2005-03-16

Developing a Design Space Model Using a Multidisciplinary Design Optimization Schema in a Product Lifecycle Management System to Capture Knowledge for Reuse

Nathaniel Luke Fife
Brigham Young University - Provo

DEVELOPING A DESIGN SPACE MODEL USING A MULTIDISCIPLINARY DESIGN OPTIMIZATION SCHEMA IN A PRODUCT LIFECYCLE MANAGEMENT SYSTEM TO CAPTURE KNOWLEDGE FOR REUSE

by

Nathaniel Luke Fife

A dissertation submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

April 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a dissertation submitted by

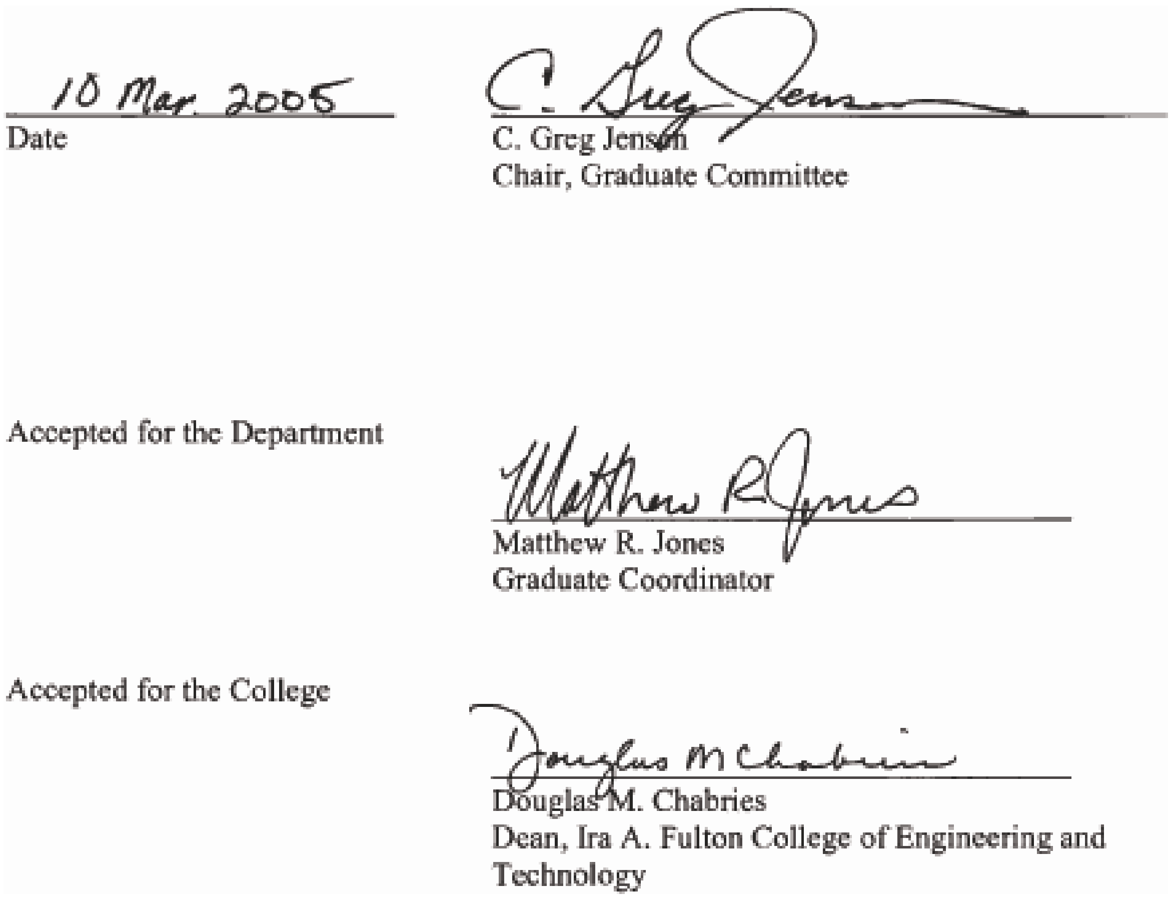
Nathaniel Luke Fife

This dissertation has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

10 Mar. 2005	[illegible]
Date	C. Greg Jensen, Chair
9 March '05	[illegible]
Date	Spencer P. Magleby
11 Mar, 2005	[illegible]
Date	Jordan J. Cox

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate’s graduate committee, I have read the dissertation of Nathaniel Luke Fife in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements: (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.



10 Mar. 2005
Date

[illegible]
C. Greg Jensen
Chair, Graduate Committee

Accepted for the Department

[illegible]
Matthew R. Jones
Graduate Coordinator

Accepted for the College

[illegible]
Douglas M. Chabries
Dean, Ira A. Fulton College of Engineering and Technology

ABSTRACT

DEVELOPING A DESIGN SPACE MODEL USING A MULTIDISCIPLINARY DESIGN OPTIMIZATION SCHEMA IN A PRODUCT LIFECYCLE MANAGEMENT SYSTEM TO CAPTURE KNOWLEDGE FOR REUSE

Nathaniel Luke Fife

Department of Mechanical Engineering

Master of Science

Parametric strategies for design automation and optimization can have a big impact on engineering design. When parametric tasks and optimization frameworks and methods are combined, theses strategies can be used to make up what is known as a multidisciplinary design optimization (MDO) schema. Knowledge of a design space can be modeled by using a MDO schema to represent the design process. However, current MDO frameworks used to create this schema lack the scope to capture enterprise wide knowledge for reuse and collaboration.

Concurrent with the development of MDO, many companies are moving toward increased use of product lifecycle management (PLM). More applications are being integrated into PLM as its usage increases; however, it has not to date been able to fully embrace the sophisticated knowledge model demands of engineering design. It has functioned primarily as a data storage and electronic email and tracking system. This thesis proposes to integrate an MDO knowledge representation in the form of a design space model with a PLM system to provide knowledge management for the product design process throughout the enterprise.

In this thesis a solution has been developed by leveraging PLM workflow management, and parametric PLM strategies. The PLM workflow management module was customized with action handlers, adding the ability to automate engineering tasks such as updating models and performing analysis. An optimization action handler was also added that iterates design processes by duplicating the entire workflow job and initiating it with updated inputs in order to explore and improve the design.

This thesis proposes a new approach to PLM and MDO framework usage that enables the complete representation of a design space with absolute, enterprise wide reuse. Because of the synergy that is created between PLM and MDO through this approach, both software providers and users in industry are looking at it as a way to achieve their greatest challenges. This thesis achieves the common knowledge representation that industry has been actively pursuing, because of this industry leaders have been impressed and believe that this approach will quickly take hold and usher in a new era for product design.

ACKNOWLEDGEMENTS

My wife, Kyla, daughter, Mariah.
Jonathan Lund, Dr. Greg Jensen, Brooke Barker

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Product development is a knowledge intensive activity. Companies generate and use vast amounts of knowledge while developing new products. This knowledge is stored in databases, reference manuals, employees’ memories and other places. The more efficient a company is at accessing and using this knowledge the better the designs are and the more profitable the product development processes become. Increasing globalization and market competitive demands are driving industry to seek out improved strategies for knowledge management. The past two decades have produced numerous knowledge management tools, but to date, companies have not been able to fully leverage these tools. Two tools in particular are product lifecycle management (PLM) and multidisciplinary design optimization (MDO). These two tools and their associated frameworks have the potential to transform product development. However companies have not been able to fully realize the associated benefits because they lack a common knowledge representation that allows full integration across the enterprise. This thesis presents an approach that defines a common knowledge representation and therefore allows for these tools to finally be used to integrate knowledge resources and make them readily available in context specific instances.

A third tool of significance is parametric strategies for design automation. Parametric strategies combined with optimization can have a big impact on engineering design by automating the individual product development tasks. When parametric tasks and optimization frameworks and methods are combined, theses strategies can be used to make up what is known as a multidisciplinary design optimization (MDO) schema. In an MDO schema, parametric methodologies are used to execute the design process. Each automated task requires inputs and produces outputs. These tasks are linked together to automate the entire design process. The MDO framework provides the ability to map the data flow between tasks and to perform optimization loops. Context specific knowledge can be uniquely stored within an MDO schema. The knowledge is captured in the form of a design space. This becomes an effective knowledge representation since it can be searched or queried within the context of the design process. However, current MDO frameworks used to create this schema lack the scope to capture enterprise wide knowledge for reuse and collaboration. Because of this MDO is used only in isolated engineering analysis situations, and has not been able to significantly impact design efficiency throughout the enterprise.

Concurrent with the development of MDO, many companies are moving toward increased use of PLM systems. More applications are being integrated into PLM as its usage increases; however, it has not to date been able to fully embrace the sophisticated knowledge model demands of engineering design. It has functioned primarily as a data storage and electronic email and tracking system. This thesis proposes to integrate an MDO knowledge representation in the form of a design space model with a PLM system to provide knowledge management for the product design process throughout the enterprise.

Currently, MDO framework solutions are not well suited for implementation within PLM systems because they use their own database application server, and therefore require significant workarounds to achieve integration. Even with PLM's increased functionality and complexity, industry uses it mainly as it did its predecessor, the Product Data Management (PDM) system, by using it to manage CAD files, with the additional feature of an internal email system and the ability to manage at a high level the automation of well defined processes. Without the ability to manage design space knowledge, PLM has not been able to achieve much more than PDM.

In this thesis, a solution has been developed that will enable knowledge management by integrating PLM automation and MDO optimization. By allowing MDO schemas to be created and managed in a PLM system, this solution makes it possible, for the first time, to capture design space knowledge for reuse and collaboration. This work creates a bridge between two engineering tools to make it possible for them to deliver their promised potential to industry.

To illustrate how the use of this approach will lead to vast improvements in product development, consider the production of a new jet engine. A typical new engine program costs a company between 100 and 500 million dollars. It involves approximately 250 engineers and lasts for 18 to 24 months with a burn rate of one million dollars per week.

One of the phases of a new engine program is concept development. Typical tasks in this phase are the creation of preliminary CAD models to give a global representation of the engine. These preliminary designs capture the main design intent to a level of detail sufficient for preliminary analysis. Generalizations and approximations are made in these models to leave out unneeded complexity.

This preliminary geometry must be meshed for analysis. Meshes must be generated so that structural, thermal and fluid analysis can be conducted. Each analysis requires specific meshes with boundary conditions, and loads relevant to the analysis mapped to them. Additionally, other properties must be applied as needed. From these analyses the performance of the design can be judged. The process of creating the geometry and analyzing its performance must be repeated until the design requirements are sufficiently met.

Once a system design has been decided on, the engine is broken down into separate modules for further design and analysis. These modules are based on engine location and function. Typical modular break down of an engine includes the fan, compressor, combustor, and turbine. At the modular or sub-system level the design process continues at a level of higher fidelity. The preliminary design is taken as the starting point for these higher fidelity models. The fidelity increases as more complex CAD models are created with more detail. These higher fidelity models take into account tolerances, nominal dimensions, and manufacturability. The model represents the designs to a level such that detailed analysis can be conducted to give a performance prediction that most closely matches reality.

To obtain detailed performance predictions sophisticated meshes must be made, and detailed information must be mapped to the meshes. Precise boundary conditions, loads, material properties, and other information must be applied to the models. The analysis results are closely reviewed and interdisciplinary relations are considered. Factors such as safety, durability, manufacturability, assembly and maintenance are also taken into account. Based on these results the design is then tweaked and analyzed so that performance can be improved. As much iteration must be performed as time allows so that the best design is proposed.

Once time has run out, the modules must be reintegrated into a system design. Interference and tolerances are taken into account and the design is updated as needed. Once the design is cleared, detailed design begins where every fillet, bolt and hole are included in the design. With this step the concept design phase ends and manufacturing planning takes over. A generic representation of this process is given in Figure 1.

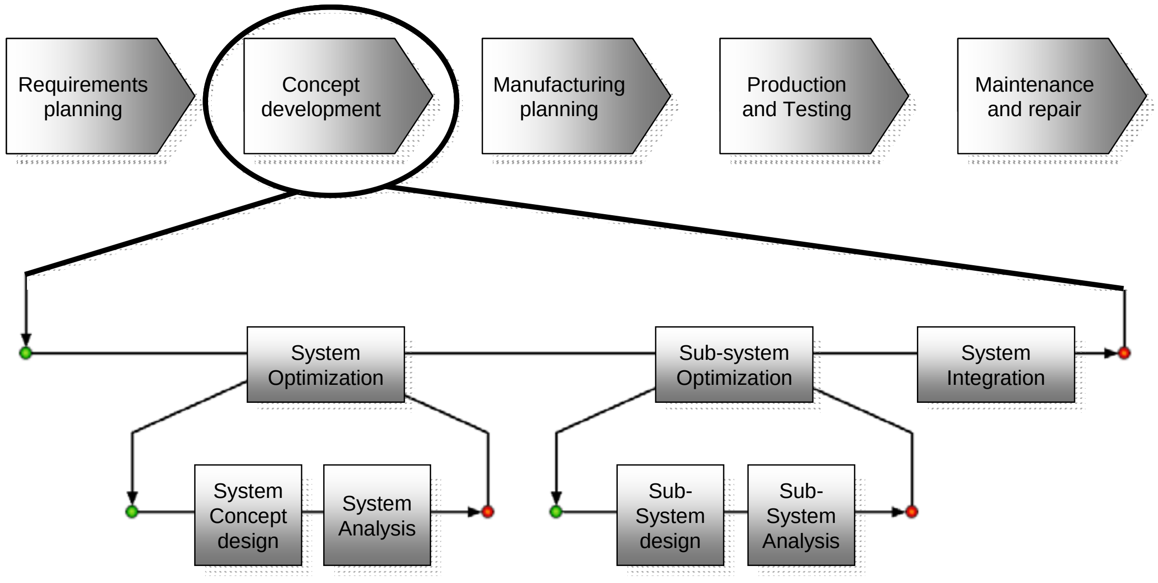


Figure 1 A generic representation of the concept development phase in a product's lifecycle.

This product development phase is much, much more complex and involved than is suggested in the preceding paragraphs. Precise details of designing an engine would take up numerous volumes. Companies have in fact generated huge amount of records documenting engine design over their decades of experience designing engines. Many strategies are used to store this information. These strategies include storage of data plots, charts, documentation standards and work standards. These are records are stored in multiple formats such as microfiche, paper documents, or computer files that are stored all over the company in filing cabinets, or on local computers. In fact, so much knowledge is stored and in such a haphazard way that, with the current method of using design knowledge, it would take years to make a new design based on this previous knowledge. However, as described above a new engine program is very expensive, and with a burn rate of one million a week, companies cannot afford to make use of their accumulated design knowledge because it would take too long. Consequentially, only a small percentage of previous knowledge is reused, resulting is engines being design mostly from scratch each time. Ironically, because previous knowledge is not used very often the same mistakes and pitfalls are fallen into every time. Figure 2 shows a representation of this situation.

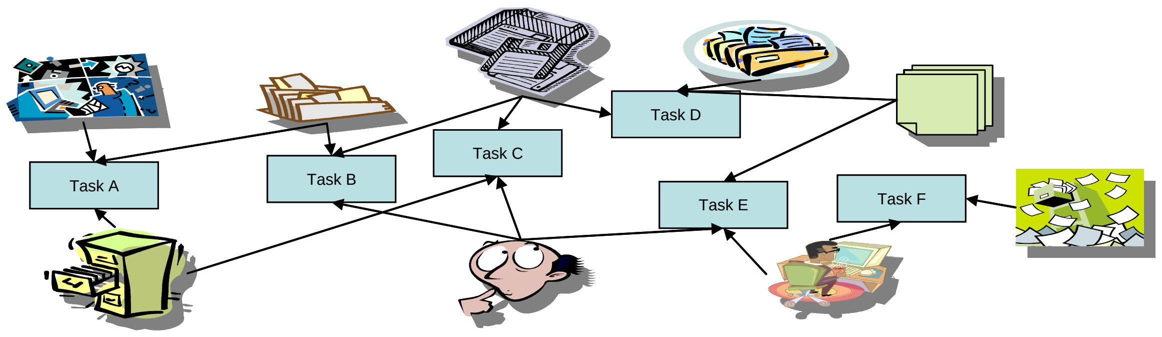


Figure 2 A representation of current practices for design knowledge. Each design task requires knowledge that has been stored hapazardly throughout a company. Knowledge is poorly organized and there is no clear method for its use.

As mentioned earlier, increasing competitiveness is driving companies to become as efficient as possible. This efficiency can be achieved through the approach presented in this thesis. Reusing knowledge can have such a large affect because a large number of product design projects in many fields do not require the creation of completely new designs, but rather variations on previous designs and therefore, reusing knowledge make a lot of sense. In jet engine design every new engine is a derivative of an existing design. In fact, every engine produced today can be classified under one of about four or five classic engine designs. Each new engine design overlaps to a great extent with previous designs. Because competition is high and so much overlap exists in every design, companies have a lot to gain by efficiently reuse design knowledge. This thesis presents an approach that allows companies to reuse knowledge. This approach is based on storing knowledge in an electronic form on an enterprise wide information system so that it is quickly accessible to all that need it. This new approach enables knowledge to be integrated into all levels of design and to be directly linked to applications that automate the design process, and optimize the design. This linkage is possible because knowledge is separated from its usage and linked directly to a central repository where everyone can use it. Figure 3 shows a representation of this approach.

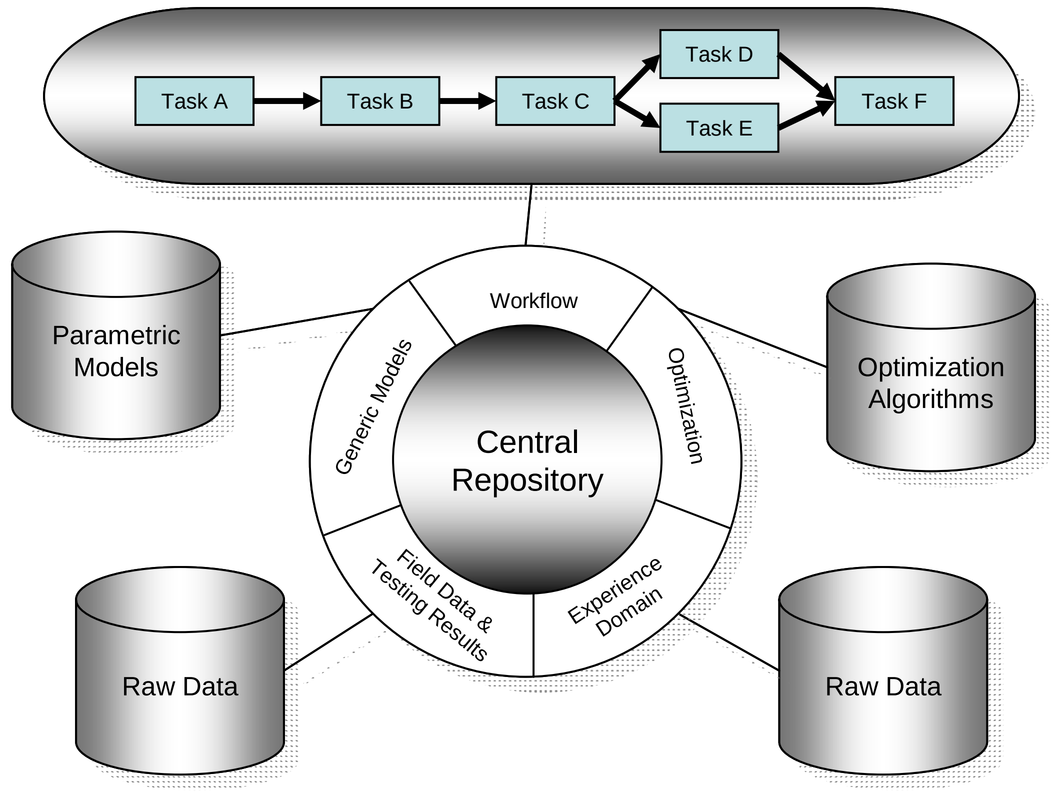


Figure 3 A representation of the knowledge management approach presented by this thesis. Knowledge is stored in an organized form that can be linked to automated design processes.

The way that knowledge is managed throughout the lifecycle can greatly affect the efficiency. For these reasons terms such as knowledge based engineering (KBE) and product lifecycle management (PLM) have become heavily used. However, a rigorous definition of KBE and PLM has proven elusive, for a very simple reason: no engineer wants to admit that they are not, in some sense, engaged in a 'knowledge-based' activity, no matter what their job entails exactly, or how they go about doing it. For this reason the reusability of knowledge in a management system must be carefully analyzed when considering how it can improve product design. The reusability of knowledge is affected mostly by the form in which it is stored and the means by which it is accessed. The knowledge management approach taken in this thesis improves product development by managing knowledge in a centrally located electronic form that is accessible and has the ability to be directly linked to applications that use the knowledge. In Table 1 a possible sampling of knowledge used in a design process is listed along with its form and means of access. The Improvement column lists the percent reusability gained by taking this thesis' approach rather that the current knowledge management. These percentages are based on dealing with engineering knowledge in design processes while working on projects within the aerospace and automotive industry and on three criterions adapted from those presented by Teare in his research of reusability [1]. These criterions are:

- Design information is undocumented.
- Design information is not accessible to other applications.
- Design information is poorly organized.

Table 1 shows the vast increases in knowledge usability gained through taking this thesis' approach.

Table 1 A comparison of the form and means of accessing knowledge currently employed with the approach taken by this thesis.

Knowledge	Current Practices		New Approach		Improvement
	Form	Access	Form	Access	
Geometry Dimensions	Technical Drawings	Microfiche	Database Table	Linked to Master Model	35%
Geometry Model	CAD File	Local File System	Master Model	Data Management System	85%
Mesh Parameters	Text File	Local File System	Database Table	Linked to Analysis Model	30%
Boundary Conditions	Text File	Local File System	Database Table	Linked to Analysis Model	30%
Loads	Text File	Local File System	Database Table	Linked to Analysis Model	30%
Tolerances	Human Knowledge	Request	Database Table	Linked to Master Model	50%
Nominal Values	Human Knowledge	Request	Database Table	Linked to Master Model	50%
Dependencies	Human Knowledge	Request	Workflow Process	Data Management System	50%
Safety Factors	Human Knowledge	Request	Database Table	Linked to Analysis Model	50%
Fatigue Life	Text File	Local File System	Database Table	Linked to Analysis Model	30%
Material Properties	Plot	Microfiche	Database Table	Linked to Analysis Model	40%
Lessons Learned	Human Knowledge	Request	Text File	Attached to Workflow	55%
Cost Effects	Text File	Local File System	Database Table	Data Management System	60%

The knowledge management approach of this thesis offers such large improvements over current practices because it is developed so that it retains company design experience in a form that can be queried, reused and archived as a historical experience domain. Product lifecycle management is seen as the ideal historical experience domain in which the knowledge is to be retained. This approach also enables active multi-team collaboration, standardization, and mass customization.

Product lifecycle management will be presented in this thesis as the ideal historical experience domain for collaborative design activities. Also, this thesis refers to design engines and their links to iterative design searches as framework tools, providing a framework where elements of the design process may be ordered together and automated. To introduce these concepts and to set the stage for the thesis, the following sections are included in this introduction:

- Objective
- Background

1.1 Objective

The objective of this thesis is to identify a knowledge representation strategy that can be implemented effectively in a PLM environment. This will create the ability for company wide design space knowledge reuse. This objective will be achieved by representing the design space in the format of an MDO schema that can interact with a PLM architecture. The feasibility of this approach will be tested using two proof-of-concepts. One proof-of-concept integrates framework capabilities into the product lifecycle management solution. The second proof-of-concept is implemented by embedding a commercial framework tool into the product lifecycle management solution. Both applications are developed to demonstrate the power and validity of embedding process integration and design optimization within a PLM system. This thesis will answer the following related questions:

- How can an MDO schema best be represented in a PLM system?
- How can PLM architecture be effectively leveraged to manage the MDO schema?
- How can the MDO schema and PLM interact to preserve reuse and modularity?

The following figure shows how this approach is taken though making use of a PLM system:

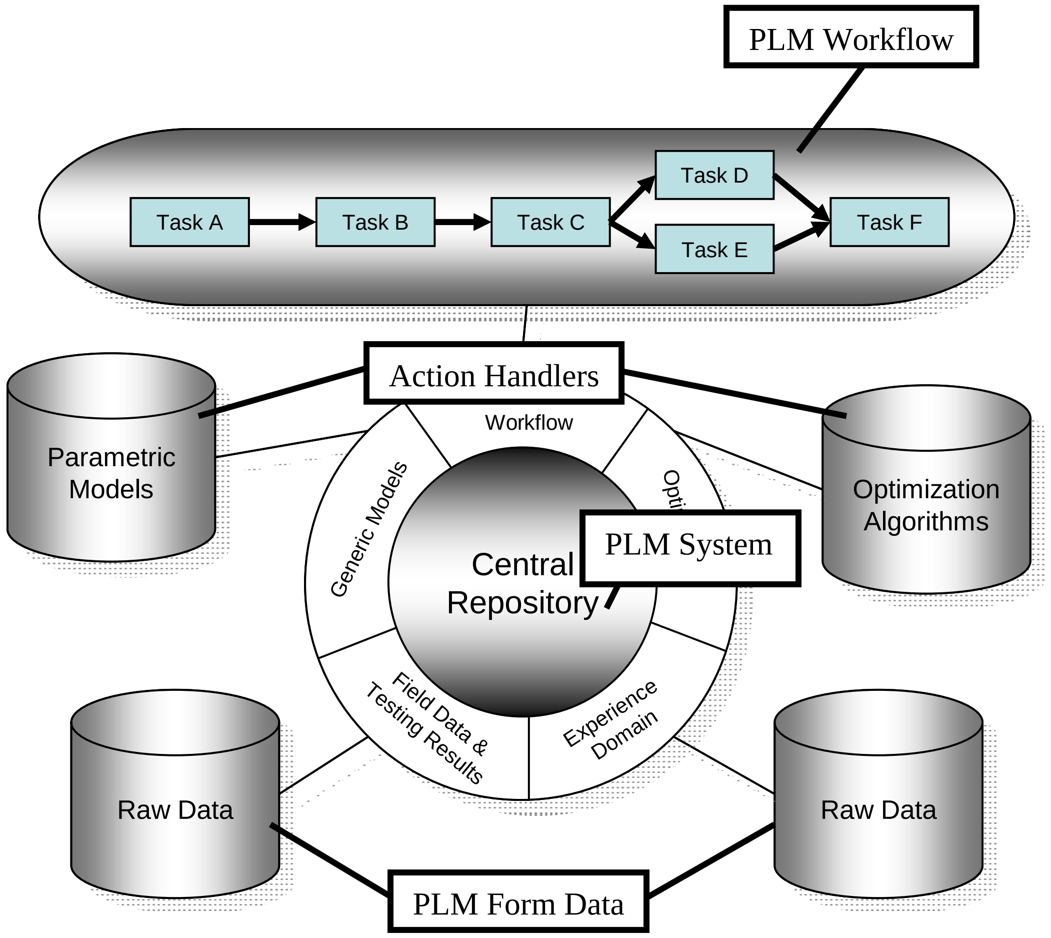


Figure 4 A representation of the approach taken in this thesis, with labels showing how each area will be implimented.

1.2 Background

This section is included to set the stage for the major issues in this thesis. Later in the body of the thesis it will be assumed that the readers have a basic understanding of the following:

- Product Lifecycle Management.
- Framework Tools.

1.2.1 Product Lifecycle Management

Product lifecycle management is a crucial element in a company's strategy for decreasing marketing time, while increasing the availability of product options and product variants. The advent of Web technologies has caused many companies to seek out the possibility to increase the collaboration between different organizations within their product lifecycle. Many solutions are being suggested to solve the challenge of improving collaboration during concept design. One such solution is product lifecycle management (PLM). Product lifecycle management is the application of Web technology to product data management (PDM). It also expands the scope of PDM to include among others, supply-chain management (SCM), enterprise resource planning (ERP), and customer relationship management (CRM). PLM systems are soon to become the working computer environment for engineering enterprise. A company's crossover to PLM may be a daunting task, but with mega-businesses like Boeing, General Motor, and Ford leading the way and insisting that key suppliers also implement PLM systems, it appears that crossovers will soon occur. [2]

In literature, PLM is said to be the key to reducing time to market and product cost, while increasing innovative content and available product options and product variants. [3] PDM is, in part, the progenitor of product lifecycle management. The goal behind PDM is that product data is stored only once, in a secure electronic vault. Information pertaining to a product may be stored along with the files. This concept allows changes to be controlled and data integrity assured. PLM takes the concept of PDM and vastly extends it to involve an entire enterprise over the product's complete lifecycle. [2] Critics may argue that product lifecycle management is too all-inclusive, and therefore bound to fail. However, companies are already receiving a return on their PLM investment.

