Integrating CAE and PDM

A First Step Towards Providing Simulation Data Management

Ken Blakely, Larry Johnson, Boma Koko
Ray Amador, and Arthur H. Fairfull

Simulation is the act of predicting product performance prior to a product's manufacture. Most often, analysts, those doing CAE (Computer-Aided Engineering), perform simulation, and in too many cases only as an adjunct— not as an integral part—of the product development process. Analysts often consider CAE "outside the process," because of the difficulty and time involved in obtaining accurate inputs.

PDM (Product Data Management) would seem to be a natural complement to CAE, by providing a repository of data that is accessed by CAE. Historically, though, that has not been the case. The CAE community generally still views incorporation of PDM as an added burden, with benefits yet unknown.

Analysts often consider CAE "outside the process."

Despite the issues inherent in integrating CAE and PDM, a subset of "Simulation Data Management," the benefits include enabling the access, verification, and control of CAE input and output data. Simulation Data Management is the next frontier in making simulation such an integral part of the product development process that it will lead and dictate the overall design ("form follows function").

CAE Data

A product design must satisfy several objectives or targets. In mechanical design, these include:

- Product performance (strength, durability, deflection, …)
- Weight and stiffness
- Safety and reliability
- Supportability and maintainability
- Salability and profitability

A mechanical designer can "adjust" three basic "design variables" to meet the design targets:

- What the product looks like – its shape
- What the product is made of – its materials
- How the product is manufactured and assembled – its manufacturing process

These design variables provide some of the product definition *content*. However, the basic design content is not sufficient to perform simulation; a subject matter expert also needs to know the environment in which the product must operate and the design targets it must meet. The operating environment defines the external loads and the interactions between moving parts of the product (called assemblies). The design targets become the constraints that the design must meet or exceed.

PDM systems have, historically, been very good at archiving product definition content, but since considering a design's *context* is equally as important, PDM must be augmented if it is to be useful with CAE.

Design knowledge needs to be captured, managed, and leveraged for reuse.

Business Case

Before looking at *how* CAE integrates with PDM, reviewing the business case provides reasons *why* this integration is so important. The "business problem statement" is:

- CAE needs accurate, auditable, consistent input data.
- Accessing and verifying input data, which often come from multiple sources, needs to be time/cost efficient.
- Design knowledge needs to be captured, managed, and leveraged for re-use.
- Although a difficult task, the tremendous amount of CAE-generated output data needs to be reconciled.
- Output needs to be available to users of the data, in the forms they require.

Consider the example of an airframe that has hundreds of load cases, each representing a different operating condition. What is the likelihood of synchronizing all the correct load cases with the correct design iteration? The answer: Not very good, especially since some load cases are dependent on the attributes of the design itself.

The benefits to business of meeting this challenge include:

- Improved access to appropriate data sources
- Higher throughput for analysts and designers, faster design verifications
- Repeatable and more consistent simulation methods, making simulation a real "process," and not an art that varies per individual
- Fewer simulation errors and design/engineering changes
- Enhanced visibility and usability of simulation results
- Trace-ability and consistency of input data and results
- Enhanced collaboration between design and simulation, within the enterprise and throughout the supply chain
- Faster and more reliable engineering decisions

Thought of in a limited way, PDM is not a "value added" job function. That is, any time spent in a PDM system looking for data is time *not* spent designing product, performing analysis, or making the product better in any way. Companies do not pay for CAE analysts to find data; rather, they pay them to perform simulation to make their product designs better. However, as shown above, the important value of tracking and storing product data resides in the knowledge that a particular set of simulation results does indeed apply to the part being built under the load conditions it will encounter in the field. Viewed in this light, PDM is not only "value added," but essential.

Context-Enabled PDM

As previously noted, the CAE community views incorporation of PDM as an added burden, not a benefit, because PDM and other systems do not provide data in the *context* in which they are used, particularly in the case of CAE data. Without this context, analysts say they do not need PDM.

