PLM-Based Parametrics for Design Automation and Optimization

Jonathan G. Lund[1], Nathaniel L. Fife[2] and C. Gregory Jensen[3]

[1]Brigham Young University, jon@jonlund.com
[2]Brigham Young University, nlf5@et.byu.edu
[3]Brigham Young University, cjensen@et.byu.edu

ABSTRACT

While PLM systems are becoming more heavily relied upon, thus far they are not standard equipped to satisfy the diverse needs of every company. Because of this, additional add-ons and customization provide increasingly important roles in satisfying business needs. This article will discuss one such added functionality. The hardware and software architecture of PLM systems lends itself extremely well to distributed automation and object reusability—a feature not extensively found in current PLM systems. A method is proposed for object reusability through leveraging PLM data structures. This method greatly facilitates development in areas such as design automation and optimization that companies are finding difficult to integrate with PLM systems. Benefits of the methodology are increased speed, better communication and collaboration, smoother release changes, and many more. A system has been created that successfully automates design, analysis, and optimization with parallel, distributed and secure data and load sharing. The implementation was rapid due to the advantages of leveraging the PLM system architecture.

Keywords: PLM, parametrics, reusability, knowledge-based engineering.

1. INTRODUCTION

Ever since the advent of commercial CAD systems in the late 60's and early 70's companies have been searching for data management tools that would manage a product from its earliest conceptual designs through its years of engineering change and through its eventual retirement and death. Product Lifecycle Management (PLM) systems have come about to satisfy some of the difficulties in managing an entire product life cycle. They are powerful tools for business organization, collaboration, and management. Business needs and customer demands are driving product requirements to high levels in many areas, which in turn require object reusability, automated design, and optimization—concepts ironically foreign to current PLM software. The irony is that the architecture of PLM hardware and software is perfectly suited to this type of computing.

This research lays a foundation that enables object reuse and automation within a PLM system. Object reusability is defined in this paper as the degree to which existing objects (viz. any data, knowledge, or task in an engineering process) to be retrieved and directly used. Also, automation is defined as the automatic running a sequence of tasks performed in an engineering product design process. Discussed in this paper is the methodology that has been developed, a prototype based on this method, and a simple use case.

The methodology uses parametric data management inside existing PLM tools in order to facilitate object reuse and automation. Today's PLM systems use a central database and client/server configuration, which makes leveraging the PLM architecture a natural consideration for capturing many advanced distributed computing features such as security, business-to-business networking, multiple platforms, process dashboard, access control lists, and more.

A prototype engineering framework was implemented as standard API customizations and extensions to Teamcenter Engineering—a popular PLM tool. The added functionality was built into the existing workflow framework as a set of action handlers associated with process tasks. This also allows one to leverage the change management system native to Teamcenter. The proof-of-concept system was developed by the authors in a few short weeks and on limited funding. Even so, it demonstrates the same advanced core capabilities as multi-million dollar commercial automation

systems that require their own entire IT infrastructure. PLM and process integration software vendors have received this concept well, and plans are being made to merge their codes to provide a far superior environment for industries struggling with data and process management.

This paper gives a high level overview of this PLM object reuse and automation enabling foundation. It explains how this foundation can be leveraged to provide design automation to existing workflow and change management systems, and a simple use case involving legacy code integration, CAD solid model generation, and analysis. Many of the concepts in this paper have been explored in other environments such as web objects and directly from CAD systems. Some of the most pertinent of these articles will be discussed here.