One company cashing in on the product lifecycle management boon is General Motors. It has achieved as much as $1 billion in cost savings while improving product quality. In terms of decreasing time to market, GM reduced product development time from six years to one year. Other companies have similarly seen the PLM benefits. [4]

1.2.2 Framework Tools

Process integration and design optimization (also termed framework software), are tools needed for engineers to achieve quicker time to market and greater product variation while achieving higher levels of quality and reliability. The key to the design and manufacturing of superior products is the generation, control and integration of all levels of engineering information. Modern engineering is dependant on the aid of computers. Fortunately, many computer applications are available to engineers. Some of these are available commercially, while others may have been developed in-house to solve a company's specific challenges. The performance characteristics of complex multi-disciplinary systems can be predicted and optimized by linking together multiple applications, each of which model different aspects or disciplines within the system. Many applications were not originally designed to be linked together. They may have been created using differing computer languages, input and output formats, or they may not even run on the same platforms. To solve these challenges, framework applications have been conceived. These applications allow data to be mapped to the different analysis applications. They also automate the process by invoking the applications in parallel or sequential order as specified by the engineer. As computing power and the availability of analysis applications increase, the need for individual members of a design project to share information and to collaborate and coordinate their activities within the framework also increases. For this reason, framework applications are being expanded to include collaboration capabilities.

CHAPTER 2: LITERATURE REVIEW

The chapter includes the results of the literature review. The subjects reviewed most generally fit into categories that answered the questions why, how, and what. These categories are discussed in that order under the following sections:

- Motivating Improved Process Knowledge Management.
- Mutual Contingencies.
- Previous Solutions.

2.1 Motivating Improved Process Knowledge Management

In recent years, emerging concepts for product design have gained credibility in enabling quicker time to market, improved quality and more product offerings. The hype surrounding these concepts has caused many companies to spend considerable effort in their implementation. This thesis presents an approach that enables companies to benefit from these concepts while lowering the cost to implement them. This chapter focuses on three of these concepts that are profited most by this thesis and have been the motivation for improving process knowledge management. The hype surrounding these concepts and their potential benefit to companies are discussed. Throughout this thesis these three concepts are used to both benchmark different tools used for the concept's implementation and to illustrate the prowess of the new approach developed by this thesis. These three key concepts are:

- Multidisciplinary Design Optimization
- Standardization
- Mass Customization.

2.1.1 Multidisciplinary Design Optimization

Multidisciplinary design optimization (MDO) is a rapidly growing body of methods, algorithms, and techniques that enable the design of complex interdisciplinary system. It can be roughly characterized as the concepts that make it possible to optimize a complex design that spans multiple disciplines. Multidisciplinary design optimization has become a major initiative of today's companies. To remain competitive customers are requiring companies to improve product performance by increasing product complexity and taking interdisciplinary interactions into account during design. Due to this, companies are forced to find ways to achieve multidisciplinary design optimization so that they can remain competitive. This section discusses the circumstances that have lead companies to MDO.

During the previous decade the motivation behind product design has become increasingly customer focused. "More and more customers are asking for products with high functionality and aesthetic design." [5] In this consumer-centric market "the demands [on companies] for shorter time-to-market and designing a product right-the-first-time are increasing." [6] In areas where a product has been well established, and where no new technologies are emerging to enable improvements, it can be very hard to eek out even the slightest improvements. Achieving improvement often requires that interdisciplinary interactions must be incorporated into the design of the product so that the result of more subtle design changes can be determined. [7]

To exploit the most potential for design improvement multidisciplinary interactions must be considered early in the design process. Otherwise modifications suggested by these interaction and their effect will become apparent only after it is too late for significant changes to be introduced. [7] For this reason MDO has emerged as the way to efficiently design highly complex, multidisciplinary systems with mutually dependent components and complex physical interactions. [8] MDO has become an incredible tool for industry.

2.1.2 Standardization

Companies are always trying to find ways to make their operations more efficient. One tactic that is increasingly gaining acclaim is standardization. Companies are striving to establish and enforce accepted procedures by which all employees work. By implementing standardization, the company's results and actions can be monitored, predicted, and repeated. By creating a system of standards by which all employees work companies aim to guarantee that everything is done according to best practices and procedures, a guaranty that is becoming increasingly difficulty to assure. By capturing and standardizing the design process, continual improvement can be achieved. [9] There is no standard method for documenting company standards, however the method employed can have a great impact on the easy of disseminating the information and enforcing the standards. [10] As companies become more global, and depend more on outsourcing, work standards and their method of employment become increasingly important in maintaining continuous improvement.

2.1.3 Mass Customization

Throughout most of human history every tool and product has had a unique, custom design. Craftsmen made each product personally. Only in the last few centuries has industry moved away from the craftsmen are toward mass production. With the introduction of the steam engine and new manufacturing methods a new era was ushered in where standard products were mass produced and costs were lowered. A new revolution is becoming available with the invention of the personal computer and the dawn of the information age. This revolution is the advent of mass customization. Mass customization is the ability to make use of the same procedures as mass production to turn out custom products. Mass production has the ability to please both industry and consumers by offering more product variants at lower costs.

Consumers can now obtain goods from a global market. In order for companies to remain competitive it is imperative that they keep their customers happy. Customers are looking for products that fit their personal needs. As needs vary, companies must have the flexibility to respond quickly to produce a variety of custom goods. [11] This ability can come through the use of mass customization.

Even while consumers are demanding more product variants they also are demanding lower prices. These previously conflicting demands are now becoming achievable. To achieve this "the low cost of mass-produced products is still essential, but it must be accompanied by products tailored to meet specific needs of various customers." [12] Mass customization is playing an ever increasing roll in bringing the low cost of mass production to custom products.

MDO, standardization and mass customization have the potential to change the face of industry. Through the use of these concepts companies obtain the power to continual achieve never before available increases in productivity, and quality. This potential however has not yet been achieved largely do to the difficulty of implementing these concepts in large companies. This thesis develops a knowledge management approach that makes the implementation of these concepts within industry's reach.

2.2 Mutual Contingencies

Companies are striving to implement multidisciplinary optimization, company standardization and mass customization. It is crucial that process knowledge management is improved to allow us to realize these goals. Managing the knowledge comprising a company's design experience has always been important. A company's design experience is the result of large investments over the company's entire existence. Historically, this valuable knowledge has been retained mainly through documentation standards. However, company initiatives are now requiring that the knowledge be retained in a way that makes the utmost use of computer tools now available. By improving process knowledge management these goals can be realized more readily, allowing companies to glean the most possible benefit from their design experience. This thesis presents an improved process knowledge management approach that makes implementing MDO, standardization, and mass customization achievable. This chapter discusses the crucial elements of the knowledge management approach, and shows how these elements relate to MDO, standardization, and mass customization, as discussed in the previous chapter. The crucial elements that make up this improved process knowledge management approach are:

- Design Process Capture and Automation
- Design Optimization
- Centralized Data Management
- Collaboration.

2.2.1 Process Capture and Automation

The initiatives of industry that have been discussed earlier are all dependant on design process automation. A product's design is made up of multiple steps that include generation of geometry, analysis procedures to predict the product's performance, the building of prototypes and their testing, and manufacturing. The steps involved in a product's design, the order they are accomplished and other related knowledge make up a design process. After a design process is defined, it is usually possible to automate most of it through the use of computer tools. Even the parts that cannot be automated can be initiated and monitored as part of an automated process. This section will discuss how the key emerging concepts for product design focused on by this thesis depend on design process automation.

The first key concept that depends on process capture and automation is MDO. MDO relies on the ability to automate engineering processes. Without automation it becomes unfeasible to perform the iterative design and analysis process that MDO requires to determine the sensitivity the design variables have with respect to each discipline. In researching literature on MDO it becomes evident that because a key feature of MDO is design-oriented analysis in each engineering discipline, it is desirable that an MDO framework be easily adaptable to a variety of existing analysis tools. [7, 8] MDO is dependant not only on an ability to automate a design process but also the agility to integrate vastly different analysis tools together to quickly capture and automate design processes.

Another key concept that relates to process capture and automation is standardization. Process standardization requires that a process can be defined and captured. To achieve continuous improvement "management attention should be directed towards creating sound processes since it is assumed that good results will follow". [9] When working toward improving product quality through standardization, it is important that a process be captured with the most detail and accuracy possible.

Mass customization is also a key concept that certainly depends on automation. It depends not only on process automation but on automating all aspects of a products lifecycle, such as supply-chain management, enterprise resource planning customer relationship management, and manufacturing.[12] Not until the design process is fully captured can it be seen to what extent the product can be customized. Automation of the process then allows for the product variants to be mass produced as requested. As shown in this discussion, process capture and automation is a crucial element of process knowledge management.

2.2.2 Design Optimization

In any design there may exist any number of free choices that are not limited by the design requirements. Design requirements may also involve sorting through multiple objectives to find the best design. These goals or objectives may include minimizing weight and cost, while maximizing strength and stability. Design requirements might also specify design constraints. To find the best design it is required that the free design choices be set at the best values. Design optimization is the method of determining the best design as easily as possible. Each of the three initiatives identified earlier depends on design optimization.

A key feature of MDO is disciplinary and system optimization methods. If a design is relatively simple and design variables affect the performance in an intuitive way, then an experienced engineer may be able to quickly choose the best design. But, “if the design variables are numerous and strongly interact and are all about the same in effectiveness, a formal mathematical optimization is the tool of choice for deciding how to change the design.” [7] Optimization is certainly the focal point and motivation behind MDO.

While standardization and mass customization do not rely on optimization, both can make use of it. “Small ongoing improvements can accumulate to an overall contribution to organizational performance.” [9] By implementing even a simple optimization loop as part of a standardized process over time those small improvements will make a difference. Design optimization must be linked to process knowledge management so that the knowledge can be used to its fullest degree.

2.2.3 Centralized Data Management

Vast amounts of data are created everyday. Computers have lead to the availability of so much data that this era has been labeled the information age. A thesis researching CAD-centric MDO was able to show that with the use of computers, thousands of CAD models representing different product variation can be generated by a single computer in a matter of hours. [13] So much data is available that becomes difficult to utilize all of it. To be useful data must be managed. Because of the importance of managing data many solutions are available. Just like anything in the universe, if data is left to its own devises, chaos and disorder will result. A company's data must be controlled in a central location so that it is not lost to chaos. In addition to combating chaos, the three initiatives identified earlier depend on centralized data management.

MDO can generate a lot of data, but it also relies on the availability of that data. Because of this, it "relies heavily on data base technology." [7] Large MDO setups greatly benefit from efficient data management. Before the data can be exploited in MDO, it needs to be extracted from existing, often multiple data sources, integrated in one data repository, validated and cleansed by removing or correcting corrupt values. "These steps take 60-70% of the time and resources of a typical [MDO] application project. Regardless of its importance, the task attracts little attention of the research community, being perceived as mundane and routine." [10] By managing all data in one centralized source and automating the MDO process, most of theses mundane tasks can be eliminated.

Standardization also benefits greatly from centralized data management. Standardized processes must be stored and managed in one single location. If this does not happen confusion will result as individuals try to sort out where to go to initiate and follow a standard work process. Managing standards in a centralized data vault brings “enhanced learning through the transmittal, accumulation and deployment of experience from one individual to another, between individuals and the organization and from one part of the organization to another.” [9]

Just as MDO and standardization can benefit from centralized data management, mass customization does likewise. Mass customization is a product of the information age and as such is information intensive. The success of mass customization is dependent on information accessibility. [12] Without centralized data management information is not as accessible as it needs to be. In implementing emerging concept for improving product design it is imperative that process knowledge be managed in a central location. This will also greatly enable collaboration.

2.2.4 Collaboration

Collaboration is the ability to work together as a team. The three initiatives identified earlier depend greatly on collaboration. Product design required that multiple teams in any location are able to work efficiently together. This section discusses the dependence that emerging product design concepts have on collaboration, starting with MDO.

MDO is based on the need to collaborate between different disciplines. These disciplines “must work in harmony to arrive at a consisted design relative to design intent.” [14] “Conceptual design issues at stake are highly interdisciplinary, and often involve collaboration from customers, designers, and engineers.” [6] “In order to coordinate activities of multidisciplinary design teams and to guarantee the interoperability among the different engineering tools, it is necessary to have efficient collaborative design environments.” [10] Any MDO framework that does not support collaboration is meaningless.

Tools used for standardization must be collaborative in nature. Every individual involved needs the ability to access and contribute to the standardization. Collaborative standardization provides “enhanced learning through the transmittal, accumulation and deployment of experience from one individual to another, between individuals and the organization and from one part of the organization to another.” [9] By giving every individual access to the standard it will be binding on everyone as it should be.

Mass customization is dependent on collaboration because products are created through a collaborative process. Mass customization must make this collaborative interaction occur as seamlessly as possible; otherwise too much time is wasted for it to be worthwhile. In mass customization “the company works directly with the customer to create a product that meets the needs of the customer.” [12] Without the ability to collaboration in a friendly and secure environment this interaction cannot occur.

The emerging concepts that are making a difference in industry, and particularly MDO, standardization, and mass customization depend not only on collaboration but also on all of the other elements mention in this chapter. It is crucial that an approach for process knowledge management that enables these emerging concepts include these crucial elements. The next chapter discusses previous solution for achieving this and shows that nothing is currently available that encompasses these contingencies.

2.3 Previous Solutions

Current strategies previously used to manage design process knowledge do not meet the needs of today’s companies. The main challenge has been to move toward greater collaborative capabilities. One researcher, Tinnsten, states that due to Internet growth, it is of interest to make use of the new opportunities for distributed collaborative computing. [15] Sobieszczanski-Sobieski and Tulinius found that multidisciplinary teams need to collaborate early on in the design so that creative ‘what if’ questions can be explored before design becomes frozen to changes. [7] In another article Sobieszczanski-Sobieski also explains that because of speed-of-light limitations on computer processing speed, complex computations need to make use of distributed concurrent computing in order to increase speed. [16] Having addressed these needs, many products are now available that allow for advanced management of design process knowledge. These products can be categorized under the following titles:

- Web Systems
- Agent Systems
- Federated Systems
- Integrated Systems
- PLM Workflow and Change Management Systems.

These products and research projects are all geared toward achieving similar goals though each takes a different route to achieve them and consequently, offer different results. This chapter has three goals: One, to discuss the ability of these solutions to address automation, optimization, centralized data management and collaboration. Another, to show the shortcomings and overall inability of the solutions to meet industry needs. And finally, to identify elements of these products that were built in this thesis. The first category to be discussed is Web systems.

2.3.1 Web Systems

Web-based design makes use of the Web's ability to combine multimedia data in order to publish design information to dispersed users. Systems based on the Web provide access to catalogue and design information, communication among design team members, and authenticated access to design tools, services and documents. Researchers have developed Web-based tools that utilize one or more of these capabilities. Web-based tools are generally coded using Java, but some make use of other languages. Two examples use Common Lisp and CORBA. As a side note, the PLM system used in this thesis makes extensive use of Java because it is robust, versatile, and enables the PLM system to be run on multiple platforms and within a Web browser. The Web is a great tool for supporting information access; however, for a concept design environment to be viable it needs to do more than simply support information access. In addition, it must support the complete integration of analysis and simulation into a design process. Web technology itself cannot satisfy these requirements. Between all of the current Web-based design tools, a large range of functionality exists, however, no one solution addresses more than one or two of the issues important to industry. The reason for this is that the Web only does not meet industry's needs. This section will discuss what has been accomplished through the use of the Web to achieve process automation, design optimization, and collaboration. Web-based solutions have been grouped together based on their attention to these areas and discussed according to these groupings. The first to be discussed are those with an emphasis on process automation.

Process automation is made possible within Web applications through the use of CORBA and ActiveX. These tools allow for programs to interoperate with multiple computers, operating systems or programming language. CORBA was used by Sony System Design Corp. to develop KA Framework. [17] KA Framework is a framework that focuses on engineering knowledge. Another project, Design for X (DFX) shell developed by Huang and co-workers uses ActiveX to allow for Web-based deployment of DFX tools. [18] DFX tools are custom tools originated by Huang and co-workers to make use of morphological charts.

Other Web-based tools allow for design optimization. WebCADET, designed by Rodgers and co-workers, uses Prolog to allow for Web-based deployment of their custom tool CADET. [19] CADET is a system that supports decision making by providing designers with feedback about alternative solutions by searching through design knowledge. This program is mostly a tool for supplying information; it does not actually automate an optimization loop.

While most Web-based design tools are for collaboration, only two are mentioned here. Zdrahal and Domingue used Common Lisp to develop WWDL [20], a tool for guiding designers around ongoing design dialogues. Other development has been implemented to make the transfer of design information easier. One of these is VRML, which is a neutral geometric representation used to display geometric models and make comments on the designs. [21] None of the Web-based tools offer the scalability and security required by industry.

2.3.2 Agent Systems

Agent technology may provide support to enhance the ability of Web technology. [6] The concept of using agents systems is used in this thesis by making use of workflow tasks in the PLM systems to provide the function of agents. Agent-based design is a loosely coupled network of problem solvers. They are engaged in active dialog with each other, working concurrently to solve problems that are beyond their individual capabilities. Agent technology has existed before the Web. According to one researcher, Parunak, agents are best suited for applications that are modular, decentralized, changeable, ill structure, and complex. He notes that agents fit into the current industrial trend towards products that are continually more complex and diverse, as well as toward increased product variety over time. [22] However, in analyzing these projects Wang et al. found that agents alone cannot solve the collaboration challenge. A possible solution would be to build a Web environment that will make the designer\agent\server interaction successful through the integration of related emerging technologies, including agents. [6] The following sections show what has been done to implement agent-based tools. These tools are discussed in the same manner the Web-based tools. That is to say, the tools are discussed grouped together according to their emphasis on process automation, design optimization, and collaboration, but also with centralized data management included as an additional grouping. The discussion is begun with process automation.

One of the earliest agent-base tools is PACT. [23] PACT includes a federated architecture using wrappers for legacy system integration and automation. This tool was useful as a proof-of-concept, and as a starting point for agent technology, but did not extend much beyond that.

Some agent-based design tools are made for optimization. One is A-Design which combines the aspects of multi-objective optimization and automated design synthesis. [24] It is of particular note because it is the best attempt to make an agent system function as a framework. It does not however have collaborative capabilities.

Another project, Concept Database, is interesting because of its use of agents to provide strategic design support for version control, workflow management and information gathering. [25] This program attempts to recreate a PLM environment by using agents. It also includes a limited framework tool. This thesis and Concept Database are similar but opposite in that it strives to add as PDM system to a framework tool, whereas this thesis strives to add a framework tool to a PLM system.

The last agent-based tool to be discussed focuses on collaboration. This tool is called SHARE. It uses a federated architecture similar to PACT. [26] It entails the development of open, network-oriented environments for concurrent engineering using email. This tool is chiefly a collaboration tool, and like all of the other agent tools does not provide a broad enough range of functionality to be used in industry.

2.3.3 Federated Systems

Under federated systems the one must prevalent is FIPER (Federated Intelligent Product EnviRonment). FIPER by Engineous is an interesting tool because it addresses the need for collaboration and framework functions. FIPER was part of a four year project co-sponsored with $21.5 million by the National Institute for Standards and Technology (NIST). “FIPER has a web-based, distributed design and integration infrastructure that allows organizations to access execute and reuse design tools and processes. Design teams may be work groups inside an organization or may be part of a global geographically dispersed network of partners.” [27] While FIPER is an exciting new tool, it creates a conflict for the large companies that are geographically dispersed. This conflict results from the fact that FIPER is not a PLM tool. Furthermore it does not interoperate with PLM tools. Large companies with *globally geographically dispersed network of partners* rely on such PLM tools. Because FIPER does not interoperate with PLM tools and offers some of the same functions as a PLM it contains duplicate structure that must also be supported by the company. Companies that are already relying on PLM systems will not be able to use any of their huge PLM investment with FIPER. Companies rely on PLM capabilities that are not available from FIPER because it is not a PLM tool. FIPER then does not become a part of the company’s collaborative tools, and is used just as its predecessor – iSIGHT. This section discusses the functions of FIPER and illustrates the duplicate structures that must be in place for a company to use it and a PLM tool.

To provide process automation, FIPER contains a workflow manager. This manager “directs the sequence of design events assembles components and controls the dataflow between steps in the design process.” [27] PLM tools also contain a workflow manager. For a company with both PLM and FIPER the question arises as to whether to use the PLM workflow or FIPER’s workflow. Again, PLM is the bigger fish and wins the debate. FIPER requires an application server to enable the collaborative process automation. PLM tools also have an application server for this purpose. FIPER’s process automation is a duplicate structure of what is available in the PLM system.

FIPER includes the ability to provide central data management. It does this by allowing for “components and data from intermediate analysis [to] be stored in a commercial back-end database.” [27] PLM tools already make use of a database for storage of data and components. FIPER can use the same database installation that the PLM tool uses, but both database usages must then be supported by the company.

For collaboration “FIPER B2B protocol allows for secure sharing of models in a federated environment.” [27] PLM tools also allow for secure sharing of models. Again, an overlap occurs and this one presents a potential security risk, because while both protocols are secure, two protocols are less secure than one. FIPER is a viable and wonderful solution for companies that do not plan to ever implement a PLM system, but for its power does not become utilized in companies that use PLM.

2.3.4 Integrated Systems

The previous sections have all discussed solutions that focus on solving the issues presented through use of tactics such as the Web or agents. However, it has been discussed that through use of these tactics no one tool has been developed addressing all of the issues presented in this thesis. This section discusses research done to integrate multiple tools together to achieve this broader goal. These research projects are those of WebBlow and research done by Klaas et al.

2.3.4.1 WebBlow

Wang et al. has done considerable research to find ways to solve the issues dealing with collaborative concept design. Through their research they have found that neither Web-based nor agent-based tools have the ability to achieve collaborative design. Based on their findings, they addressed the challenge with an integrated approach. Their approach was to develop a distributed multidisciplinary design optimization (MDO) environment called WebBlow. [28] This project strove to integrate the Web with agents in order to automatically access and manipulate information while enabling seamless interaction between designers, agents, and servers. While the application was initiated for blow molding applications, the methodologies and system architecture is extendable to any application where collaborative and distributed MDO is required. While this tool is a large advancement in distributed MDO, it does not offer the robustness and security of a PLM tool. This section will discuss the novel elements of the project as related to the issues of this thesis.

The major work includes developing a Web-based user interface for design and implementation, agent-based computing resource management. Through the Web users can setup the design process and monitor the progress. The agents allow for the automation of the process. This novel setup allowed for distributed processing.

WebBlow made heavy use of XML as a means to transfer data between agents. The system relied on this XML-based data management to store the data and distribute it to the various elements of the tool. User input, as well as analysis results were all stored using a central XML-based data management system.

Information was passed between the user interface and the several agents through use of the XML files. Because of the ease to which XML can be transmitted over the web collaboration was greatly improved over other tools.

2.3.4.2 Klaas

Research conducted by Klaas et al. was the most relevant to this thesis. This thesis will build on their research to embed numerical analysis capabilities into an enterprise-wide information system. [29] Their product is still under development, but the ideas shared in their reports are very insightful. Some of what they mention is related to improving PLM elements that already exist and others are elements that must be added to PLM. As of date, no literature is found that addresses these areas of future work. [21] This thesis will build on their work by developing the areas that they identified as future work

Klaas et al. stresses the point that in order to effectively use numerical simulation in product designs, an automated simulation environment must interact seamlessly with product data management (PDM) and workflow systems. Workflow management can be used to coordinate and automate the execution of processes. Along the same lines, commercial CAE and legacy tools must be integrated such that these managers can provide them with needed input data based on the problem description, and also to transfer results back into the PDM system. Klaas additionally mentioned the need for development of generic automatic simulation models.

They state that by making use of an enterprise’s PDM system, data redundancy can be eliminated, revision control will be provided, and accessibility and security will be guaranteed. Klaas identified information structures and management as two areas for future development. Management needs to be initiated so that problem description data modifications that become apparent during the simulation may be directly stored and retrieved into the PDM. 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.

Collaborative engineering crucially depends on up-to-date data. By using the PLM system this need of collaborative engineering will automatically be supported. The workflow can be monitored to track progress and identify bottlenecks. A workflow process can be either initialized by a direct request or automatically based on the need to update parameter estimates due to an upstream design modification.

2.3.5 PLM Workflow and Change Management

PLM workflow and change management are the current tools used in PLM systems for a company to manage the processes taken by employees in everyday work and when changes need to be made to products. While these tools are used to automate these processes and allow for collaboration in a global company, they do not provide the functionality to support the issues presented in this thesis.

Workflow and change management maintain a passive role in the automation of processes. They rely on users to perform the specific tasks described in the process. They automate the process by automating the assigning and notification of tasks to be performed by participants. The automation needed by MDO, and mass customization is an active automation. The tasks in the workflow management must perform their assigned duties themselves. Currently standard workflow and change management do not provide this support. Additionally, both of these tools are much too rigid to be useful in MDO and mass customization. It is important in concept design to be able to easily update and change the automated workflow, but the standard tools require that only system administrators can edit or design processes.

Workflow and change management do not provide design optimization. Never before this thesis has an optimization loop been implemented within a PLM system. This is the major shortcoming of workflow and change management for MDO.

CHAPTER 3: METHOD

Currently, no solution for advanced management of design process knowledge achieves all the needs of a company to realize multidisciplinary design optimization, standardization, and mass customization. While all have strengths, none can offer the complete solution that an integration of product lifecycle management and a framework can offer. The concept design needs of PLM can be met in a large part by the capabilities of a framework. Likewise, the collaborative and distributed computing needs of frameworks can be met in a large part by the capabilities of PLM. In this thesis framework capabilities will be added to PLM because PLM lacks less than what frameworks lack. Also, it is expected that companies will already have a PLM system. Because of this, to add a framework with collaborative capabilities to a company's suit of software tools would mean that there would be unneeded overlap in software. Therefore, PLM is a bigger part of a company than framework software it is concluded that is it better to add framework functionality to a PLM system.

The method used to develop this tool is the major contribution of this thesis. Figure 5 show a diagram of the major parts that must be developed. These pieces are:

- Customizing a central repository to support the framework integration.
- Creation of a process automation or workflow module.
- Integrating generic automation models.
- Integration of design optimization.
- Linkage into a centralized enterprise wide data.

The following sections describe the pieces that must be created in developing these major parts. After these areas have been described a description of the method use to test the feasibility of the concept is included.

