Enhancing Simulation Value using Abaqus with SIMULIA SLM - A Customer's Perspective

Chris Pieper

Kimberly-Clark Corporation

Abstract: The emergence of simulation data management software packages provides an opportunity to both streamline simulation processes and further leverage the impact of simulation results. The nimble mechanism for process automation offered by SIMULIA SLM (Simulation Lifecycle Management) product reduces simulation turnaround by establishing connections between and managing simulation stages while allowing interactive components, such as Abaqus/CAE, to provide rich functionality. A strategy of combining a server based management system with local interactive components allows new and existing simulation processes to be quickly encapsulated into a streamlined tool. The management system (SLM) tracks data pedigree and provides sophisticated search/retrieval tools for simulation related data. The results are faster simulation cycles (build/run/analyze) and improved quality control of simulation activities (formalized process and associated control mechanism). Achieving the correct balance between adaptability and rigidity within a simulation process is a key aspect to consider as SLM is configured to a specific target user group. For example, expert users can take advantage of process streamlining and data management with full access to interactive application capability providing maximum flexibility. At the other extreme, a directed process with intentionally limited choices may be appropriate for a non-expert user. Combining a configurable process and data management tool (SLM) and customized interactive components (i.e., Abaqus/CAE) provides a common environment for teams to realize full value from simulation efforts. This paper uses two applications using SLM in conjunction with interactive tools to provide formalized simulation processes to illustrate the advantages of such a strategy.

1. Introduction

Computer simulations commonly require and generate voluminous amounts of digital data, these data represent a significant challenge to organize, share and manage. Further, as researchers, businesses and industries rely more completely on simulation based results to support decisions and aid learning, it is increasingly important to track data pedigree to insure conclusions are based on valid input. Many current business practices incorporate simulation, as a result, standardized analysis procedures are developed to insure quality results are delivered and used appropriately to aid business function. These factors are common to a large cross section of industrial and educational organizations and as such, many share the desire for a commercial solution that is robust, adaptable and scalable to meet their needs. SIMULIA SLM (SLM hereafter) product has proven to be a good solution to these problems and is being used to manage our Abaqus simulation data. This paper is intended to summarize our experience in using SLM (VR2009x) to

address the specific functional needs from the perspective of an analyst and simulation process developer.

2. Simulation data management

The management of simulation associated data is a necessary task for simulation of all types and at all levels. As the magnitude of simulation activity increases within an organization, the challenge of data management grows correspondingly. Additional demand is placed on managing data as analysis teams grow and become decentralized. For many organizations, it is common to involve multiple sites, working teams and individuals in the production and utilization of simulation data. The demand for efficient simulation data management quickly grows beyond the simple organization of a file structure, such as manually managing project files on one's hard drive. To meet these growing demands, specialized software solutions are appearing to relieve individual analysts or information technology teams from some of the burden of effectively managing large volumes of data.

2.1 Using SLM

SLM provides an infrastructure and interface to allow users to set up and automate simulation data management. Like any data management system, configuring SLM requires appropriate forethought and planning before implementation. Decisions from defining the foundational data organizational structure to determining which documents should be overwritten or versioned when changed must be made throughout the configuration process. Fortunately, configuration decisions can be changed if another strategy is preferred, but it is best to create a thorough plan prior to configuring the tool.

2.2 SLM data structure

2.2.1 Workspace definition

Defining the foundational data structure is typically the first step toward organizing data in a logical way to allow easy and efficient access. In SLM, the top level data structure is called a "workspace." A workspace may contain one or more data folders which can hold multiple levels of additional data organization. One of the first decisions regarding SLM data architecture is to decide how to organize these top level data containers. One choice might be to organize workspaces based on functional teams. Another choice may be to organize based on simulation discipline (e.g., structural, fluid, business, etc.). Still another choice may be to align workspace definition with the organization, business, institution or individual. Defining the workspace structure sets the foundation upon which the rest of the data structure is built. For example, we chose to define workspaces that correspond to analysis disciplines and utility categories (specifically, StructuralProductSimulations, FluidDynamicSimulations, Software and Templates).

