7-2008

ENHANCEMENT OF INFORMATION MANAGEMENT CAPABILITIES IN MDO FRAMEWORK

Santosh Hiriyannaiah
Clemson University, shiriya@clemson.edu

ENHANCEMENT OF INFORMATION MANAGEMENT CAPABILITIES IN MDO FRAMEWORK

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mechanical Engineering

---

by
Santosh Hiriyannaiah
August 2008

Accepted by:
Dr. Gregory M. Mocko, Committee Chair
Dr. Georges Fadel
Dr. Joshua D. Summers

ABSTRACT

Multidisciplinary Design Optimization (MDO) frameworks have been developed to facilitate the integration of disciplinary analysis codes and optimization techniques. Recent advances in MDO frameworks have addressed issues related to data exchange, distributed computing, process integration and trade study. However, managing, storing and sharing MDO problem information have not yet been fully addressed. In this research a software configuration is proposed. The configuration is built upon a structured repository, common file system and software applications. The configuration is integrated into a commercially available MDO framework to manage, store and share MDO problem information. A common file system proposed in this research provides a structure to store MDO components and enable sharing of components over the network. The ModelCenter framework is selected for the integration of the repository based on the evaluation of the MDO frameworks against a set of extended information management requirements. The repository is a relational database which provides an information model to store information related to MDO problems. A Java interface is utilized to provide access to the structured repository and the common file system in the ModelCenter framework. Java applications are developed to demonstrate the benefits offered by the proposed repository and the common file system. The proposed features and the Java applications are tested for the functionality and performance utilizing IEEE software testing standards.

DEDICATION

This thesis is dedicated to my parents who have a major contribution for what I am today and my sister who has helped and supported me to achieve my goals.

ACKNOWLEDGMENTS

I would like to thank Dr. Mocko for his continuous support and guidance without which this research would not have taken the present form. I would also like to express my gratitude to my committee members Dr. Fadel and Dr. Summers for their support and valuable advice.

I would like to thank Ben Caldwell, Pavan Kumar, Chiradeep Sen, Santosh Tiwari and Carl Lamar who have provided me timely feedback and have helped me review this thesis.

TABLE OF CONTENTS

Page

Table of Contents (Continued)

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

List of Figures (Continued)

Figure Page

CHAPTER ONE

INTRODUCTION

In recent years, emphasis has been on the advances that can be achieved with the interaction between two or more disciplines [1]. Many disciplines interact with each other to solve a complex engineering design problem. The design of an aircraft involves interaction between specialized disciplines such as aerodynamics, propulsion and structural analysis to mention a few, often times with conflicting objectives and constraints. The overall desired objective to increase aircraft's performance is divided into many sub level objectives which are solved separately by the specialized disciplines. In this process several analysis models are generated from each discipline and are brought together to optimize for the overall desired objectives using several optimization techniques. One such methodology which integrates analysis models and optimization techniques to solve engineering design problems involving multiple disciplines is called Multidisciplinary design optimization (MDO). A hardware and software architecture that enables integration, execution and communication among diverse disciplinary processes is referred as MDO framework [21]. The key requirements for architecting MDO frameworks and supporting features are identified by Salas in the early 1990s. These requirements include: architectural design; problem formulation; problem execution; and information access. Several commercially available and research-based software frameworks have been developed to enable disciplinary analysis code, geometric design models, and optimization routines to be coupled. The frameworks have fulfilled the requirements to varying degree. These frameworks include ModelCenter, iSIGHT-FD,

and modeFRONTIER. Several issues in these frameworks associated with integration, execution and communication have been addressed with significant contributions and advancements made by the MDO community. Advances in technologies such as distributed computing and data exchange are being incorporated in these frameworks. Analysis Server and CenterLink by Phoenix Integration and Fiper from Engineous software are examples of such advancements[10, 19, 20]. These developments help in the best utilization of resources and automate information exchange thereby making it easier for the designer.

While current frameworks enable models to be linked and information to be exchanged between heterogeneous codes, they do not provide sufficient information management capabilities. Current MDO frameworks do not provide a structured representation and information model for capturing information, such that designers can easily store, organize, and retrieve previous MDO decisions and projects for reuse. Thus, the principle objective in this research is to develop a structured repository (database) for capturing MDO related information to facilitate reuse, reconfiguration and exchange of MDO problems. The structured repository will enable information across disciplines to be shared such that the designer is benefited with the necessary prior information required to setup and solve an MDO problem.

1.1 Research Questions and Validation Plans

In order to achieve the above stated principle objective the following research questions have been proposed. Addressing these research questions not only help to achieve the principle objective, but also provide a road map for the research.

Research Question 1: What are the information management requirements of MDO framework to support reuse and reconfiguration?

Research Question 1 focuses on the identification of requirements for managing MDO related information. The correlations between current frameworks and the requirements enable gaps to be identified. These gaps include retrieval and reconfiguration of existing MDO problems; capture and storage of information for the integration of disciplinary analysis models; representing constraints and requirements in formulating MDO problems

Research Question 2: What are the features that need to be integrated into the MDO framework to enhance information management capabilities?

Research Question 2 focuses on developing features to enhance information management capabilities in MDO framework. This is done first, by evaluating currently available frameworks against the information management requirements and selecting a suitable framework for extension. Second, by identifying the drawbacks of the selected framework configuration and finally by identifying structured repository and common file system as features that help address these drawbacks and enhance the information management capabilities

Research Question 3: What is the structure of the information model to enable efficient reuse and reconfiguration in MDO problems?

The focus of Research Question 3 is to design and develop the conceptual information model of the repository. The information model provides MDO information to be stored in a structure and retrieved. This enables designers to reuse and reconfigure the MDO problems with the help of information from the repository.

Research Question 4: How will the repository be interfaced/ integrated with an MDO framework in general and ModelCenter/Analysis server specifically?

Research Question 4 focuses on the integration of the structured repository and the common file system in ModelCenter/Analysis Server configuration. A Java application is developed to connect the repository and to incorporate method calls provided by the ModelCenter and Analysis Server API's.

1.2 Thesis overview

A comprehensive set of requirements obtained from literature to understand the development of MDO frameworks is described in Chapter 2. Currently available features addressing these requirements are discussed in detail and need for reuse and reconfiguration of MDO problems is explained. Information management requirements are extended to support reuse and reconfiguration in Chapter 3. The three frameworks under study in this research are evaluated against these requirements and a suitable framework is selected. The drawbacks of the selected framework configuration are identified. A new ModelCenter – Analysis Server configuration is proposed to address the stated drawbacks. The proposed configuration is built upon a structured repository,

common file system and software applications. The design and development details of the proposed configuration are discussed in Chapter 4. The repository is implemented with information from a walkway beam structure analysis example problem in Chapter 5. The quality and performance of the features in the proposed configuration are tested and demonstrated under a scenario in Chapter 6. Finally the thesis is concluded in Chapter 7 by presenting a research summary which includes the advantages of enhancing information management capabilities and the future opportunities this research leads to.

CHAPTER TWO

LITRATURE REVIEW

The development of model integration frameworks and dedicated MDO software packages has been addressed from a research perspective (i.e., DAKOTA, FIDO, MIDAS) and from commercial software solution perspectives (i.e., iSIGHT-FD, ModelCenter, modeFRONTIER and LMS OPTIMUS). Additionally, many of the commercial software packages have evolved from research thrusts at universities and government research laboratories into commercially available software solutions. The available software solutions have strengths and shortcomings in the context of formulating engineering design problems, integrating disparate design and analysis models, representing mathematical solutions, and subsequently solving the MDO problems. These software frameworks differ in their ease of integration with existing design support tools, their human computer interface, their ability to capture the design intent, incorporating changes, providing optimization routines, tracking the information generated and many more. However, underlying each of the frameworks is the same set of core requirements. These requirements are grouped into Architectural Design requirements; Problem Formulation requirements; Problem Execution requirements and Information Access requirements [21]. The list of requirements is generated based on a review of the existing model integration framework literature, available common functionality from several software packages, and leveraging from information and knowledge reuse of complementary engineering domains. In this context a

comprehensive set of requirements for MDO software frameworks is developed and discussed in the following section based on a critical review of existing literature [15, 17, 21].

