically. This paper will focus on just these two techniques: BPM workflow modeling and data mapping.

4. BPM Graphical Modeling

Business Process Modeling consists of creating workflows that resemble real world business processes. A workflow is a graph that resembles a series of steps that need to take place in order to get some unit of work done. The graph typically consists of nodes representing states and edges representing transitions between states. There are many enhancements and variations to this graph originating from both academics and industry.

In the academic world the graph structure has been expanded to include Petri Nets. Petri Nets are a mathematical way of describing distribution systems so they go way beyond simple workflow modeling. In this model the transition is considered a node and the edges between States and Transitions are considered arcs. The entire graph represents data flow. Transition nodes will collect Tokens (or a piece of data) and pass that data to a connected State which will consume the Token and create another Token as output to be stored temporarily in an outwardly connected Transition node. These graphs can be massively parallel allowing many arcs between a Transition and State node. This is why Petri Nets are ideal for modeling the concurrent behaviors in distributed systems [6].

Petri Nets are often thought to be overly complex for modeling simple business processes. So other more simplified models have been created to allow simple workflow design while maintaining a reasonable level of functionality. There are essentially two ways of describing this process modeling from an industrial standpoint. The first being an Abstract business process which only partially specifies a process and can be used to provide visibility but does not provide much more use. For the purpose of this paper we will focus on the other type of process modeling which is executable business processes. These processes are intended to be used directly by a system that monitors and facilitates their execution. Industrial approaches to executable business processes typically focus on creating several node types as well as adding actions and tasks. Actions can be considered something as simple as e-mail someone an update about the workflows progress or as complex as calling some specialized code to do some specific work. A task is typically an indication that workflow is stopped and waiting for user input to continue. In this way these models allow both human and automated work to be coordinated and tracked. Often these graphs can be stored as XML in a standardized language such as the Business Process Execution Language (BPEL).

Unfortunately many industry workflow modeling tools do not focus on data flow. Data passing and exchange is either expected to be added by programmers after the model has been created, or is never really defined explicitly. If the data is neglected in these ways it is harder for project managers to understand the amount of work that will be necessary to enable the system to exchange data. In our system we include the use of data mapping tools that still maintain a level of data awareness.

During the execution of a node data is consumed, created, or modified. Each action or block of algorithmic code is likely to have a data structure that is a particularly "natural" fit for that problem. Some go as far as saying that “you should get your data structures correct first, and the rest of the program will write itself.” Programmers who write difficult algorithms want their data a certain way. Interoperability middle ware should be flexible in the way it deals with data and not enforce certain semantics onto how data is structured. Actions are often represented by tasks executed by legacy systems that are difficult to change and modify for every required use. This makes it seam reasonable that from each actions viewpoint the data consumed would be in its desired form. If this is the case then very little code is necessary to re-use existing actions and legacy systems.

5. System Interoperability using BPM

There are various places in a process definition where data integration can be addressed. A natural place for this is in the transition of a workflow from one node to another. Our approach positions the translation of data directly in the overall workflow as an intermediate process. The reason for this choice is that data translation is not always an automated task and can require human intervention for things like data validation. The integration workflow consists of translation steps as well as data validation steps. By making the data translation and validation phase a first class business process it is our opinion that work will become more systematic and visible to everyone involved.

Some interoperability middle ware advocates the interjection of a mediation or abstraction layer between data sources and data consumers. By programming

each action to exchange data with the mediation layer connecting systems becomes merely a matter of data translation and not so much a concern of data access. These systems will often be outfitted with an Application Router [7]. The Application Router will provide hardware that acts as the single point of access for internal applications as well as exchanging information with business partners. Data translation has also been mitigated in some of these systems by enforcing the use of shared objects commonly referred to as "Business Objects". This enables the easiest form of system integration since exchanging business objects that share the same definition in all systems does not require any translation. Although this represents an idealized integration model it does not maintain the aforementioned problems of legacy systems and flexible data structures. The system we propose does not make any assumptions about the objects that will be exchanged but instead builds a process that will allow developers to quickly identify data translation needs as well as facilitate the translation process.

These approaches have also been standardized. For example, the Web Service Business Process Execution Language (WS-BPEL) is another standard language built on top of BPEL and it allows behavior based Web Service to be included in the process definition. All processes that are defined in WS-BPEL import and export data using Web Service interfaces only. These interfaces are defined using the Web Service Description Language (WSDL). One goal of this language is to "provide data manipulation functions for the simple manipulation of data needed to define process data and control flow" [8].

Our system takes this approach one step further to allow data manipulation or mapping to be performed at the business analysis level. In order to allow this to take place actions must describe what data they consume as input and what data they create as output. This meta-data is then provided to the mapping tasks of our design to facilitate data integration.