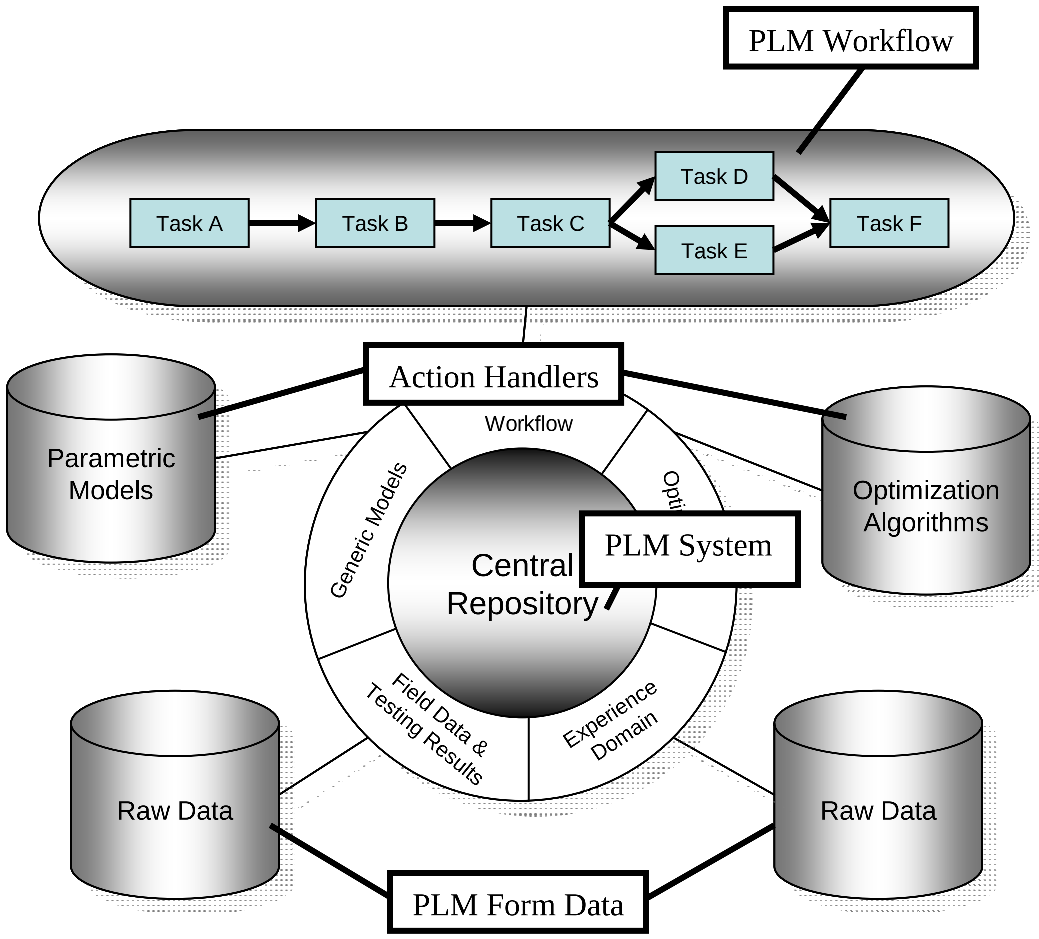


Figure 5 A representation of the knowledge management approach presented by this thesis. Knowledge is stored in an organized form that can be linked to automated design processes.

3.1 Design Process Automation and Optimization

Process automation was achieved within a PLM system by customizing workflow management. There are two ways that workflow was customized to achieve process automation. These two tactics were one, the internal method, where framework capabilities were added completely internal to the PLM system and two, the external method, where PLM workflow was customized to integrate external framework tools.

Both tactics have specific strengths and differing application for different situations. These strengths will be discussed in the results chapter. The implementation was made using the PLM system Teamcenter. The concepts and code written is also applicable to other PLM tools. All that would need to be changed for different tools is the data transfer functions to be updated for other tools and the action handlers to be registered according to the other tool's documentation. The method for creating both the internal and the external tools are now described

3.1.1 Internal to PLM

As discussed in the review of research conducted by Klaas et al. it is possible to provide framework capabilities to a PLM system. This section will discuss how the internal integration was developed. The main elements needed in this development are:

- Automation Modules.
- Data Mapping.
- Design Optimization.
- User Interface.

3.1.1.1 Automation Modules

The nature of PLM workflow tasks was changed by building each automation module right into the task. Standard workflow tasks are passive. They rely on users to perform what is assigned. Once the user indicates that they have performed the task, the workflow then moves to the next task. Active workflow tasks must be integrated into the workflow. A task itself will perform its assigned duty and then the next task will be initiated. This customization of Teamcenter Engineering Workflow is accomplish through the use of Teamcenter's API, the Integration Tool Kit (ITK), and makes use of ITK's Engineering Process Management (EPM) functions. Dynamically linked libraries using EPM functions can be linked to standard Teamcenter libraries. These functions allow action handlers to be registered to Teamcenter. Action handlers are the actions that can be assigned to tasks in the workflow. Registered custom action handlers can be assigned to workflow tasks in order to control their behavior. In this way, the workflow can be customized to include any action that a developer can program. Using the workflow, empowered by data mapping and custom action handlers, process integration and automation is possible. Among others, action handlers can be made to update parametric CAD models, to mesh and analyze the models and to optimize the performance of the product. The feature tree of this integration for the test case is shown in Figure 6. Several generic modules are required for design process automation. These modules are:

- Geometry Update/Creation.
- Mesh Generation.
- Analysis.

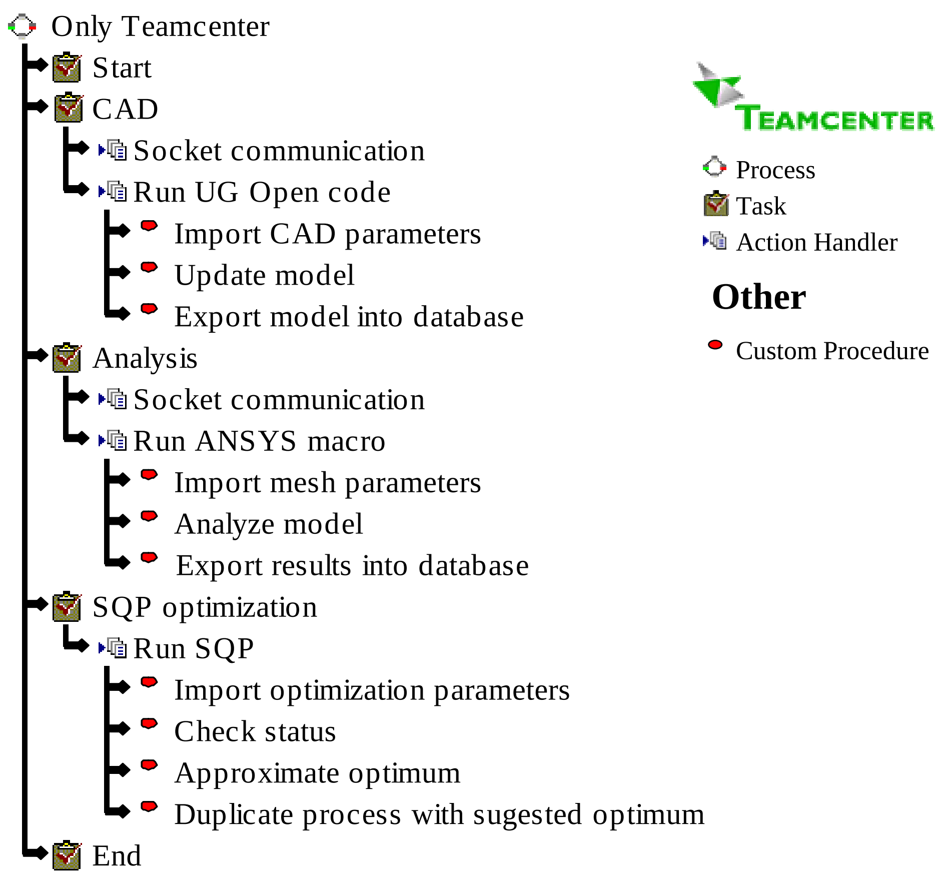


Figure 6 The feature tree for directly integrating framework capabilities into Teamcenter.

3.1.1.1.1 Geometry

A generic geometry module was needed to create the ability to update a CAD model with data taken directly from the PLM database without any intermediate files or steps. The development of this module was divided into two main parts – the ability to import data to the model and the ability to update the model based on that data. The ability to import data was achieved through the use of a socket. A socket consists of a server and clients that can transfer data between each other. The server listens on a port for data transferred from a client. When data is transferred, the server program can then use the data and communicate the results back to the client. The information transferred over the socket can be formatted in such a way that allows requests for information retrieval and storage to be made. With the server processing requests via ITK functions, a client can issue request to the server and make use of the returned output in any functions. A socket was required for the CAD automation because Teamcenter header files and libraries conflict with those of the CAD program, Unigraphics. This means that ITK functions cannot be used in the same dynamic linked libraries as functions from Unigraphics' application program language, UG Open. The use of a socket also makes the program generic so that if the module is to be used in another PLM tool, only the server side changes and the client can remain the same.

The second part of the module, updating of the geometry, was fairly simple once the data was available. In the CAD automation the client can request the needed parameters and the relevant parametric part file, then using UG Open functions it can update the part with the new parameters.

3.1.1.1.2 Mesh Generation

Mesh generation can be achieved in multiple ways. The creation of a generic mesh generation routine is out of the scope of this research. Development of a completely generic mesh generation tool is a major undertaking that has not yet fully been realized by researchers. This module is very important to a design process and the creation of a generic module would greatly simplify the move between geometry and analysis, however a generic module is not required because code can be created on a specific case by case basis. For this reason, code was made to generate a mesh for the analyzed geometry. Inputs to the routine were commutated from the PLM database by the use of a socket, as discussed in the previous section. With the geometry updated and meshed the only module remaining was analysis.

3.1.1.1.3 Analysis

The ability was created to import and export analysis data directly to and from the PLM database. To do this an action handler was assigned to perform an analysis and transfer data by the use of a socket. The analysis software used in the test case was ANSYS. Macros to perform the analysis can be made using ANSYS's macro language. The macros can link to Tool Command Language (Tcl) code that can communicate over the Teamcenter socket to request analysis parameters such as mesh information, and boundary conditions. The analysis is then performed and results are communicated back to Teamcenter over the socket. Each of the modules for design automation required that data be transferred from the PLM database to the application code. Extracting data from the database and mapping that data between the modules are important issues that are preformed in the background during the automation; the next section presents how these issues were addressed.

3.1.1.2 Data Mapping

Each of the modules created to perform automation require data. This section presents the issues involved with supplying this data so that PLM benefits can be realized. These are discussed in the following sections:

- Using the PLM database
- Extracting and Importing data from the database
- Mapping data to and between modules.

3.1.1.2.1 Database

PLM systems store data in a database. Everything that is stored in the database by the PLM system is then managed by the PLM system. Because it is desired that all data used in design be managed by the PLM system so that it can be part of the PLM advantages. This section describes how to access data in the PLM managed database.

It has been found that it is not effective to directly access the PLM data in the database. [30] This statement is illustrated in Figure 7. Although it is possible for to directly access the data using SQL, the process contains risk because full knowledge of how the PLM system sets its data would need to be known, and such information is not available. For instance, to change one value in the database interactively through the PLM system may change the data in over ten tables in the database. If one tried to do this and neglected to update one of the tables that the PLM would update, the database could be corrupted and all data may be lost. Hence, it is more efficient to let the PLM system update its own data. By using this process data may be accessed for use.

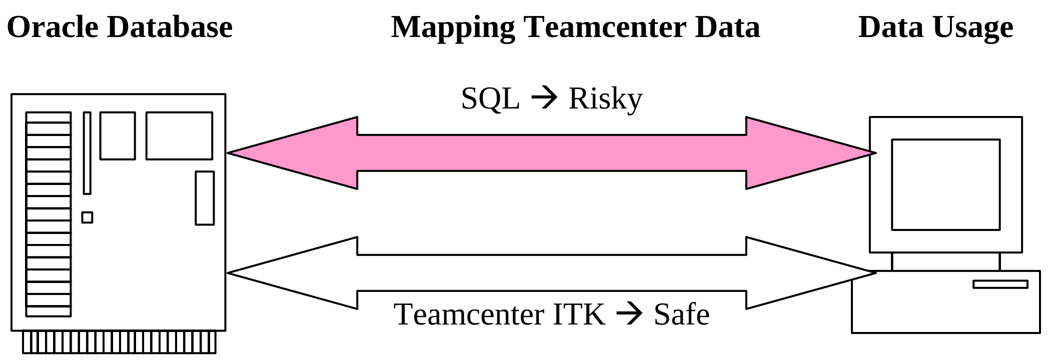


Figure 7 Mapping Teamcenter data directly through SQL is risky. It is safer to use Teamcenter ITK functions to retrieve and store data from the database.

3.1.1.2.2 Extraction and Insertion

The internal method allows all data to be used internal to the PLM system. As discussed earlier PLM data should be accessed only through the use of the PLM system. The methods for doing this are different for each PLM system and can be found in that system's documentation. For this implementation, Teamcenter's ITK includes the needed functions that allow a programmer to access data through Teamcenter. Functions used can be found in the documentation as part of Teamcenter's persistent application and workspace object memory (POM/AOM/WSOM) functions. Through an intricate use of these functions data can be accessed and stored in the database.

The data structure can be set up interactively or programmatically using the ITK. To do this, Teamcenter provides information classes and forms. Information is held in the classes. Forms allow a set of that data to be interactively viewed and changed. The data structure is illustrated in Figure 8. Interactively or programmatically, classes and forms accessing data in the classes must be created, and data must be put into the classes. If this is being done programmatically, forms and classes must be saved and unloaded from memory to the database. Teamcenter handles the saving and unloading of the forms and classes if they were created interactively. Once the data is stored it is available to be accessed.

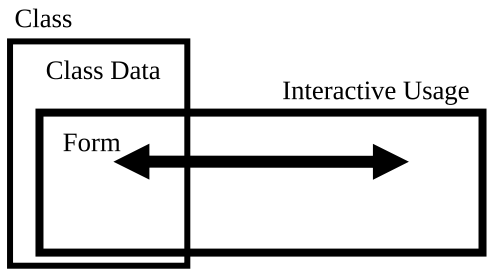


Figure 8 Teamcenter memory structure. Classes store metadata, and forms allow a set of that data to be accessed interactively

When programmatically accessing the data, these classes and forms must be loaded into memory from the database. Most memory allocated by ITK functions to access data must be freed using the appropriate methods as specified in the documentation. Working with ITK allocated memory can lead to unpredictable results. To avoid this, it is suggested to follow the subsequent procedure to deal this with dynamic memory. First, manually allocate memory. Then, retrieve the data using the ITK function. Immediately following this, copy the data into the manually allocated memory, and free the memory allocated using an ITK function. Finally, free the manually allocated memory when through with the data. This process is shown in the following code where an array of doubles is retrieved from the database:

```
//manually allocate memory
double* my_doubles = new double[length];

//created temporary pointers for the ITK function
double* tmp = 0;
unsigned char *junk1;
unsigned char *junk2;

//ITK function to retrieve into "tmp" an array of doubles from positions
// 0 to "length" in the "attr_id" field of the "class_instance" class
POM_ask_attr_doubles ( class_instance, attr_id, 0, length, &tmp, &junk1,
&junk2 );

//assign the data in "tmp" into "my_doubles"
for(int i=0;i<length;i++) {
   if(junk1[i] || junk2[i])
         my_doubles[i] = 0;
   else
         my_doubles[i] = tmp[i];
}

//immediately clean up ITK allocated memory
SM_free(junk1);
SM_free(junk2);
SM_free(tmp);

/***add code here to make use of "my_doubles"***/

//manually clean up memory
delete [] my_doubles;
```

After the data is copied into personal allocated memory, the ITK allocated memory must be freed immediately. This is due to the unpredictable behavior that can occur when there are multiple variables containing memory allocated by ITK functions that are still in memory. Figure 9 shows this problem and the suggested method for managing dynamically allocated memory. Care must also be taken in loading, saving and unloading data from the database. Lund has made good suggestions on the procedure used to make data mapping robust when accessing data from the database. With the ability to access data from the database available, other functionality will be developed to use the data.

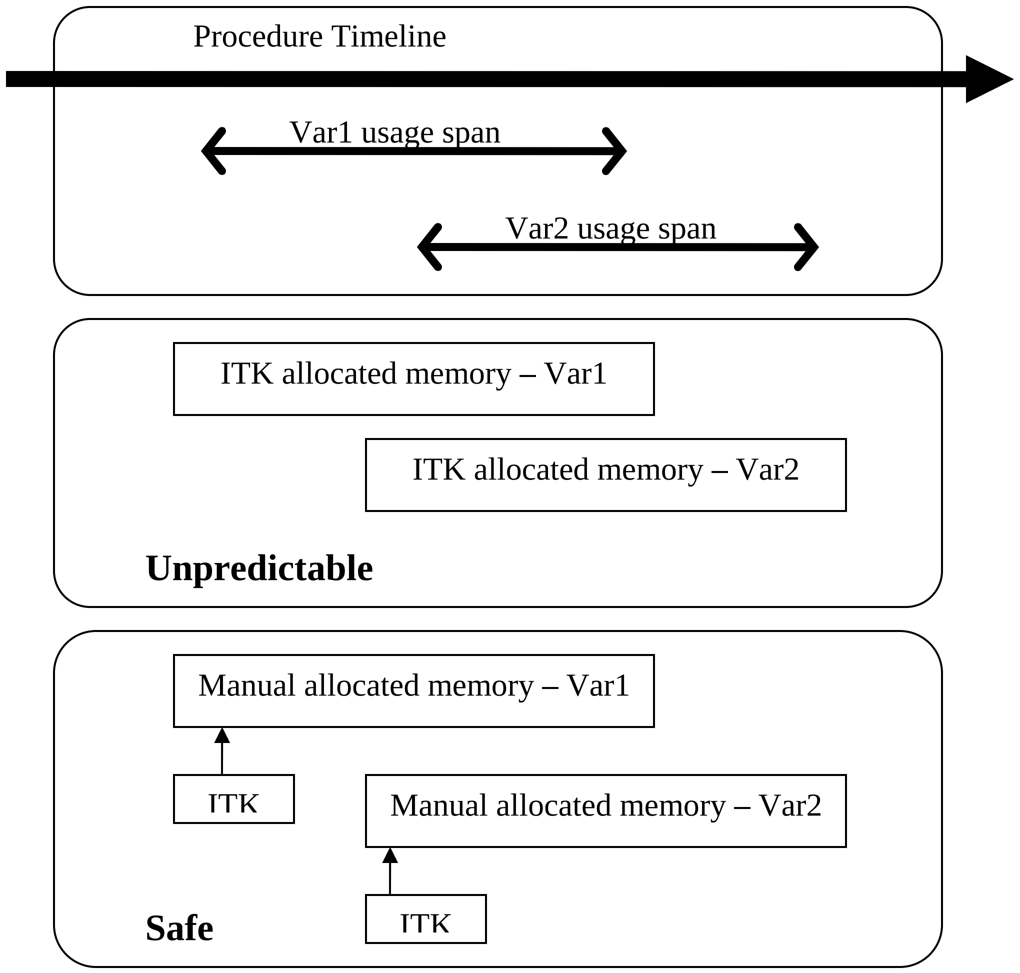

Procedure Timeline
Var1 usage span
Var2 usage span
ITK allocated memory – Var1
ITK allocated memory – Var2
Unpredictable
Manual allocated memory – Var1
ITK
Manual allocated memory – Var2
ITK
Safe

Figure 9 Overlapping ITK allocated memory can cause unpredictable behavior. It is better to use ITK functions to feed manually allocated memory.

3.1.1.2.3 Management

When running an automated process, data needs to flow between the different modules. There is no mechanism built-in to a PLM to associate data to specific tasks, and make the data available to the task itself to use. Data mapping capabilities must be integrated into PLM workflow. In his ongoing master's research at Brigham Young University, Lund has made large contributions to developing the ability of data mapping in a PLM system. A small portion of his method will be discussed, but a more in depth discussion can be found in his thesis work. [30] The ability to manage data mapping in PLM workflow was created by building on current PLM methods. Teamcenter engineering allows for data to be attached to a process. As tasks are performed, users can manually access the attached data. Data mapping for an automated process was created by attaching a folder to the process. An argument passed into the action handler of each action requiring an input specifies the name of a form containing input data for the action. When the action is run it will look for the form in the attached folder. In the same way outputs were stored in the folder. Through this process, data mapping was accomplished between modules.

Other than mapping data between modules another type of data management for attributes was required. Attributes are parameters of a geometric feature that are not associated with geometry dimensions. Examples of attributes are mesh density, load conditions, boundary conditions, material properties, and geometry names. A generic system needed to be created allowing attributes to be assigned and continually associated to the specific geometric features as they move from one analysis to another. The work of creating a generic management system is beyond the scope of this thesis, but for implementing the test case, code was written in the ANSYS macro to accomplish attribute management for this specific case.

3.1.1.3 Design Optimization

A PLM system has no built-in ability to automatically make decisions to improve a design. Decision support within PLM requires the integration of optimization algorithms. This section discusses the issues concerned with embedding an algorithm into a PLM system and the advantages associated with it. A discussion of the generation of new algorithms or researching and comparing the efficiency of possible algorithms is not included and is not part of the scope of this thesis.

3.1.1.3.1 Algorithms

The optimization algorithm used is a sequential quadratic program (SQP). SQP algorithms are described in optimization literature. These algorithms find the gradient at a point and find the optimum point of a quadratic function based on the gradient. They then move the design to that optimum point and repeat the process. Once the optimum point remains fixed within a tolerance, or a maximum number of iterations are reached, the point is said to be the optimum and the algorithm is terminated. The SQP algorithm can be summarized in the following equations. [31] When solving the general problem

$$\text{Min} \quad f(\vec{\mathbf{x}}) \tag{3}$$

$$\text{s.t.} \quad g_i(\vec{\mathbf{x}}) - b_i \geq 0 \quad \text{i = 1, … , n} \tag{4}$$

$$g_i(\vec{\mathbf{x}}) - b_i = 0 \quad \text{i = n+1, … , m} \tag{5}$$

the quadratic approximation at point $\vec{\mathbf{x}}^{k+1}$ is:

$$\text{Min} \quad f_a(\Delta\vec{\mathbf{x}}) = f(\vec{\mathbf{x}}^{k+1}) + \nabla f(\vec{\mathbf{x}}^{k+1})^T \Delta\vec{\mathbf{x}} + \frac{1}{2}\Delta\vec{\mathbf{x}}^T \nabla_{\mathbf{x}}^2 L(\vec{\mathbf{x}}^{k+1}, \vec{\lambda})\Delta\vec{\mathbf{x}} \tag{6}$$

s.t. $g_{i,a}(\Delta\vec{\mathbf{x}}):$ $\quad g_i(\vec{\mathbf{x}}^{k+1}) + \nabla g_i(\vec{\mathbf{x}}^{k+1})^T \Delta\vec{\mathbf{x}} \geq b_i$ i = 1, … , k (7)

$g_i(\vec{\mathbf{x}}^{k+1}) + \nabla g_i(\vec{\mathbf{x}}^{k+1})^T \Delta\vec{\mathbf{x}} = b_i$ i = k+1, … , m. (8)

The values of $\Delta\mathbf{x}$ and $\vec{\lambda}$ are computed by solving:

$$\nabla f_a(\Delta\vec{\mathbf{x}}) - \lambda \nabla g_a(\Delta\vec{\mathbf{x}}) = 0 \tag{9}$$

$g_a(\Delta\vec{\mathbf{x}}) = 0$ for i=1, … ,m. (10)

The Lagrangian Hessian matrix, $\nabla^2_{\mathbf{x}} L$ of the first iteration is the identity matrix but otherwise is approximated using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update:

$$\mathbf{N}^{k+1} = \mathbf{N}^k + \frac{\Delta\vec{\mathbf{x}}^k (\Delta\vec{\mathbf{x}}^k)^T}{(\Delta\vec{\mathbf{x}}^k)^T \vec{\gamma}^k} - \frac{\mathbf{N}^k \vec{\gamma}^k (\vec{\gamma}^k)^T \mathbf{N}^k}{(\vec{\gamma}^k)^T \mathbf{N}^k \vec{\gamma}^k} \tag{11}$$

where the Lagrangian Hessian matrix to be found is $\mathbf{N}^{k+1}$, $\mathbf{N}^k$ is the previous iteration's Lagrangian Hessian matrix, and $\vec{\gamma}^k$ is found in using the following equations:

$$\vec{\gamma}^k = \nabla L_{\mathbf{x}}(\vec{\mathbf{x}}^{k+1}, \vec{\lambda}) - \nabla L_{\mathbf{x}}(\vec{\mathbf{x}}^k, \vec{\lambda}) \tag{12}$$

$$\nabla_{\mathbf{x}} L = \nabla f(\vec{\mathbf{x}}) - \sum_{i=1}^{m} \lambda_i \Delta g_i(\vec{\mathbf{x}}) . \tag{13}$$

3.1.1.3.2 Paradigm

As stated earlier, a PLM system has no built-in ability to automatically make decisions to improve a design. Embedding an optimization algorithm into PLM required the use of a new optimization paradigm. Current optimization algorithms perform assuming that the optimization is the overarching program. It supplies the inputs as well as initiates, monitors, and retrieves the output from the automation. It makes decisions about input changes and analyzes the outputs in order to find the optimal design. All tasks are performed optimization-centric. The new paradigm is PLM-centric. 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 the outputs with respect to the inputs and suggest an improved design by instantiating a duplicate of its own process. This paradigm lends itself well to parallel and distributed processing. If multiple design processes are spawned by the optimization task, each of these processes would then run parallel. A genetic algorithm could spawn an entire generation of design processes to be run simultaneously. Likewise, a gradient-based algorithm could at once run all of the processes that are needed to approximate the gradient.

3.1.1.4 User Interface

The internal method allows for collaborative user interaction. It offers an unprecedented ability to setup, monitor and visualize results of an automated optimization process within a completely collaborative environment. An important part of any application that requires human interaction is a user interface. User interface design has a great impact on both the usability and the user perception of an application. In general if a program works like a charm but has a poorly designed interface, it will not be appreciated to an extent anywhere near it’s potential. This section discusses the interfaces needed in the customization to integrate a framework internally in a PLM.

3.1.1.4.1 Setup

An automated optimization is setup through the use of PLM’s standard workflow designer. As mentioned earlier, workflow is a rigid tool and as such, the workflow designer is meant for seldom use by administrators. While it is usable and performs the function needed, it does not allow for the agility required because this interface was not meant for high traffic use. Future work should include the creation of a more usable interface. Suggested changes would be to have the action and rule handlers grouped in categories specific to their function and have them be selectable by icons. Also the interface should be created so that each action handler’s arguments can be seen with a description of their potential values included. Last of all, it needs to be editable in a way that would allow engineers to access to it without requiring administrative abilities.

3.1.1.4.2 Dashboards

The current PLM workflow dashboard was leveraged for use as an unprecedented tool for monitoring the progress of automated optimization processes. Dashboards allow the status of the process to be monitored by users and managers. PLM workflow has a built in dashboard to monitor this progress. Anyone can view the process dashboard and quickly surmise the progress of the process. This dashboard may be viewed by anyone with permission in any location.

3.1.1.4.3 Visualization

Data and results of the process can be viewed though normal PLM methods. This is more advanced for CAD models, where lightweight images can be quickly viewed. However other data forms can only be viewed as pure data or by using an external tool. Work needs to be done to create visualization tools for viewing optimization results, and other data forms within the PLM system.

3.1.2 External to PLM

The second methodology is to integrate an external framework tool into the PLM system. Through implementing this tactic on the test case it was found that the external method is best suited for applications where agility and ease of use are required. Because a commercial tool was used, this tactic provided more support and a better user interface, making it much more usable as shown by the low score received for the number of required specifications that needed as shown in Table 3. The following sections support these conclusions as they discuss how this tactic was developed and the findings obtained when implementing the test case. The two main areas of development are:

- Executing the external application from within the PLM system.
- Linking the external run to the PLM such that all inputs and results can be access in the PLM system.

3.1.2.1 Execution

PLM workflow management was used to integrate iSIGHT into the PLM system. iSIGHT was chosen because of its wide acceptance in industry and because of the ease at which it was integrated into the PLM system. Additionally, iSIGHT has only the functions needed by a framework with few other collaborative or data management functionality that would overlap with and cause redundancy with PLM functionality. A standalone version of FIPER was considered, but not chosen because at the time and now, though advertised to be available, no standard supported application programming interface is available.

To accomplish the workflow customization to run iSIGHT, a new custom task action handler was made to export to the user's local computer an iSIGHT description file specified in the handler's arguments and included in the folder attached to the process. It then makes a system call to run iSIGHT in batch with the locally stored description file. The action handler then stops the progress of the workflow process until the iSIGHT optimization is completed. The handler is notified of the completion of iSIGHT by continually looping on a one-second pause until a file is written to by the iSIGHT run signifying its completion. Once the optimization is completed the handler continues the process' progress. In this way it is possible for the iSIGHT run to be included as a task in a larger workflow process.

An iSIGHT run was created that involved creating an executable to read in CAD parameter from a file, and update and save a parametric CAD model using the parameters. The development of an iSIGHT simcode was also necessary to wrap the executable and map the input parameters into the file that the executable will read. An ANSYS macro was made to read the mesh parameters in from a file. Also, an iSIGHT simcode similar to the previous one was created to map the mesh parameters into the mesh parameter’s file. These simcodes were implemented to run in sequential order. An optimization loop was also specified that matched the loop implemented in the internal method as explained above. The feature tree for this methodology including both Teamcenter and iSIGHT components are shown in Figure 10.