2.1 General requirements

Architectural requirements: Architectural requirements are generated to develop a method or style for designing a framework. These requirements are in terms of extensibility of the framework, incorporation of standards into the framework, use of existing legacy codes and collaborative design support.

- Incorporation of standards: Standards like message passing interface, database access and languages incorporated into the framework help reduce the maintenance cost and also preserve investments. [21]
- Extensibility: Modification of disciplinary analysis codes, integration of new processes into the system and incorporating changes in the design problem makes the framework more flexible. Incorporating new developments and technology will help the designer to continuously improve the design process. [21]
- Incorporation of legacy codes: Legacy codes exist in various forms. These codes are tested and proven over years; they are improved and expanded over time. Incorporation of these codes into the framework supports code reuse and also helps in achieving best results. The designer will be able to use codes with no changes required when incorporated in a framework. [21]
- Support for collaborative design: The architecture of the framework is important in collaborative design. The architecture should support versioning of documents to

prevent duplication of documents and to update the designers with the new version. Effective utilization of available resources supports complex multidisciplinary interactions. Multiple discipline designers can collaborate and work on the same problem by sharing the files over the network and utilizing the updated documents. It enables them to access the problem data at the same time. [17]

Problem Execution requirements: These requirements are generated in order to maintain and facilitate the execution of MDO problems. The key problem execution requirements are in terms of distributed computing, automated data transfer and automated problem execution

- Automation of problem execution: Input file preparation, the execution of disciplines and optimization methods, data extraction from output files and data transfer between processes should be automated in a framework. This helps in reducing design cycle time and also eliminates human intervention when not required. [17]
- Parallel processing: Distributed computing enables integrated product design, collaboration between multidisciplinary design teams and increase in computational speed. Parallel processing helps designers to work from different workstations and share design ideas at the same time. [17, 21]
- Creation of a wrapper: Creation of a wrapper helps in automating the data transfer and integrating various analysis codes from different disciplines. The framework should provide tools for creating wrappers generated by appropriate input files; invoke disciplinary programs and should automatically extract the output of interest [17].

Problem Formulation requirements: These requirements are generated to ease the formulation of MDO problems and to emphasize reuse and reconfiguration to make the framework flexible.

- Variable fidelity configurable models: Saving the assembly of linked codes and design exploration tools from the framework enables reuse. By facilitating the deletion or replacement of elements of various level of fidelity in the model, flexibility can be achieved. [15]
- Ease to reconfigure: Reconfiguration in problem formulation includes replacing existing processes with new ones, deleting processes or adding new ones to the application. Reconfiguration helps users to explore alternative views of the problem. Incorporating customizable tools and plug in components supports reconfiguration in a framework. [17]

Information Management requirements: Information management requirements pertain to structured storage and retrieval of optimization problems implemented in MDO frameworks and supporting analysis models. Specifically, these requirements define the architecture and interfaces for managing information.

- Modularity: The concept of modularity adapted in a framework makes the code more manageable and understandable. Modularity means that components of analysis and optimization tasks can be constructed from a library of interchangeable modules. Modularity also helps in understanding the design better and incorporates changes if necessary. [15, 21]

- Database Management: A central database for maintaining data used by multiple disciplines enables efficient numerical analysis, process restart capabilities and reuse of codes. It also encourages multidisciplinary analysis to reduce the number of translation routines needed. The option of defining which data is written to and read from the database, helps designers to manage and share information. An efficient search system enables effective retrieval of information from the database. [17]
- Plug and play user interface: The user should have the option of selecting from a set of analysis codes, linking and executing the codes in a process. This pick and place option illustrates a plug and play user interface. The components linked should be able to be broken, repaired or expanded as design requires. [14]

2.2 Available features in MDO frameworks

Three commercially available software frameworks are evaluated against the requirements described in Section 2.1. This evaluation is based on several information sources including user manuals [7-11, 20], software usage experiences [18], and informal interviews with graduate students currently conducting research utilizing the frameworks. The evaluation is summarized in Table 1. The software frameworks include primary and supporting software systems. The software includes:

- ModelCenter 7.1, Analysis Server 5.1, and CenterLink from Phoenix Integration
- iSIGHT-FD and FIPER from Engineous, and
- modeFRONTIER 3 from ESTECO

ModelCenter

Model Center is a visual environment for process integration developed by Phoenix integration. ModelCenter helps designers to integrate similar codes together and perform complex design analysis. The architecture includes simple Graphical User Interface which makes it easy to link applications, several trade study tools and optimization techniques. The architecture also includes a Java based software server called Analysis Server and a web based server called CenterLink to help support distributed computing, wrapping of software tools and data management. Java and COM API's are provided to call ModelCenter from external applications. These API's helps the designer to custom write their interface programs specific to their design process. Legacy codes can be incorporated with the help of a wrapper. An excel wrapper and a file wrapper wrap the analysis codes and make them available to designers by publishing them on a network with the help of an analysis server. ModelCenter allows wrapping analysis programs and running them in an automated fashion, linking multiple programs together to form systems engineering models, Perform trade studies on the models and Archive results from multiple trade studies into a single project [20]. Input and output from different components can be linked in ModelCenter Environment with the help of a key feature called Link Editor. Similar key features available in ModelCenter are auto link and link checker which makes the linking easier. The Scheduler feature helps in knowing which components need to be run and when. It allows mapping of design model graphically and run them in parallel. Valid and invalid are the two states given by the scheduler to show which model needs to be run. The plug-in tool kits are available to

extend the functionality of the framework. This is done by providing various plug-in types like trade study plug-Ins, component plug-Ins and data analysis plug-Ins. Variable influence profiler and prediction profiler features enables the designer to effectively formulate MDO problems. These features help the designer to analyze the most impacting input design variable on the output and choose the design parameters carefully. The other support tools available in the Model center are the optimization tool and data explorer tool. Visualization and analysis of the results in the trade study can be done using data explorer. These analysis results can be saved in an SQL database to share the results with other designers in the team. Optimization tool allows gaining insight into key design parameters and their impact in design process to efficiently find optimal design.

modeFRONTIER

modeFRONTIER is a multiobjective optimization and design environment which allows the designers to couple commercial analysis codes together in a design process. This optimization framework is developed by ESTECO. modeFRONTIER is implemented using Java and Inter platform communication is achieved using CORBA. It supports integration of various CAE tools such as CAD Finite Element Structural Analysis and Computational Fluid Dynamics (CFD) software. Problem formulation is done in a modular fashion with the help of node library. Node library contains node types or components which are used to formulate a process. Components can be picked and placed from the node library into the workflow as modeFRONTIER has a plug and play interface. Graphical process flow and Logic flow facilitates designer to keep track on changes and to ensure proper linking between components. Optimizer in

modeFRONTIER is referred to as scheduler. Base schedulers, advanced scheduler and evolutionary strategy schedulers are the three types of scheduler available. modeFRONTIER provides a plug-In interface for the third party optimization algorithms to be integrated into the framework. The Projects are multi-platform. Software like Catia, Pro-Engineer and Matlab can be directly integrated using their respective software nodes available in the node library. Other software can be executed using batch mode node. Summary of the design problem formulated and executed can be created by report module. After creating the report it can be shared with other designers using email node. Information about specific runs can be viewed by Runtime design space. Decision making tools such as goal programming, weighted sum of objectives and non linear utility function are provided to help to convert user intuitions or design knowledge into algorithms that extract best solutions.