In research conducted to explore the ability of web technology to assist in conceptual design, Wang et al. assert that for a conceptual design environment to be viable it needs to do more than simply support information access. It also needs to support the complete integration of analysis and simulation into a design process. Web technology itself cannot satisfy these requirements; however, agent technology may provide support to enhance the ability of Web technology.[10] While this paper's findings are similar to Wang's, it differs from their approach of creating an environment from scratch to address the challenge of conceptual design. The proposed method makes use of workflow tasks in existing PLM systems to provide the function of agents and the web. The approach used by Wang et al. was used to develop a distributed multidisciplinary design optimization (MDO) environment called WebBlow.[11] Their project strived to integrate the Web with agents in order to automatically access and manipulate information while enabling seamless interaction between designers, agents, and servers. The system is composed of software agents, Applets, and Servlets. Information is passed between the user interface and the several agents using XML. The major work includes developing a web-based user interface for design and implementation, agent-based computing resource management, and XML-based data management. While the application was initiated for blow molding applications, the methodologies and system architecture are extendable to any application where collaborative and distributed MDO is required.

Klaas et al. discuss the concept of embedding numerical analysis capabilities into an enterprise-wide information system.[4] To accomplish this they indicate the need to build upon the data management and workflow architecture of the information system. They also discussed the areas of PLM that are missing or inadequate for embedding these capabilities in PLM systems. One area mentioned was information structures and management. This is the ability to map data between the PLM system and commercial CAE and legacy tools, to provide them with needed input data, based on the problem description. Another area required is an attribute system. An attribute is information that describes material properties, loads and boundary conditions. The definition of a simulation problem requires a system to be developed that associates the geometric data and problem description with attributes. Klaas additionally mentioned the need for development of generic automatic simulation models. As of date, no literature is found that addresses these areas of future work. The work of this paper materializes many of the concepts presented by Klaas et al. and reinforces various advantages spoken of in their literature.

2. PLM ARCHITECTURE AND REUSABLE KNOWLEDGE

PLM systems run as application servers that manage files using extensive related information in relational databases. A great deal of information is stored strictly as database information with no files, such as bills of materials and account information. This type of storage allows users to search and retrieve a large number of records very rapidly. The client-server nature of the system is advantageous as well, adding support for version control, access over the web and even server-to-server linking. This article will show that these key points of PLM systems are perfect for providing distributed access to knowledge-based models such as spreadsheets and CAD data.

Relational databases function as a series of tables analogous to spreadsheets. Each table has a defined number of columns that expand by adding rows. Columns are defined as types such as floats, doubles, strings, and integers much like variables in a C++ class. Each row in a table can then be compared to an instance of the class. This object-oriented view of databases is very powerful when considering what is needed by parametric CAD systems. CAD features and relations embody a class, or definition, that display or represent a set of dimensions. This is an important advantage of parametric CAD because with one class definition and many sets of parameters one can represent many design variations. Currently, few designers are afforded the training, time or understanding to make use of this modern approach and consequently store each design as a whole part file. Another reason for this can be the current problem with data storage strategies and their inability to handle this form of parametric data.

Modern databases have evolved and have taken the name of relational database management systems (RDBMS). In these databases, often one of the columns will represents a relationship to a row in another table. This setup is similar to a pointer in a C++ class that refers to an instance of another class. These relationships can also exist as parameter sets for solid models that allow interesting and normally challenging aspects of CAD such as discrete component and assembly optimization. This will be discussed in further detail in the optimization section.

Ideally, reusable objects such as spreadsheets, parts, and meshes should be standardized and version-controlled, and then treated as a complete class or application much like a compiled software program. These objects or applications then become data engines (processors) with defined inputs and outputs. For example, when given a set of parameters, a version of a parametric solid model will generate a version of a design (Fig. 1). Each design can be stored as a set of parameters in database tables instead of a full part file. Multiple designs can be compared quickly by viewing design parameters using database queries rather than opening multiple models.