2.2.2 Folder definition

Data organization within workspaces can be similar across workspaces or defined uniquely within each workspace. A workspace can only contain folders at its top hierarchical level; therefore, the data structure within each workspace is determined using folders. Establishing a sensible folder structure is very important, especially at higher levels in the data structure hierarchy. In our case, we chose to define top level folder structures that were best suited for each workspace category. For example, within the StructuralProductSimulations workspace, the top level folders include, ComputerCode, Libraries, Projects and SandBox. Multiple folder levels may be appropriate to represent the data structures in a way that is intuitive to users. While data object search capabilities allow broad or narrow data searching, it is helpful to take advantage of a logical data structure to provide context and to speed manual browsing for data. Figure 1 shows several levels of the data structure that we use to manage much of our simulation data.

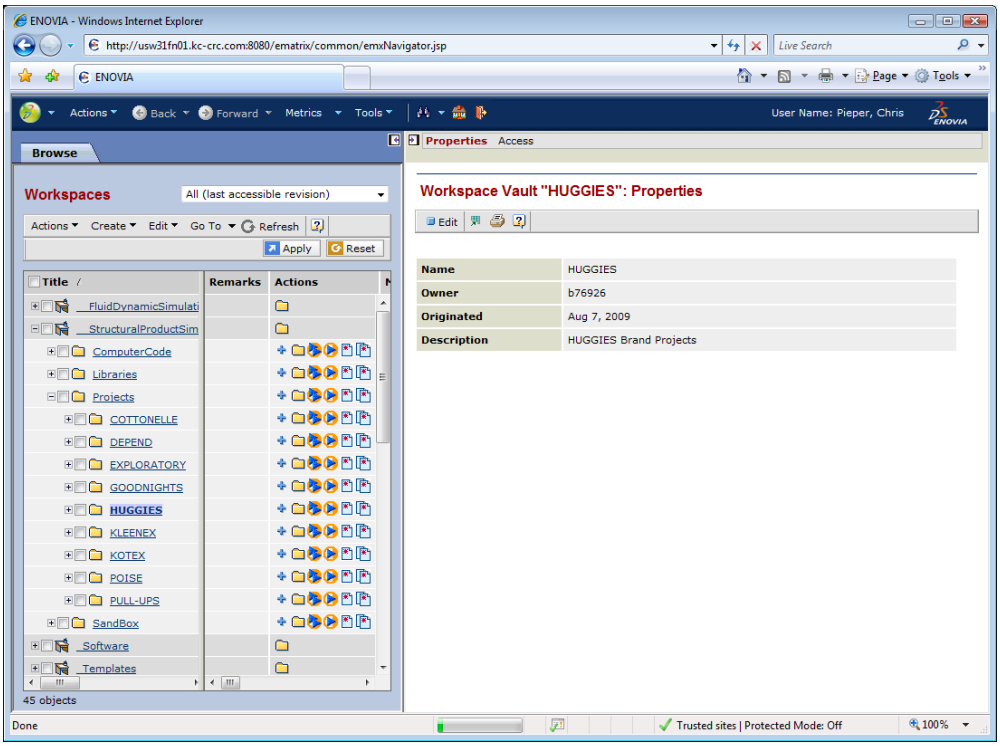

Figure 1. Example simulation data structure

2.3 Simulation Data Objects

While the intent of this paper is not to provide a tutorial on how to use SLM, some understanding of terminology and data objects is appropriate to appreciate how SLM meets the challenges of

simulation data and process management. The concepts of workspaces and folders are not too different from other well known data storage structures. The power of SLM goes far beyond providing a means of logical data structure; it is specifically suited to manage simulation data. The following definitions will provide a language to discuss SLM's design and how specialized objects can be used to construct custom functional data systems.

2.3.1 SLM object definitions

A partial list of definitions of some of the unique SLM terminology is described in the following (SLM objects are shown in bold letters).

Process – object used to collect an ordered set of **Activity** objects that is used to direct a process flow.

Activity – object used to manage a specific computing activity, it is associated with a **Connector** and contains several categories of **Documents** including Specifications, Internal Data and Results (typically corresponding to input and output of a specific **Activity** within a **Process**). An activity also has specific export and import rules defining how to manage documents and files when the activity starts and completes.