6. Graphical Data Mapping

BPM Workflows model the states (nodes) and feasible transitions between them. Graphical data mapping is focused on the data that is shared between various states or nodes in the workflow model.

The translation process is facilitated by a graphical programming technique to assist in Data Mapping. In the simplest case there exists a data translation from one object to another in a one to one fashion. An example of this case can be seen when exchanging the object Person(name, age) to the object Client(username, years_of_age). In this case the user can create an edge in the graph linking name and username as well as age and years_of_age since they refer to the same data. Programs have existed for some time that assists in Data Mapping in a graphical way, one such example is MapForce. This program works quite well for translation tasks that are considered in isolation. There are quite a few systems which allow icons (Nodes) representing data transforms to be placed on a canvas and connect them to data fields, to create a translation graph [9].

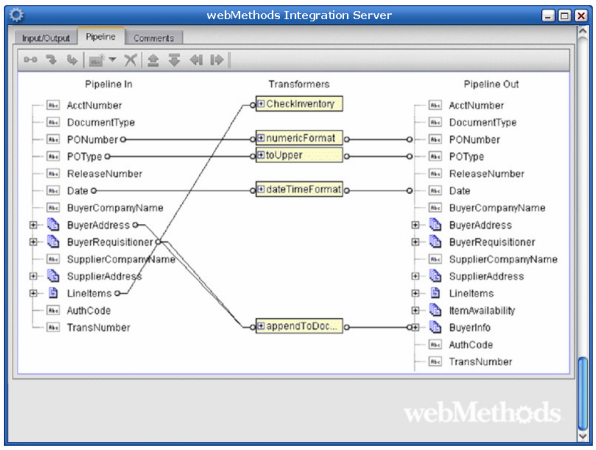

Figure 1. WebMethods Integration Server, using a graphical mapping tool. Source: http://www1.webmethods.com/images/products/screenshots/wm_ESP_BI_2.jpg

Another example is WebMethods Flow [Figure 1] which is intended to be used in a Service Oriented Architecture and has a number of related features of our intended System.

An important feature, which is not provided by the webMethods approach is computer assisted intelligent data mapping.

7. Intelligent Data Mapping Assistance

CDM technologies developed a system called the Intelligent Mapping Toolkit (IMT) [10], which has had success in providing assistance to data mappers at large organizations such as the US Transcom. This assistance comes in the form of suggested mappings between two data sources. This technology has identified several ways to provide suggested mapping that go beyond simple one to one name based mapping.

Most schema matching systems perform 1:1, linguistic, elemental, and structural schema matching; some use auxiliary resources [12]. Some apply information retrieval and machine learning techniques

(e.g., SemInt uses neural networks to cluster attributes and find likely mappings [13]). At the core, all matching methods must contend with syntactic and semantic variations of the schemata vocabulary. Common syntactic variations include abbreviations (e.g., Arpt vs. Airport) and conventions (e.g., AirportCode, vs. Airport_Code). Semantic variations include the use of *synonyms* (e.g., code vs. id), *hypernyms* and *hyponyms* (e.g., vessel vs. ship), *meronyms* (e.g., first and last name vs. name), and *omonyms* (stud [part] vs. stud [horse]). Syntactic variations can be addressed by exploiting methods for assessing string similarity. These vary from finding exact matches to using edit distances. In contrast, semantic variations cannot be effectively addressed using conventional string matching techniques. Instead, *auxiliary knowledge resources* (e.g., thesauri, linguistic ontologies) must be used. The use, development, and maintenance of knowledge resources with suitable coverage and validity pose challenging issues, which we address in IMT. The large variations in schemata vocabulary motivate the adoption of a multi-pronged approach for matching, which is the approach we take in IMT, where several configurable linguistic and structural matching agents are applied to each pair of schema elements to assess their similarity [10].

The unique approach taken by IMT has provided data mappers with a tool that enables them to do their job much more rapidly. One identified shortfall of this tool is that it requires users to use it in isolation from the context of the needed interoperability much the same way the graphical mapping tools did as well. By combining graphical mapping tools with IMT and business process modeling it is our hope that we will greatly enable system integration at a business analysis level.