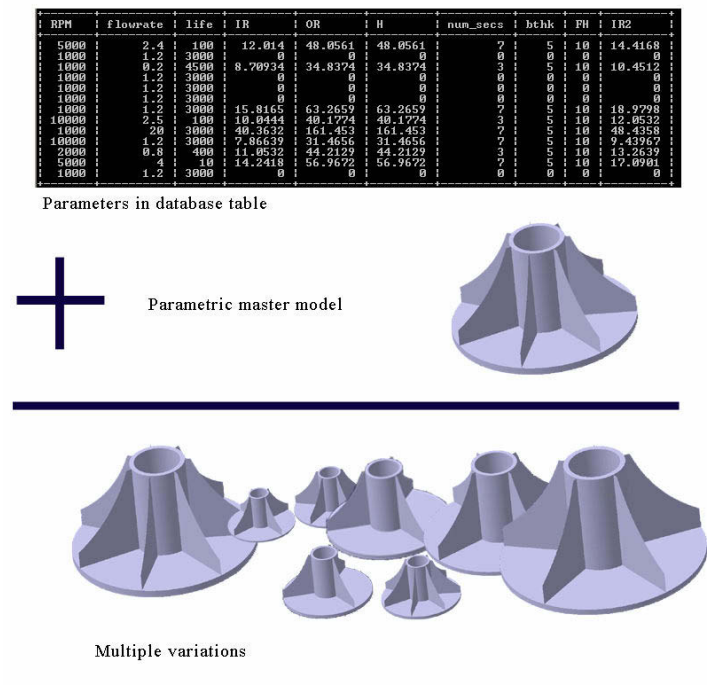

Fig. 1. Database instantiation

Another advantage of parametric use in PLM is the significant reduction of network bandwidth and storage requirements. One of the most common complaints with PLM systems is the speed sacrificed by transporting everything over the network. While PLM can decrease design time significantly through its organizational benefits, it feels slow and frustrates users. In the simple impeller example depicted in Figure 1, parameter storage and network requirements for saving a CAD model are several orders of magnitude greater than saving its parameter set. In other cases, it could easily be six to eight orders of magnitude greater. Companies could store version-controlled master models locally and then import datasets from the network. This becomes especially important in situations involving large numbers of iterations such as a large assembly optimization and design space characterization. Specific performance implications for entire PLM systems are currently being researched.

Design representation in a database also allows multiple objects to share datasets. For example, if the parameters to design an impeller were needed in a spreadsheet for cost calculations, the cost spreadsheet could be opened, the values imported from the *same database table as the solid model*, cost calculations performed and saved back into a cost table and the spreadsheet then closed *without saving*. Why would the spreadsheet not be saved? Once again, a piece of company knowledge is being used as an "application" for data transformation. Like the parametric model, the

spreadsheet itself could be version-controlled and stored locally. This approach is highly favorable for enforcing business rules and overall standardization.

For most companies, a great deal of company knowledge resides in legacy programs and is difficult to change. Using legacy programs in a parametric PLM environment is another example of data transformation. From a process flow standpoint, there is no difference between transforming data with a spreadsheet and old FORTRAN programs. Company knowledge can be stored and available for instant usage through creating parametric applications that are stored and managed in a PLM system.

3. AUTOMATION

The principle of parametric reuse has many beneficial applications, particularly in process automation. Once an engineering activity can be planned and organized into a repeatable process, reusable parametric models can make automation possible. Process automation can free engineers of mundane and redundant tasks. This freedom will allow them to be more creative, explore new designs, and improve existing designs. By automating engineering processes, higher product-quality and reliability can be achieved. Tools for providing process automation are available; however, none of these tools has the data management or collaborative capabilities of a PLM system. Automation and data management can be achieved through use of the PLM system. PLM workflow managers can be used to assist engineering design process automation within the PLM system.

PLM workflow is used by managers as a structured way to automate their activities. It generally has the built-in ability to automate the managerial process of assigning tasks to others and notifying participants of progress. The standard role of workflow is passive rather than active because it allows management to automate the assigning of tasks, but it does not actually complete tasks. By customizing workflow and making use of parametric engineering models, workflow can take on an active roll in process automation.