Document – object used to contain one or more files. A **Document** may be versioned or non-versioned (content within non-versioned documents are overwritten when changed). A document may be used in one location only or referenced and shared by multiple locations within the SLM data structure.

Connector – object used to interface with an application running on a host computer; the details of application interface are defined in a general sense within the **Connector** object and more specifically in a host configuration file used by the SLM application server.

Template – a prototype object with a defined structure to allow easy replication (instantiation) of commonly used objects (**Process** or **Activity** objects).

2.3.2 SLM objects

The working entities in SLM are referred to as objects just as entities used in object oriented programming (OOP) because this is what they are. An OOP object is a container for data as well as for the methods used by the object. SLM objects typically contain data content and have methods defined that allow them to interact with other objects or provide specific functionality. Consistent with OOP, users of the object do not need to understand the internal details; rather one only needs to understand the object's purpose and how to use it. Fortunately, the complexities of the SLM objects are hidden allowing a user to focus on arranging and connecting objects at a very high level to achieve the desired functionality. In most cases, a user will primarily interact with the system interface and work with existing data structures and processes.

2.3.3 Remote connectivity

Activities managed by SLM can run either on a user's local computer or on any other appropriately configured and connected remote computer. SLM utilizes a network protocol called secure shell (or SSH) to pass data and initiate remote execution. A user's local computer becomes an execution hub by providing the necessary authentication between the SLM application server and any remote execution host. This strategy requires appropriate setup between the user's local computer and any needed remote execution hosts. The advantage of this strategy is that it is very secure and robust. Disadvantages include the need to complete necessary configurations both on the user's local computer as well as any remote hosts the user wishes to have access to. A onetime setup is required on all computing resources used in conjunction with SLM.

2.3.4 Process and activity definition

SLM characterizes users by assigning them different roles and interacts with each user based on their role assignments. Two of these roles are Simulation Methods Developer and Simulation Analyst. A methods developer is responsible for arranging and configuring SLM objects as necessary to support simulation data and process management for the analysts who perform simulation tasks using SLM. In our early experience with SLM, the author served as the primary methods developer, analyst as well as having some overall system administration responsibilities. Serving in these different capacities gave the author a broad view and understanding of the SLM tool, its capabilities and potential applications. This paper reflects aspects of the author's experiences in each capacity.

Given this basis, the remainder of the paper will focus on two specific process implementations and some of the underlying strategies that have proven successful for our application needs.

3. Applications

3.1 General analysis process

In the spirit of OOP design principals, we began our implementation by going from very general to specific as we built up a framework to support our data and process management applications starting with a generic simulation process as shown in figure 2.

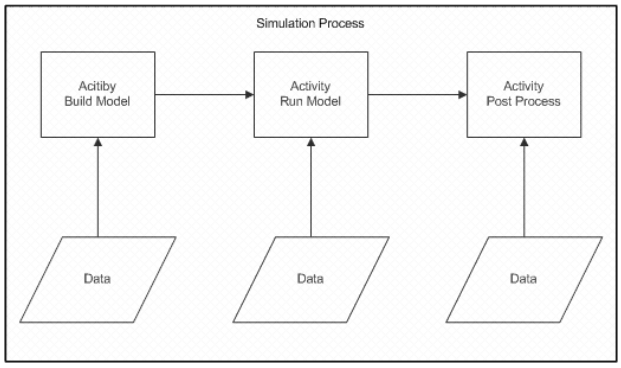

Figure 2. Generic simulation process

In SLM this general structure consists of a simulation process involving three activities. As the arrows of this simple diagram suggest, the activities are related to each other such that output from one activity is used as input to another activity. Many simulation processes can be described using this basic building block. For data management, SLM provides the connectivity between activities relieving the user from being concerned with file handling. Once the process is defined in SLM, each time it is run, the user can remain focused on the technical aspects of the simulation and leave most logistics to the data management tool.

3.2 Abaqus analysis process

One of the most common simulation activities at Kimberly-Clark Corporation involves structural analysis performed using the Abaqus suite of tools. A typical Abaqus analysis involves data inputs including geometry and material behavior specifications, model data such as a model database and Abaqus input file and analysis data such as an output database, post processed images and report files. With Abaqus analysis being one of our key data management challenges, it was a logical initial focus for our SLM implementation and evaluation.