iSIGHT-FD

iSIGHT-FD is a software framework developed by Engineous software helps improve the productivity in a design process. It automates the manual design process by integrating and coupling multidisciplinary simulation codes. The architecture of iSIGHT-FD includes Multidisciplinary optimization Language (MDOL), Tcl language interpreter engine and easy to use Graphical User Interface. Interpreter engine has the capability to receive and send commands and allows creating customized expressions at run time [7-10]. Design Gateway, Runtime Gateway, Component editors and Library are the four categories of interface available. These four interfaces help in formulation, execution, creating and publishing models. The Task plan functionality available in the Design

Gateway allows design drivers to be added to the work flow. The design drivers available in iSIGHT-FD are approximations, DOE, Monte Carlo simulation, and optimization tools. A special feature called pointer automatic optimizer is available which makes appropriate choices and determine which algorithms as well as their control parameters are most successful for the design. A set of resources can be connected to iSIGHT-FD using FIPER. It provides thin client interface using WebTop and also provides communication with WebLogic application Server. This facilitates for distributed computing to run jobs in a heterogeneous computing environment and provides access to Libraries and Databases. Application Control System (ACS) controls and manages these internal operations in FIPER. Model selector functionality allows the user to select several models and run them parallel. A process controller controls various job executions and communicates with the database. Database and description file store history of the project and a compressed XML files stores the characteristics of the component used in the design problem. The database component in iSIGHT-FD allows access to the database to store input and output values obtained after execution. The supported databases are MS access, oracle, DB2, MySQL and SQL Server. Data Exchanger is a unique feature in iSIGHT-FD that reads, writes and manages the data between two objects (parameter-text file, text file-text file). Engineering Data Mining (EDM) provides for post processing capabilities for multi objective optimization results to understand design better.

Table 2. 1: Identification of features addressing general requirements

General requirements	ModelCenter 7.1	modeFRONTIER 3	iSIGHT-FD 2.5
Architectural	Java and COM API's Simple intuitive GUI's Legacy codes can be incorporated with the help of wrapper	Java language Distributed object model RMI Inter platform communication using CORBA	Tcl language interpreter Simple intuitive GUI's MDOL
Problem execution	Link editor Analysis Server Center link Scheduler Plug-in tool kit	Batch mode execution for other software Projects are multi-platform	FIPER Parallel processing Web Top&Web Logic Runtime Gateway
Problem formulation	Analysis Wrapper File wrapper and Excel wrapper Plug and play interface Variable influence profiler Prediction profiler	Plug and play interface Node Library Graphical process flow and Logic log Plug in interface for third party algorithms Scheduler	Design Gateway Pointer automatic optimizer Plug and play interface Version control capability Design drivers
Information management	Data explorer SQL Database	Email node Report module Runtime design space view Remote monitoring for optimization progress	Database component ACS Data Exchanger Publisher

2.3 Need for Reuse and reconfigurability of MDO problems

Several researchers have described design as a decision making process and selection of design parameters represents decision [13]. Multidisciplinary design optimization is a decision making process where design decisions are the optimization based representation. Effective decisions can be taken if the designer formulating an MDO problem is given a set of choices of analysis code and optimization techniques along with information about them. These set of choices can be design variables that are

linked to various output parameters; several constraints that are imposed on an objective function; decisions taken by other designers; information about analysis code used and optimization techniques performed. Information about MDO problems stored can later be utilized to reuse and reconfigure the MDO problem. It is important to note that reuse and reconfiguration are not mutually exclusive. In this context the key definitions of reuse and reconfiguration follows.

Need For reuse

Several researchers have addressed the idea of reuse to help reduce integration gaps and to facilitate knowledge sharing. Grosse et al.[12] define reusability as ability to reuse an analysis model for the same applications by someone other than the model developer. Mocko and colleagues [16] discuss about reusability of behavioral models to benefit engineering design by capturing design analysis knowledge in a repository. While definitions change slightly with domains, reuse in the context of MDO problems can be defined as repetitive use of MDO components (analysis and optimization) and MDO project files stored in a file system for formulating similar design problems. Storing the MDO components would facilitate efficient reuse for formulating a new MDO problem. An example scenario to better understand reuse concept is presented below.

Reuse scenario

Consider a design of component from multiple disciplines where two experts are handling the same design problem i.e. an analyst and a designer. Analysts create disciplinary analysis code and store the file in a file system. Designers use this code to link it with an optimizer to form an MDO problem. The MDO problem is then executed

and the result is then analyzed using various trade study tools to study the effect of the design variables on the desired output. Therefore the code generated by an analyst is reused by the designer to formulate an MDO problem.

Need For reconfiguration

According to Alexandrov and Lewis [2, 3] most of the MDO formulations share the basic computational components, comprising output/input couplings and attendant sensitivity information. Here idea of reconfiguration is addressed in a mathematical approach. They define reconfiguration as a straight forward transformation among problem formulations with a single operation. In an MDO problem, reconfiguration can be defined as reassembling of analysis and optimization components to form a new design problem, with addition or deletion of components/objects to incorporate changes in design. In order to further explain the reconfiguration concept an example scenario is presented below.

Reconfiguration scenario

The MDO problem formulated by a designer is saved as a project file in a local file system with a specific file naming convention. This can only be accessed by other designers in the team if they are permitted access. When a requirement to create a similar design problem occurs such as change in optimization method, the saved project file is opened in a framework and changes are made by adding or deleting several design components. The same analysis file can be used with a different optimizer. Also if the design requirement changes, where analysis files used in the previous project can be used with minor modifications. For example analysis of a rectangular beam changes to

analysis of an I-beam. Thus, instead of formulating a new problem an existing project file can be used and reconfigured to meet new design changes.

Reuse and reconfiguration of MDO problem have potential to save time; reduce computational costs and also to speed up the formulation process. The main advantage of reuse and reconfiguration in MDO problem is (1) to gain knowledge about the outcome of the previous run, (2) to predict the impact of design parameters on the outcome if the designer is using similar analysis code, (3) to support the use of same analysis component to run in different optimization methods. To facilitate for efficient reuse and reconfiguration in MDO problem, capturing and storing information and meta information is essential. This requires an efficient information management system. However, information management tools available in the current frameworks are limited; resulting in difficulties with reuse and reconfiguration of design problem formulated/developed in MDO frameworks.

CHAPTER THREE

FRAMEWORK EVALUATION

As discussed in Chapter 2 problem formulation and information management are the key requirements that are not fully addressed. The main focus of this chapter is to emphasize the above mentioned key requirements to extend the information access capabilities in MDO frameworks. This is done by first applying and expanding on the general characteristics of the key requirements to facilitate reuse and reconfiguration of MDO problems. Subsequently, from these extended requirements, current MDO frameworks are evaluated and a suitable framework is selected for the extension. Drawbacks from the selected framework configuration are identified and new features to enhance information access capabilities are proposed.

3.1 Extended information management requirements

Database management: Capturing, storing, sharing and managing information related to MDO problems in a database helps in collaborative design [17]. Providing access to the information associated with the MDO problem and its components should facilitate intelligent querying of MDO components for reuse and MDO problems for reconfiguration. Database should store information shared by many disciplines such as design objectives, critical design considerations and requirements, important design parameters, design constraints, optimization technique used, results from previous runs, designer's rationale, decisions taken and component locations in the file system. Database management systems should help the designer to effectively utilize these MDO

information stored in a database during the three phases of problem formulation i.e. before formulation- when designer selects components for a particular MDO problem; during formulation- when linking of components to form an MDO problem; after formulation- when trade study and analysis is carried out.