8. System Design

We present a system which combines BPM Modeling, and intelligent data mapping to allow system developers, domain experts and business analysts to efficiently collaborate in a shared environment. The diagram in [Figure 2] shows the architecture of the system. At the top level, an existing open source BPM platform is utilized to provide the required process

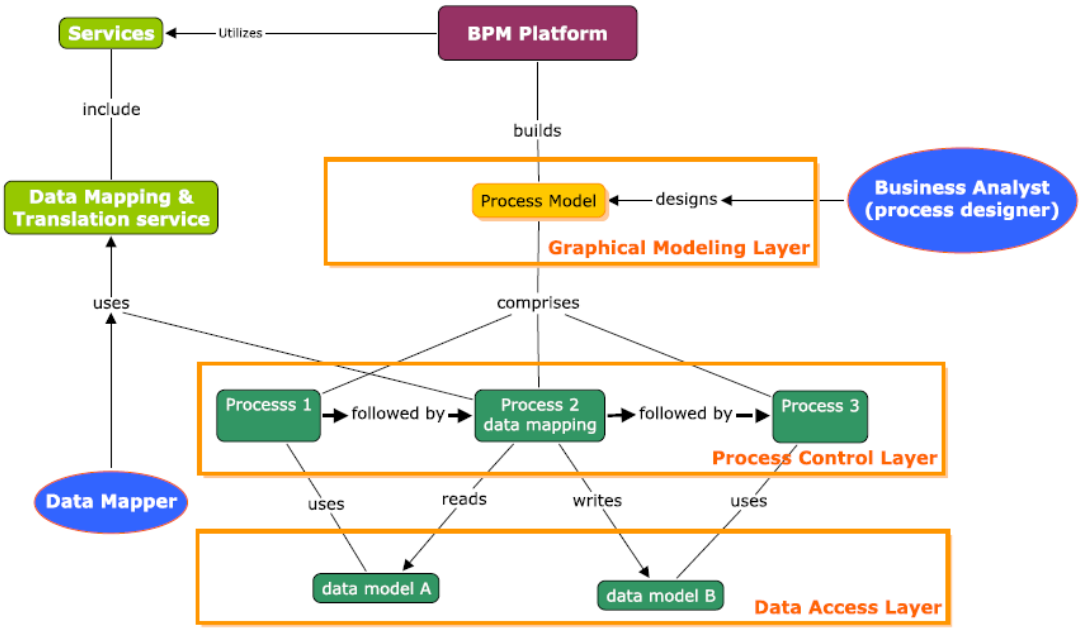

Figure 2: Conceptual design of integrating data mapping with process control

control functionality. At the graphical modeling layer, graphical tools allow the business analyst to model the business processes and submit it to the process control layer. Data mapping and translation services also exist at this level. The graphical mapping tools can be used by a human user to define new mappings. Mapping processes can automatically exploit existing mappings between previously mapped data sources. The IMT is included as part of the translation service, which is used as external service that can be invoked through the process control mechanism within the BPM platform. The data access layer manages the data sources that are used by the various tasks. Data sources may include databases, web services, or plain text files. This layer catalogs the metadata for each data source and holds previously created mappings between any pair of data sources, for future use.

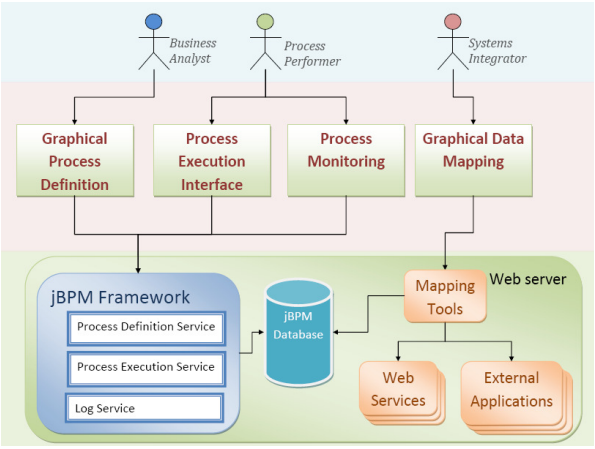

Figure 3. The Integration Platform

The diagram in [Figure 3] shows the types of users of the integration platform, the components used by each user type, and the internal relationships among the components and the BPM platform.

To demonstrate how interoperability is achieved within this framework, consider the following example. Process 1 and Process 3 in [Figure 2] can be currently existing processes. The only requirement to interface with these systems is that there is a description of each interface's inputs and outputs. This can be done using WSDL and data schemes for each object being exchanged. This is typically a programmer's task. These definitions act as the input to the Data Mapping and Translation Services and provide the process modeler with necessary data to allow proper validation to be built into the data mapping stage, the second process in the diagram.

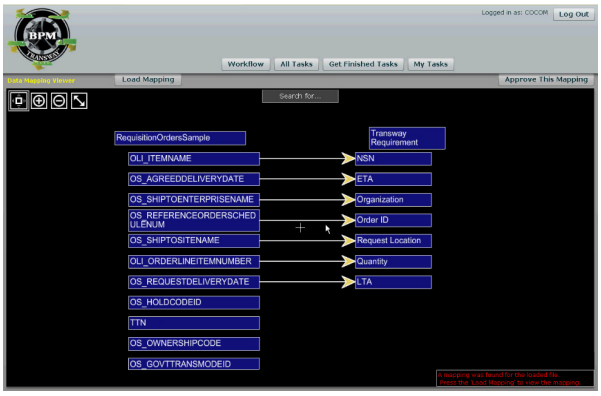

Figure 4. The graphical mapping tool.