3.2.1 Virtual product simulation process

Our implementation began by tailoring the generic simulation building block to specifically support our Abaqus virtual product simulation process. For this, one of our most well developed processes, the general process is extended by adding additional activities and enhancing the degree to which the process logistics are automated. Each activity, shown in the SLM interface in figure 3, has an associated connector used to launch a particular program with any necessary launch parameters.

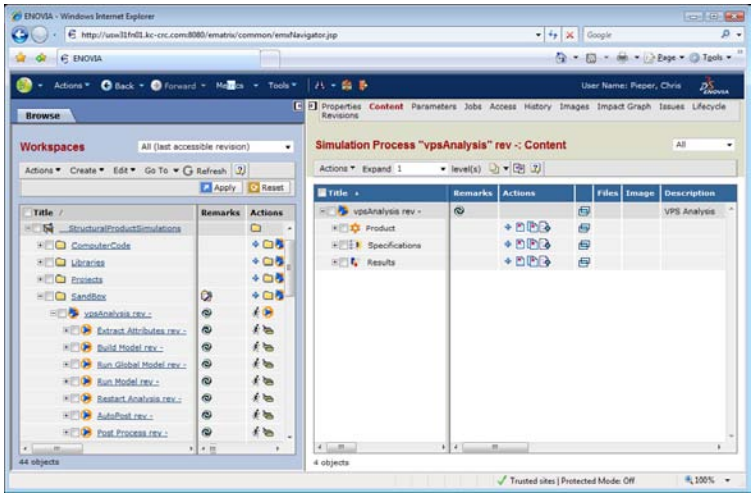

Figure 3. Automated virtual product simulation shown in the SLM Interface

3.2.1.1 Extract Attributes

The first activity in our process, called "Extract Attributes," initiates a Python script which is used to parse key metadata from a provided XML input file. The XML file is used to define simulation attributes that are useful to characterize the simulation and to aid retrieval in subsequent searches. The XML file is provided as input to the activity and is also used by other subsequent activities. A single copy of the file is stored in SLM and referenced where needed. This insures that files are synchronized between activities and eliminates redundant storage. Upon completion of this activity the values of the simulation process attributes are updated – even the name of the simulation process is built and assigned based on a combination of key attribute values. SLM includes a mechanism to pass parameters to and retrieve parameter updates from an activity. These parameters may be linked to attributes allowing visibility to search engines providing efficient search mechanisms.

3.2.1.2 Build Model

The next phase of the process involves a combination of automated tasks, driven by automation scripts, and interactive model definition using Abaqus/CAE. This activity is performed on the user's local computer. When initiated, SLM creates a temporary local directory and populates it with all needed input and script files. Abaqus/CAE is automatically launched and if a previously created model file exists, it is automatically loaded into CAE. The user can either initiate this activity directly or request a series of activities automatically sequenced and managed by SLM by initiating the simulation process. In our application, this activity may be performed once or multiple times until the model is completely and correctly defined. Outputs from this activity include the model file and Abaqus input file(s). File management is automatically handled by SLM based on how the activity's file import and export rules are defined.

3.2.1.3 Run Global Model

The next phase of our standardized simulation process is to execute a global model which will be used to drive submodel boundary conditions in a subsequent main model. The global model can be executed on the user's local computer or remotely depending on the user's preference and the size of the model. SLM creates a temporary directory on the execution computer, downloads the necessary input files and launches Abaqus. When the Abaqus run is complete, SLM uploads selected files (e.g., *.odb and *.prt files) and removes the temporary run directory and its contents from the execution computer. The relevant global model files are loaded into a document for use by subsequent activities within the process.

3.2.1.4 Run Model

The input file generated in the Build Model activity together with the global model files generated in the Run Global Model activity are downloaded by SLM into a newly created temporary directory on the execution computer. This activity typically involves running on multiple cpus on a computing cluster. In our case, SLM works in concert with a job manager (Platform LSF) to assign the job to the next available resources (in accordance with the run job parameters provided by SLM). This activity is more complex than some others because the responsibility for job execution was handed off to a job manager. In this case, SLM actually launches a "wrapper script" that initially submits the Abaqus job to and tracks the job status by communicating with the job manager on a regular interval. This wrapper script keeps SLM informed of the job status while the job is either queued or running. When the job finishes and the job manager delivers all associated files to the temporary directory, SLM uploads the requested files to be stored in designated documents. These documents are referenced in subsequent activities where needed.