Workflow processes are made up of tasks, where each task is a collection of action and rule handlers. Rule handlers provide the ability to guide processes by preventing execution of subsequent tasks until upstream dependencies have been completed. An action handler provides the automated action. For example, action handlers can be used to assign tasks to responsible parties by placing task instructions in that parties' user or group inbox. The combination of action and rule handlers make up a task that can be used in a workflow process. PLM workflow allows custom action and rule handlers to be added to the standard collection of handlers. By adding handlers to perform automation tasks, workflow can be extended to perform engineering process automation.

In order to add custom action handlers, Teamcenter's API was used to add functions to the shared library, libuser_exits.dll. This is a three step programming process, as follows (unimportant details omitted for brevity):

1) Create the function that will run upon task execution:

```
int excel_handler(EPM_action_message_t* message){        // message is created by workflow
     char* workbook = get_file(message);                 // extract filename argument
     FormArgs * iargs = getInputForm(message);           // get handle to input data
     int i=0;
     Com_Stuff excel;                                            // get handle to Excel
     excel.openFile(workbook);                                   // open specified file
     char* name = iargs->getName(i);
     while(name[0] !=  ' ' && name[0] != '\0') {                 // insert each value
            VARIANT a = iargs->getVariant(i);
            excel.putCellValue(name,a);
            VariantClear(&a);
            i++;
            name = iargs->getName(i);
     }
     char* value = NULL;
     name = oargs->getName(i);
     while(name[0] !=  ' ' && name[0] != '\0') {                 // retrieve results
            value = excel.getCellChar(name);
            oargs->setValue(name,value);
            i++;
```

```
            name = oargs->getName(i);
        }
        attach_output(message, oargs->form->form_instance);        // upload results
        excel.closeExcel                                           // close
        return 0;
    }
```

2) Register the handlers in the automatic startup script (user_init.c):

```
        EPM_register_action_handler (                              // add handler to library
                "DFM-excel",                                       // name
                "Puts values from form into a spreadsheet",        // description
                excel_handler);                                    // function address
```

3) Add custom handler to workflow tasks to trigger execution (in GUI):

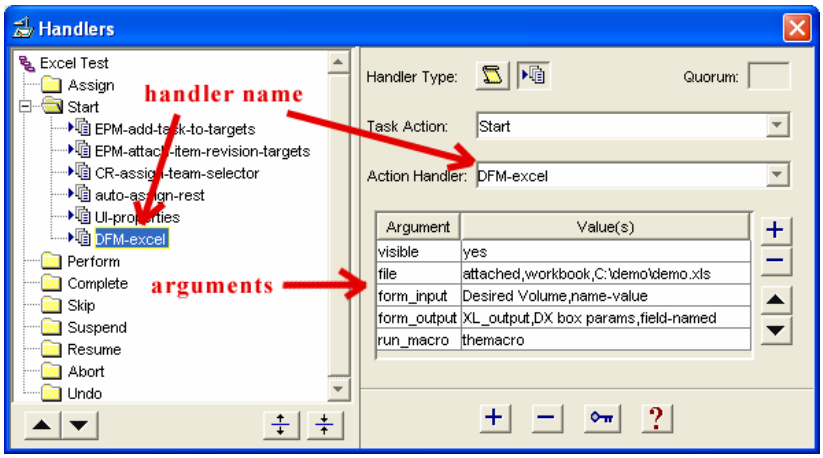

Fig. 2. Task configuration with custom handler

A workflow process can be designed to automate knowledge-based components of engineering processes. When such a workflow process is run, each of the parametric components that it contains will automatically be instantiated. Process data and artifacts are stored and managed in the PLM system. The workflow tool allows process tasks to be run in a sequential or parallel chronology. In typical parallel configurations, federated computers act as servers that must register themselves with a central control system, and then respond directly to requests through socket communication. In contrast, the proposed method for distributed processing handles jobs in job queues residing in the PLM system. Jobs are assigned to one or many computers that then act as agents that periodically check for queued jobs. When a job is completed, the output data is imported to the database as needed and the job queue item is flagged as being satisfied. This method provides several advantages, such as much simpler setup, superior activity logs, better error-handling and built-in security. One drawback is the response time of agents due to the lack of direct socket connection.