A well-executed program of Simulation Data Management manages the *context* of the data as well as the *content*. These data can be stored in PDM systems, materials databases, loads data management systems, ERP, and other systems suitable for storing and managing the data content. However, these data must be available to the user in the *context* of the task being performed. An analyst should be able to see, *without any extraneous information or process steps*, all the data and functions needed to review, apply, or perform the task at hand. Therefore, Product Data Management should be put in place as a *necessary resource* for any analyst, enabling getting the job done efficiently and accurately.

When properly presented with data in context, the CAE analyst:

- No longer has to search for data
- Does not directly use a PDM system to put in data or take it out
- Does not spend time doing bookkeeping to track the relationships among part versions, geometry, meshes, loads, analyses, and results

All these "nuisance functions" (from the perspective of the analyst), which used to occur as a normal part of the job, are now taken care of by an application that manages the analysis context of the data, regardless of the content.

No longer perceived as a burden that provides benefit to everyone except the analyst, PDM and CAE-PDM integration now have *real value* to the analyst because the overall system provides Simulation Data Management, addressing the *business context* needed to get the job done right.

An Architectural Framework for CAE-PDM Integration

Just as careful consideration of the business process, the tasks that comprise that process, and the computer applications that accomplish those tasks provide solutions, so will a careful approach to implementing CAE-PDM integration provide real value. Process improvements frequently involve the integration of systems and applications in order to reduce manual labor and inherent inaccuracies in using information across multiple business departments and multiple systems. These types of integrations require an organized approach in order to achieve the desired business benefits and goals.

Managing the context of the data, as well as the content, is key.

In planning and executing a program of CAE-PDM integration, careful definition of the architecture of the systems to be integrated, and the manner in which all the pieces will be integrated, ensures success. Architecture is simply a documented strategy of how pieces of the system are to be constructed, how the inter-connections are to be made. This is similar to documenting the architecture of a building (for example, documenting how the frame is to be constructed with specifications of typical corner junctions).

In documenting the architecture of a building, there is no single representation that can describe every detail of the structure. For example, a building may require a framing layout and a plumbing layout, each of which must be related to the other. Computer systems are no different. In these systems, each "layout" is referred to as an "Architectural Aspect."

Addressed in any integration strategy are a number of Architectural Aspects:

- *Business Architecture* (designates the business process, potential opportunities for improvement, and the most meaningful metrics for success)
- *Application Architecture* (inventories the applications to execute the business process)
- *Application Integration Architecture* (establishes patterns required for integrating new and existing software and systems)
- *Service (Function) Architecture* (defines the essential services required by the applications)
- *Execution Architecture* (describes how applications use the services)
- *Administrative Architecture* (defines an incrementally-developed and maintainable system)
- *Physical Architecture* (assesses the speed of computers and network links to assure robustness of the system)

Simulation Data Management is a made-to-order system.

Today, Simulation Data Management is not a COTS (commercial off-the-shelf) software system; rather, it is a made-to-order system, built around the needs and requirements specific to each customer and implemented as a services engagement. Such a services engagement is performed in five steps:

1. Requirements Definition Analysis: collecting data about the desired and current business process
2. Architecture design: creating the architecture to match the desired business process
3. Creation of the Simulation Data Management System: creating and implementing the system
4. Education and Training: familiarizing the users with the system and its data
5. Measurement: assessing the achieved business process improvement

Crucial because they compose the foundational elements of any successful CAE-PDM integration, the Architectural Aspects need particular attention in the planning process.

Business Architecture

The Business Architecture, the fundamental Architectural Aspect, is used to determine the specification of all other architectural aspects. All of the specifications of the technical Architectural Aspects must ultimately be traceable to the Business Architecture.

The goal of the Business Architecture's definition is to ensure that the system supports the business functions efficiently. The Requirements Definition Analysis collects information that will drive the specification of each of the Architectural Aspects. Based on the results, the system's architecture should be designed to create a robust, maintainable Simulation Data Management system that can evolve with the company's needs over time.

The Business Architecture description contains three main elements:

1. *The structure of the (future) business process* – The tasks of interest are described with their information consumption and production.
2. *A task-to-application tool mapping* – All major and minor applications required for a task are listed, along with the data consumption and production by these applications.
3. *A description of the data sharing requirements* – The types, access privileges, and

typical volumes of data shared among business units, tasks, and external enterprises such as customers, partners, and vendors are documented.