Modularity in problem formulation: Formulation of MDO problem in a modular structure helps the framework to be flexible in incorporating changes. Modularity is breaking down of the components of a multidisciplinary design problem and information associated with it, such that reusable components can be identified. Modularity enhances transparency in problem formulation such that designers are able to view the dataflow and linking between components. This also helps in re-linking of components of a design problem.

Reconfiguration capability: Adding or replacing components in an MDO problem facilitates designer to explore alternative views in design. This helps in achieving desirable results in considerably less time and also avoids creation of new problems when a similar problem already exists. Loose coupling between MDO components ensures easy re-linking and re-assembling to incorporate changes in design. Features for automatic linking of the new component should be supported by the framework. Providing MDO components information to the designer is essential for a reconfiguration process.

Intelligent Search and retrieval: An interface providing information about the components stored in the file system enables intelligent search of that component. Querying service for the retrieval of MDO components enables the designer to effectively utilize the components already available for problem formulation. The framework should

support of such interface and querying services such that the components are intelligently search and retrieved.

Plug and play interface: A framework should have a plug and play interface which would allow the user to pick and place components from a file system into the design problem.

3.2 Evaluation of MDO Frameworks

Based on extended information management requirements from previous section, three readily available commercial frameworks are evaluated (see Table 2). This evaluation is based upon the features available in these frameworks that meet the extended requirements.

Table 3.1: Evaluation of information management requirements against MDO frameworks

Requirements	Model Center	mode FRONTIER	iSIGHT-FD
(1) Database Management	⭘	○	⭘
(2) Modularity in problem formulation	⭘	●	⭘
(3) Reconfiguration capability	⭘	○	⭘
(4) Intelligent search and retrieval	⭘	○	⭘
(5) Plug and Play interface	●	●	●
●- fully met ⭘ - partially met ○ - not met			

(1) Database Management requirement

ModelCenter - partially meets

ModelCenter partially meets the database management requirement. A centralized grid computing system called CenterLink supports database functionality by storing the trade study results in a standard SQL database for future reference. However this is

limited to trade study results and there in no much scope of storing information about MDO components. Analysis Server enables the analysis application to be reused by publishing them over a network. This application can be manipulated only by the analysts who create it and designers who have prior knowledge about the application. The information about the application is only made available by a naming convention, if followed. There is no file information system which gives information about where the file is stored.

modeFRONTIER - not met

modeFRONTIER does not meet the database management requirement as there are no database features available to store and retrieve the information about MDO problems. The files are saved in the local file system as a project file. These files are not available to any designer who doesn't have an access to the local file system as there are no distributed computing technologies in modeFRONTIER. This makes it difficult for data exchange and for the files to be shared across networks.

iSIGHT-FD - Partially met

iSIGHT-FD partially meets the database management requirement. Database currently available in iSIGHT-FD stores only input and output values obtained after execution. A library exists which stores all the design models which can be later reused by the designers to use it appropriately in their models. Fiper integration in iSIGHT-FD provides powerful distributed computing which accesses the library to allocate available resources. However there is no file system which helps in reuse of the components of design problem.

(2) Modularity in problem formulation

ModelCenter - Partially met

ModelCenter only partially supports modularity. Wrapping technology in Analysis Server allows analysis files to be decomposed into 3 parts input; output and executable. The analysis executable code is considered as a black box. The designer can only manipulate the code in terms of input and output and not the code parameters. In ModelCenter, MDO problem formulated has only two modules analysis module and optimization module. With this, the only modification done is by varying with the design variables, objectives constraints and optimization technique. The designer would not know what the code includes and thus is only limited to reuse analysis code as a black box. The information about the component can be viewed in the work flow, which facilitated the user to visualize the information flow and the link editor helps to define the data flow.

modeFRONTIER- Fully met

modeFRONTIER formulates the work flow in a modular way. It provides input variable icon, output variable icon, objective icon, constraint icon (referred as nodes) and a scheduler thereby making it easy for the designer to understand the workflow. Re linking input and output variables can be easily done as the work flow is set up in a modular fashion. The executed design problem is saved as a project file and the whole project file has to be used to make it reusable. Reconfiguring the existing design problem is a cumbersome procedure in modeFRONTIER.

iSIGHT-FD-Partially met

Component generator enables the wrapping of the analysis files similar to the one in ModelCenter. The design problem is formulated in a modular fashion by dragging and dropping components into the workspace. The design problems are saved as project files and can be retrieved and used as a whole problems and reconfiguring it is a difficult process. Change in design required re parsing of input and output files which is almost equivalent to creating a new design problem. The Library feature provides brief description about the component which can be replaced. Hence it partially meets the modularity requirement.

(3) Reconfiguration capability

ModelCenter - Partially met

ModelCenter environment provides good linking between components and can be re-linked with ease with the help of auto link and link editor features. The components can be created added and replaced easily but only with the help of knowledge of the designer, who knows what to add and why to replace a component. There is no feature to provide information to the designer about the components before the files are opened in ModelCenter.

modeFRONTIER- Not met

Once the work flow is fully defined it can be re-linked with the help of link creator. But if the variables are to be changes then a whole new process of data mining has to take place in order to make the project executable. There is a logic log window which helps the designer to formulate the design problem. However it is limited to only the components added to the work flow. It does not suggest addition of new component

as there is no information system or an expert system like knowledge repository supported in modeFRONTIER.

iSIGHT-FD-Partially met

iSIGHT-FD partially meets the reconfiguration capability requirement. The component can be easily added or replaced in the work flow in Design Gateway. The design models once formulated can be reconfigured only in the design work flow. This reconfiguration is done by re linking of the new variables and parsing the output files. But it does not provide any information to the user which would help reconfiguration. Hence it still is a difficult process to edit the existing problem to form a new design problem.

(4) Intelligent Search and retrieval

ModelCenter - Partially met

ModelCenter partially supports for intelligent search and retrieval. A server browser facilitates for access to the Analysis Server file system from ModelCenter environment where the designer can view the models before using them. But this is effective only when the designer has prior information about the files stored in the server. But it just shows the files in that folder. It does not show what those files contain and does not tell how to use those files in a design problem. There is no information system to provide information about the usability of that component. Moreover if the files are not saved in a proper file naming convention it would be difficult to identify the files which need to be used again in a new design problem.

modeFRONTIER - Not met

modeFRONTIER does not meet intelligent search and retrieval requirement. There are no search techniques incorporated for the retrieval of components required in design problem. The only way of retrieving is by storing the file in a particular folder and opening it from the framework. modeFRONTIER does not provide any interface for search and retrieval of components.

iSIGHT-FD-Partially met

iSIGHT-FD partially meet intelligent search and retrieval requirement. It does provide description about the component after the components are opened. There are no querying services for the retrieval of component stored in the file system. The library feature provides a brief description about the component and also stores the version of the component.

(5) Plug and play interface

ModelCenter - Fully met

ModelCenter work environment enables plug and play interface such as server browser, where the user can pick/select components from Analysis Server and place them into the work flow window.

modeFRONTIER -Fully met

modeFRONTIER enables plug and play interface by allowing the users to pick the components ore nodes from the node library and place them into the work flow window.

iSIGHT-FD- Fully met

iSIGHT-FD's interface provides efficient plug and play interface where the user can select from a set of components, pick and place them into the work flow window with the help of a design gateway. Drag and drop functionality is also available for the published components to add them to the workflow

3.3 Suitable Framework for Research in Reuse and Reconfigurability

From the evaluation made ModelCenter is chosen for extending the information management capability. The developments incorporated in ModelCenter in terms of component sharing over the network by Analysis Server and interface to this server by server browser makes ModelCenter a suitable framework to extend information management capability. This is achieved by developing a structured repository and a file information system to provide information about the components stored in the server file system (see Figure 3.1). Phoenix integration's CenterLink which is a powerful support tool for grid computing can also be utilized for the design process execution. Java and COM API's provided in ModelCenter helps in creating an application to call the ModelCenter and Analysis Server method externally. These API's also helps to custom write their interface programs specific for reuse and reconfiguration of MDO problems. With these functionalities provided by Phoenix integration in ModelCenter, it gives us sufficient opportunity to extend the information management capabilities and emphasize on reuse and reconfiguration of design models to help the designer to efficiently formulate the design problem.