4. OPTIMIZATION

Performing design optimization using the PLM system requires a variation of the optimization paradigm. The current optimization paradigm assumes that the optimization is the overarching control program. It supplies the inputs to, initiates, monitors and retrieves output from the automation. The optimizer makes decisions about input changes and analyzes outputs in order to find the optimal design. Everything is performed optimization-centric, yet our paradigm is PLM-centric. Now the PLM system performs the automation and stores the inputs and outputs; the optimization is only a task at the end of the process. Its function is solely to analyze outputs with respect to inputs and suggest an improved design by instantiating a duplicate of its own process. The PLM system's workflow module is then responsible to run the new process with the suggested input.

Discrete optimization is greatly simplified when implemented within the PLM system. As mentioned before, references or pointers to other database objects can be stored as part of a model's dataset. This allows process parts, files, or any

other PLM object to be swapped by simply changing the value of the reference. When supplied with a list of options, an optimizer can select other parts by simply changing the reference value.

PLM-centric automation also lends itself to parallel and distributed processing. Just as tasks in the process automation can be run in parallel and assigned to run on distributed processors, entire iterations of the optimization can be run in parallel and distributed to multiple processors. If multiple design processes are spawned by the optimization task, each of these processes would then be run in parallel. A genetic algorithm could spawn an entire generation of designs to be run simultaneously. Likewise, a gradient-based algorithm could run all of the processes at once that are needed to approximate the gradient.

5. TEST CASE

A prototype process was created that utilized a number of the methods described here using the API of Teamcenter Engineering.

Fig. 3 shows a flowchart of the process used in this test case. The process uses an output file from a FORTRAN program as input data for the creation of a preliminary airfoil design. The data transformations take place through action handlers that are initiated by Teamcenter's workflow module. The airfoil definition is extracted from the flowpath text file and formatted for another legacy program that generates airfoil parameters for a solid model. Unigraphics then creates the model and the job is passed to the analysis team in a separate location where it is loaded and analyzed by ANSYS. The results are fed back to the system and inserted into the process folder. The success of this proof-of-concept process shows that PLM system architecture is a solid foundation for knowledge-based applications.