The Business Architecture description shows the business functions that require common data access; data viewing; automated data exchange; data access according to design geometry, materials, loads, and location; and automated updates and notifications corresponding to task completion and design cycles.

The current or "to be" business process must be defined in the Requirements Definition Analysis. Using an effective technique, Architecture Aspect planners define various "usage scenarios," showing data inputs, operations, and information outputs.

Application Architecture

The Application Architecture provides the structure for creating applications that directly serve the business process. An important consideration for planners is bridging the gap between business abstractions and information technology abstractions. The business process model inventories discrete tasks represented in a context natural for efficient operation. Since the existing software applications may not provide what is needed, the goal of the Application Architecture is to maximize the rapid development of applications directly serving the business process, thus permitting cost effective changes of applications as necessary to match any changes in the business process.

When legacy or discrete applications and databases are involved, it is often possible to develop or employ software "wrappers" (interfaces) that will provide a uniform interface enabling these separate systems to interact. Modern PDM systems are beginning to provide PDM Enablers Interfaces, a standard defined by the Object Management Group. As a consequence of these wrappers, programmers can write very small, targeted PDM applications that leave the bulk of the processing in the service infrastructure (see Service Architecture below). Many PDM vendors are offering or developing access to their systems through these interfaces.

Application Integration Architecture

The primary goal of the Application Architecture is to facilitate the rapid assimilation of stand-alone applications into a system with robust interoperability. The Application Integration Architecture, then, specifies the "glue" needed to perform integration. The seamlessness of integration involving legacy software packages is thereby enhanced — or limited — by the richness of each application's Application Programming Interface (API).

When possible, an API makes available specific functions of an application to other systems. Sometimes an application requires standard input files, in which case front-end software, or "wrappers," must be written to query other systems and retrieve the desired information so that the application can operate in its usual fashion. If results need to be made available to the system, services need to be defined to access the results in a uniform way across all consuming applications.

A good system will evolve with the company over time.

Service (or Functional) Architecture

The system requires a common set of infrastructure services to support all applications. The set of services includes product data management, data parsing and browsing, image viewing, version control; access privilege control, workflow, and more. Applications developed as isolated tools often contain aspects of these services implemented to various levels.

The Service Architecture moves the work out of the applications and into the shared ser-

vices. Depending on the Execution Architecture (see below), these shared services can be in various forms, including link libraries, dynamic libraries, DLLs, and CORBA services. Then, since the applications are no longer required to contain duplicate business rules and algorithms that are not shared, duplication is eliminated. Where possible, the system employs standard services, such as the PDM Enablers Standard referred to above.

A three-tiered approach provides long-term stability.

Execution Architecture

The Execution Architecture describes how an application uses the services catalogued in the Service Architecture. For greatest flexibility, a three-tiered client server architecture is preferred (Figure 1). Implementation of this three-tiered approach insulates changes in the object service from the application service and removes hardware dependencies, thus greatly increasing the maintainability of the system. Using this three-tiered approach provides long-term stability for the system. The three levels are:

- **Application Level:** When possible, the application is primarily responsible for the user interface. No specific work is done in this user interface module. The work is done through sharable services in the Service Architecture. Otherwise, any business policy or algorithm implemented by the application would be unusable in other applications, and if they needed to be used elsewhere, they would have to be duplicated, creating opportunity for error and severe problems for software maintenance.
- **Message Broker Level:** The specific application involved does not need to worry about the "who, what, or where" of the service provider. At the object service level, production quality service does not worry about who is using it or why. The broker serves as the communicator and mediator between the service requester and the service provider. This frees the application and the server of any considerations of each other's implementation language, operating system, network protocol, or location. Many brokering technologies are available: CORBA, COM/DCOM, Java-RMI, SOAP, and others.
- **Object Service Level:** This service level is defined in the Service Architecture (see above).