3.2.1.5 Restart Analysis

Optionally, we have the ability to run a restart analysis to consider multiple use cases or changes in loads, interactions or boundary conditions. To accomplish this, we first go back to the Build Model activity and generate a restart input file. When we finish, SLM uploads the restart input file and stores it in a document within the Build Model activity where it is referenced in the Restart Analysis activity. Several files are needed in addition to the restart input file for an Abaqus restart analysis. The needed files, strategically stored in documents within their creating activities, are referenced in this activity allowing SLM to download them to a temporary directory created on the execution computer. When the restart run is finished, SLM uploads the requested files and stores them in documents within the Restart Analysis activity. In this case, we do not overwrite some of the original files to maintain a distinction between the original run and the restart analysis. If only the restart results are desired, unwanted files can be manually deleted to reduce storage requirements.

3.2.1.6 Auto Post

When a simulation is complete, a set of standardized post process results are generated using Abaqus/Viewer in batch mode running an automation script. This activity is orchestrated by SLM where it creates a temporary directory and populates it with required files including the automation script. When the batch post processing is complete, SLM collects the requested files and uploads

them into documents within the Auto Post activity. Key output documents are also referenced in data categories within the simulation process level where key inputs and outputs from the overall simulation process are collected providing a summary of the simulation status and associated files.

3.2.1.7 Post Process

A final activity is optionally available to allow custom post processing by running Abaqus/Viewer interactively on the user's local computer. This activity allows custom post processing activities to supplement the standardized output generated in the previous fully automated activity. When the interactive post processing session is completed, SLM collects and stores requested output in prescribed documents within the Post Process activity. Many of these documents are also referenced at the simulation process level providing enhancing the summary of the process input and output.

3.2.2 Using a simulation template

Creating a complex SLM data objects, such as a process or activity, often involves significant effort. Fortunately, SLM provides a mechanism to retain the underlying data structure of select objects in the creation of templates. For example, the previously described simulation process evolved over several generations to the point that it provides the functionality that is best suited for a specialized class of simulations that are performed regularly. Converting the fully developed simulation process into a simulation template allows future projects to be quickly setup and executed following a standardized practice outlined within the SLM simulation process. The logistical burdens of file management and process flow are orchestrated by SLM allowing the analyst to focus on the current simulation task. When a new simulation process is needed, a user simply instantiates a new process using a previously developed template and a new simulation process is generated. Figure 4 shows the simulation template instantiation interface. This template was designed with several simulation attributes exposed for the user to populate these fields when a new simulation is created. In this example, the simulation process was designed to automatically update many of the simulation attributes (during the Extract Attributes activity described above), thus the burden on the user is minimized requiring only two fields to be populated during instantiation.

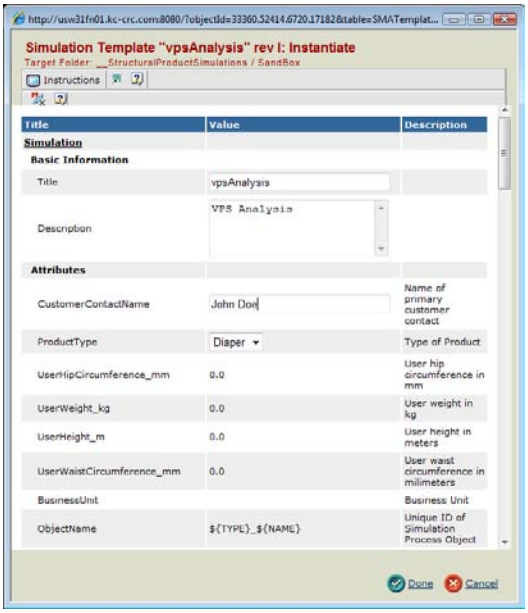

Figure 4. Example simulation process instantiation interface

Templates are created from already defined objects using SLM's "Save As Template" functionality. A template object has a title, description and may also include detailed instructions to inform a user how to use the template to generate a SLM object. The ability to create and use templates to define new simulation processes or activities facilitates standardized analysis practices and reduces simulation project effort offering a boost in both analysis throughput and quality.