3.4 ModelCenter software and hardware configuration

The current software and hardware configuration in ModelCenter, Analysis Server, and CenterLink is as shown in Figure 3.1. The (1) Analysis Server is a Java-based software server which "publishes" available analysis models over the network. The models are located on the (2) Server File System and are published using (A) Java-based Wrapper technologies.

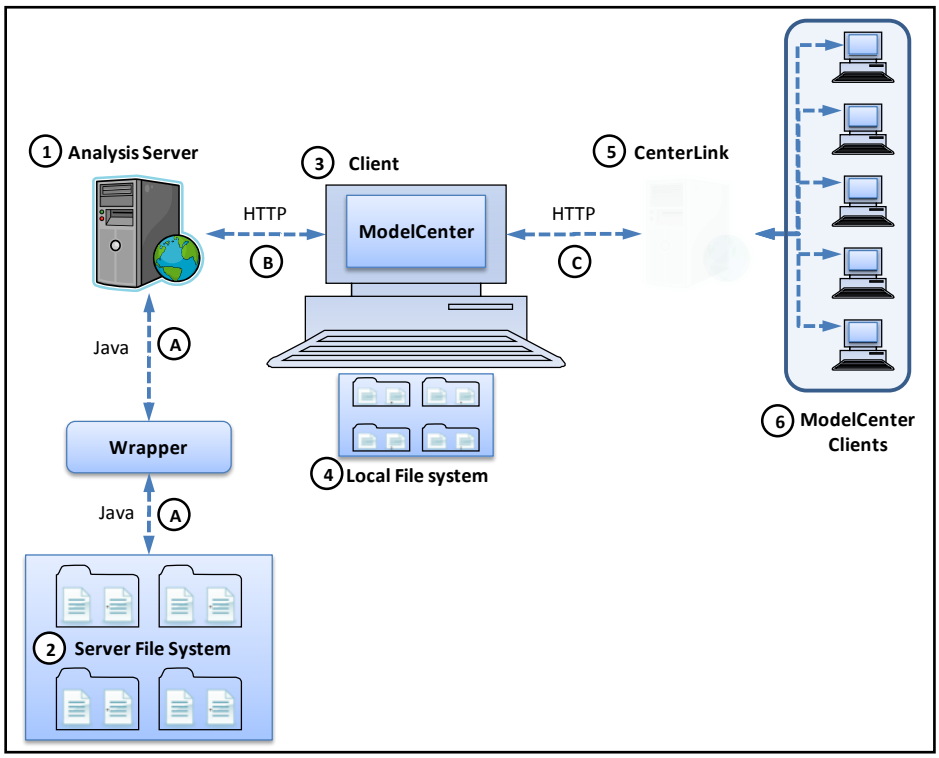


Figure 3.1: Current ModelCenter configuration

The (3) ModelCenter client is a process integration engine which utilizes models from Analysis Server through a (B) protocol similar to HTTP. Multiple analysis models can be accessed from multiple servers. Information about the models including: name, author, date, and a brief description are available in ModelCenter (see Figure 3.2).

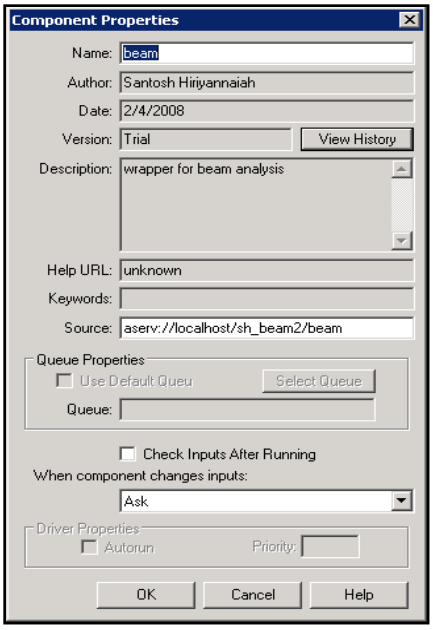


Figure 3.2: Information describing analysis models available on Analysis Server

The Analysis Server and ModelCenter follow a client-server architecture that enables distributed resourced to be utilized. Analysis applications are served from the Analysis Server to the ModelCenter client, where the analysis models are integrated into a design process. These design processes enable trade studies, optimization, and general model integration to be performed. This is then saved as a project file on the (4) Local file system of the ModelCenter client. Computationally expensive engineering design problems can be solved more quickly by taking advantage of distributed resources. CenterLink (5) is an environment to manage and distribute the job execution over several (6) ModelCenter clients available over the network.

3.5 Usage of analysis model in current ModelCenter – Analysis Server configuration

In the current ModelCenter – Analysis Server configuration, the plug-ins available are limited for the direct integration of the analysis models using analysis software. Models are wrapped with the help of file wrapper utility and stored in the Analysis Server. The file wrapper is a text file with a .fileWrapper extension that contains information about how to execute the analysis [20]. It provides instructions to generate an input file, run analysis software in batch mode, and parse the result to an output file.

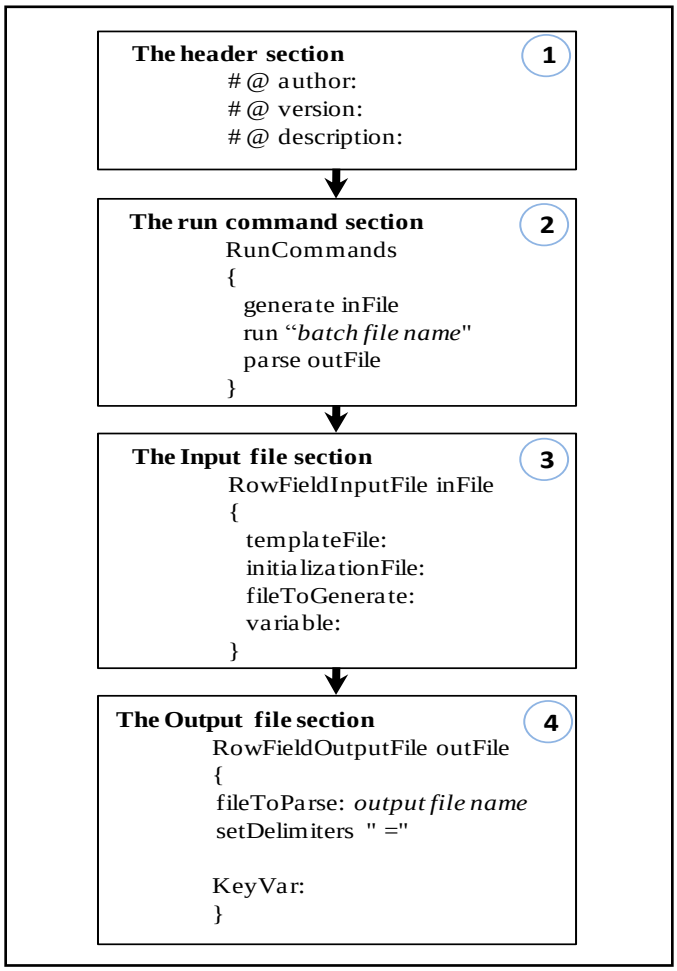


Figure 3.3: Sections of file wrapper

The structure of the four main sections in the file wrapper provided by the ModelCenter documentation is as shown in Figure 3.3. Section (1) is the header section which includes creation details of the file wrapper such as author name, version and description. Section (2) is the run command section which calls the batch file to run analysis in batch mode. Section (3) is the input file section which contains the name of the analysis file to be used and the corresponding input variable details. Section (4) is the output file section which contains the name of the output file generated by the analysis model along with the output variable details.