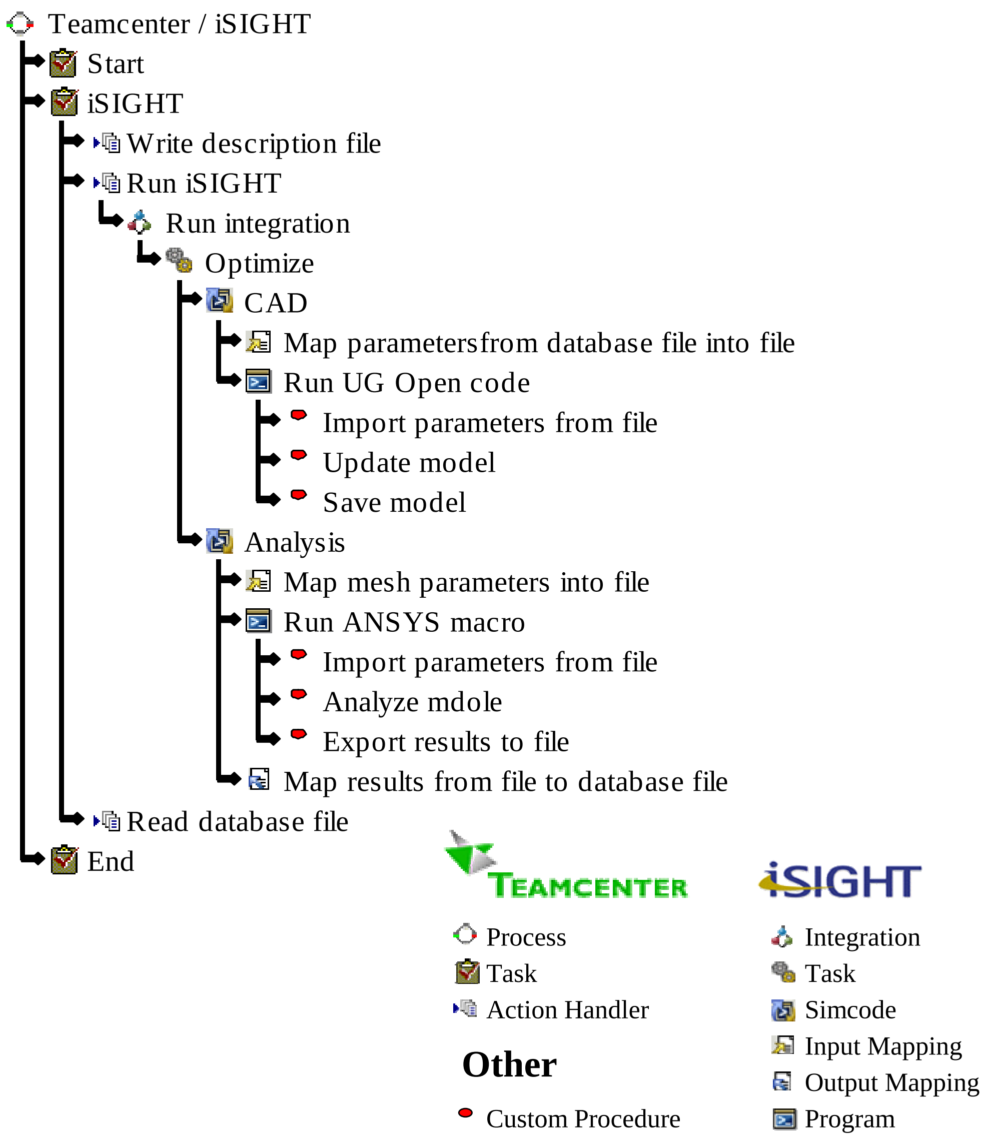


Figure 10 The feature tree for integrating iSIGHT into Teamcenter Engineering Workflow.

3.1.2.2 Linking

Linking the iSIGHT data to the PLM system was implemented by using standard file parsing procedures. iSIGHT uses an ASCII formatted description file to store the preferences and input information to be used in the iSIGHT run. iSIGHT stores its run data in an ASCII formatted database file. In order to supply iSIGHT with necessary information and to put data back into Teamcenter, the action handler was made to parse the description before iSIGHT was run to supply it with user setup information. Then, once iSIGHT completed, the action handler continued to parse the database file for results and store them in the PLM database.

3.2 Centralized Data Management and Collaboration

Centralized data management is achieved through the use of the PLM system. All data transactions by both the internal and external methods have used PLM methods to transfer and store data. Because of this, all of the data is automatically managed by the PLM system. PLM systems also have collaborative ability built in. That is the beauty of using a PLM system. Everything done by the framework integration is securely accessible though the entire enterprise because of the use of a PLM system.

3.3 Test Feasibility

To test the feasibility of the concepts those have been developed. The concept is deemed feasible if its objective is meant. The objective as stated in the Introduction is that it can perform a design optimization based engineering results obtained from an automated processes involving geometry creation and analysis. The test case and comparison metrics devised to prove the concepts are discussed in this section.

3.3.1 Test Case

A simple design process was created as a test case used to compare the two methodologies. The elements of this process were created using currently available parametric procedures. As illustrated in Figure 11, the process will consist of a parametric CAD model of an I-beam with inputs of height (h), width (h), web thickness (b), and flange thickness (l). The analysis was a simple stress analysis of a cantilever beam under a bending load with inputs of mesh density and load, and outputs of maximum displacement (δ_{max}) and weight (f). The optimization minimized weight while keeping the displacement under a critical amount. The problem was posed as:

Minimize:

$$f(\mathrm{h,w,b,l}) \tag{1}$$

Subject to:

$$g = \delta_{critical} - \delta_{max}(\mathrm{h,w,b,l}) > 0. \tag{2}$$

The design variables were height, width, web thickness and flange thickness. This process was automated and evaluated using the Teamcenter customized workflow. It was also automated using an iSIGHT automation that was then integrated into Teamcenter. The comparison metrics were used to evaluate the two methods on the basis of usability, robustness, and easy of implementation and quality of results.

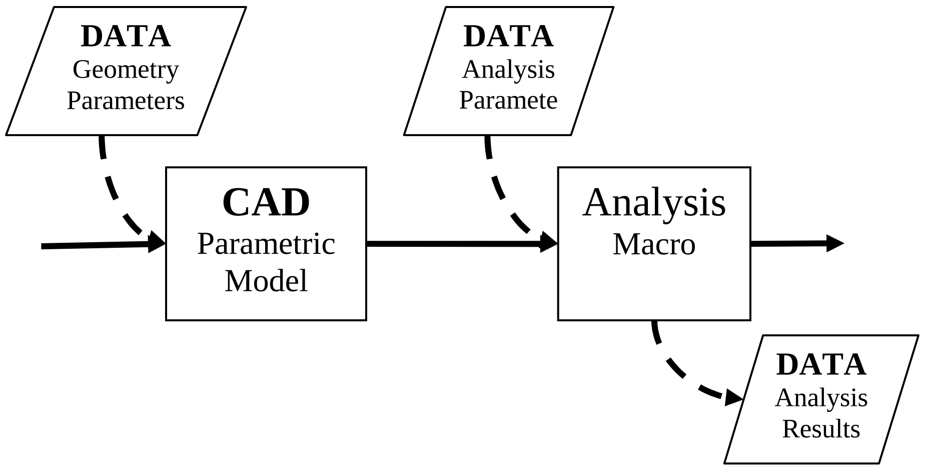


Figure 11 Simple design process used to demonstrate framework integration.

3.3.2 Comparison Metrics

The two tactics were compared using the following metrics. Four weighted criteria were measured and then the results were added. The weights used to scale the criteria were chosen so that the metrics better represented the performance of each methodology. The method with the lower total was the better method. The criteria and their respective weights are as follows:

Table 2 Comparison metrics for evaluating level of integration into the PLM system and ease of use.

Criteria	Weights
Data Mapping Operations	1
File Conversions	2
Parameter Specifications Required	5
Options Not Available Inside the PLM	10

These metrics are descriptive of the level of integration and usability of each method. The method that has the least data mappings and file conversions will be a tighter integration. It will also be more robust and faster for comparable operations. The method requiring fewer parameter specifications per run and that does not require the user to go outside of the product lifecycle management system to specify parameters will be easier to use and a tighter integration. The weights were chosen by comparing each criterion against each other. It was determined through this comparison that the number of specifications not available inside of the product lifecycle management made the largest contribution to ease of use and level of integration. For this reason it has the largest weight. The number of data mappings was determined to contribute least in determining the better method. Hence, it has the lowest weight. The other weights were determined in a similar manner. Through these metrics the better method can be determined.

CHAPTER 4: RESULTS AND DISCUSSION OF RESULTS

Currently, no solution for advanced management of design process knowledge achieves all the needs of a company to realize multidisciplinary design optimization, standardization, and mass customization. While all have strengths, none can offer the complete solution that an integration of product lifecycle management and a framework can offer. The concept design needs of PLM can be met in a large part by the capabilities of a framework. Likewise, the collaborative and distributed computing needs of frameworks can be met in a large part by the capabilities of PLM. In this thesis framework capabilities will be added to PLM because PLM lacks less than what frameworks lack. Also, it is expected that companies will already have a PLM system. To add a framework with collaborative capabilities to a company's suit of software tools would mean that there would be unneeded overlap in software. Because PLM is a bigger part of a company than framework software it is concluded that is it better to add framework functionality to a PLM system.

4.1 Results from Development of the Proofs-of-Concept

It was found that two tactics could be used to achieve the PLM – framework integration. These tactics consist of one, embedding a commercial framework, iSIGHT (Engenious Software Inc.) into the product lifecycle management system and two, integrating design/analysis applications and an optimization algorithm into the product lifecycle management system’s workflow action handler. A generic case study is implemented and used to compare the tactics. Recommendations are made based on the level of integration and ease of implementation. This chapter shows that a PLM – framework integration solves the challenges associated with performing MDO, standardization, and mass customization because it enables a company to:

- Capture and Automate Design Processes.
- Optimize Designs.
- Manage All Data in a Centralized, Secure Fashion.
- Collaborate With All Participants.

4.1.1 Design Process Automation and Optimization

Process automation was achieved within a PLM system by customizing workflow management. There are two ways workflow was customized to achieve process automation. These two tactics were one, the internal method, where framework capabilities were added completely internal to the PLM system and two, the external method, where PLM workflow was customized to integrate external framework tools. Both tactics have specific strengths and differing application for different situations. To make recommendations on the usage of the different tactics, both were applied to the same test case and compared through the use of comparison metrics, as explained in the Method. The implementation was made using the PLM system Teamcenter. The concepts and code written is also applicable to other PLM tools. All that would need to be changed for different tools is the data transfer functions to be updated for other tools and the action handlers to be registered according to the other tool's documentation. The results of the implementation comparison are show here in Table 3, (more details on the test case are found in the appendix):

Table 3 Summary of implementation results

Metric	Internal	External
Data Mappings	14	21
File Conversions	4	14
Parameter Specifications	160	105
Options unavailable in PLM	0	200
Total	178	340

These result and conclusions are drawn from the results are discussed in the following sections:

- Internal Method.
- External Method.

4.1.1.1 Internal to PLM

As discussed in the review of research conducted by Klaas et al. it is possible to provide framework capabilities to a PLM system. By following this tactic it was found that the internal method was best suited for applications where every element of a design process needs to be monitored and controlled within the PLM system. This method is best suited for these applications because it provides a tighter integration and more control over the data as shown by the lower score in Table 3 for data mappings, file conversions, and option unavailable in PLM. This section will discuss the findings that lead to the conclusions just stated. The main elements evaluated in the internal method are:

- Automation Modules.
- Data Mapping.
- Design Optimization.
- User Interface.

4.1.1.1.1 Automation Modules

The creation of automation modules in the PLM system resulted in a very tight integration of each of the automated process steps into the PLM system. Such a tight integration was possible because each module was built right into PLM workflow tasks, thereby changing the very nature of these tasks. Standard workflow tasks are passive. They rely on users to perform what is assigned. Once the user indicates that they have performed the task, the workflow then moves to the next task. Active workflow tasks must be integrated into the workflow. A task itself will perform its assigned duty and then the next task will be initiated. Using the workflow empowered by data mapping and custom action handlers, process integration and automation is possible. Among others, action handlers can be made to update parametric CAD models, to mesh and analyze the models and to optimize the performance of the product. The feature tree of this integration for the test case is shown in Figure 12 and the break down of the test case score is given in Table 4. Several generic modules are required for design process automation. These modules are:

- Geometry Update/Creation.
- Mesh Generation.
- Analysis.

These modules make up a large proportion of the steps involved in a design process. If these modules can be made generic enough that they can be applied to vastly different models and programs, then the design automation setup will have the level of agility needed for continual improvement and exploration.

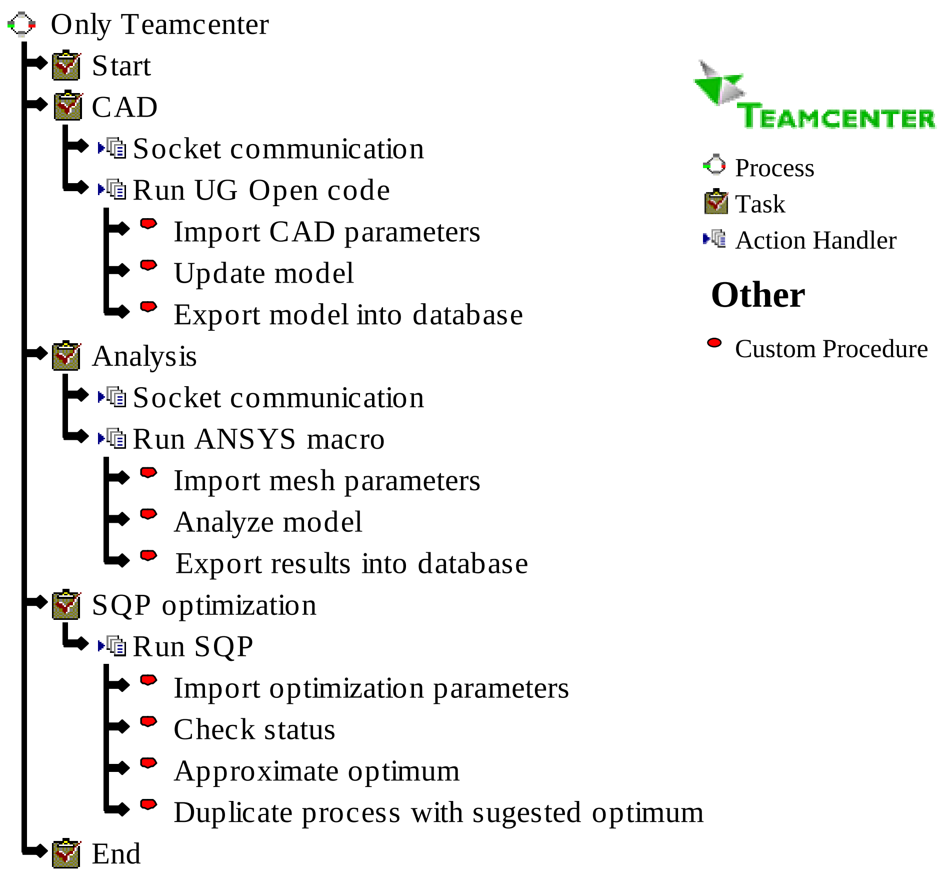


Figure 12 The feature tree for directly integrating framework capabilities into Teamcenter.

Table 4 Results of implementing the internal method

Workflow / Handler	Start	Run UG code			ANSYS macro			Run SQP						
		Import Params	Update Model	Export model	Import Params	Analyze	Export Results	Import Params	Check status	Apprixmate Optimum	Duplicate process	Sum	Weight	Total
Data Mappings		3			1		2	5			3	14	1	14
File Conversions				1	1							2	2	4
Required Specifications	5							11	16			32	5	160
Option unavailable												0	10	0
														178 Total

Geometry

Creation of a generic geometry module resulted in the ability to update a CAD model with data taken directly from the PLM database without any intermediate files or steps. This ability integrated the geometric module very tightly into the PLM system by contributing to only three data mappings and one file creation in the test case implementation. The use of a socket also makes the program generic so that if the module is to be used in another PLM tool, only the server side changes and the client can remain the same.

Mesh Generation

The mesh generation was also tightly integrated with very few data transfer procedures. Mesh generation can be achieved in multiple ways. In implementing the test case code was made to robustly generate a mesh for the analyzed geometry. Inputs to the routine were commutated from the PLM database by the use of a socket, as discussed in the previous section.

Analysis

The analysis module showed the ability to greatly reduce the number of data mappings needed. Because of this it was a very tight integration. The need for input and output files was eliminated by creating the ability to import and export analysis data directly to and from the PLM database. Because of this, the analysis module only contributed three data mappings and one file conversion during the test case implementation.

4.1.1.1.2 Data Mapping

The internal method gives the PLM system the most control over data mapping. Each of the modules created to perform automation require data. The internal method enables every detail of data mapping to be specified from within the PLM system. This section presents the issues involved with supplying this data so that PLM benefits can be realized. These are discussed in the following sections:

- Using the PLM database
- Extracting and Importing data from the database
- Mapping data to and between modules.

Database Data Extraction and Insertion

PLM systems store data in a database. Everything that is stored in the database by the PLM system is then managed by the PLM system. Because it is desired that all data used in design be managed by the PLM system so that it can be part of the PLM advantages. The internal method allows all data to be used internal to the PLM system. This enables the automation modules to be tightly integrated and eliminates the need of using files to store and communicate data. As discussed earlier PLM data should be accessed only through the use of the PLM system. The methods for doing this are different for each PLM system and can be found in that system's documentation.

4.1.1.1.3 Design Optimization

The internal method provides to a PLM system, for the first time ever, the ability for the PLM system to control an optimization loop. It also enables optimization data to be managed automatically by a PLM system. A PLM system has no built-in ability to automatically make decisions to improve a design. Decision support within PLM requires the integration of optimization algorithms. This section discusses the issues concerned with embedding an algorithm into a PLM system and the advantages associated with it. A discussion of the generation of new algorithms or researching and comparing the efficiency of possible algorithms is not included and is not part of the scope of this thesis.

4.1.1.1.4 User Interface

The internal method allows for collaborative user interaction. It offers an unprecedented ability to setup, monitor and visualize results of an automated optimization process within a completely collaborative environment. An important part of any application that requires human interaction is a user interface. User interface design has a great impact on both the usability and the user perception of an application. In general if a program works like a charm but has a poorly designed interface, it will not be appreciated to an extent anywhere near it's potential. This section discusses the interfaces needed in the customization to integrate a framework internally in a PLM.

Setup

An automated optimization is setup through the use of PLM's standard workflow designer. As mentioned earlier, workflow is a rigid tool and as such, the workflow designer is meant for seldom use by administrators. While it is usable and performs the function needed, it does not allow for the agility required because this interface was not meant for high traffic use. Future work should include the creation of a more usable interface. Suggested changes would be to have the action and rule handlers grouped in categories specific to their function and have them be selectable by icons. Also the interface should be created so that each action handler's arguments can be seen with a description of their potential values included. Last of all, it needs to be editable in a way that would allow engineers to access to it without requiring administrative privileges.

Dashboards

The current PLM workflow dashboard was leveraged for use as an unprecedented tool for monitoring the progress of automated optimization processes. Dashboards allow the status of the process to be monitored by users and managers. PLM workflow has a built in dashboard to monitor this progress. Anyone can view the process dashboard and quickly surmise the progress of the process. This dashboard may be viewed by anyone with permission in any location.

Visualization

Data and results of the process can be viewed though normal PLM methods. This is more advanced for CAD models, where lightweight images can be quickly viewed. However other data forms can only be viewed as pure data or by using an external tool. Work needs to be done to create visualization tools for viewing optimization results, and other data forms within the PLM system.

4.1.1.2 External to PLM

The second methodology is to integrate an external framework tool into the PLM system. The feature tree for this implementation is shown in Figure 13. Through implementing this tactic on the test case it was found that the external method is best suited for applications where agility and ease of use are required. Because a commercial tool was used, this tactic provided more support and a better user interface, making it much more usable as shown by the low score received for the number of required specifications that needed as shown in Table 3. The following sections support these conclusions as they discuss the findings obtained when implementing the test case. The two main areas of development are:

- Executing the external application from within the PLM system.
- Linking the external run to the PLM such that all inputs and results can be access in the PLM system.

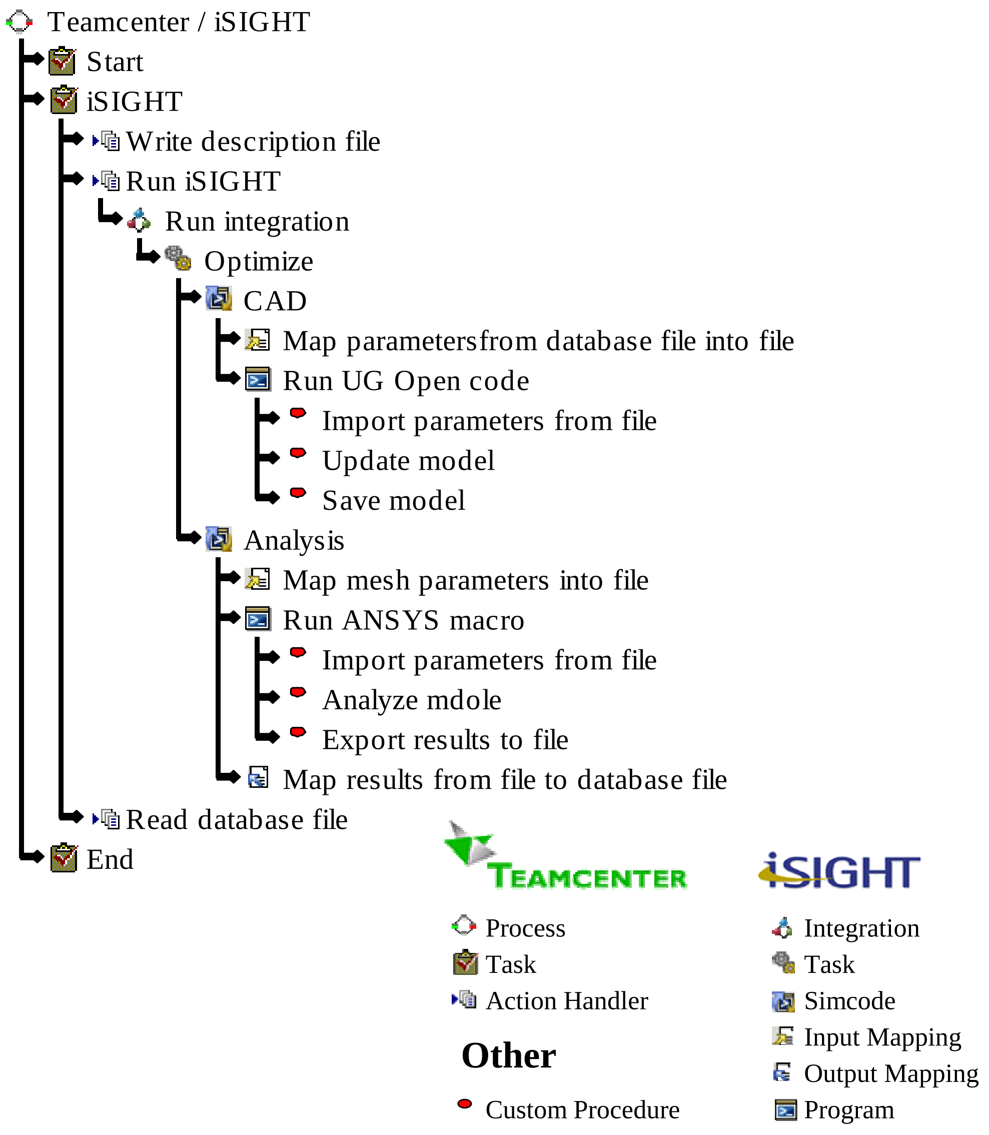


Figure 13 The feature tree for integrating iSIGHT into Teamcenter Engineering Workflow.

4.1.1.2.1 Execution

Execution of an automated optimization enables extreme setup agility, because the use of a commercial tool execution of an already setup process is very easy. This is shown by the results of the test case implementation as broken down in Table 5.

Table 5 Results of implementing the external method

Worflow	Start	Execute iSIGHT											Read file				
Handler		Dscr File	Run iSIGHT														
iSIGHT			Optimize														
Integration				CAD				Analysis									
Simcode				Map	Run UG code			Map	ANSYS macro			Map					
Program					Import Params	Update Model	Export model		Import Params	Analyze	Export Results			Sum	Weight	Total	
Data Mappings		4		3	3			1	1		2	2	5	21	1	21	
File Conversions		1		1			1	1	1			1	1	7	2	14	
Required Specifications	1	11	9											21	5	105	
Option unavailable		11	9											20	10	200	
																340	Total

PLM workflow management was used to integrate iSIGHT into the PLM system. iSIGHT was chosen because of its wide acceptance in industry and because of the ease at which it was integrated into the PLM system. Additionally, iSIGHT has only the functions needed by a framework with few other collaborative or data management functionality that would overlap with and cause redundancy with PLM functionality. A standalone version of FIPER was considered, but not chosen because at the time and now, though advertised to be available, no standard supported application programming interface is available.

4.1.2 Centralized Data Management and Collaboration

Centralized data management is achieved through the use of the PLM system. All data transactions by both the internal and external methods have used PLM methods to transfer and store data. Because of this, all of the data is automatically managed by the PLM system. PLM systems also have collaborative ability built in. That is the beauty of using a PLM system. Everything done by the framework integration is securely accessible though the entire enterprise because of the use of a PLM system.

CHAPTER 5: CONCLUSION

The objective of this thesis was to identify a knowledge representation strategy that can be implemented effectively in a PLM environment. This created the ability for company wide design space knowledge reuse. This objective was be achieved by representing the design space in the format of an MDO schema that can interact with a PLM architecture. The feasibility of this approach was tested using two proof-of-concepts. One proof-of-concept integrated framework capabilities into the product lifecycle management solution. The second proof-of-concept was implemented by embedding a commercial framework tool into the product lifecycle management solution. Both applications were developed to demonstrate the power and validity of embedding process integration and design optimization within a PLM system. This thesis answered the following related questions:

- How can an MDO schema be represented in a PLM system?
- How can PLM architecture be leveraged to manage the MDO schema?
- How can the MDO schema and PLM interact to preserve reuse and modularity?

5.1 Representing an MDO Schema in a PLM System

It is concluded through the results obtained from the test case that an MDO schema should be represented in a PLM system by creating action handlers within PLM workflow to perform automated engineering tasks required by the MDO process. Parametric automation modules can be created as action handlers to update CAD models, generate analysis meshes and to perform analysis.

When these action handlers are linked together in a workflow process it represents an ideal representation of the product design. The entire design space can be represented by this workflow process if it is defined such that it can be implemented as an optimization loop. As such, an action handler should be created to perform optimization on the automated process. Through the use of PLM workflow action handlers to provide automation and optimization, this design space can be represented in a PLM system as an MDO schema. This representation, however, in itself does not insure that the PLM system manages the MDO schema. The next section discusses how the PLM system can be leveraged to manage the MDO schema.

5.2 Leveraging PLM Architecture to Manage an MDO Schema

It is concluded through the results obtained from the test case that by leveraging PLM workflow, and form architecture the PLM system can be used to manage MDO schemas. To manage the MDO schema the PLM system must have access to the schema’s inputs and results. Additionally, this data must be accessible to PLM users. By storing this data as PLM forms the PLM system and its users will be able to have access to manage the data.

To further increase the data organization for more control, and easier management PLM workflow process attachment architecture should also be used. MDO schema data stored in PLM forms can be referenced in folders attached to workflow processes. By attaching the PLM forms containing a schema’s data to the schema’s workflow process the data becomes linked to the schema, and both schema and data can be managed together. The next section discusses the conclusions for achieving interaction between the MDO schema and the PLM system.

5.3 Interaction Between the MDO Schema and the PLM System

It is concluded through the results obtained from the test case that interaction between the MDO schema and PLM system should be achieved through the use of socket communications. Interaction required between the schema and PLM system consists of data communication and status notification. A socket communication is the most ideal way to handle this interaction because it promotes modularity.

Modularity is achieved because the communication and interaction functionality needed between the MDO schema and PLM system can be contained within the server of the socket. The server then becomes a distinct module that every parametric automation module can use to interact with the PLM system. In the event that a new automation module is created it can be created as a client that communicates with the socket server. Because the server module communicates with the PLM system, that functionality does not need to be recreated in the client automation module. Also, if an automation module needs to be used with another PLM system the module will only need to be changed so that it communicates with a different server module that communicates with the new PLM system. The combined ability to represent an MDO schema in a PLM system such that the PLM system can manage it and preserve modularity presents an enormous opportunity to industry which will be discussed in the final conclusions.