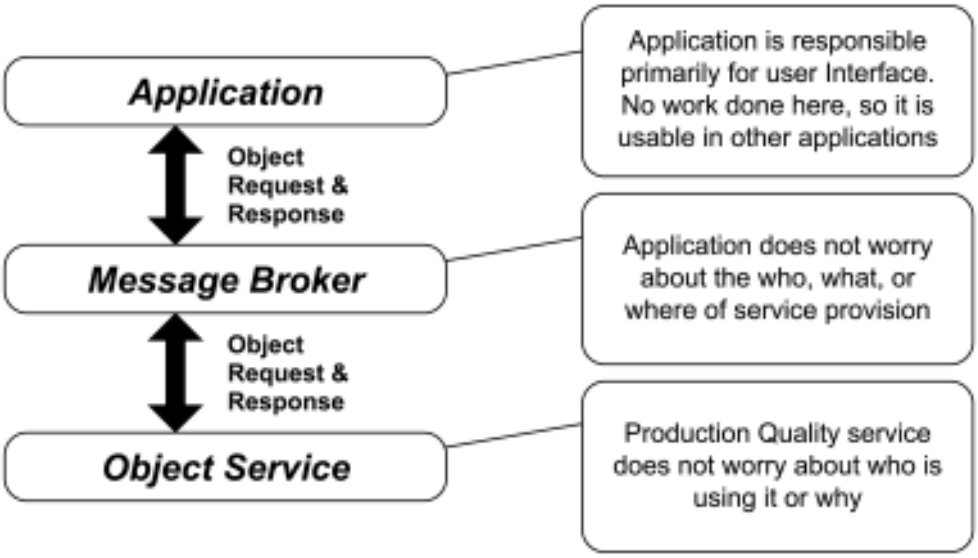

Figure 1: ***Execution Architecture Example – A Three-tiered Client Server Approach***

The Execution Architecture may use this three-tiered binding (Application-Message Broker-Object Service), the older style Client/Server two-tiered binding (Application-Object Service), a one-tiered binding (Direct Application Linking), or a combination of any of these bindings in the provision of the services. A business organization selects the best architecture based upon its business requirements.

Some computing environments intrinsically employ particular messaging transports. Microsoft applications use COM/DCOM coupled with DLL architectures. In many circumstances, these can be coupled with CORBA environments to permit standard Visual Basic programming while preserving system independence. In other cases, the applications can use CORBA through standardized COM/CORBA bridges. After architectural boundaries between the protocols are carefully considered, organizations generally decide to use a combination of approaches.

Administrative Architecture

As cost of ownership is such an important factor, businesses should consider the system's cost over its entire life-cycle. Not only must the system be cost-effective to purchase, it must also be extendible and maintainable. Technology and business processes evolve continuously, as do software systems. The architecture designers must, therefore, prepare for continuous development and multiple, continuous upgrades and major releases, rather than for a "final end state." Such a "blend and build" strategy for architecture definition, implementation, and administration is preferred. An *incremental development strategy* is key. The evolving design must include plans for reusability of the existing infrastructure.

Physical Architecture

Architecture designers should consider platform diversity, network bandwidth limitations, data storage device limitations, and all hardware limitations in the design. The projected computational and data storage requirements indicate the hardware and integration technology required to implement an integrated environment for the business.

When relatively small portions of data in a large file are required, it makes sense to use an API jacket to transmit only the information needed. If most of the file is required, or the file is needed repeatedly, it makes sense to transfer the large file when the speed of the network is also considered. Similarly, fine-grained object interfaces that are highly distributed generally do not perform as well as large-grained object interfaces. Architecture designers must balance object size, operation complexity, and network bandwidth.

Applying Architecture to CAE-PDM Integration

The volume of data in CAE can be many times the amount of data in CAD. Consequently, the infrastructure involved in the management of the various architectural aspects of the data is far more complex. To understand this, one simply needs to list the various disciplines involved with generating, consuming, and managing all types of simulation data.

The best architecture is based upon business requirements.

For consistent, repeatable simulation, the infrastructure needs to include a good way of handling materials data management starting with testing, jurying (approving), and exporting to the various analysis packages. Equally important, the system design needs to address loads data management starting from external loads through the allocation of internal loads and individual component analysis. Another important aspect the system must address is the correlation of virtual simulation with physical simulation to increase the organization's confidence in simulation results, thereby reducing expensive physical testing.