Once each process definition's interfaces are properly defined and it is indicated that they should be integrated the system would provide data mappers with the opportunity to examine the data mapping task and provide any human user input that would be necessary.

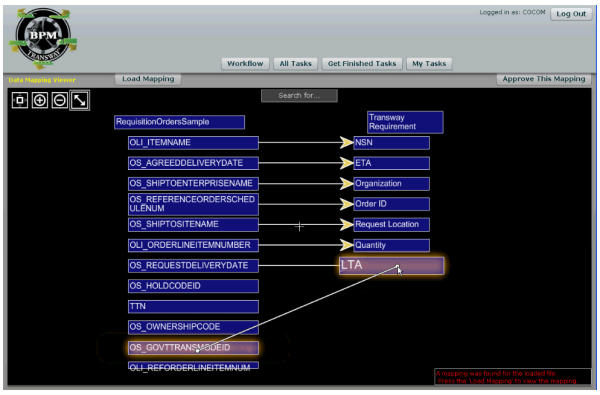

Figure 5. A mapping is suggested automatically by IMT.

By enhancing the data mapping results with suggestions from the Intelligent Mapping Toolkit in a graphical way we increase the productivity of data mappers.

9. Conclusion

We present an approach and a system design for building an integration platform, which supports syntactic and semantic interoperability based on an existing open source BPM framework. We add graphical mapping tools to facilitate building mappings among data sources. We also utilize intelligent mapping tools to provide assistance in selecting and

creating mappings based on the semantics of the data sources.

Such a system provides visibility of the interoperability problem. By exposing the tasks necessary in system interoperability to users in a graphical way it allows all individuals involved to see the amount of work that will be necessary for them to do their task. In doing so it allows system designers to quickly assess the possibilities of developing more enhanced automated assistance for human users such as IMT. In the future we hope to utilize this system to explore such possibilities through the integration of existing systems with varying data requirements.

10. References

[1] Andreas Tolk; James A. Muguira "The Levels of Conceptual Interoperability Model" 2003 Fall Simulation Interoperability Workshop Orlando, Florida, September 2003.

[2] Tolk, A.; Pullen, J.M., "Using Web services and data mediation/storage services to enable command and control to simulation interoperability," *Distributed Simulation and Real-Time Applications, 2005. DS-RT 2005 Proceedings. Ninth IEEE International Symposium on* , vol., no., pp. 27-34, 10-12 Oct. 2005

[3] Bernard P. Zeigler, Saurabh Mittal, Xiaolin Hu, "Towards a Formal Standard for Interoperability in M&S/Systems of Systems Engineering", Critical Issues in C4I, AFCEA-George Mason University Symposium, May 2008

[4] John Vlissides. "Pattern Hatching: Notation, Notation, Notation". C++ Report, April 1998.

[5] JBOSS-jBPM User guide. Online at: http://docs.jboss.com/jbpm/v3/userguide/graphorientedprogramming.html

[6] Robert Christoph Riemann. *Modelling of Concurrent Systems: Structural and Semantical Methods in the High Level Petri Net Calculus*. PhD thesis, Universität Hildesheim & Université Paris-Sud U.F.R. Scientifique d'Orsay, 1998.

[7] Whitepaper: "Integration Appliances: Deliver Application Integration Projects in Days" Cast Iron Systems, Inc. 2006

[8] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[9] G. S. Cunningham, K. M. Hanson, G. R. Jennings, Jr., and D. R. Wolf. "An object-oriented implementation of a graphical-programming system,", in Medical Imaging: Image Processing, M. H. Loew, ed., Proc. SPIE 2167, pp. 914-923. 1994

[10] Zang, M., Gray, A., Kriege, J., Pohl, J., Gupta, K.M., & Aha, D.W. IMT: A mixed-initiative data mapping and search toolkit. To appear in Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence. Chicago, IL: AAAI Press. 2008

[11] Gupta, K.M., Zang, M., Gray, A., Aha, D.W., & Kriege, J. Enabling the interoperability of large-scale legacy systems. To appear in Proceedings of the Twentieth Innovative Applications of Artificial Intelligence Conference. Chicago, IL: AAAI Press. 2008

[12] Rahm, E., and P. A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," VLDB Journal 10, 4. Dec. 2001, pp. 334-350

[13] Wen-Syan Li and Chris Clifton. SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. Data & Knowledge Engineering, Volume 33, Issue 1, April 2000. Pages: 49 - 84