5.4 Final Conclusions

Increasing globalization and market competitive demands are driving industry to seek out improved strategies for knowledge management. Concurrently, engineering software providers (specifically PLM and MDO framework providers) have been selling their products to industry claiming that they can solve these challenges; but to date, companies have not been able to fully leverage these tools. One of industries greatest challenges is to capture a common knowledge representation of their product's design space that allows full integration across the enterprise so that as market needs shift they can quickly pinpoint the design to meet these needs. This thesis proposes a new approach to PLM and MDO framework usage that enables the complete representation of a design space with absolute, enterprise wide reuse. Because of the synergy that is created between PLM and MDO through this approach, both software providers and users in industry are looking at it as a way to achieve their greatest challenges. This thesis achieves the common knowledge representation that industry has been actively pursuing, because of this industry leaders have been impressed and believe that this approach will quickly take hold and usher in a new era for product design.

REFERENCES

1. Teare, S. “A procedure for Structuring Product Information for Reusability”, *Master’s Thesis BYU*, 2000.

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

3. www.ugs.com, Posted Sept. 2004.

4. Johnson, C. Gavilanes, J. "Quick and below budget", *InTech*, v 50 n 5, pages 51-54, May 2003.

5. Rahse, Wilfried, Hoffmann, Sandra “Product design - Interaction between chemistry, technology and marketing to meet customer needs” *Chemical Engineering and Technology*, v 26, n 9, pages 931-940, September, 2003.

6. 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, pages 981-996, November 2002.

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

8. Stelmack, Marc A., Batill, Stephen M., Beck, Bryan C. “Design of an aircraft brake component using an interactive multidisciplinary design optimization framework”, *Journal of Mechanical Design, Transactions of the ASME*, v 122, n 1, pages 70-76, March 2000.

9. Berger, “Continuous improvement and kaizen: Standardization and organizational designs” *Integrated Manufacturing Systems*, v 8, n 2, pages 110-117, 1997.

10. Kulhavy, Rudolf. “Data-centric decision support”. *Proceedings of the American Control Conference*, v 4, pages 3395-3400, 2002.

11. Pine, J.B., 1993, “Mass Customizing Products and Services,” *Planning Review*, Vol. 21, No 4, pp. 6-13.

12. Roach, Gregory, "The Product Design Generator—A Next Generation Approach To Detailed Design", Brigham Young University, Dissertation, August 2003.

13. Hogge, D, "Integrating Commercial CAx Software to Perform Multidisciplinary Design Optimization", *Master's Thesis BYU*, 2002.

14. Tappeta, R.V., Nagendra, S., Renaud, J.R. "Multidisciplinary design optimization approach for high temperature aircraft engine components", *Structural Optimization*, v 18, n 2-3, pages 134-145Oct, 1999.

15. Carlson, P. "A distributed computing system used for concurrent optimization methods on a violin top", *Structural and Multidisciplinary Optimization*, v 25 n 5-6, pages 453-458, December 2003.

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

17. Numata J. Lei, B. Iwashita, Y. "Information management for knowledge amplification in virtual enterprise". *IEEE International Engineering Management Conference, Managing Virtual Enterprises: A Convergence of Communications, Computing, and Energy Technologies.* pages 281-285, 1996.

18. Huang G. Lee S. Mak K. "Web-based product and process data modeling in concurrent 'design for X'". *Robotics and Computer-Integrated manufacturing*; v 15 n 3, pages 53-63, June 1999.

19. Caldwell, N. Rodgers, P. "WebCADET: Facilitating distributed design support". *London, UK: IEE Colloquium on Web-based Knowledge Servers*. n 307, pages 9/1-9/4, 1998

20. Zdrahal Z, Domingue J. "The World Wide Design Lab: an environment for distributed collaborative design". *In Proceedings of International Conference on Engineering Design*, Tampere, Aug. 19-21, 1997.

21. Klein M. "Capturing geometry rationale for collaborative design". *In Proceedings of the IEEE Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE'97*). Pages 24-28, 1997.

22. 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*. pages 1-18, 1998.

23. Cutkosky, M. Engelmore, R. Fikes, R. Genesereth, M. Bruber, T. Mark, W. Tenenbaum, J. Weber, J. “PACT: An experiment in integrating concurrent engineering systems”. *IEEE Computer*; v 26(1), pages 28-37, 1993.

24. Campbell, M. Cagan, J. Kokvsky, K. “A-Design: an agent-based approach to concept design in a dynamic environment”. *Research in Engineering Design* v 11, pages172-192, 1999.

25. Varma, A. Dong, A. Childambaram, B. Agogino, A. Wood, W. “Web-based tool for engineering design”, *Working Paper*, 1999.

26. Toye, G. Cutkosky, M. Leifer, L. Tenenbaun, J. Glicksman, J. “SHARE: A methodology and environment for collaborative product development”. *In proceeding of Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Computer Society Press*. Pages 33-47, 1993.

27. http://www.engineous.com/product_FIPER.htm, Posted Feb. 2005.

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

29. Klaas, O. "Embedding reliable numerical analysis capabilities into an enterprise-wide information system", *Engineering with Computers*, v 17 n 2, pages 151-161, 2001.

30. Lund, J. “Parametric Product Lifecycle Management” *Master’s Thesis BYU*, 2005.

31. Goldfarb, D. Idnani, A. “A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs” *Math Programming*, v 27, pages 1-33, 1983.

32. Kim, Y. Kang, S. Lee, S. and Yoo, S. “A distributed, open, intelligent product data management system”. *International Journal of Computer Integrated Manufacturing*, v 14 n 2, pages 224–235, March/April 2001.

33. Koonce, D. "A hierarchical cost estimation tool", *Computers in Industry*, v 50 n 3, pages 293-302, April 2003.

34. Alexandrov, N. M. Lewis, R. "Analytical and computational aspects of collaborative optimization for multidisciplinary design", *AIAA Journal*, v 40 n 2, pages 301-309, February 2002.

35. Robinson, C. "Good goals lead to better data", *Machine Design*, v 66 n 10, pages 51-60, May 1994.

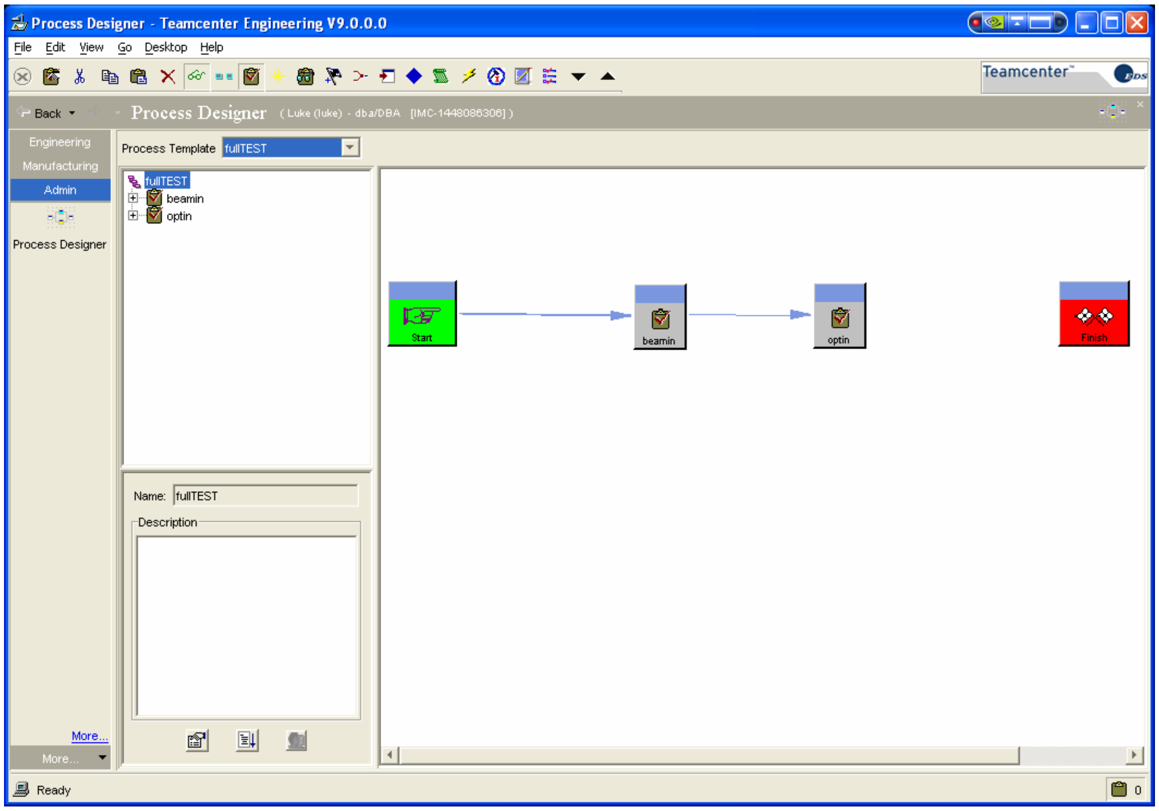
36. Meade, L. Presley, A. Rodgers, K. "Tools for engineering the agile enterprise", *IEEE International Engineering Management Conference, Managing Virtual Enterprises: A Convergence of Communications*, *Computing*, pages 381-385, 1996.

37. 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, pages 491-500, October/December 2003.

38. Rogers PA, Huxor AP, Caldwell. “Design support using distributed Web-based AI tools”. *Research in Engineering Design*, v 11, pages 31-44, 1999.

APPENDIX

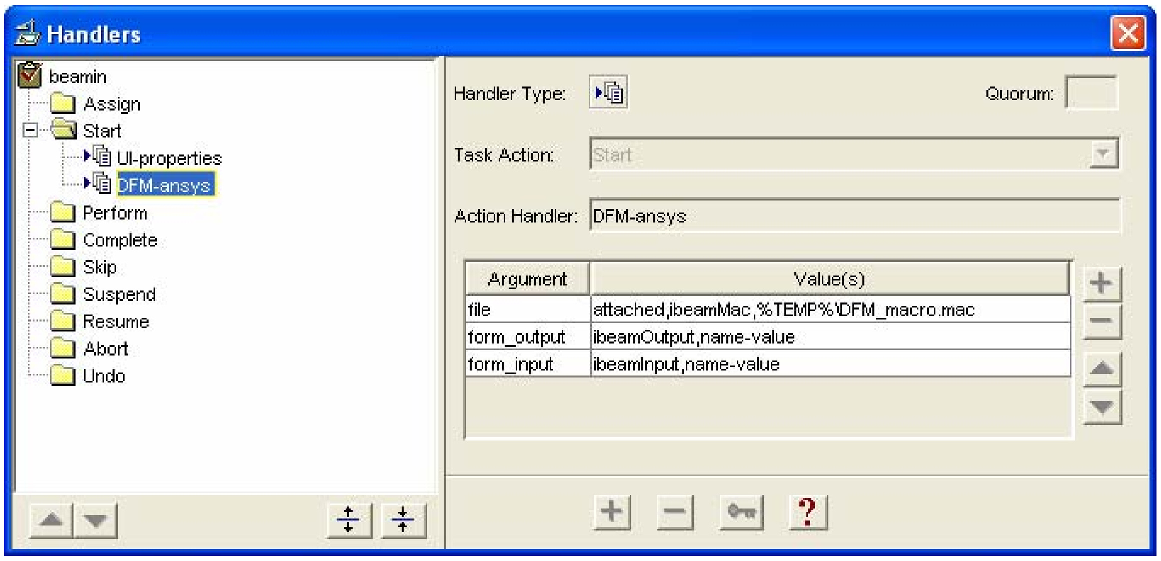
APPENDIX A: INTERNAL INTEGRATION


Process Designer - Teamcenter Engineering V9.0.0.0
File Edit View Go Desktop Help
Teamcenter
Back
Process Designer (Luke (luke) - dba/DBA [IMC-1448086306])
Engineering
Manufacturing
Admin
Process Designer
fullTEST
beamin
optin
Description
More...
More...
Start
beamin
optin
Finish
Ready
0

Process Template: fullTEST

Name: fullTEST

Figure 14 The PLM workflow process designer. Design for the internal method includes an ANSYS task and an optimization task.



Handlers

- beamin
  - Assign
  - Start
    - UI-properties
    - DFM-ansys
  - Perform
  - Complete
  - Skip
  - Suspend
  - Resume
  - Abort
  - Undo

Handler Type:

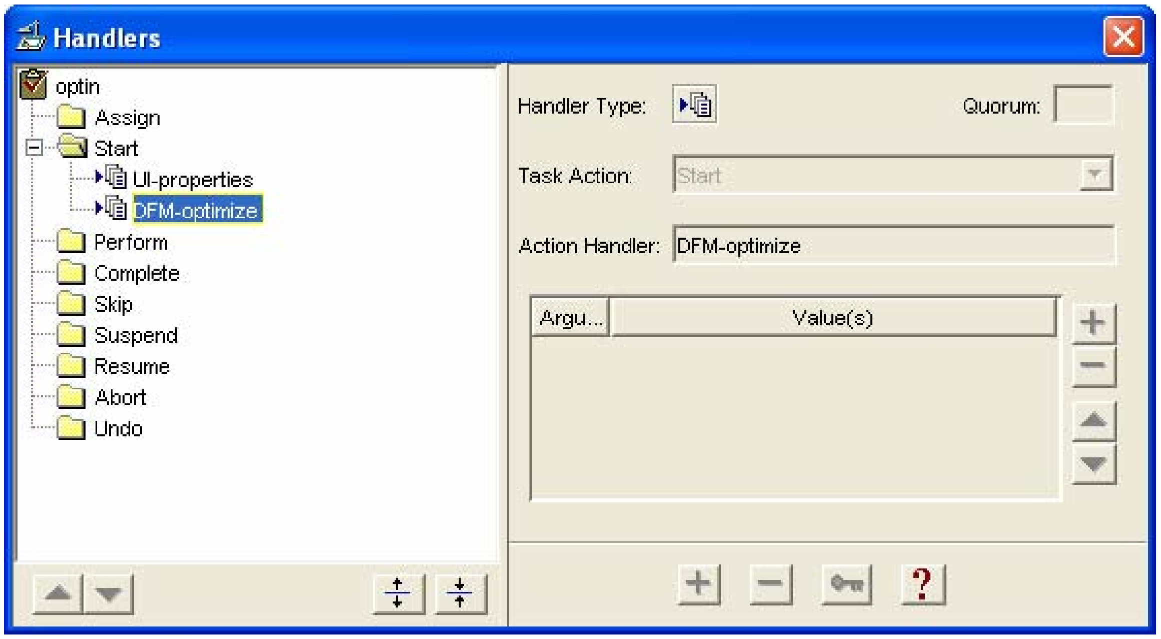
Quorum:

Task Action: Start

Action Handler: DFM-ansys

Argument	Value(s)
file	attached,ibeamMac,%TEMP%\DFM_macro.mac
form_output	ibeamOutput,name-value
form_input	ibeamInput,name-value

Figure 15 ANSYS task action handler. The handler accepts three arguments: The macro file. The output form. The input form.



Handlers

- optin
  - Assign
  - Start
    - UI-properties
    - DFM-optimize
  - Perform
  - Complete
  - Skip
  - Suspend
  - Resume
  - Abort
  - Undo

Handler Type:

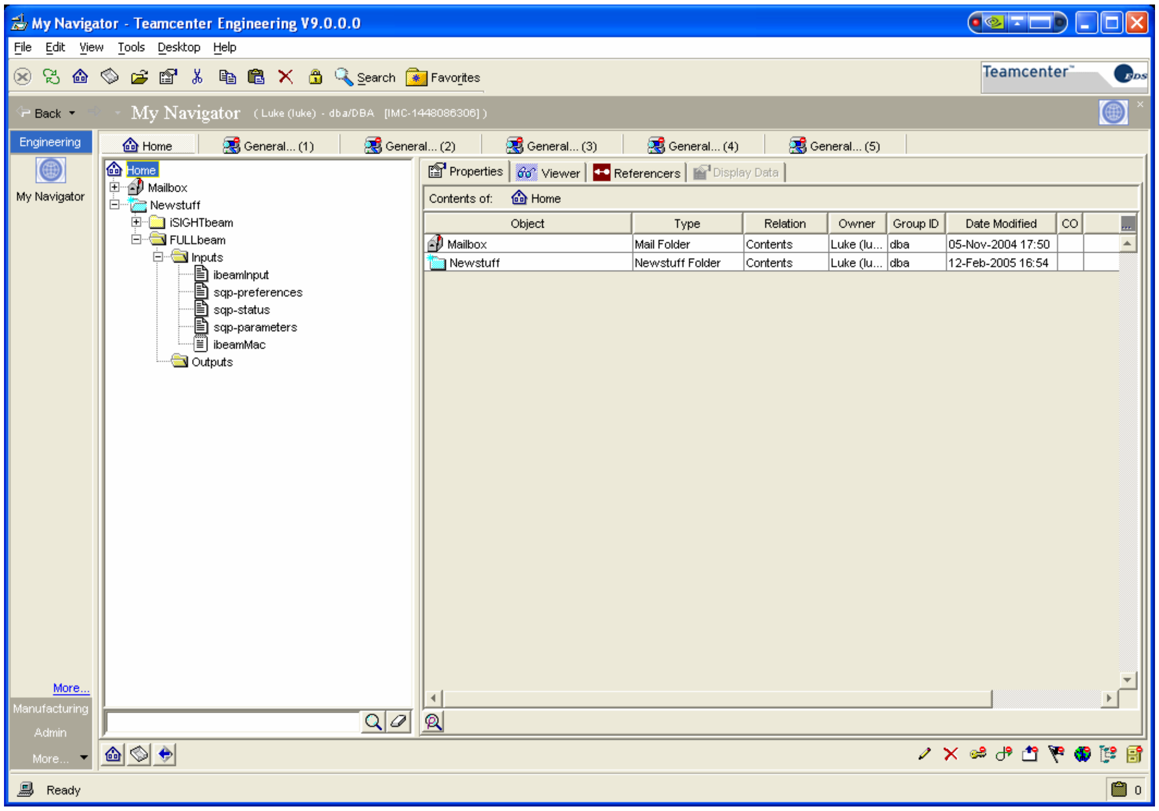
Quorum:

Task Action: Start

Action Handler: DFM-optimize

Argu...	Value(s)

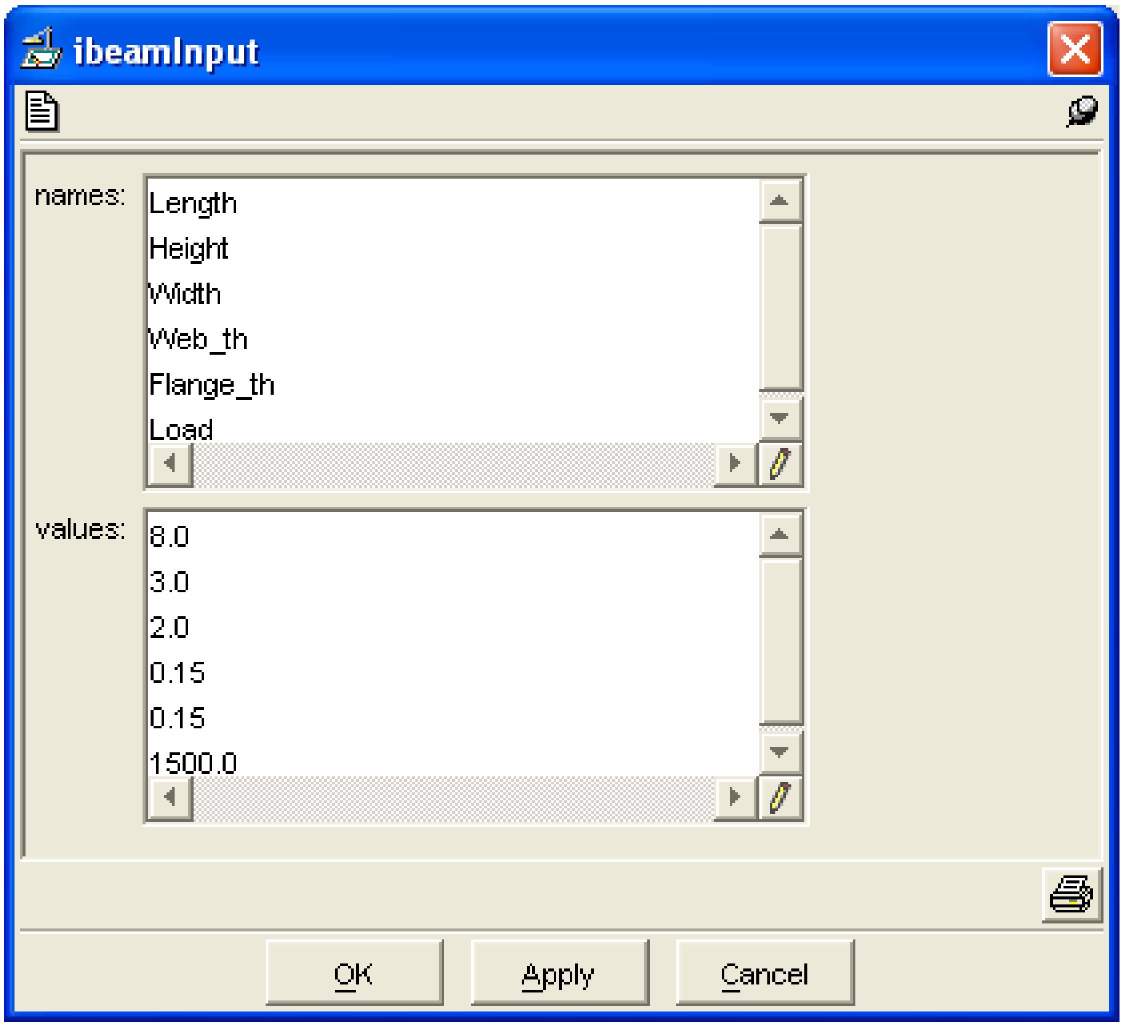
Figure 16 The optimization task action handler. This handler accepts no arguments, because it recieves all needed information from a parameters, preference, and status for attached to the process.



- Home
  - Mailbox
  - Newstuff
    - iSIGHTbeam
    - FULLbeam
      - Inputs
        - ibeamInput
        - sqp-preferences
        - sqp-status
        - sqp-parameters
        - ibeamMac
      - Outputs

Object	Type	Relation	Owner	Group ID	Date Modified	CO
Mailbox	Mail Folder	Contents	Luke (lu...	dba	05-Nov-2004 17:50	
Newstuff	Newstuff Folder	Contents	Luke (lu...	dba	12-Feb-2005 16:54	

Figure 17 Internal method attached folders, and forms. The Inputs folder contains the I-beam inputs, the three forms required by the optimization, and the ANASYS macro.



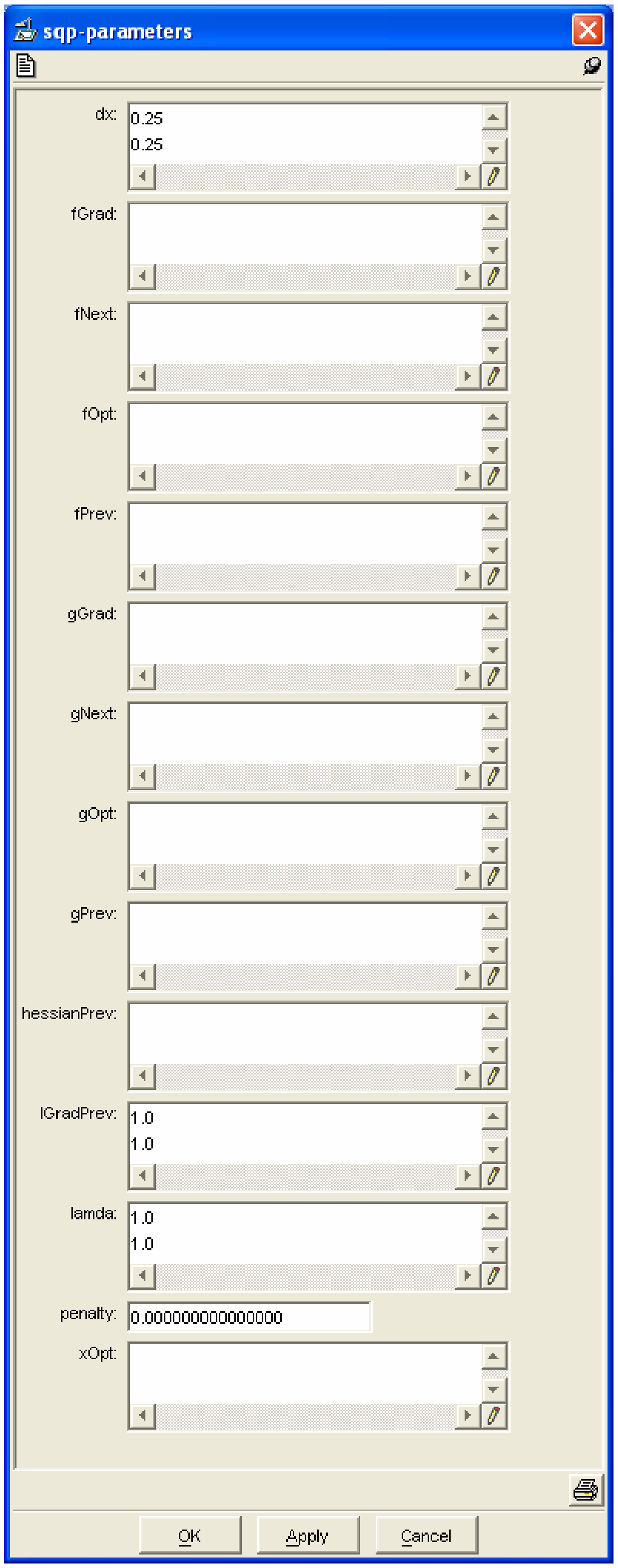
names:

Length
Height
Width
Web_th
Flange_th
Load

values:

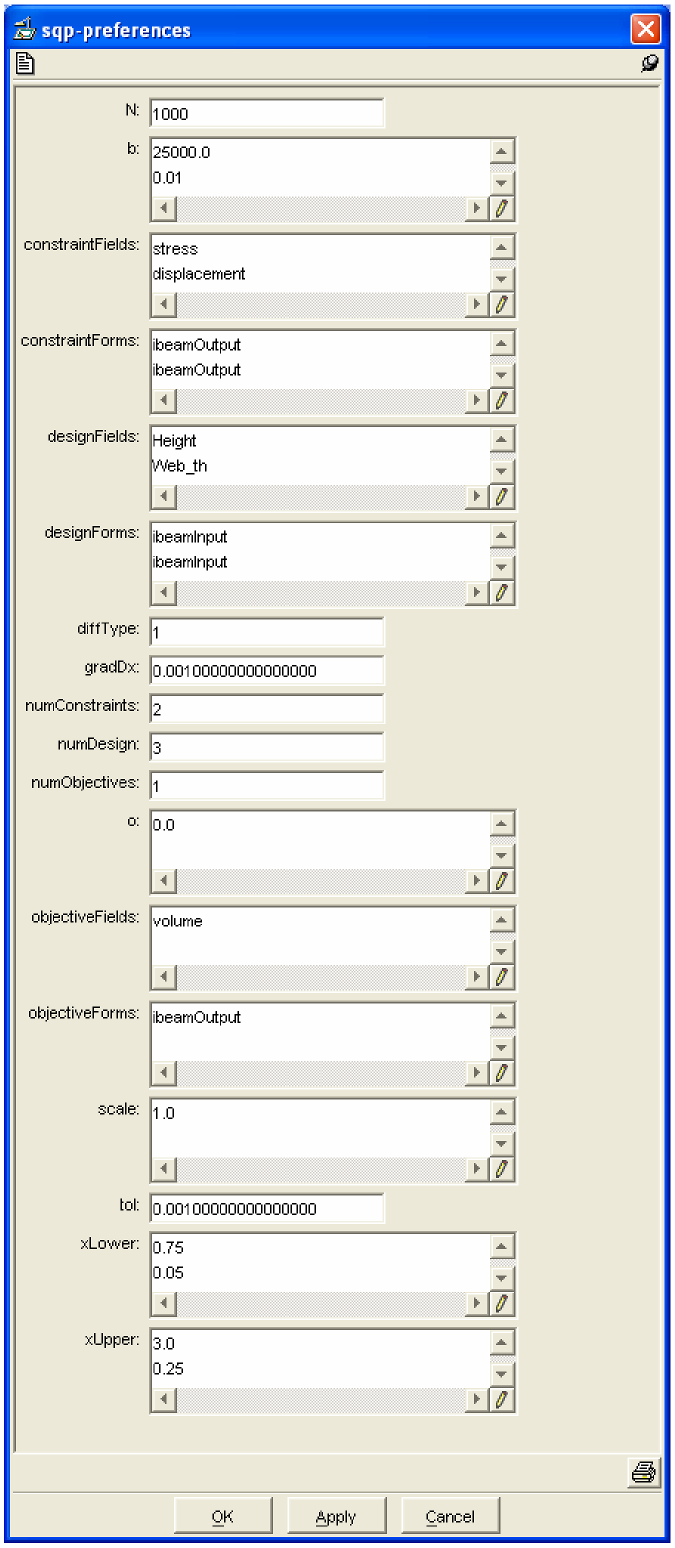
8.0
3.0
2.0
0.15
0.15
1500.0

Figure 18 Internal method input form.


sqp-parameters

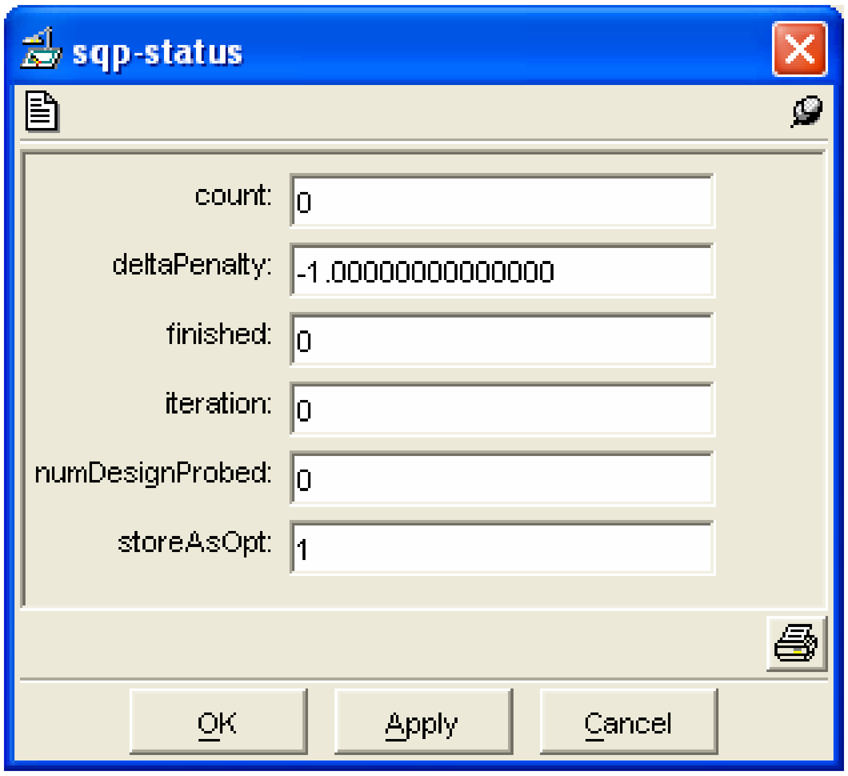
dx:	0.25 0.25
fGrad:	
fNext:	
fOpt:	
fPrev:	
gGrad:	
gNext:	
gOpt:	
gPrev:	
hessianPrev:	
lGradPrev:	1.0 1.0
lamda:	1.0 1.0
penalty:	0.000000000000000
xOpt:	

Figure 19 Internal method optimization parameters form initial setup. As the optimization runs parameters needed by the optimization are retained in this form.