The steps involved in the creation of the file wrapper are as shown in the flow chart in Figure 3.4. File wrapper creation is a 9 step process which involves main steps such as identification of information associated with the analysis model, writing the four sections of the file wrapper with proper syntax and saving the file wrapper in Analysis Server. The information required to create the file wrapper are input and output variable details; name of the output file generated by the analysis model; name of the batch file to execute analysis in batch mode and the name of the input file to be generated by the wrapper. This information has to be identified by the user creating the file wrapper which required prior knowledge about the analysis models. It is also important to ensure that the file wrapper, along with the corresponding batch file and analysis models should be stored in the same folder in the Analysis Server for the file wrapper to be executed in ModelCenter.

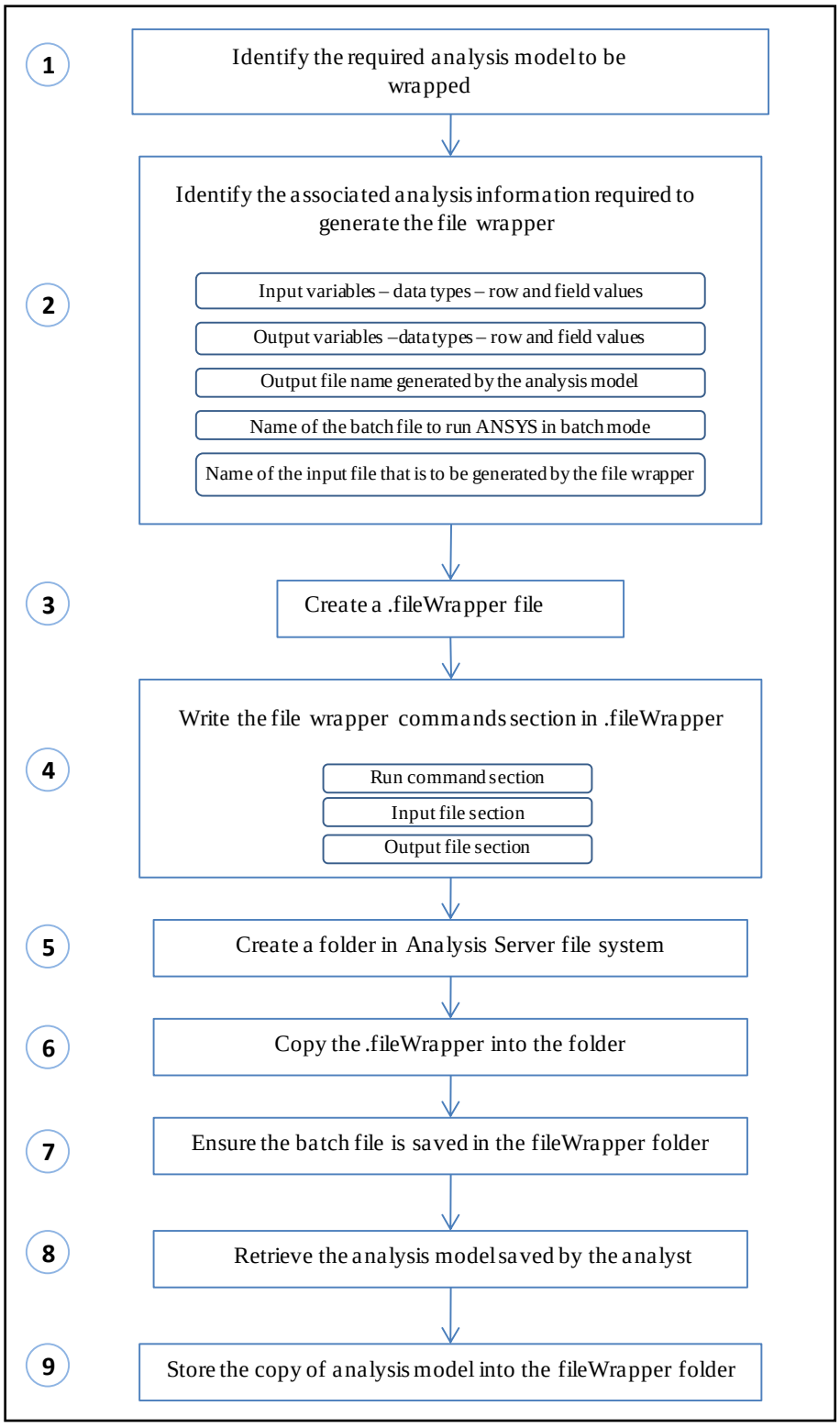


Figure 3.4 : Flowchart to create a file wrapper

The ANSYS analysis model is executed by the file wrapper in a batch mode. This requires the batch file to be created and saved in the file wrapper folder. The steps involved in the creation of batch file are shown in the flowchart in Figure 3.5. Creation of the batch file for the file wrapper is a 9 step process in which main steps such as identification of the analysis software executable location and the writing of the batch command with proper syntax.