3.3 Custom code application

While Abaqus analysis represents a large portion of simulation work done in our company, we also conduct a significant number of simulations using other modeling software. The same SLM mechanisms that support Abaqus simulations are available for other applications as well. In many cases, application connectors have already been defined and are available for commercial simulation codes and common programs. Even with the growing variety of commercial simulation offerings, there are still applications that are best addressed using custom or in-house developed software. SLM handles these cases as well. In structure, all simulation processes and activities are very similar regardless of which simulation code is used within. Specific simulation codes are supported by creating an application connector to interface SLM with the application. Using this methodology, SLM handles both commercial and custom software the same way; thus, SLM is capable of managing nearly any software application (anything that can be run from a command line can be managed by SLM).

Custom developed software often come without high level tools to prepare inputs (preprocessing) or evaluate outputs (post processing). While this problem is not directly related to SLM, it is a problem that must be addressed when defining a standardized simulation process and making this process available to multiple users. The following example illustrates a strategy that we have used to provide a user friendly preprocessor that is managed using SLM.

3.3.1 Java preprocessor

In our application, we have an internally developed code, written in FORTRAN[1], which reads a series of input files, performs desired calculations and generates a series of output files. A key challenge in this process is the generation of input files. Due to the unique nature of the simulation, there is no off the shelf product capable of providing all needed input generation functionality. Consequently, we must develop our own means to generate these files. Since our intention is not to develop a full featured commercial software product, we desire a strategy that provides the necessary function to allow our users to work efficiently while keeping development and maintenance costs low. The strategy we chose to develop involves using Java[2], running on the user's local computer, to collect or generate all needed input files.

This strategy has several advantages including low development overhead, the ability to deploy the Java application with SLM (application versions controlled by SLM) and running the application locally provides fast interactive response (versus running on a remote server through a web interface). This method requires local users to utilize a Java Virtual Machine (JVM) to run the Java application. This is no issue as the JVM is an instance of the Java Runtime Environment (JRE) which is included in our company's standard enterprise desktop installation (the JVM is also required for SLM). In practice, SLM provides all necessary Java code in the form of a Java Archive (JAR) file. This eliminates the need for users to have multiple applications installed locally on their computer and allows the application version to be managed by SLM. Figure 5 shows the SLM Run Activity interface corresponding to the launch of the Java preprocessor. SLM utilizes a "Java" connector, which is configured to properly launch Java on the user's local computer, providing an argument telling Java to read its instructions from a JAR file.

[1] FORTRAN is a programming language that is especially suited to numeric computation and scientific computing

[2] Java is a programming language originally developed by James Gosling at Sun Microsystems. For further information, see http://www.java.com

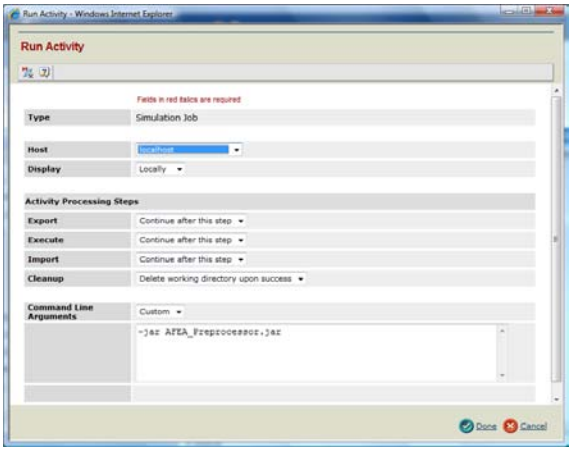

Figure 5. SLM Run Activity interface for the launch of the Java preprocessor

As usual, SLM creates a temporary directory on the user's local computer and populates it with the JAR file (which contains all necessary Java code and libraries) and any additional input or data files that are required by the preprocessor application. Figure 6 shows what the Java preprocessor looks like running on the user's local computer.