The list of disciplines that generate simulation data, and the complexity of each discipline,

is far more extensive and complex than that in the list above. For instance, in the aerospace industry, engineers determine external loads for various maneuvers of an aircraft at various altitudes by examining various combinations of cases, with the number of cases sometimes exceeding 10 million. From these, engineers employ filtering processes used to derive the most critical cases, which can number in the thousands. This number expands again as combinations are used to determine multi-axis loading for various analyses (linear static, fatigue, etc.).

The infrastructure needs to manage the volume of CAE data.

Considering a system that requires the consolidation of all these data into one physical homogeneous system is not practical. A more desirable solution allows for autonomy of the various heterogeneous systems while providing the *content* of the various represented systems to the user in the *context* of the business task at hand. The desired system needs to embrace the notion of federated systems, in which each system is best-of-breed and works well with the other system(s) to efficiently execute the business processes of the organization.

As an illustration, consider the example of integrating a materials database system with PDM, as illustrated in Figure 2. The Architectural Aspects include:

- **Business Architecture:** Enterprise processes use materials data that have been "blessed" (approved and archived).
- **Application Architecture:** In addition to the materials database and PDM systems, other applications access the enterprise materials data as needed.
- **Service Architecture:** The materials database, fitted with a set of services for accessing material properties, accesses the PDM system, which has a PDM Enablers standard interface.
- **Application Integration Architecture:** Special functions are defined and implemented so that whenever the physically common data are modified, they are modified in both systems.
- **Execution Architecture:** A CORBA-compliant, object-request broker isolates the applications from the service layers.
- **Administrative Architecture:** Changes to the implementation, or location of the PDM or materials databases, do not require changes to the applications that use them, thus making maintenance cost effective.

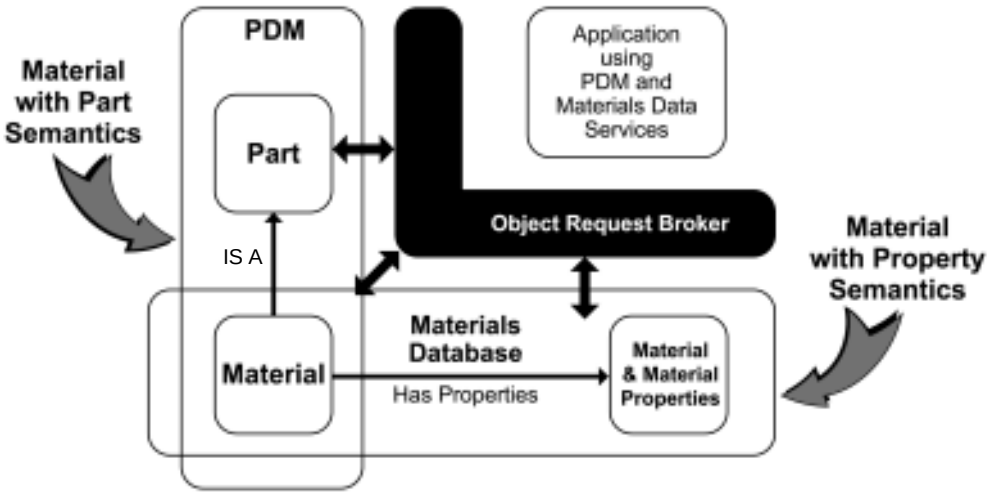

Figure 2: ***Federation through the Application of Multiple Architectural Aspects***

The results of this integration are:

- The PDM and materials database systems can be used independently, as they were before the integration.
- Applications using the CORBA service interface use both systems as though they were one. The application sees no application boundaries.
- The material properties can go through an approval process just as a part can go through an approval process.
- Since Parts support "Alternates" and "Substitute," materials now also can have a concept of "Alternates" and "Substitutes."
- The "Where Used" function of the PDM system, which normally locates the assemblies where parts are used, can now be used to locate where materials are used in the product as well.

Summary: Towards Simulation Data Management

Simulation Data Management encompasses more than CAE-PDM integration. It involves the integration of simulation systems, simulation data, and enterprise applications, all linked and usable by the CAE analyst and by those who supply data or retrieve results. While CAE-PDM integration is a subset of Simulation Data Management, the Architectural Aspects compose a subset of CAE-PDM integration. Both Simulation Data Management and CAE-PDM integration embody a combination of software and implementation services, specific to each company's business needs. The architectural approach is common, of course, but the overall implementation is company-specific.