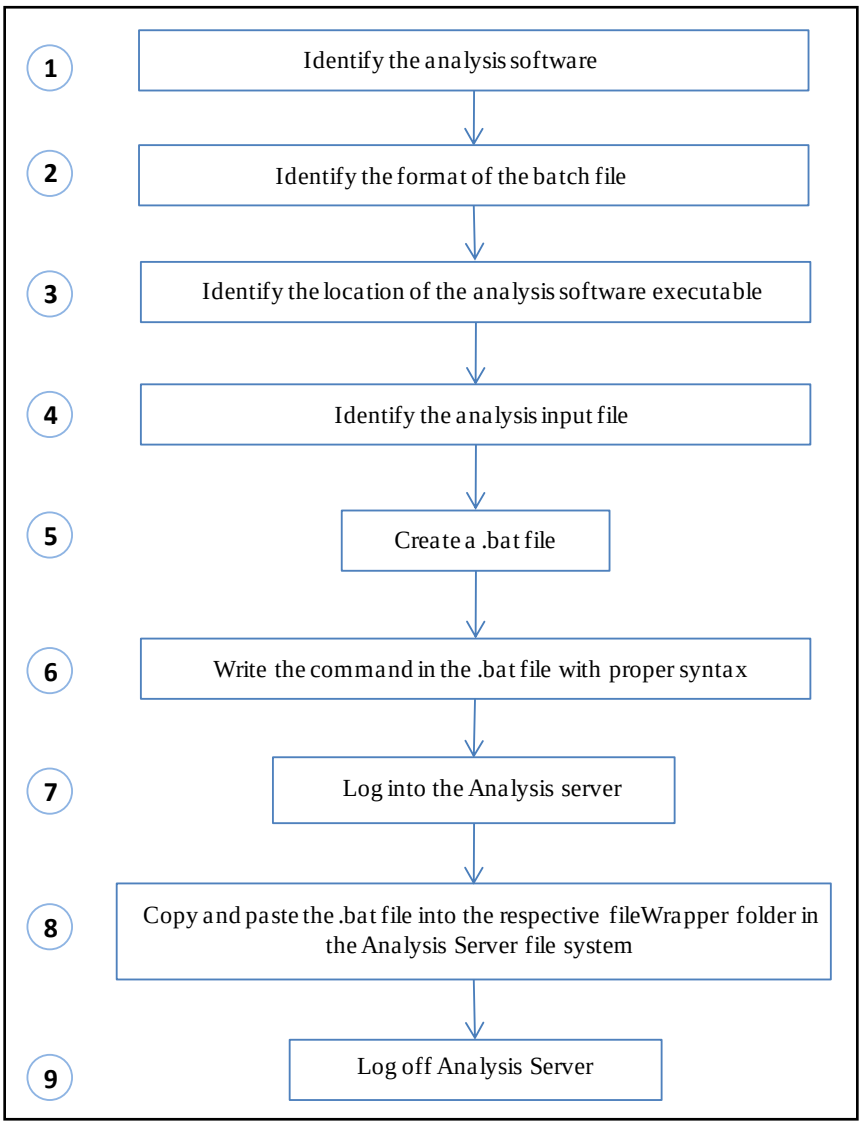


Figure 3.5 : Flowchart for batch file creation

Thus in order to prepare analysis model for execution in ModelCenter it takes a total of 18 steps (see Figure 3.6) , 9 steps for file wrapper creation and 9 steps for Batch file creation. These 18 steps have to be carried out for each analysis model if it has to be integrated and executed in ModelCenter.

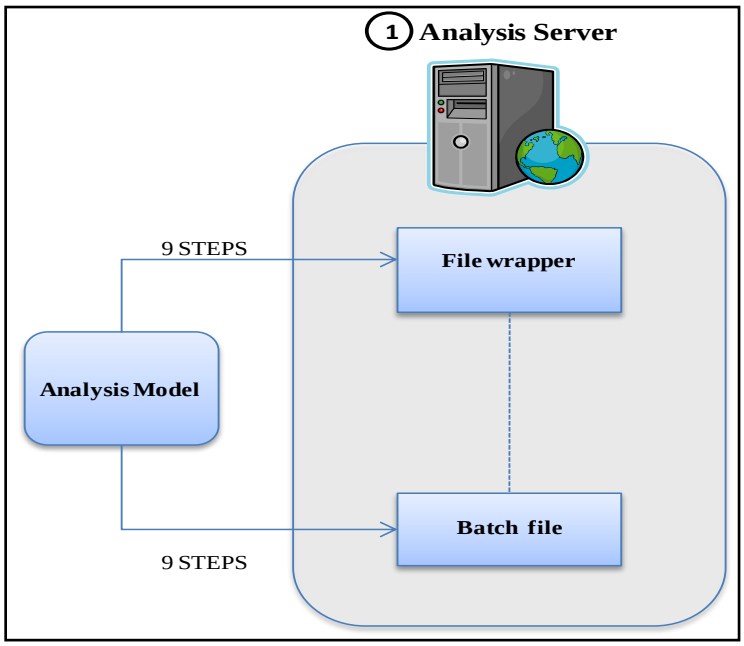


Figure 3.6 : Overview of steps involved in preparing ANSYS analysis model for execution

3.6 Drawbacks of Current Configuration

The configuration detailed in the previous section enables designers and analysts to share models over a distributed network, better utilize computer resources, and integrate various design process intelligently. The file wrapper and the batch file creation enables for the integration of analysis models in ModelCenter. However there are certain drawbacks in this configuration. These drawbacks are listed below.

- The designer formulating an MDO problem should have prior knowledge about analysis component used in formulation. There is no structured repository to store the information and provide this knowledge to the designer during formulation.
- Without a lengthy file naming convention or a brief description about the components, it would be difficult for the designer to identify the right MDO component to be reused.
- Database features available are limited. They only capture trade study details. They do not capture information about components of the MDO problem, so that the designer can decide which components need to be used to create his design problem.
- The manual creation of file wrapper and batch file are error prone and time consuming. Without proper file wrapper and batch file syntax, the analysis model cannot be used in ModelCenter.
- The information about a particular analysis model which is entered into the file wrapper template and the batch file information entirely depends on the designer's prior knowledge and the familiarity with the analysis model. Without the prior knowledge it would be very difficult and time consuming to get the information by opening and looking into the analysis model.

3.7 Features to support reuse and reconfiguration of MDO problems

Structured repository: The primary function of a structured repository is to capture and store the information generated such as design objectives, critical design considerations and requirements, important design parameters, design constraints, optimization technique used, results from previous runs, designer's rationale, and

decisions taken while formulating a design problem. The knowledge data generated from a process is helpful for determining the purpose of a particular analysis code; the optimization process methods used; the input and output data for the process; and knowledge about the components available in the analysis code, including those used and those which was not.

File information system: File information system is a subset in the structured repository. The primary function of this file information system is to store the information about components /files used in a MDO problem. It includes properties of a file and also keeps track of the changes made to the components in a design problem. Example of which include elements used in an analysis code, instances of various objects created to formulate a design problem, optimization techniques and other components used in a design process.

CHAPTER FOUR

PROPOSED MODELCENTER – ANALYSIS SERVER CONFIGURATION AND IMPLEMENTATION

The focus of this chapter is to provide design details of the proposed ModelCenter and Analysis Server configuration (see Figure 4.1). The main components in the configuration are (1) the structured repository, (2) the common file system and (3) Java applications. The repository is implemented using a relational database. It provides a structured information model to store and organize the MDO information. The common file system enables shared access and provides a structure to store analysis models and project files. Java applications are developed to complement the benefits offered by the structure repository and the common file system. Together the structured repository and the common file system enhance the information management capability in ModelCenter and Analysis Server configuration, by capturing Meta information about the MDO files organized and stored in the common file system. The subsequent sections in this chapter are organized to discuss the details of structured repository design; the structure of the common file system; and the development of java applications.

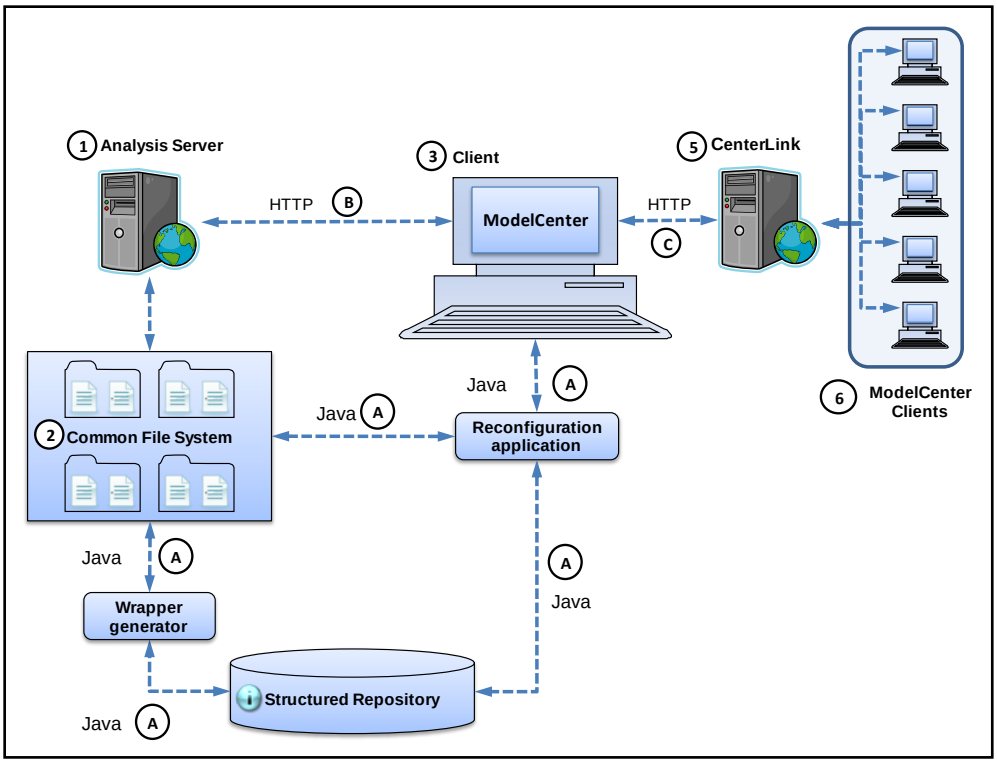


Figure 4.1 : Proposed ModelCenter - Analysis Server configuration

4.1 Structured repository

The structured repository (see Figure 4.2) is a relational database which stores information about analysis models, input and output variables, MDO projects, optimization techniques used, file locations, and file creation. In order to manage, store and retrieve information from the database, the MYSQL database management system is utilized in this research. The design of the repository is explained in details in the subsequent sections.