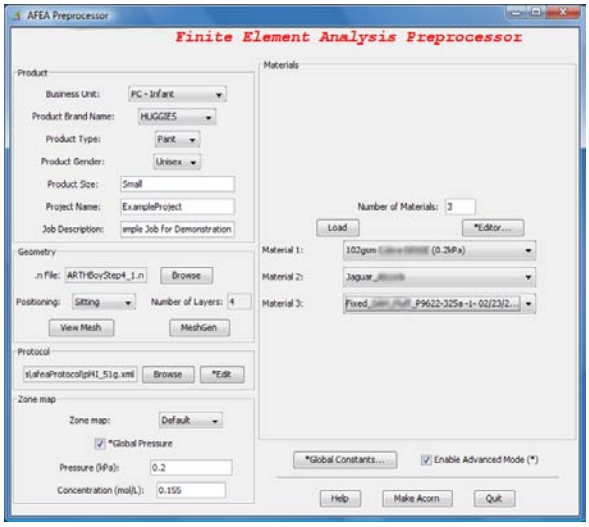

Figure 6. Java preprocessor running on user's local computer

3.3.2 Custom application simulation process

From an SLM standpoint, the simulation process involving a custom simulation application is no different than any other simulation process. The setup, control and data management structures

are the same for any simulation application. The differences lie in the specific documents that are defined or referenced and the application connector(s) involved. Our evaluation of using SLM to drive a non-commercial application demonstrated that even a custom application can easily leverage SLM's data and process management capability. Combining this with the ability to manage and launch a Java application that runs locally provides great advantage over a fully manual process. Once a simulation process is defined and the details established by a methods developer, the user can utilize the SLM interface to manage processes involving a variety of different simulation applications. This allows a user to leverage familiarity with the SLM interface for multiple applications rather than being required to be proficient with multiple application interfaces.

4. Evaluation of SLM

Our experience with SLM began as an evaluation to see if it would meet our primary need of Abaqus data management. Our evaluation suggests that SLM will be able to manage our growing volumes of Abaqus simulation data and also provides a mechanism to streamline the execution of our simulation processes while offering a standardized procedure that multiple analysts can use improving the quality and value of our simulation investments. We expect SLM to provide the ability to trace data pedigree and efficiently search for simulation results based on simulation attributes to reduce redundant data storage and allow reuse of simulation results.

4.1 SLM strengths

Key features of SLM that I consider strengths include its well established foundational infrastructure as it inherits much underlying functionality from its parent product, ENOVIA. From SLM's initial offering, the strategy to deliver essential functionality in a standard product avoids the need for costly customization, unique installations and maintenance. Providing wide functionality along with mechanisms for the customer to configure objects to meet unique needs or preferences is also a key strength. This allows customers specialists the freedom to develop unique solutions either with or without SIMULIA support. The logical layout of the user interface makes learning to use SLM intuitive. Standardized icons place in logical proximity to target objects provide most frequently used functionality with a single mouse click. Overall, we find a significant set of strengths that align with our simulation management strategies making SLM a viable solution for our simulation data management needs.

4.2 SLM difficulties

Being an immature application, the version of SLM we tested (VR2009x) did have some functional areas that should be improved. Some of the identified difficulties have already been dealt with in more current versions of SLM. The difficulties we encountered with the tested version are listed here to identify some challenges that we had to work around. We fully expect these difficulties to be addressed (in some cases, may have already been addressed) in future releases. Some of the frustrations we faced are summarized in the following.

4.2.1 Object access

One key frustration has been the need for explicit access control associated with SLM objects[3]. While this high level of control may be appropriate for some installations, in cases where such strict control is not desired, it can become a major source of user frustration.

4.2.2 Reliance on SSH

The strategy of utilizing SSH to facilitate file transfers and execution authorization comes with some constraints that hinder expanding usage. Not only is the initial setup of SSH connectivity to all necessary computing hosts tedious, this communication strategy does not support activity execution on a remote host while the local computer is taken out of the loop. This becomes significant if a laptop user wishes to submit a long running job and then remove the laptop from the network temporarily. Doing so breaks the communication chain between SLM and the executing computer removing the channel for data return. This is not a problem when an appropriate wrapper script is used; however, wrapper scripts can be difficult to write for general simulation applications. In our case, we utilize a wrapper script that is specifically designed to submit Abaqus jobs via Platform/LSF. In this situation, there is no constraint preventing a laptop user from temporarily disconnecting from the network. In other situations without an appropriate wrapper script, this is a limitation.