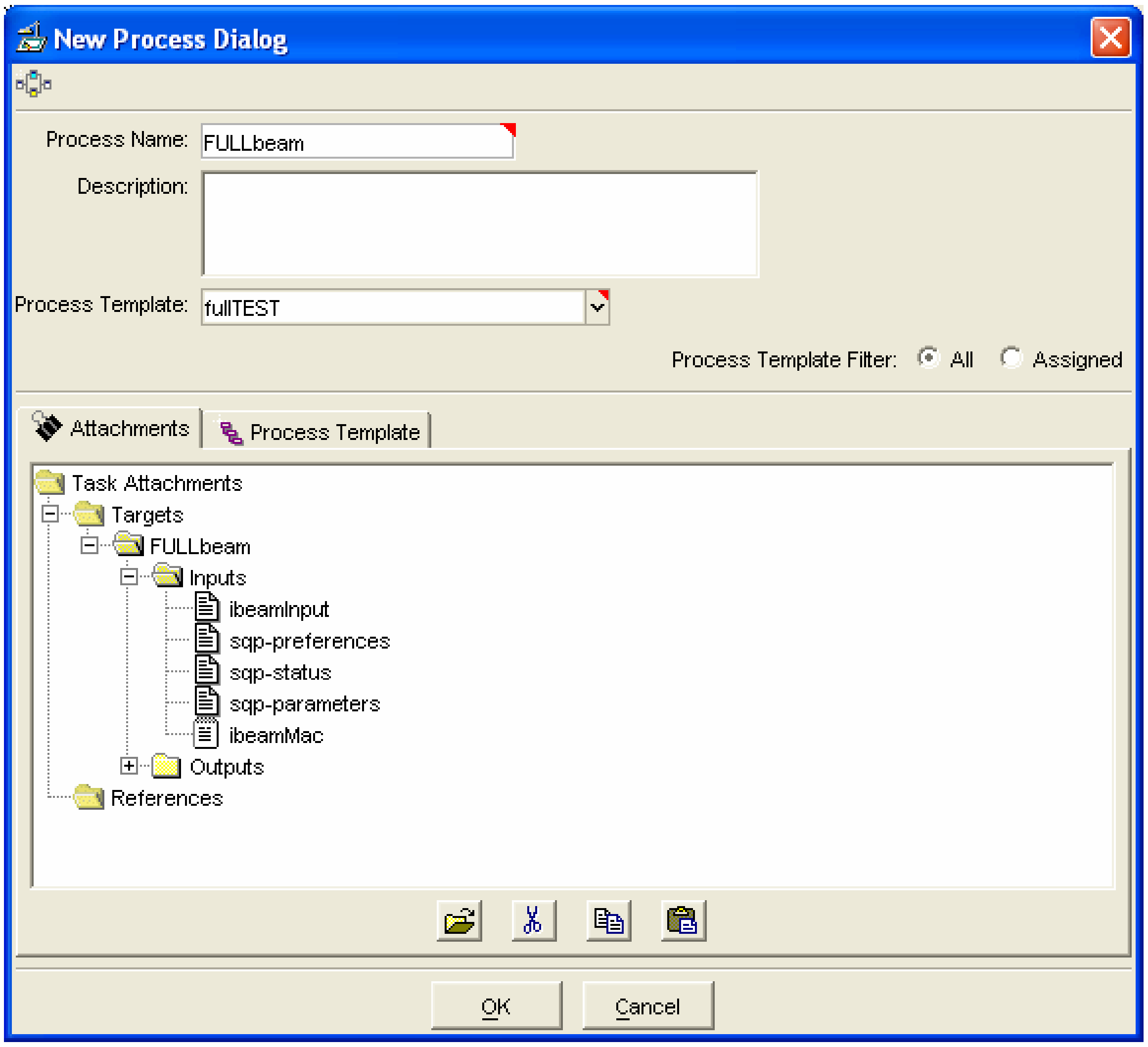
N:	1000
b:	25000.0 0.01
constraintFields:	stress displacement
constraintForms:	ibeamOutput ibeamOutput
designFields:	Height Web_th
designForms:	ibeamInput ibeamInput
diffType:	1
gradDx:	0.00100000000000000
numConstraints:	2
numDesign:	3
numObjectives:	1
o:	0.0
objectiveFields:	volume
objectiveForms:	ibeamOutput
scale:	1.0
tol:	0.00100000000000000
xLower:	0.75 0.05
xUpper:	3.0 0.25

Figure 20 Internal method optimization prefences form. This form contains the user's optimization preferences and setup.



count:	0
deltaPenalty:	-1.00000000000000
finished:	0
iteration:	0
numDesignProbed:	0
storeAsOpt:	1

Figure 21 The internal method optimization status form. The optimization uses this form to retain information about the optimization's current status.



New Process Dialog

Process Name: FULLbeam

Description:

Process Template: fullTEST

Process Template Filter: [✓ All] Assigned

Attachments | Process Template

```
Task Attachments
  Targets
    FULLbeam
      Inputs
        ibeamInput
        sqp-preferences
        sqp-status
        sqp-parameters
        ibeamMac
      Outputs
  References
```

OK | Cancel

Figure 22 Dialog to initiate a new process from the internal method template created in the process designer. The attached folder is shown.

ANSYS Macro Code:

```
/com,starting

FINISH
/CLEAR
!/CWD,'C:\Documents and Settings\Nathaniel\My Documents\School\isightSide\ANSYS'
~eui,'source [file join C:/IMAN0900/bin/DFM_ansys.tcl]'
~eui,'DFM::getValue Length'
~eui,'DFM::getValue Height'
~eui,'DFM::getValue Width'
~eui,'DFM::getValue Web_th'
~eui,'DFM::getValue Flange_th'
~eui,'DFM::getValue Load'

! Load IGES file
/AUX15
! ~UGIN,ibeam,prt,'..\CAD\',SOLIDS,1,0 !***Edit this line

! Go into the preprocessor
/prep7
!RECTNG,4,-4,2,1.5,
RECTNG,-Width/2,Width/2,Height/2,Height/2-Flange_th
RECTNG,-Width/2,Width/2,-Height/2,-Height/2+Flange_th
RECTNG,-Web_th/2,Web_th/2,Height/2-Flange_th,-Height/2+Flange_th
AADD,ALL
VOFFST,4,Length, ,


! Define element types
ET,1,MESH200
KEYOPT,1,1,6
KEYOPT,1,2,0
ET,2,SOLID45

! Define material properties
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,30e6
MPDATA,PRXY,1,,.3
MPDATA,dens,1,,.0007


! Create a volume if necessary
allsel,all
*get,volumeCount,volu,,count
*if,volumeCount,eq,0,then
   nummrg,kp,7e-4,7e-4,,low
   va,all
*endif


! Get the front area number (loadArea)
areaNum=0
minCentZ=1000
allsel,all
*get,areaCount,area,,count
*do,i,1,areaCount,1
   asel,all
   areaNum=arnext(areaNum)
   asel,s,,,areaNum
   asum
   *get,centZ,area,,cent,z
   *if,centZ,lt,minCentZ,then
      minCentZ=centZ
      loadArea=areaNum
   *endif
*enddo

! Get the back area number (fixArea)
```

```
areaNum=0
minCentZ=0
allsel,all
*do,i,1,areaCount,1
   asel,all
   areaNum=arnext(areaNum)
   asel,s,,,areaNum
   asum
   *get,centZ,area,,cent,z
   *if,centZ,gt,minCentZ,then
      minCentZ=centZ
      fixArea=areaNum
   *endif
*enddo


! Mesh the top area
myesize=Web_th/2
*if,Web_th,gt,Flange_th,then
   myesize=Flange_th/2
*endif

asel,s,,,loadArea
lsla,s
*get,lineCount,line,,count
lineNum=0
*do,i,1,lineCount,1
   lsla,s,
   lineNum=lsnext(lineNum)
   lsel,s,,,lineNum
   lesize,lineNum,myesize
*enddo


asel,s,,,loadArea
TYPE,1
MAT,1
REAL,
ESYS,0
SECNUM,
MSHAPE,0,2D
MSHKEY,0
amesh,all


! Set number of divisions on the lines
lsel,all
asel,s,,,loadArea
asel,a,,,fixArea
lsla,u
lesize,all,,,8,,,,,0 ! This puts n divisions on all lines selected


! Sweep mesh the volume
allsel,all
TYPE,2
MAT,1
REAL,
ESYS,0
SECNUM,
!*
MSHAPE,0,3D
!*
VSWEEP,all


! Constrain root face of airfoil
DA,fixArea,ALL,


! Find number of nodes in the web area
```

```
asel,s,,,loadArea
nsla,s,1
cm,loadNodes,node
*get,nodeCount,node,,count
nodeNum=0
minX=Web_th/2
numNodesToLoad=0
*do,i,1,nodeCount,1
   cmsel,s,loadNodes
   nodeNum=ndnext(nodeNum)
   nsel,s,,,nodeNum
   *get,xloc,NODE,nodeNum,loc,x
   *if,xloc,le,minX,then
      *if,xloc,ge,-minX,then
         numNodesToLoad=numNodesToLoad+1
      *endif
   *endif
*enddo

nodeForce=Load/numNodesToLoad


! Apply forces to nodes in the web area
asel,s,,,loadArea
nsla,s,1
cm,loadNodes,node
*get,nodeCount,node,,count
nodeNum=0
minX=Web_th/2
numNodesToLoad=0
*do,i,1,nodeCount,1
   cmsel,s,loadNodes
   nodeNum=ndnext(nodeNum)
   nsel,s,,,nodeNum
   *get,xloc,NODE,nodeNum,loc,x
   *if,xloc,le,minX,then
      *if,xloc,ge,-minX,then
         F,nodeNum,FY,nodeForce
      *endif
   *endif
*enddo



! Solve it
/solu
allsel,all
solve




! Output solution analysis objectives to PLM

! Measure volume of ibeam (representative of mass)
/prep7
vsel,all
vsum
*get,vol,volu,,volu
~eui,'DFM::setValue volume vol'

! Get max principal stress
/post1
nsort,s,1,0
*GET,logtmax,SORT, ,MAX
~eui,'DFM::setValue stress logtmax'

! Get max displacement
nsort,u,sum,0
*GET,dymax,SORT, ,MAX
~eui,'DFM::setValue displacement dymax'
```

```
~eui,'DFM::closeConnetion'

!create plot
/SHOW,JPEG
/VIEW,1,1,2,3
/ANG,1
/AUTO,1
/RGB,INDEX,100,100,100,0
/RGB,INDEX,0,0,0,15
/VCONE,ALL,45.0
/DEV,PSFN,NINC
/gfile,400
PLNSOL,S,EQV
/SHOW,CLOSE
~eui,'set jobname [ans_getvalue ACTIVE,,JOBNAM]; set imagename [string trim $jobname];
file copy -force $imagename.jpg results.jpg; file delete -force $imagename.jpg;'
```

Tcl Code:

```
namespace eval DFM {
  variable myPID [pid]
  variable channel
  variable serverHost "127.0.0.1"
  variable serverPort "27016"
  variable arg
  variable field
  variable valueToSet
  variable luke
  variable msg

}

proc debug {arg} {
  puts $arg
}

proc DFM::getArg { arg } {
  set DFM::arg $arg
  set DFM::msg "$DFM::myPID:getarg:$DFM::arg"
  DFM::msgSend
  vwait DFM::luke
}

proc DFM::getValue { arg } {
  set DFM::arg $arg
  set DFM::msg "$DFM::myPID:getvalue:$DFM::arg"
  DFM::msgSend
  vwait DFM::luke
}

proc DFM::setValue { field valueToSet } {
  set DFM::field $field
  set DFM::valueToSet $valueToSet
  set value [ans_getvalue PARM,$valueToSet,VALUE]
  set DFM::msg "$DFM::myPID:setvalue:$DFM::field:$value"
  DFM::msgSend
}

proc DFM::msgSend { } {
  puts "TCL client sent <<<$DFM::msg>>>"
  puts $DFM::channel "$DFM::msg"; flush $DFM::channel
}

proc DFM::msgHandler { } {
  set data [gets $DFM::channel]; flush $DFM::channel
  puts stdout "TCL client recieved <<<$data>>> for message <<<$DFM::msg>>>"; flush stdout
  if { [string first "setvalue" $DFM::msg] != -1 } {
    set data [gets $DFM::channel]; flush $DFM::channel
  } elseif { [string first "getarg" $DFM::msg] != -1 } {
        ans_sendcommand *set,$DFM::arg,$data
  } elseif { [string first "getvalue" $DFM::msg] != -1 } {
        ans_sendcommand *set,$DFM::arg,$data
  }
  set DFM::luke ready
  unset DFM::luke
}

proc DFM::closeConnetion { } {
       puts "Closing TCL client"
   catch {close $DFM::channel}
}

#open get socket
if {[catch {socket $DFM::serverHost $DFM::serverPort} DFM::channel]} {
  puts stdout "Failed to connect to server at $DFM::serverHost $DFM::serverPort"; flush
stdout
  exit
} else {
  fconfigure $DFM::channel -blocking 0
  fileevent $DFM::channel readable "DFM::msgHandler"
}
set data [gets $DFM::channel]; flush $DFM::channel
```