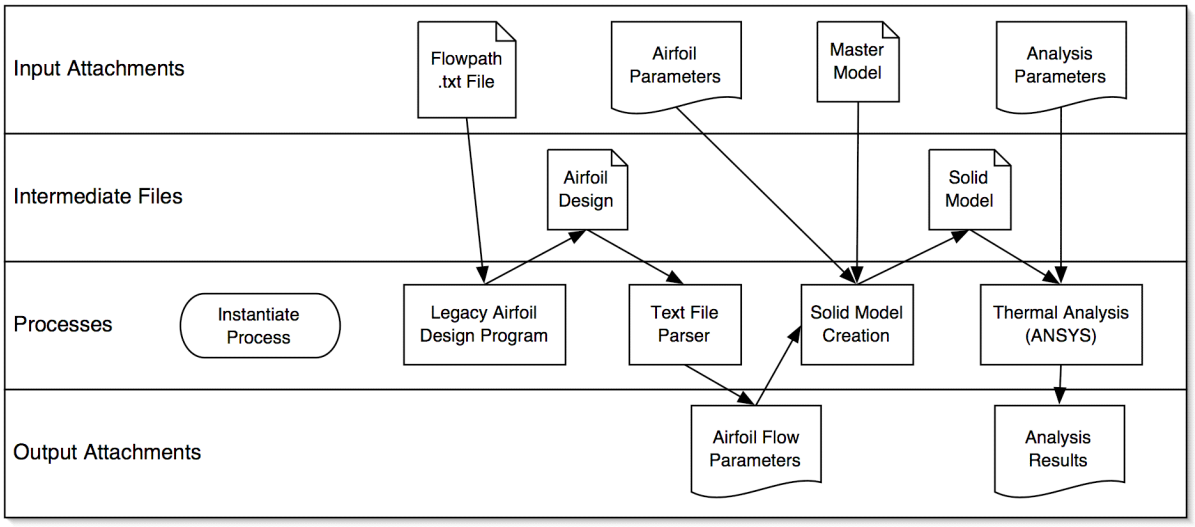

Fig. 3. The flow diagram of the test case

5.1 Defining database tables

Database tables can be interactively defined using the *Form Type Wizard* as shown in Fig. 3. Once the form type is defined, instances can be created and populated with data. For the airfoil design case the form object named *Airfoil Definition* was inserted into the *Inputs* subfolder of the *Airfoil* folder. This folder will be attached to the workflow process that performs the automation.

5.2 Creating the workflow process template

Workflow process templates are created using Teamcenter's *Process Designer* module. These processes are made up of custom tasks arranged as a flowchart. Knowledge is implemented into each task using extensible action and rule handlers. Once this process is complete, it can be saved as a template in the database. This template becomes a standard reusable workflow object that essentially standardizes the company's product development process.

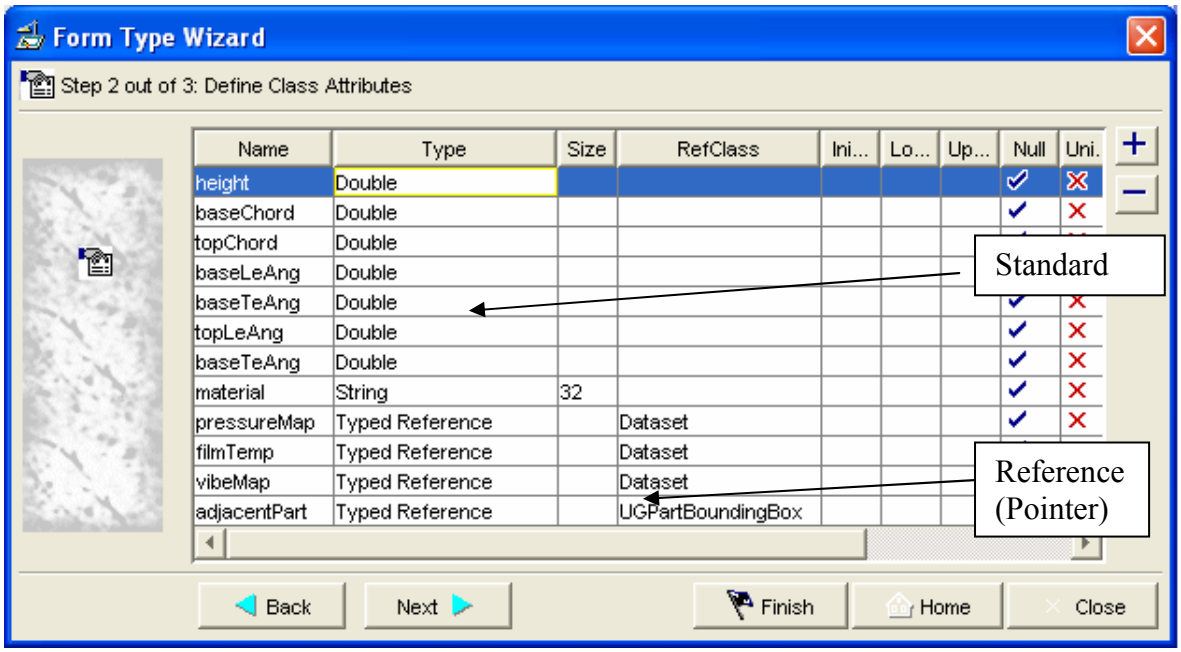

Fig. 4. Database Table Definition through GUI

5.3 Instantiating the process

Using the *New Process Dialog*, new processes can be created based on existing templates. Each instantiation of a process template becomes its own database object. Data to be associated with processes are inserted into a folder that is attached to the new process instance. Currently in the PLM system used, data cannot be attached directly to each task. For this reason data must be included in the folder attached to the process instance. This folder must contain an *Inputs* and an *Outputs* folder (see Fig. 5). The custom action handlers created for this process will search for inputs in the *Inputs* folder of the attached folder and will insert outputs into the *Outputs* subfolder. Through this method data can be mapped from task to task.