According to SIMULIA, SLM 2010x offers new functionality to allow jobs to re remotely executed using the "Simulation Execution Engine" (SEE formally the Engenious Fiper product). This functionally no longer relies on SSH for communication and allows an SEE server to manage all job functionality and return all results back to the SLM server. SEE still requires configuration on the machines that will be running jobs but does not require an SSH authentication setup.

4.2.3 JRE version dependency

Java versions (specifically, the JRE) may be set up to automatically update or may be managed on a corporate level. In one instance, we found when our Java version was automatically updated, SLM would no longer work. According to SIMULIA, the apparent Java version dependency is due to an error in Java 6 update 10 (which has been corrected in more recent updates). SIMULIA further reports that Java 6 update 14 and above will work with the tested version of SLM (VR2009X) and more recent versions of SLM should not see this incompatibility issue.

[3] A strategy to automatically modify access permissions using configured triggers on the create actions of the Simulation objects has been suggested by SIMULIA. This mechanism will allow full flexibility in assigning access permissions to objects allowing either strict or looser default access control.

4.3 Evaluation outcome

Our evaluation of SLM showed that the product meets our current needs for Abaqus simulation data and process management and based on the, we chose to adopt SLM to support our Abaqus simulation data and processes. We will continue to evaluate SLM's ability to manage other applications and processes as we transition all of our Abaqus analysis into SLM over the next year. To accommodate our migration, we are scaling up our infrastructure to support our vision of a multi-site collaborative network with all Abaqus simulation work managed by SLM. We expect SLM to continue to improve in both functionality and ease of use, the adoption of additional simulation applications will continue consistent with SLM's rate of improvement.

5. Infrastructure

During our evaluation, we utilized an infrastructure that was known to be unsuitable for a long term installation. Now that our evaluation is completed, we are transitioning to a more robust and functional architecture. Figure 7 illustrates the architecture that we believe will provide high functionality and data access to our distributed teams around the world.

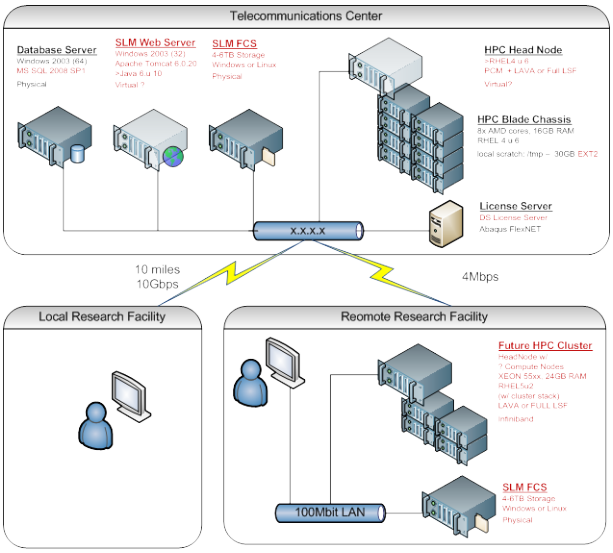

Figure 7. SLM infrastructure to support production SLM implementation

Longer term, we envision extending this infrastructure to incorporate our local SIMULIA office to support our ongoing collaboration on project work. All of the details for this future installation are still being worked out, but the ability to closely interact with SIMULIA as an external partner was a consideration in our choice of SLM to manage our Abaqus simulation data.

6. Summary

SLM is still a relatively young code and as such, there are several features that are not available or are non-optimal. However, because SLM is built on top of a very mature PLM infrastructure (ENOVIA), even its early versions offer an impressive collection of high level function. Our evaluation of SLM motivated us to adopt it for our Abaqus simulation data management. This choice was based on its demonstrated functionality and customer configurable design. My experience as a methods developer, analyst and system administrator has given me a unique perspective and allowed me to more fully appreciate the significance of SLM's capabilities. A successful SLM implementation does require an appropriate infrastructure, forethought and time to learn and configure; we expect the results to be well worth the effort.