Optimization Code:

```
#include "sqp.h"


//the forms
Form* Stats;
tag_t stats;
Form* Prefs;
tag_t prefs;
Form* Param;
tag_t params;
tag_t root_task;
tag_t dup_task;

//found in preference form
int numConstraints;
int numDesign;
int numObjectives;
double tol;
int N;
int diffType;
double gradDx;
double *b;
double *o;
double *scale;
double *xUpper;
double *xLower;

//found in status form
int storeAsOpt;
int finished;
int count;
double deltaPenalty;
int iteration;
int numDesignProbed;

//found in parameters form
double penalty;
double **gPrev;
double **gNext;
double **gGrad;
double **fPrev;
double **fNext;
double **fGrad;
double **hessianPrev;
double **dx;
double *lamda;
double *gOpt;
double *fOpt;
double *lGradPrev;
double *xOpt;

//found in other attached forms
double *f;
double *g;
double *xRun;

//not read in
int total;
int num2probe;
double **hessian;
double **coeficiants;
double **gama;
double **gamaT;
double **dxT;
double *lGrad;
double *xNext;
double *other;
double *solution;
```

```
//----------------------------------------------
// allocates memory for a 2D array
//----------------------------------------------

void setup2DArray(double **&array, int x, int y)
{
   array = new double*[x];
   for(int i=0;i<x;i++) {
      array[i] = new double[y];
         for(int j=0;j<y;j++)
                array[i][j]=0;
   }
}

//-----------------------------------------
// deallocates memory for a 2D array
//-----------------------------------------

void destroy2DArray(double **&array, int x)
{
   for(int i=0;i<x;i++)
      delete [] array[i];
   delete [] array;
}

//-----------------------------------------
// constructor for optimization
//------------------------------------------

int opt()
{
       int i;
       //allocate memory
       setup2DArray(gNext, numDesign, numConstraints);
       setup2DArray(gPrev, numDesign, numConstraints);
       setup2DArray(gGrad, numDesign, numConstraints);
       setup2DArray(fPrev, numDesign, numObjectives);
       setup2DArray(fNext, numDesign, numObjectives);
       setup2DArray(fGrad, numDesign, numObjectives);
       setup2DArray(hessianPrev, numDesign, numDesign);
       setup2DArray(dx, numDesign, 1);
       g                      = new double[numConstraints];
       b                      = new double[numConstraints];
       lamda          = new double[numConstraints];
       gOpt           = new double[numConstraints];
       for(i=0;i<numConstraints;i++) {
          g[i] = 0;
          b[i] = 0;
          lamda[i] = 0;
          gOpt[i] = 0;
       }
       f                      = new double[numObjectives];
       o                      = new double[numObjectives];
       scale          = new double[numObjectives];
       fOpt           = new double[numObjectives];
       for(i=0;i<numObjectives;i++) {
          f[i] = 0;
          o[i] = 0;
          scale[i] = 0;
          fOpt[i] = 0;
       }
       xNext          = new double[numDesign];
       xRun           = new double[numDesign];
       xUpper         = new double[numDesign];
       xLower         = new double[numDesign];
       lGradPrev      = new double[numDesign];
       xOpt           = new double[numDesign];
       for(i=0;i<numDesign;i++) {
          xNext[i] = 0;
          xRun[i] = 0;
          xUpper[i] = 0;
          xLower[i] = 0;
          lGradPrev[i] = 0;
          xOpt[i] = 0;
       }

       return 0;
}

//------------------------------
// constructor for sqp
//------------------------------

int sqp()
{
       //variables NOT read in but ONLY calculated and used in calculations in part of
function
       setup2DArray(hessian, numDesign, numDesign);
       setup2DArray(gama, numDesign, 1);
       setup2DArray(gamaT, 1, numDesign);
       setup2DArray(dxT, 1, numDesign);
       setup2DArray(coeficiants, total, total);
       lGrad      = new double[numDesign];
       for(int i=0;i<numDesign;i++) {
          lGrad[i] = 0;
       }
       solution   = new double[total];
       other      = new double[total];
       for(i=0;i<total;i++) {
          solution[i] = 0;
          other[i] = 0;
       }

       return 0;
}

//------------------------------
// destructor for sqp
//------------------------------

int Tsqp()
{
   destroy2DArray(hessian, numDesign);
   destroy2DArray(gama, numDesign);
   destroy2DArray(coeficiants, total);
   destroy2DArray(gamaT, 1);
   destroy2DArray(dxT, 1);
   delete [] lGrad;
   delete [] solution;
   delete [] other;

   return 0;
}

//------------------------------
// destructor for optimization
//------------------------------

int Topt()
{
   destroy2DArray(dx, numDesign);
   destroy2DArray(hessianPrev, numDesign);
   destroy2DArray(gNext, numDesign);
   destroy2DArray(gPrev, numDesign);
   destroy2DArray(fNext, numDesign);
   destroy2DArray(fPrev, numDesign);
   destroy2DArray(gGrad, numDesign);
   destroy2DArray(fGrad, numDesign);
   delete [] g;
   delete [] xNext;
   delete [] xRun;
   delete [] xOpt;
   delete [] xUpper;
   delete [] xLower;
   delete [] b;
   delete [] lamda;
   delete [] gOpt;
   delete [] lGradPrev;
   delete [] f;
   delete [] o;
   delete [] scale;
   delete [] fOpt;

   return 0;
}

//-----------------------------------
// convinience functions for reading
//-----------------------------------

double* getRefStringDoubles(POM_Field* refs, POM_Field* strings) {
       double* tmp = new double[refs->length];
       for(int i=0;i<refs->length;i++) {
              tag_t tmp_tag_holder = refs->get_ref_at(i);
              POM_Class *tmp_class = new POM_Class(tmp_tag_holder, POM_no_lock);
              char* tmp_field = strings->get_value_at(i);
              tmp[i] = tmp_class->getField(tmp_field)->getDouble();
              delete tmp_field;
              delete tmp_class;
       }
       return tmp;
}

double* getFormName_FieldNameDoubles(EPM_action_message_t* message, POM_Field* names,
POM_Field* fields) {
       double* tmp = new double[names->length];
       for(int i=0;i<names->length;i++) {
              char* tmp_name = names->get_value_at(i);
              tag_t tmp_tag_holder = get_attachment_byname(message, tmp_name);
              delete [] tmp_name;
              Form *tmp_class = new Form(tmp_tag_holder, POM_no_lock);
              char* tmp_field = fields->get_value_at(i);
              tmp[i] = tmp_class->data->getField(tmp_field)->getDouble();
              delete [] tmp_field;
              delete tmp_class;
       }
       return tmp;
}

int setFormName_FieldNameDoubles(tag_t folder, POM_Field* names, POM_Field* fields,
double* values) {
       for(int i=0;i<names->length;i++) {
              char* tmp_name = names->get_value_at(i);
              tag_t tmp_tag_holder = get_attachment_byname(folder, tmp_name);
              delete [] tmp_name;
              Form *tmp_class = new Form(tmp_tag_holder, POM_modify_lock);
              char* tmp_field = fields->get_value_at(i);
              tmp_class->data->getField(tmp_field)->setValue(values[i]);
              delete [] tmp_field;
              delete tmp_class;
       }

       return 0;
}


double* getFormName_NameValueDoubles(EPM_action_message_t* message, POM_Field* forms,
POM_Field* valueNames) {
       double* tmp = new double[forms->length];
       for(int i=0;i<forms->length;i++) {

              //get the name of the name-value form for the i-th value
```

```
                char* tmp_formName = forms->get_value_at(i);

                //use the name to get the name-value form
                tag_t tmp_tag_holder = get_attachment_byname(message, tmp_formName);
                if(tmp_tag_holder == NULL_TAG){
                        printf("Cannot find %s amonge attachments\n",tmp_formName);
                        return NULL;
                }
                delete [] tmp_formName;
                Form *tmp_class = new Form(tmp_tag_holder, POM_no_lock);

                //get the value name for the i-th value
                char* tmp_valueName = valueNames->get_value_at(i);

                //search for value name in the names field of the name-value form
                POM_Field *NameValue_Names = tmp_class->data->getField("names");
                POM_Field *NameValue_Values = tmp_class->data->getField("values");
                for(int j=0;j<NameValue_Names->length;j++){
                        char* jth_name = NameValue_Names->get_value_at(j);
                        if(strcmp(tmp_valueName,jth_name) == 0){

                                //get the corresponding value in the values field of the
name-value form
                                char* value_to_store = NameValue_Values->get_value_at(j);

                                //convert to double and store the value in the i-th
position in the returning array
                                tmp[i] = atof(value_to_store);
                                delete [] value_to_store;
                                delete [] jth_name;
                                break;
                        }
                        delete [] jth_name;
                }

                delete [] tmp_valueName;
                delete tmp_class;
        }
        return tmp;
}

int setFormName_NameValueDoubles(tag_t folder, POM_Field* forms, POM_Field* valueNames,
double* values) {
        for(int i=0;i<forms->length;i++) {

                //get the name of the name-value form for the i-th value
                char* tmp_formName = forms->get_value_at(i);

                //use the name to get the name-value form
                tag_t tmp_tag_holder = get_attachment_byname(folder, tmp_formName);
                if(tmp_tag_holder == NULL_TAG){
                        printf("Cannot find %s amonge attachments\n",tmp_formName);
                        return 1;
                }
                Form *tmp_class = new Form(tmp_tag_holder, POM_modify_lock);
                                                    //TRACK: class_instance here was the
error!! POM_no_lock changed to POM_modify_lock

                //get the value name for the i-th value
                char* tmp_valueName = valueNames->get_value_at(i);
                //printf("setting form <%s> name <%s> to value
<%lf>\n",tmp_formName,tmp_valueName,values[i]);
                delete [] tmp_formName;

                //search for value name in the names field of the name-value form
                POM_Field *NameValue_Names = tmp_class->data->getField("names");
                POM_Field *NameValue_Values = tmp_class->data->getField("values");
                        //TRACK: class_instance
                for(int j=0;j<NameValue_Names->length;j++){
                        char* jth_name = NameValue_Names->get_value_at(j);
                        if(strcmp(tmp_valueName,jth_name) == 0){
        //TRACK: position

                                //set the corresponding value in the values field of the
name-value form
                                char buf[100];
                                sprintf(buf,"%lf",values[i]);
        //TRACK: val
                                //printf("set Form Name Value from found match\nsetting %s
at %d\n",buf,j);
                                NameValue_Values->set_value_at(buf,j);
        //TRACK: val   TRACK: position  TRACK: class_instance
                                //printf("set Form Name Value from spq cleaning memory from
temperary string\n");
                                delete [] jth_name;
                                //printf("set Form Name Value from spq breaking from
search\n");
                                break;
                        }
                        //printf("set Form Name Value from spq searching for
%s\n",tmp_valueName);
                        delete [] jth_name;
                }

                //printf("set Form Name Value from spq cleaning memory from the value name
string\n");
                delete [] tmp_valueName;
                //printf("set Form Name Value from spq cleaning memory from the form\n");

                delete tmp_class;
        }

        return 0;
}



//-------------------------------------
// read info for sqp
//---------------------------------------

int read(EPM_action_message_t* message)
{
        int i,j;

        //**********read in preferences*************

        //get form
        prefs = get_attachment_byname(message,"sqp-preferences");
        Prefs   = new Form(prefs, POM_no_lock);

        //get values
        numDesign              = Prefs->data->getField("numDesign")->getInt();
        numConstraints         = Prefs->data->getField("numConstraints")->getInt();
        numObjectives  = Prefs->data->getField("numObjectives")->getInt();
        N                              = Prefs->data->getField("N")->getInt();
        diffType               = Prefs->data->getField("diffType")->getInt();
        tol                    = Prefs->data->getField("tol")->getDouble();
        gradDx                 = Prefs->data->getField("gradDx")->getDouble();

        //allocate memory
        opt();

        xUpper  = Prefs->data->getField("xUpper")->getDoubles();
        xLower  = Prefs->data->getField("xLower")->getDoubles();
        b              = Prefs->data->getField("b")->getDoubles();
        o              = Prefs->data->getField("o")->getDoubles();
        scale   = Prefs->data->getField("scale")->getDoubles();
        xRun    = getFormName_NameValueDoubles(message,Prefs->data-
>getField("designForms"),Prefs->data->getField("designFields"));
```

```
        f               = getFormName_NameValueDoubles(message,Prefs->data-
>getField("objectiveForms"),Prefs->data->getField("objectiveFields"));
        g               = getFormName_NameValueDoubles(message,Prefs->data-
>getField("constraintForms"),Prefs->data->getField("constraintFields"));
        delete Prefs;


        //***********read parameters*************
        //get form
        params = get_attachment_byname(message,"sqp-parameters");
        Param   = new Form(params, POM_no_lock);

        //get values
        if(count > 0 ) {
                hessianPrev    = Param->data->getField("hessianPrev")->getDoubles2D();
                fNext   = Param->data->getField("fNext")->getDoubles2D();
                fPrev   = Param->data->getField("fPrev")->getDoubles2D();
                fGrad   = Param->data->getField("fGrad")->getDoubles2D();
                gNext   = Param->data->getField("gNext")->getDoubles2D();
                gPrev   = Param->data->getField("gPrev")->getDoubles2D();
                gGrad   = Param->data->getField("gGrad")->getDoubles2D();
                gOpt    = Param->data->getField("gOpt")->getDoubles();
                xOpt    = Param->data->getField("xOpt")->getDoubles();
                fOpt    = Param->data->getField("fOpt")->getDoubles();
                penalty = Param->data->getField("penalty")->getDouble();
        }
        lamda   = Param->data->getField("lamda")->getDoubles();
        lGradPrev= Param->data->getField("lGradPrev")->getDoubles();
        double* dx1D = new double[numDesign];
        dx1D    = Param->data->getField("dx")->getDoubles();
        for(i=0;i<numDesign;i++)
                dx[i][0] = dx1D[i];
        delete [] dx1D;
        delete Param;


        //***********read in status**************

        //get form
        stats = get_attachment_byname(message,"sqp-status");
        Stats   = new Form(stats, POM_no_lock);

        //get values
        storeAsOpt             = Stats->data->getField("storeAsOpt")->getInt();
        finished               = Stats->data->getField("finished")->getInt();
        count                  = Stats->data->getField("count")->getInt();
        iteration              = Stats->data->getField("iteration")->getInt();
        numDesignProbed = Stats->data->getField("numDesignProbed")->getInt();
        deltaPenalty   = Stats->data->getField("deltaPenalty")->getDouble();
        delete Stats;

        //calc other status vars
        total = numDesign+numConstraints;
        num2probe = numDesign;
        if(diffType == 0)
                num2probe += numDesign;


        //store relavant run data
        if(storeAsOpt == 1)
        {
          for(i=0;i<numDesign;i++)
                  xOpt[i] = xRun[i];
          fOpt[0] = f[0];
          for(i=0;i<numConstraints;i++)
                  gOpt[i] = g[i];
        } else {
          i = (numDesignProbed-1)%numDesign;
          switch(diffType)
          {
          case -1:
                  fPrev[i][0] = f[0];
                  for(j=0;j<numConstraints;j++)
                        gPrev[i][j] = g[j];
                  break;
          case 0:
                  if((numDesignProbed-1)<numDesign)
                  {
                        fNext[i][0] = f[0];
                        printf("fNext = %lf\n",fNext[i][0]);
                        for(j=0;j<numConstraints;j++)
                           gNext[i][j] = g[j];
                  } else {
                        fPrev[i][0] = f[0];
                        for(j=0;j<numConstraints;j++)
                           gPrev[i][j] = g[j];
                  }
                  break;
          case 1:
                  fNext[i][0] = f[0];
                printf("fNext = %lf\n",fNext[i][0]);
                  for(j=0;j<numConstraints;j++)
                        gNext[i][j] = g[j];
                  break;
          default:
                  fPrev[i][0] = f[0];
                  for(j=0;j<numConstraints;j++)
                        gPrev[i][j] = g[j];
                  break;
          }
        }

        return 0;
}

//-----------------------------
// writes info for sqp
//-----------------------------

int write(tag_t folder){
        tag_t cpParams, cpStats, cpPrefs;
        //write stats
        cpStats = get_attachment_byname(folder,"sqp-status");
        Stats   = new Form(cpStats, POM_modify_lock);
        Stats->data->getField("iteration")->setValue(iteration);
        Stats->data->getField("count")->setValue(count);
        Stats->data->getField("finished")->setValue(finished);
        Stats->data->getField("storeAsOpt")->setValue(storeAsOpt);
        Stats->data->getField("deltaPenalty")->setValue(deltaPenalty);
        Stats->data->getField("numDesignProbed")->setValue(numDesignProbed);
        delete Stats;

        //write params
        cpParams = get_attachment_byname(folder,"sqp-parameters");
        Param   = new Form(cpParams, POM_modify_lock);
        double *dx1D = new double[numDesign];
        for(int i=0;i<numDesign;i++)
                dx1D[i] = dx[i][0];
        Param->data->getField("dx")->set_array(dx1D, numDesign);
        delete [] dx1D;
        Param->data->getField("fOpt")->set_array(fOpt, numObjectives);
        Param->data->getField("penalty")->setValue(penalty);
        Param->data->getField("xOpt")->set_array (xOpt, numDesign);
        Param->data->getField("lGradPrev")->set_array (lGradPrev, numDesign);
        Param->data->getField("lamda")->set_array (lamda, numConstraints);
        Param->data->getField("gOpt")->set_array (gOpt, numConstraints);
        Param->data->getField("gGrad")->set_array (gGrad, numDesign, numConstraints);
        Param->data->getField("gPrev")->set_array (gPrev, numDesign, numConstraints);
        Param->data->getField("gNext")->set_array (gNext, numDesign, numConstraints);
        Param->data->getField("fGrad")->set_array (fGrad, numDesign, numObjectives);
        Param->data->getField("fPrev")->set_array (fPrev, numDesign, numObjectives);
        Param->data->getField("fNext")->set_array (fNext, numDesign, numObjectives);
        Param->data->getField("hessianPrev")->set_array (hessianPrev, numDesign,
numDesign);
        delete Param;

        //write prefs
        cpPrefs = get_attachment_byname(folder,"sqp-preferences");
        Prefs   = new Form(cpPrefs, POM_modify_lock);
printf("storing new design variables into spq-prefences\n");
        setFormName_NameValueDoubles(folder,Prefs->data->getField("designForms"),Prefs-
>data->getField("designFields"),xNext);                     //DEBUG: function of no
return
        delete Prefs;


        return 0;
}


//------------------------------
// calculate the gradiants
//------------------------------

int calcGradiants()
{
   int i,j;

   switch(diffType)
   {
   case -1:
      //calc backward
      for(i=0;i<numDesign;i++)
      {
         fGrad[i][0] = backward(fPrev[i][0],fOpt[0],gradDx);
         for(j=0;j<numConstraints;j++)
            gGrad[i][j] = backward(gPrev[i][j],gOpt[j],gradDx);
      }
      break;
   case 0:
      //calc center
      for(i=0;i<numDesign;i++)
      {
         fGrad[i][0] = center(fPrev[i][0],fNext[i][0],gradDx);
         for(j=0;j<numConstraints;j++)
            gGrad[i][j] = center(gPrev[i][j],gNext[i][j],gradDx);
      }
      break;
   case 1:
      //calc forward
      for(i=0;i<numDesign;i++)
      {
         fGrad[i][0] = forward(fOpt[0],fNext[i][0],gradDx);
         for(j=0;j<numConstraints;j++)
            gGrad[i][j] = forward(gOpt[j],gNext[i][j],gradDx);
      }
      break;
   default:
      //calc backward
      for(i=0;i<numDesign;i++)
      {
         fGrad[i][0] = backward(fPrev[i][0],fOpt[0],gradDx);
         for(j=0;j<numConstraints;j++)
            gGrad[i][j] = backward(gPrev[i][j],gOpt[j],gradDx);
      }
      break;
   }

/*   printf("\ngadients f");
   for(i=0;i<numDesign;i++){
          printf("\n\t%7lf",fGrad[i][0]);
   }
   printf("\ngradiant g");
   for(i=0;i<numDesign;i++){
          printf("\n");
          for(j=0;j<numConstraints;j++){
                 printf("\t%7lf",gGrad[i][j]);
          }
   }*/

   return 0;
}


//---------------------------------------------
// calculate the ghessian of the langranian
//---------------------------------------------

int calcHessian()
{
   int i,j;

   if(iteration > 0)
   {
          for(i=0;i<numDesign;i++)
          {
                lGrad[i] = fGrad[i][0];
                for(j=0;j<numConstraints;j++)
                {
                      lGrad[i] -= lamda[j]*gGrad[i][j];
                }
          }
/*printf("\nlGrad");
for(i=0;i<numDesign;i++)
       printf("\n\t%7lf",lGrad[i]);*/

      for(i=0;i<numDesign;i++)
      {
         gama[i][0] = lGrad[i] - lGradPrev[i];
         gamaT[0][i] = gama[i][0];
         dxT[0][i] = dx[i][0];
      }

      //BFGS approximation of Hessian of the Lagrangian
      double **temp1, **temp2, **temp3, **temp4;
      setup2DArray(temp1, numDesign, numDesign);
      setup2DArray(temp2, numDesign, numDesign);
      setup2DArray(temp3, numDesign, numDesign);
      setup2DArray(temp4, numDesign, numDesign);

      //third part
      multiply(hessianPrev,dx,temp1,numDesign,numDesign,1);
      multiply(temp1,dxT,temp2,numDesign,1,numDesign);
      multiply(temp2,hessianPrev,temp3,numDesign,numDesign,numDesign);
      multiply(dxT,hessianPrev,temp1,1,numDesign,numDesign);
      multiply(temp1,dx,temp2,1,numDesign,1);
      for(i=0;i<numDesign;i++)
         for(j=0;j<numDesign;j++)
            temp1[i][j] = -temp3[i][j]/temp2[0][0];

      //second part
      multiply(gama,gamaT,temp2,numDesign,1,numDesign);
      multiply(gamaT,dx,temp3,1,numDesign,1);

      for(i=0;i<numDesign;i++)
         for(j=0;j<numDesign;j++)
            temp4[i][j] = temp2[i][j]/temp3[0][0];

      //add them with the previous hessian
      add(temp1,temp4,temp2,numDesign,numDesign);
      add(hessianPrev,temp2,hessian,numDesign,numDesign);

      destroy2DArray(temp1, numDesign);
      destroy2DArray(temp2, numDesign);
      destroy2DArray(temp3, numDesign);
      destroy2DArray(temp4, numDesign);

   } else {
            for(i=0;i<numDesign;i++) {
                  for(j=0;j<numDesign;j++) {
                        if(i == j)
                              hessian[i][j] = 1;
                        else
                              hessian[i][j] = 0;
                  }
            }
   }
/*   printf("\nhessian matrix");
   for(i=0;i<numDesign;i++){
         printf("\n\t");
         for(j=0;j<numDesign;j++){
               printf("%7lf  ",hessian[i][j]);
         }
   }*/
   printf("\n");
   return 0;
}


int solve()
{
   int i;
   //int j;

   //fill coeficiants and other arrays
   for(i=0;i<total;i++)
   {
      if(i<numDesign)
         other[i] = -fGrad[i][0];                   //TODO fix this for multiple
objectives
      else
         other[i] = -g[i-numDesign];
      for(int j=0;j<total;j++)
         if(i<numDesign)
            if(j<numDesign)
               coeficiants[i][j] = (hessian[i][j]+hessian[j][i])*.5;
            else
               coeficiants[i][j] = -gGrad[i][j-numDesign];
         else
            if(j<numDesign)
               coeficiants[i][j] = gGrad[j][i-numDesign];
            else
               coeficiants[i][j] = 0;
   }


   //solve for dx's and lamda's
   if(gauss(coeficiants,other,total,solution,.01)==1)
   {
      printf("\n\nERROR in gauss\n\n");
      for(i=0;i<total;i++)
         if(i<numDesign)
            solution[i] = .5*dx[i][0];
         else
            solution[i] = lamda[i-numDesign];
   }


   return 0;
}



//-----------------------------------------------
```

```
// calculate the penalty and change in penalty
//--------------------------------------------

double calcPenalty()
{
        double temp=0;
        for(int i=0;i<numObjectives;i++)
                temp += scale[i]*f[i];
        for(i=0;i<numConstraints;i++)
                if(g[i] > b[i])
                        temp += lamda[i]*g[i];
        double delta = penalty - temp;
        if(delta < 0)
                penalty = temp;

        return delta;
}

//----------------------------------------------------
// changes values for calculating next gradiant point
//----------------------------------------------------

int storeGradPt()
{
   int i;
   for(i=0;i<numDesign;i++)
   {
      xNext[i] = xOpt[i];
      if(i == numDesignProbed%numDesign)
      {
         switch(diffType)
         {
         case -1:
            xNext[i] -= gradDx;
            break;
         case 0:
            if(numDesignProbed<numDesign)
               xNext[i] += gradDx;
            else
               xNext[i] -= gradDx;
            break;
         case 1:
            xNext[i] += gradDx;
            break;
         default:
            xNext[i] -= gradDx;
            break;
         }
      }
   }
   numDesignProbed++;

   return 0;
}

//----------------------------------------------------
// changes values for calculating next optimum point
//----------------------------------------------------

int storeOptPt()
{
   int i,j;
   deltaPenalty = 1;
   numDesignProbed = 0;
   iteration++;
   storeAsOpt = 1;

   for(i=0;i<numConstraints;i++)
      lamda[i] = solution[numDesign + i];
/*printf("\nlamda");
for(i=0;i<numConstraints;i++)
```

```
       printf("\n\t%7lf",lamda[i]);*/

   double max = 0;
   for(i=0;i<numDesign;i++)
   {
         lGradPrev[i] = fGrad[i][0];
         for(j=0;j<numConstraints;j++)
         {
               lGradPrev[i] -= lamda[j]*gGrad[i][j];
         }
      dx[i][0] = solution[i];
      xNext[i] = xOpt[i] + dx[i][0];
      if(xNext[i] > xUpper[i])
         xNext[i] = xUpper[i];
      if(xNext[i] < xLower[i])
         xNext[i] = xLower[i];
      if(fabs(dx[i][0]) > max)
         max = fabs(dx[i][0]);
   }
/*printf("\nlGradPrev");
for(i=0;i<numDesign;i++)
       printf("\n\t%7lf",lGradPrev[i]);*/

   if(max < tol || count > N )
      finished = 1;

   for(i=0;i<numDesign;i++)
      for(j=0;j<numDesign;j++)
         hessianPrev[i][j] = hessian[i][j];




   return 0;
}

//-------------------------------------------------------
// changes values for calculating next penalty
//-------------------------------------------------------

int storePenaltyPt()
{
   double max = 0;
   for(int i=0;i<numDesign;i++)
   {
      dx[i][0] = dx[i][0]*0.5;
      xNext[i] = xOpt[i] - dx[i][0];
      if(xNext[i] > xUpper[i])
         xNext[i] = xUpper[i];
      if(xNext[i] < xLower[i])
         xNext[i] = xLower[i];
      if(fabs(dx[i][0]) > max)
         max = fabs(dx[i][0]);
   }

   if(max < tol)
      finished = 1;

   return 0;
}


//------------------------------
// finite differencing algorithms
//------------------------------

double forward(double f, double fNext, double dx)
{
   return ( fNext - f ) / dx;
}
```

```
double backward(double fPrev, double f, double dx)
{
   return ( f - fPrev ) / dx;
}

double center(double fPrev, double fNext, double dx)
{
   return ( fNext - fPrev ) * .5 / dx;
}




//-----------------------------------------
// linear solving by gauss elimination
//-----------------------------------------

int gauss(double **a,double *b, int n, double *x, double tol)
{
   int i=0, j=0;
   double *s = new double[n];
   int er=0;
   for(i=0;i<n;i++)
   {
      s[i] = fabs(a[i][0]);
      for(j=1;j<n;j++)
         if(fabs(a[i][j])>s[i])
            s[i]=fabs(a[i][j]);
   }
   eliminate(a,s,n,b,tol,er);
   if(er == -1)
   {
      return 1;
   }
   substitute(a,n,b,x);
   return 0;
}

int eliminate(double **a, double *s, int n, double *b, double tol, int er)
{
   int i=0,j=0,k=0;
   double factor;
   for(k=0;k<n-1;k++)
   {
      pivot(a,b,s,n,k);
      if(fabs(a[k][k]/s[k])<tol)
      {
         er = -1;
         return 1;
      }
      for(i=k+1;i<n;i++)
      {
         factor = a[i][k]/a[k][k];
         for(j=k+1;j<n;j++)
         {
            a[i][j] = a[i][j] - factor*a[k][j];
         }
         b[i] = b[i] - factor*b[k];
      }
   }
   if(fabs(a[k][k]/s[k]) < tol)
   {
      er = -1;
      return 1;
   }
   return 0;
}

int pivot(double **a, double *b, double *s, int n, int k)
{
   int i=0,j=0,p=k;
   double big = fabs(a[k][k]/s[k]);
   double dummy;
   for(i=k+1;i<n;i++)
   {
      dummy=fabs(a[i][k]/s[i]);
      if(dummy > big)
      {
         big = dummy;
         p = i;
      }
   }
   if(p != k)
   {
      for(j=k;j<n;j++)
      {
         dummy = a[p][j];
         a[p][j] = a[k][j];
         a[k][j] = dummy;
      }
      dummy = b[p];
      b[p] = b[k];
      b[k] = dummy;
      dummy = s[p];
      s[p] = s[k];
      s[k] = dummy;
   }
   return 0;
}

int substitute(double **a, int n, double *b, double *x)
{
   int i=0,j=0;
   double sum=0;
   x[n-1] = b[n-1]/a[n-1][n-1];
   for(i=n-2;i>=0;i--)
   {
      sum=0;
      for(j=i+1;j<n;j++)
      {
         sum += a[i][j] * x[j];
      }
      x[i] = (b[i] - sum)/a[i][i];
   }
   return 0;
}

//-------------------------------
// matrix operations
//-------------------------------

int add(double **a, double **b, double **sum, int ni, int nj)
{
   for(int i=0;i<ni;i++)
      for(int j=0;j<nj;j++)
         sum[i][j] = a[i][j] + b[i][j];
   return 0;
}

int multiply(double **a, double **b, double **product, int ai, int aj, int bj)
{
   for(int i=0;i<ai;i++)
      for(int j=0;j<bj;j++)
      {
         product[i][j] = 0;
         for(int k=0;k<aj;k++)
            product[i][j] += a[i][k]*b[k][j];
      }
   return 0;
}
```

```
void storeInfo(void){
	/*
	tag_t record_tag = get_item("luke_test", "Form");
	Form* record_form = new Form(record_tag, POM_modify_lock);

	char buf[32];
	sprintf(buf,"%lf",f[0]);
	record_form->data->getField("f")->set_value_at("buf",1);

	sprintf(buf,"%lf",g[0]);
	record_form->data->getField("g")->set_value_at("buf",1000);

	sprintf(buf,"%d",iteration);
	record_form->data->getField("iteration")->set_value_at("buf",1000);

	sprintf(buf,"%d",count);
	record_form->data->getField("run")->set_value_at("buf",1000);

	sprintf(buf,"%lf",xRun[0]);
	record_form->data->getField("x1")->set_value_at("buf",1000);

	sprintf(buf,"%lf",xRun[1]);
	record_form->data->getField("x2")->set_value_at("buf",1000);
	delete record_form;*/
	printf("\n\t\t\t\t%d_%d  f=%7lf  g=%7lf  x1=%7lf
x2=%7lf\n",iteration,count,f[0],g[0],xRun[0],xRun[1]);
}
```

APPENDIX B: EXTERNAL INTEGRATION

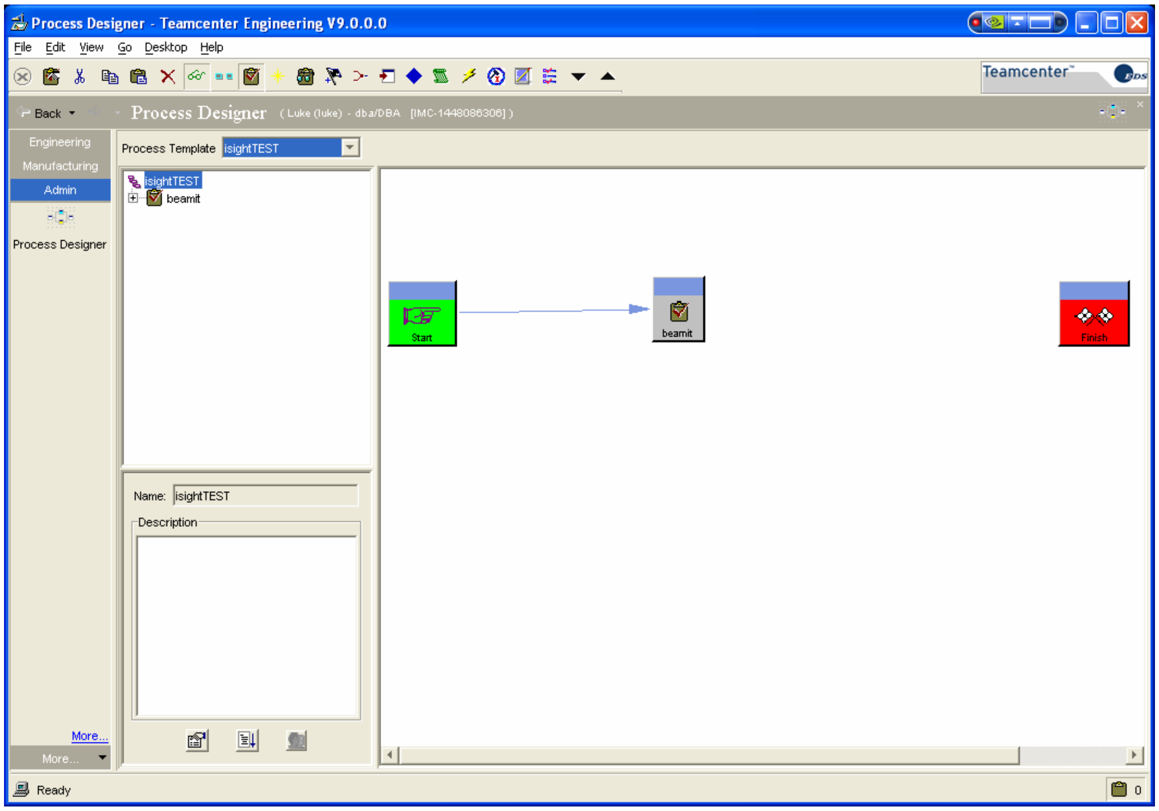
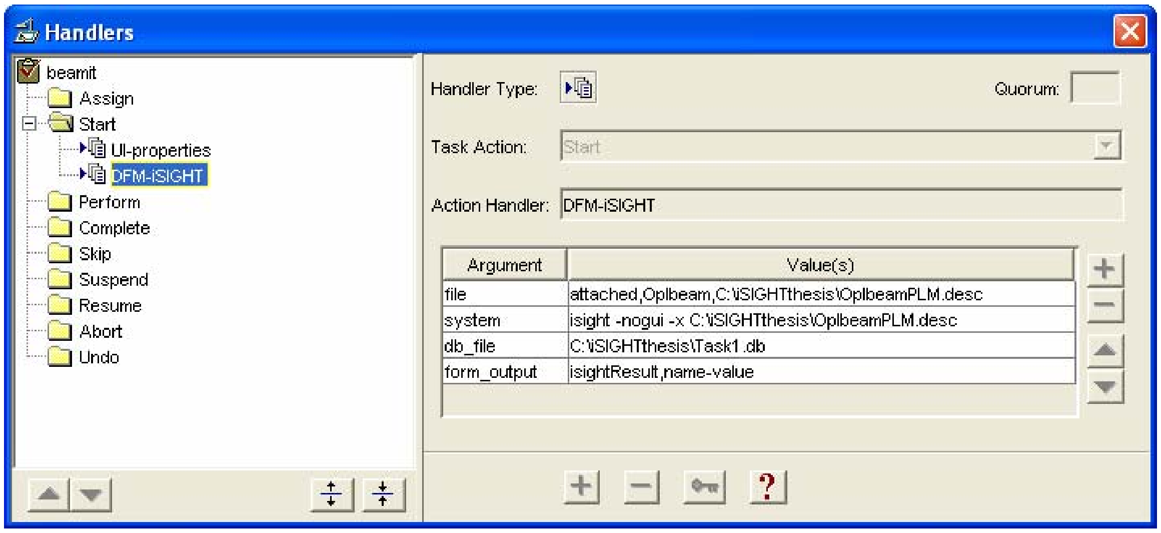

Process Designer - Teamcenter Engineering V9.0.0.0
File Edit View Go Desktop Help
Teamcenter
Back
Process Designer (Luke (luke) - dba/DBA [IMC-1448086306])
Engineering
Manufacturing
Admin
Process Designer
Process Template isightTEST
isightTEST
beamit
Name: isightTEST
Description
Start
beamit
Finish
More...
More...
Ready

Figure 23 The PLM workflow process designer. Design for the external method includes only an iSIGHT task.