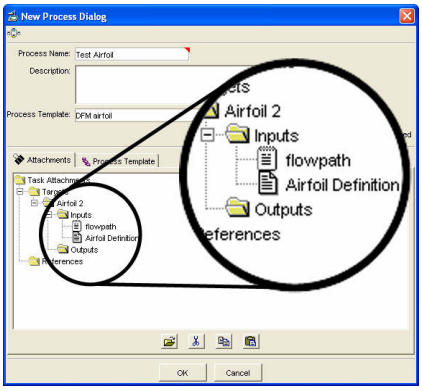

Fig. 5. Process Instantiation with Attachments

5.4 Monitoring progress

A dashboard is available to track the progress of the process. The process is located for tracking in the *Tasks to Track* folder in the *User InBox* for any users specified as being responsible for the process.

Fig. 6 shows a view of an associated user's Inbox with the associated process dashboard. Icons and a color scheme show the progress. In this scheme a flag indicates that the task is complete, a stop light shows that the task is currently underway and an hourglass indicates that the task has not yet been reached.

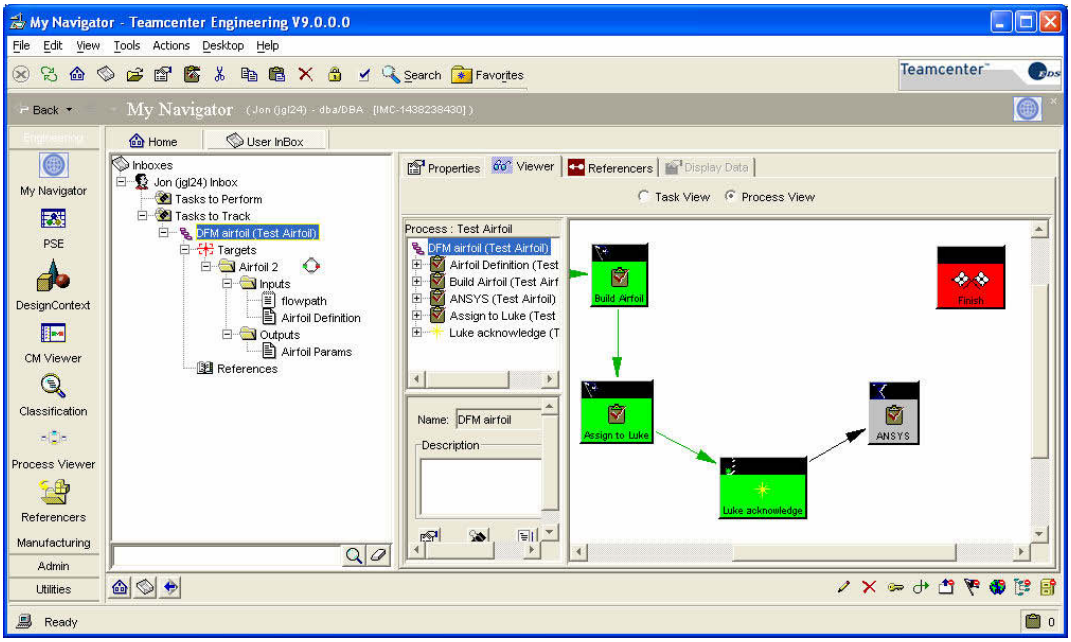

Fig. 6. Process Tracking through User Inbox

5.5 Reviewing results

Once the process is completed, the results may be obtained by viewing the forms inserted into the *Output* folder of the attached folder. This data is stored in the database and is included in the data management system.