When Simulation Data Management and CAE-PDM integration are implemented in the right way ("right" means using the architectural aspects described herein), the benefits are many. Companies that include education, training, and culture change management along with the technical aspects of these kinds of implementations increase the likelihood of a truly successful process or tool change initiative.[1] For CAE analysts who manage their simulation results by storing them on their hard disks — there is a better way! With the right architecture and implementation, Simulation Data Management makes simulation an integral part of the product development process, enabling simulation to drive the design. That is the ideal, and we are getting closer to it.

Integrating CAE and PDM is a first, necessary, and important step towards providing true Simulation Data Management. It will yield improved access to appropriate data sources; higher throughput for analysts and designers; faster design verifications; repeatable and more consistent simulation methods; fewer simulation errors and design/engineering changes; enhanced visibility and usability of simulation results; trace-ability and consistency of input data and results; enhanced collaboration between design and simulation, within the enterprise and throughout the supply chain; and faster, more reliable engineering decisions. Businesses that realize these benefits will have a competitive advantage — they will respond more efficiently and effectively to their customers' requirements, and will produce better quality products.

Federated systems efficiently execute business processes.

For more information, contact:
Ken Blakely, Sr. V.P.
MSC.Software
714-444-5103, kenb@mscsoftware.com

[1] Carol S. Melton, "Cultural Solutions for Successful Implementation," *Integrated Enterprise* 2 (Winter 2001): 19.

About the Authors

Ken Blakely is Sr. Vice President at MSC.Software, responsible for the company's global sales and marketing, software development, and professional services. Mr. Blakely has been at MSC.Software since 1983, and has held positions in business development, marketing, and software development. He is the author of more than 40 technical articles on simulation and test-analysis correlation. He earned a Bachelors degree in Engineering and a Masters degree in Structural Dynamics from the University of California at Los Angeles.

Larry Johnson is Chief Architect at MSC.Software. Since joining MSC in 1997, he has been responsible for the design and coordination of integration architecture for the seamless deployment of MSC products and services in customer environments; facilitation of collaborative advanced technology programs; and representation of MSC in software standards groups. He currently serves on the Object Management Group's (OMG) Board of Directors and is Co-Chair of the OMG's Manufacturing Domain Task Force. For 15 years prior to joining MSC, Mr. Johnson was CAD/CAM/CIM Infrastructure Architect at Texas Instruments, Inc. He earned a Masters degree in Chemistry from Indiana University, Bloomington.

Boma Koko, a Senior Director at MSC.Software, has over 19 years of experience in the development of information technology (IT) solutions for engineering and manufacturing organizations worldwide. He has held positions in Product Data Management development and implementation, manufacturing consulting, and software engineering. Prior to joining MSC.Software, Mr. Koko worked for Unigraphics Solutions, where he was responsible for, amongst other things, the development of IMAN (Unigraphics Solutions's PDM product). Mr. Koko earned a Bachelors degree in Aeronautical Engineering from Embry Riddle Aeronautical University and a Masters degree in Civil Engineering from Rensselaer Polytechnic Institute, Troy, New York.

Ray Amador, P.E. is Director, Professional Services at MSC.Software, responsible for the company's knowledge capture software and process automation professional services. Mr. Amador has been at MSC.Software since 1990, and has held positions in business development, product management, and software tools development. Formerly, Mr. Amador was the president of a systems engineering consulting company and author of more than 100 technical articles on structural dynamics, optics, stress, and thermal analysis and testing. He earned a Bachelors degree in Mechanical Engineering and a Masters degree in Aeronautics and Astronautics from Stanford University.

Arthur H. Fairfull, ***Ph.D.***, Product Manager for Simulation Data Management at MSC.Software, is responsible for product definition and business development in this field. Dr. Fairfull has specialized in engineering information management for 14 years, and has been with MSC.Software since 1992. He earned a Bachelors degree in Mechanical Engineering from the University of Glasgow, and a Doctorate in the Impact Behaviour of Composite Structures from the University of Liverpool.