Handler Type:

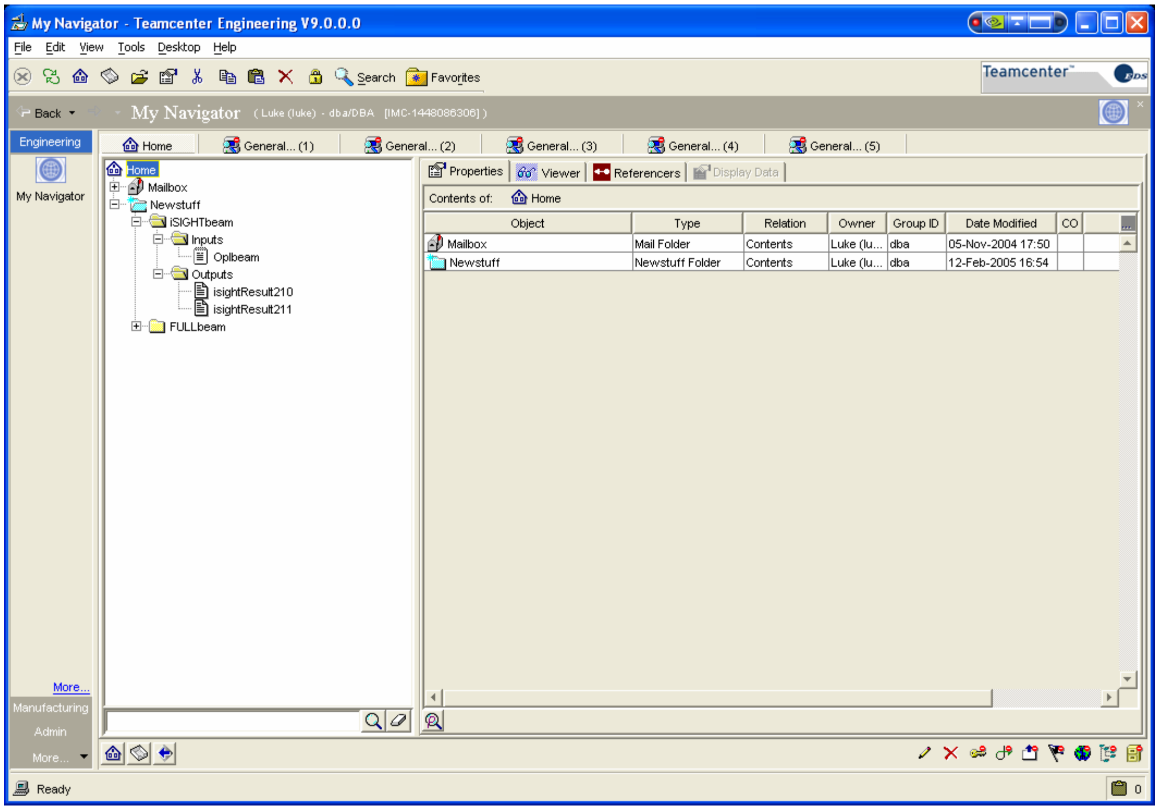
Quorum:

Task Action: Start

Action Handler: DFM-iSIGHT

Argument	Value(s)
file	attached,Oplbeam,C:\iSIGHTthesis\OplbeamPLM.desc
system	isight -nogui -x C:\iSIGHTthesis\OplbeamPLM.desc
db_file	C:\iSIGHTthesis\Task1.db
form_output	isightResult,name-value

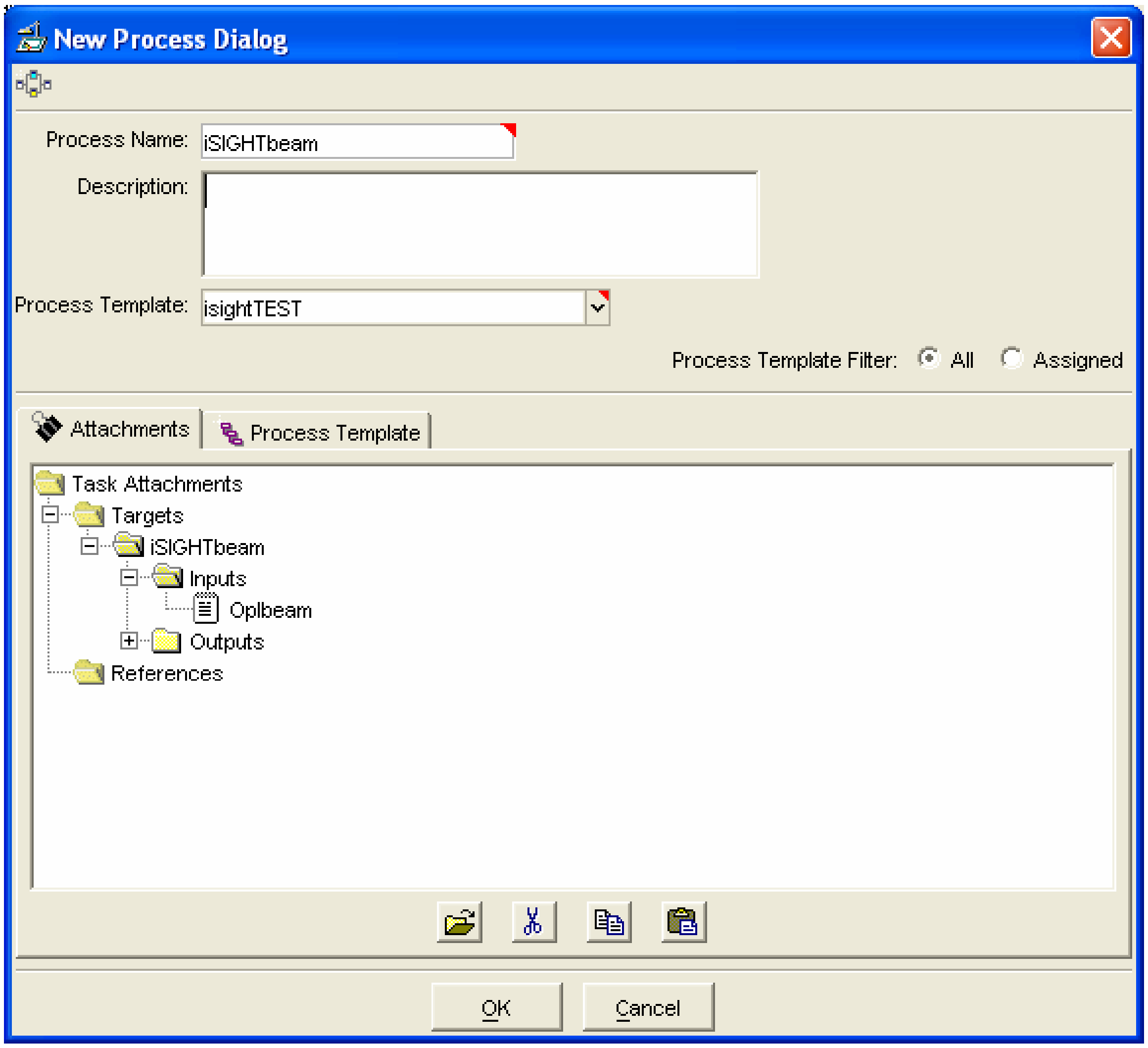
Figure 24 The iSIGHT task action handler. The handler accepts four arguments: The iSIGHT description file. The system call to execute the iSIGHT run. The form where the outputted results are stored.



Contents of: Home

Object	Type	Relation	Owner	Group ID	Date Modified	CO
Mailbox	Mail Folder	Contents	Luke (lu...	dba	05-Nov-2004 17:50	
Newstuff	Newstuff Folder	Contents	Luke (lu...	dba	12-Feb-2005 16:54	

Figure 25 External method attached folders and forms. The Inputs folder contains only the iSIGHT description file. The Outputs folder need not contain anything.



New Process Dialog

Process Name: iSIGHTbeam

Description:

Process Template: isightTEST

Process Template Filter: [✓ All] Assigned

Figure 26 Dialog to initiate a new process from the external method template created in the process designer. The attached folder is shown.

iSIGHT Description File:

```
MDOLVersion: 9.0
CompilerOptions: warn

Task Task1

    TaskHeader Task1
        Version: 1.0
        Evaluation: doestudy surface
        ControlMode: user
        RunCounter: 29
        BoundsPolicy: adjustvalue
        CheckPoint: unknown
    End TaskHeader Task1

    Inputs Task1
        Parameter: Length Type: real InitialValue: 8.0
        Parameter: Height Type: real InitialValue: 3.0
        Parameter: Webth Type: real InitialValue: 0.0501
        Parameter: Width Type: real InitialValue: 2.0
        Parameter: Flangeth Type: real InitialValue: 0.089685
        Parameter: Load Type: real InitialValue: 1500.0
    End Inputs Task1

    Outputs Task1
        Parameter: Volume Type: real
        Parameter: Stress Type: real
        Parameter: Displace Type: real
    End Outputs Task1

    SimCode ibeamANSYS
        InputFiles ibeamANSYS
            FileDescription paramstxt
                FileType: standard
                TemplateFile: "params.template"
                InputFile: "params.txt"
                Parameters
                    Length Height Webth Width Flangeth Load
                Instructions
                    require  Length Height Webth Width Flangeth Load
                        find "Length= " ignore
                        replace word with $Length
                        find "Height= " ignore
                        replace word with $Height
                        find "Width= " ignore
                        replace word with $Width
                        find "Web_th= " ignore
                        replace word with $Webth
                        find "Flange_th= " ignore
                        replace word with $Flangeth
                        find "Load= " ignore
                        replace word with $Load
                End Instructions
            End FileDescription paramstxt
        End InputFiles ibeamANSYS

        OutputFiles ibeamANSYS
            FileDescription ibeamout
                FileType: standard
                OutputFile: "ibeam.out"
                Parameters
                    Volume Stress Displace
                Instructions
                    find "IBeam Volume:  " ignore
                    read Volume as "%f"
                    provide $Volume
                    find "Max longitudinal stress:   " ignore
                    read Stress as "%f"
                    provide $Stress
                    find "Max displacement: " ignore
                    read Displace as "%f"
                    provide $Displace
                End Instructions
            End FileDescription ibeamout
        End OutputFiles ibeamANSYS

        SimCodeProcess ibeamANSYS
            ScriptLanguage: DOSBatch
            Script
                C:\Progra~1\AnsysI~1\v81\ANSYS\bin\intel\ansys81.exe -b -p ansysrf -j
JobName -i C:\iSIGHTthesis\ibeam.mac -o C:\iSIGHTthesis\LogFileName.out
            End Script
            ProcessType: transient
            Environment: unrestored
            ElapseTime: 5m
            Prologue
                WriteInputSpecs: paramstxt
            Epilogue
                ReadOutputSpecs: ibeamout
        End SimCodeProcess ibeamANSYS

    End SimCode ibeamANSYS

    TaskProcess Task1
        Control: [
            ibeamANSYS
        ]
    End TaskProcess Task1

    Optimization Task1
        PotentialVariables:
            Length Height Webth Width Flangeth Load
        Variables:
            Webth Flangeth Height
        VariableScaling
            Parameter: Length ScaleFactor: 1.0
            Parameter: Height ScaleFactor: 1.0
            Parameter: Webth ScaleFactor: 1.0
            Parameter: Width ScaleFactor: 1.0
            Parameter: Flangeth ScaleFactor: 1.0
            Parameter: Load ScaleFactor: 1.0
        InputConstraints
            Parameter: Height LowerBound: 0.75 UpperBound: 3.0
            Parameter: Webth LowerBound: 0.05 UpperBound: 1.0
            Parameter: Flangeth LowerBound: 0.05 UpperBound: 1.0
        PotentialObjectives:
            Volume Stress Displace Length Height Webth Width Flangeth Load
        Objectives
            Parameter: Volume Direction: minimize Weight: 1.0 ScaleFactor: 1.0
        OutputConstraints
            Parameter: Stress UpperBound: 25000.0 Weight: 1.0 ScaleFactor: 1.0
            Parameter: Displace UpperBound: 0.01 Weight: 1.0 ScaleFactor: 1.0

        OptimizePlan midterm
            DefaultUpperBound: 1.0E15
            UseScaling: yes
            OptimizeStep Step1
                Technique: "Generalized Reduced Gradient - LSGRG2"
                Prologue
                    RestoreBestSolution: no
                    RerunTask: no
                Epilogue
                    RestoreBestSolution: yes
                    RerunTask: no
                Options
                    ConvergenceEpsilon: 0.001
                    GradientStepSize: 0.001
            Control: [
                Step1
            ]
        OptimizePlan SQP
            DefaultUpperBound: 1.0E15
            UseScaling: yes
            OptimizeStep Step1
                Technique: "Sequential Quadratic Programming - NLPQL"
                Prologue
                    RestoreBestSolution: no
                    RerunTask: no
                Epilogue
                    RestoreBestSolution: yes
                    RerunTask: no
                Options
            Control: [
                Step1
            ]

        # PLAN TO BE CONFIGURED BY ADVISOR:
        OptimizePlan PriorityRankedPlan
            Control: [
            ]
    End Optimization Task1

    DesignOfExperiments Task1
        Plan DOEPlan1
            Technique: "CentralComposite"
            Factors
                ParameterList
                    Type: control
                    Parameters
                        Height BaseLine: 2.6 Levels: values [ 2.3 2.4 2.6 2.8 3.0 ]
Alpha: 2.0 LowerLevel: 2.4 UpperLevel: 2.8
                        Webth BaseLine: 0.0501 Levels: values [ 0.0499268 .05 0.0501 1.0
1.6953268 ] Alpha: 1.732 LowerLevel: .05 UpperLevel: 1.0
                        Flangeth BaseLine: 0.089685 Levels: values [ 0.02095058 .05
0.089685 1.0 1.66635058 ] Alpha: 1.732 LowerLevel: .05 UpperLevel: 1.0
                End ParameterList
            End Factors
        End Plan DOEPlan1

        Study surface
            Plan: DOEPlan1
            Responses
                Outputs:
                    ObjectiveAndPenalty
            End Responses
            Actions
                Objective: ObjectiveAndPenalty
                Direction: minimize
            End Actions
            ResultsFile: "doe_Study.surface"
            Prologue
                Tcl
                End Tcl
            Epilogue
                Tcl
                End Tcl
        End Study surface
    End DesignOfExperiments Task1

    TaskPlan Task1
        StopTaskPlanOnError: no
        Control: [
            midterm
            SQP
        ]
    End TaskPlan Task1

    DataStorage Task1
        Restore: no
        DataLog: "Task1.db" Mode: overwrite
        DataLookUp: "Task1.db"
        MatchMode: Exact
        Levels: all
        StoreGradRuns: yes
        StoreApproxRuns: yes
    End DataStorage Task1

End Task Task1
```

External Method ANASYS Macro Code:

```
/com,starting

FINISH
/CLEAR
!/CWD,'C:\Documents and Settings\Nathaniel\My Documents\School\isightSide\ANSYS'
/INPUT,params,txt

! Load IGES file
/AUX15
! ~UGIN,ibeam,prt,'..\CAD\',SOLIDS,1,0 !***Edit this line

! Go into the preprocessor
/prep7
!RECTNG,4,-4,2,1.5,
RECTNG,-Width/2,Width/2,Height/2,Height/2-Flange_th
RECTNG,-Width/2,Width/2,-Height/2,-Height/2+Flange_th
RECTNG,-Web_th/2,Web_th/2,Height/2-Flange_th,-Height/2+Flange_th
AADD,ALL
VOFFST,4,Length, ,


! Define element types
ET,1,MESH200
KEYOPT,1,1,6
KEYOPT,1,2,0
ET,2,SOLID45

! Define material properties
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,30e6
MPDATA,PRXY,1,,.3
MPDATA,dens,1,,.0007


! Create a volume if necessary
allsel,all
*get,volumeCount,volu,,count
*if,volumeCount,eq,0,then
   nummrg,kp,7e-4,7e-4,,low
   va,all
*endif


! Get the front area number (loadArea)
areaNum=0
minCentZ=1000
allsel,all
*get,areaCount,area,,count
*do,i,1,areaCount,1
   asel,all
   areaNum=arnext(areaNum)
   asel,s,,,areaNum
   asum
   *get,centZ,area,,cent,z
   *if,centZ,lt,minCentZ,then
      minCentZ=centZ
      loadArea=areaNum
   *endif
*enddo

! Get the back area number (fixArea)
areaNum=0
minCentZ=0
allsel,all
*do,i,1,areaCount,1
   asel,all
   areaNum=arnext(areaNum)
   asel,s,,,areaNum
   asum
   *get,centZ,area,,cent,z
   *if,centZ,gt,minCentZ,then
      minCentZ=centZ
      fixArea=areaNum
   *endif
*enddo


! Mesh the top area
myesize=Web_th/2
*if,Web_th,gt,Flange_th,then
   myesize=Flange_th/2
*endif

!asel,s,,,loadArea
!lsla,s
!*get,lineCount,line,,count
!lineNum=0
!*do,i,1,lineCount,1
!   lsla,s,
!   lineNum=lsnext(lineNum)
!   lsel,s,,,lineNum
!   lesize,lineNum,myesize
!*enddo

! Mesh the top area
asel,s,,,loadArea
lsla,s
*get,lineCount,line,,count
lineNum=0
*do,i,1,lineCount,1
   lsla,s,
   lineNum=lsnext(lineNum)
   lsel,s,,,lineNum
!   *get,lineLength,LINE,lineNum,LENG
!   *if,lineLength,eq,Height-2*Flange_th,then
!      myesize = lineLength/3
!   *endif
   lesize,lineNum,myesize
*enddo


asel,s,,,loadArea
TYPE,1
MAT,1
REAL,
ESYS,0
SECNUM,
MSHAPE,0,2D
MSHKEY,0
amesh,all


! Set number of divisions on the lines
lsel,all
asel,s,,,loadArea
asel,a,,,fixArea
lsla,u
lesize,all,,,16,,,,,0 ! This puts n divisions on all lines selected


! Sweep mesh the volume
allsel,all
TYPE,2
MAT,1
REAL,
ESYS,0
SECNUM,
!*
```

```
MSHAPE,0,3D
!*
VSWEEP,all

! Constrain root face of airfoil
DA,fixArea,ALL,


! Find number of nodes in the web area
asel,s,,,loadArea
nsla,s,1
cm,loadNodes,node
*get,nodeCount,node,,count
nodeNum=0
minX=Web_th/2
numNodesToLoad=0
*do,i,1,nodeCount,1
   cmsel,s,loadNodes
   nodeNum=ndnext(nodeNum)
   nsel,s,,,nodeNum
   *get,xloc,NODE,nodeNum,loc,x
   *if,xloc,le,minX,then
      *if,xloc,ge,-minX,then
         numNodesToLoad=numNodesToLoad+1
      *endif
   *endif
*enddo

nodeForce=Load/numNodesToLoad


! Apply forces to nodes in the web area
asel,s,,,loadArea
nsla,s,1
cm,loadNodes,node
*get,nodeCount,node,,count
nodeNum=0
minX=Web_th/2
numNodesToLoad=0
*do,i,1,nodeCount,1
   cmsel,s,loadNodes
   nodeNum=ndnext(nodeNum)
   nsel,s,,,nodeNum
   *get,xloc,NODE,nodeNum,loc,x
   *if,xloc,le,minX,then
      *if,xloc,ge,-minX,then
         F,nodeNum,FY,nodeForce
      *endif
   *endif
*enddo



! Solve it
/solu
allsel,all
solve




! Extract information to an output file
! Output solution analysis objectives to a file
*cfopen,ibeam,out
*vwrite
Ansys output file
*vwrite
(' ')
! Measure volume of airfoil (representative of mass)
/prep7
vsel,all
```

```
vsum
*get,vol,volu,,volu
*get,centX,volu,,cent,x
*get,centY,volu,,cent,y
*get,centZ,volu,,cent,z

*vwrite,vol
IBeam Volume: %14.7G



! Get max principal stress
/post1

nsort,s,1,0
*GET,logtmax,SORT, ,MAX

*vwrite,logtmax
Max longitudinal stress: %14.7G


! Get max displacement
nsort,u,sum,0
*GET,dymax,SORT, ,MAX

*vwrite,dymax
Max displacement: %14.7G
*cfclose
/eof
/VIEW,1,1,2,3
/ANG,1
/AUTO,1
/RGB,INDEX,100,100,100,0
/RGB,INDEX,0,0,0,15
/VCONE,ALL,45.0
/DEV,PSFN,NINC
/gfile,400
PLNSOL,S,EQV
```

APPENDIX C: ANALYSIS AND OPTIMIZATION RESULTS

Analytical Solution for I-beam stress and deflection:

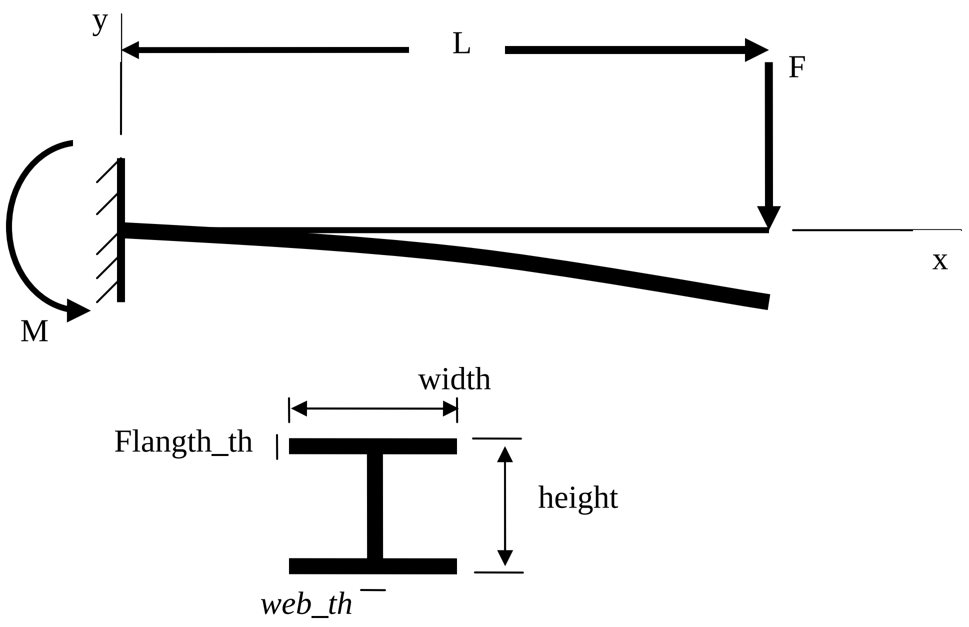


Figure 27 Cantilever beam - end load.

The following equations are used to find the maximum stress and deflection in the I-beam:

$$M_{\max} = FL \tag{14}$$

$$I = \frac{width \cdot height^3 - (width - web\_th) \cdot (height - 2 \cdot flange\_th)^3}{12} \quad (15)$$

$$\sigma_{\max} = \frac{M_{\max} c}{I} \quad (16)$$

$$y_{\max} = -\frac{FL^3}{3EI} \quad (17)$$

These equations were used to find the theoretical stress and displacement. These values were then compared to the numerical stress and displacement found using ANSYS. The results obtained varied proportionally for multiple designs which qualifies the use of the numerical model in the optimization.

Optimization Results:

Table 6 The optimal design. The design is at the minimum web thickness and the maximum height. The deflection constraint is binding.

Flange_th	Web_th	Height	Volume	Stress	Deflection
0.08968	0.05000	3.00000	3.99815	21092.98179	0.01000

The following contour plots illustrate the design space.

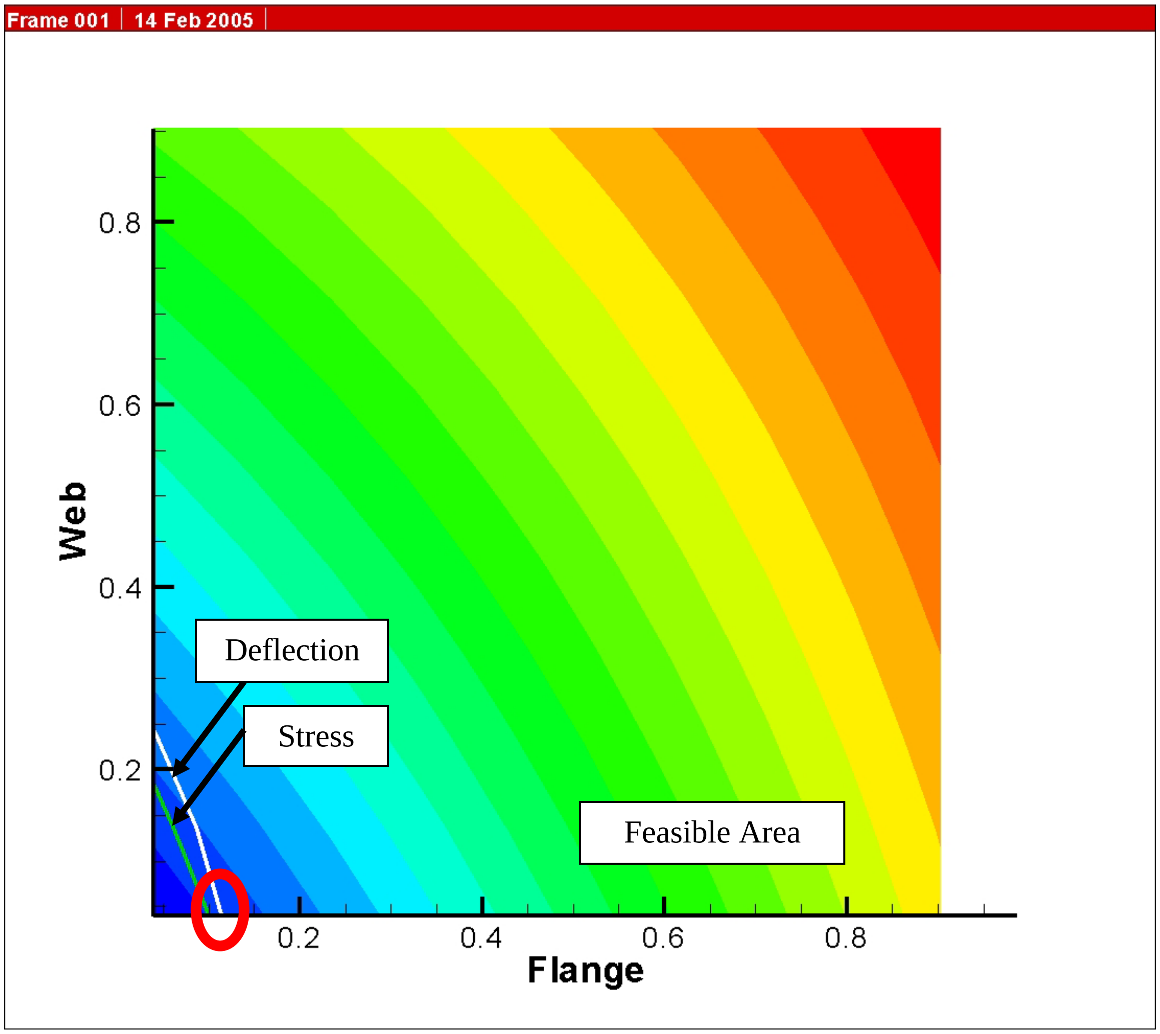


Figure 28 A slice of the design space at Height = 3.0. Volume is contoured with values decreasing toward the lower left corner. Deflection and Stress constraint boundaries are shown as lines. The opimal design is circled.

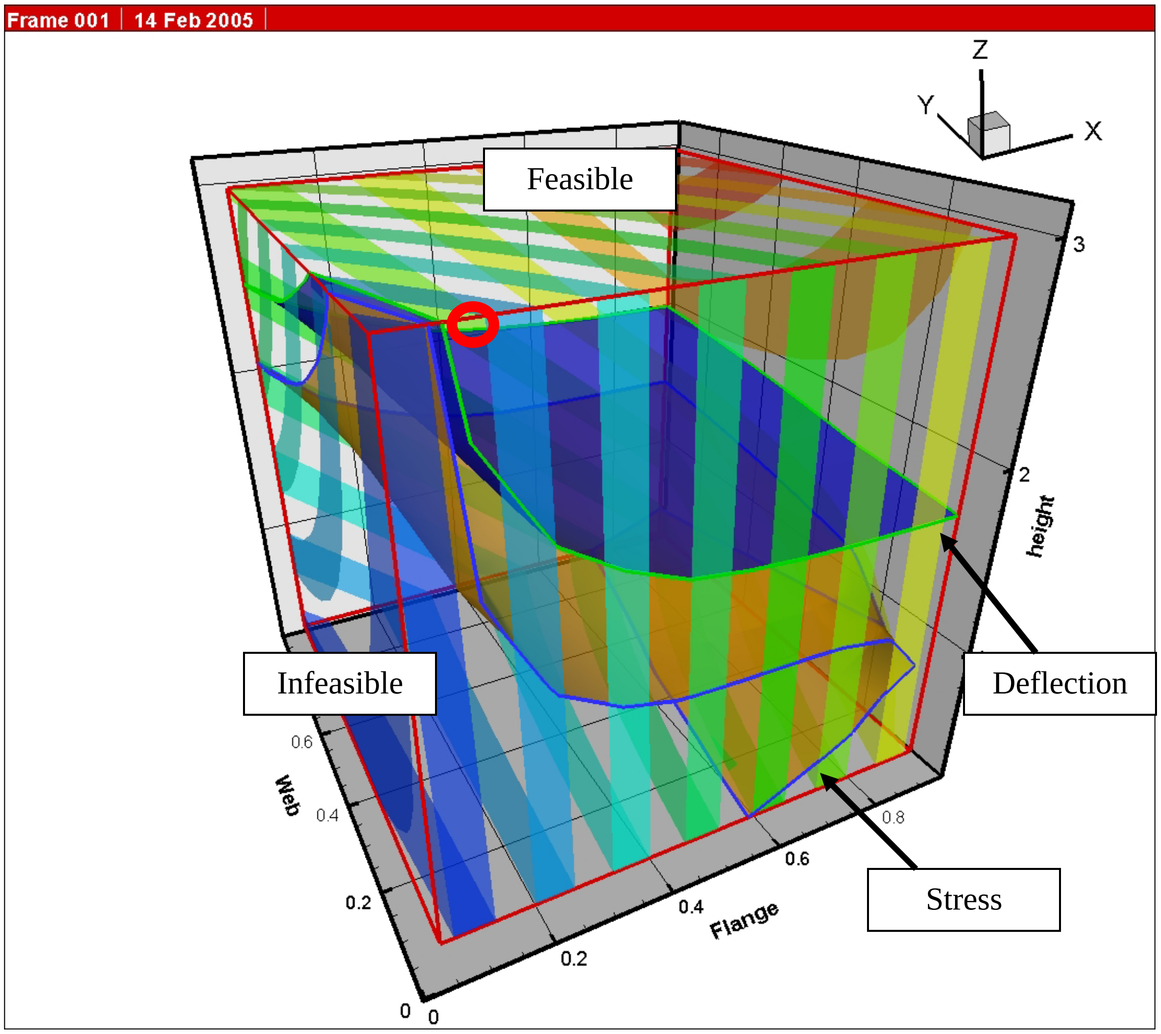


Figure 29 The design space. The space is contoured by Volume with lower corner at the origin having the smallest volume. The two surfaces displayed within the space represent the deflection and stress constraint boundaries. The optimal design is circled.