6. CONCLUSION

PLM system architecture has proven to be a solid foundation for knowledge-based applications. This is shown by the quick development of the prototype process discussed as the test case. The reason that it could be created so quickly and with such complexity was because the PLM system was already available as the foundation. For PLM systems to supply an even firmer foundation, work needs to be done to allow for data to be attached directly to tasks within a workflow process and not just to the process itself. This will allow greater flexibility and easy in mapping data between tasks.

Although object reusability is not readily available in commercial PLM systems, it can be added and used with API assistance. The API allows for workflow functionality to be extended to automating the instantiation of these reusable objects. Future work is recommended to make datasets that link the parametric models to the parameter data stored in database tables. Viewing these datasets would automatically update and show the model results. These would be made so that objects upstream of the selected dataset would be updated as needed so that the most current data results would be viewed. In this way the reusable objects could also be used outside of the workflow module.

Performing automation and optimization within PLM systems allows the added benefit of highly advanced data management and actually improves PLM system performance. Distributed computing features of the PLM such as security, B2B, multiple platforms, process dashboard, access control lists for the automation were captured from existing software. The PLM system performance of these tasks was improved because the reusable objects required that only the parameter dataset be transferred.

A simple test case was constructed in very little time that demonstrates the same capabilities found in the most advanced process integration frameworks. The reusable knowledge was stored in the PLM system where it is available to be used by any in the company. Different components of the process were run on dispersed computers and user interaction was included where desired. Added features could include a user interface for mapping data and the ability to graph the results of the process for monitoring optimization progress. There are many possibilities in this area and we expect this to be the wave of the future.

7. REFERENCES

[1] Anon. “PLM: What does it mean? What do you want it to mean?” *British Plastics and Rubber*, n MAY, May 2004, pp 35-36.

[2] Aziz, H. Gao, J. Maropoulos, P. Cheung, W. "Application of product data management technologies for enterprise integration", *International Journal of Computer Integrated Manufacturing*, v 16 n 7-8, October/December 2003, pp 491-500.

[3] Caldwell, N. Rodgers, P. “WebCADET: Facilitating distributed design support”. *London, UK: IEE Colloquium on Web-based Knowledge Servers*. n 307, 1998, pp 9/1-9/4.

[4] Klaas, O., Shephard, M. "Embedding reliable numerical analysis capabilities into an enterprise-wide information system", *Engineering with Computers*, v 17 n 2, 2001, pp 151-161.

[5] Koonce, D. "A hierarchical cost estimation tool", *Computers in Industry*, v 50 n 3, April 2003, pp 293-302.

[6] Parunak, H. “What can agents do in industry, and why? AN overview of industrially-oriented R&D at CEC, Cooperative information agents II: learning, mobility and electronic commerce for information discovery on the Internet”. *In: Klusch M, Weiss G, editors. Second International Workshop, CIA’98, Paris, France: Springer*. 1998, pp 1-18.

[7] Sobieszczanski-Sobieski, J. Tulinius, J. "MDO can help resolve the designer's dilemma", *Aerospace America*, v 29 n 9, September 1991, pp 32-35, 63.

[8] Sobieszczanski-Sobieski, J. "Multidisciplinary design optimization (MDO) methods: Their synergy with computer technology in the design process", *Aeronautical Journal*, v 103 n 1026, August 1999, pp 373-382.

[9] Teamcenter online help collection.

[10] Wang, L.Shen W. Xie H. Neelamkavil, P. A. "Collaborative conceptual design - State of the art and future trends", *CAD Computer Aided Design*, v 34 n13, November 2002, pp 981-996.

[11] Wang, Y. D. Shen, W. Ghenniwa, H. "WebBlow: A Web/agent-based multidisciplinary design optimization", *Computers in Industry*, v 52 n 1, September 2003, pp 17-28.