The main functionalities of the structured repository are listed below:

- Provide information model to store MDO information
- Manage the storing and sharing of MDO information

- Allow users to update MDO information
- Allow users to retrieve MDO information

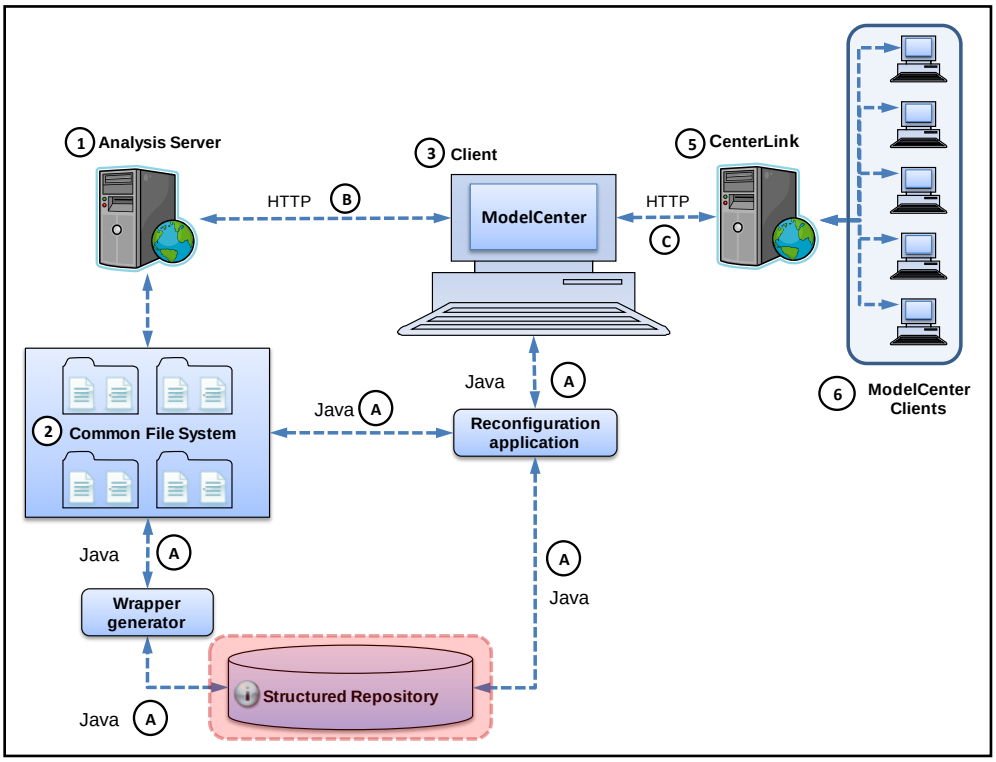


Figure 4.2 : Structured repository in proposed ModelCenter-Analysis Server configuration

4.6.1 Design of the structured repository

The design and development of the structured repository involves four phases as shown in Figure 4.3. Phase 1 is the conceptual database design, Phase 2 is the Transaction design, Phase 3 is the data model mapping and Phase 4 is the transaction implementation details. Phase 2 is executed in parallel with Phase 1 in order to include the characteristics of the transactions in the conceptual database design.

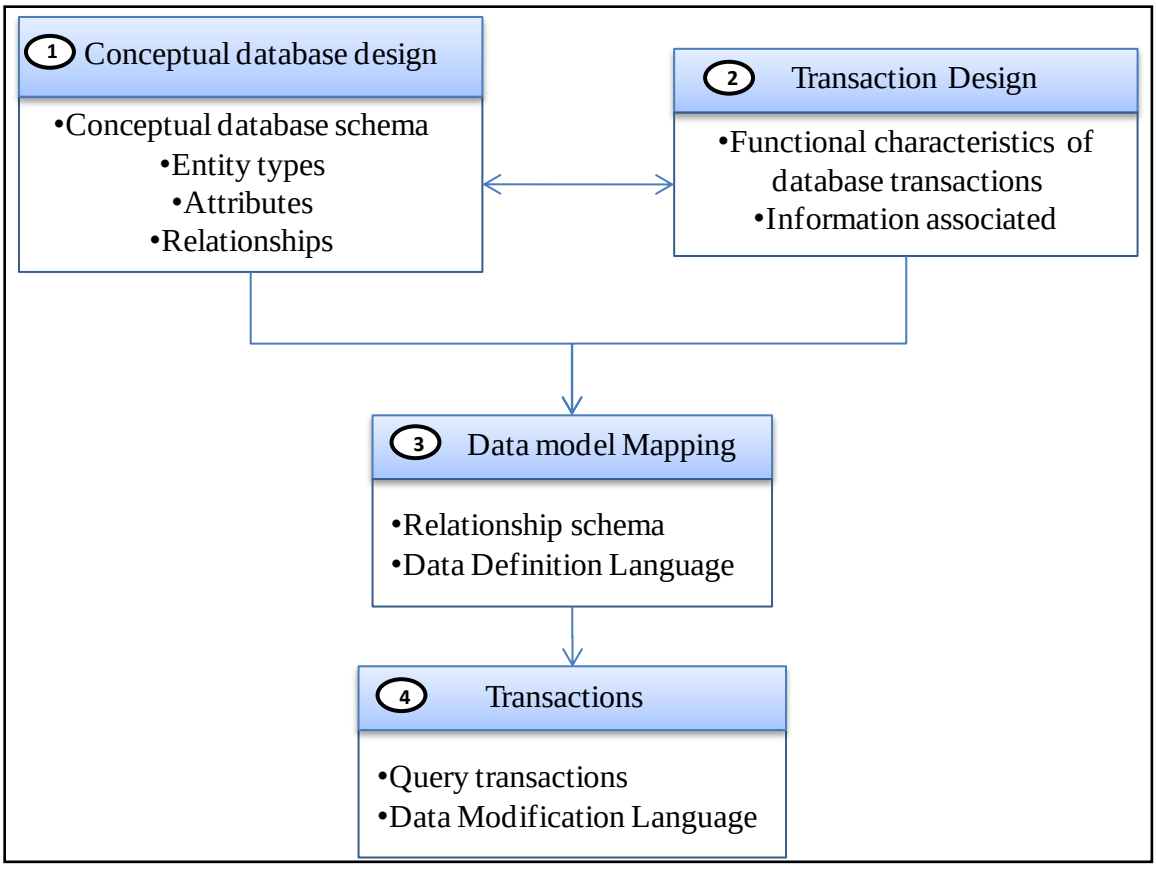


Figure 4.3: Phases in structured repository development

PHASE 1: The conceptual database design phase includes conceptual database schema and identification of basic components of the schema. The conceptual database schema provides a high level overview of the information model. The three basic components of the schema are entity types, attributes and relationship types provide the structure to store the MDO information. Unified Modeling Language (UML) class diagrams are utilized to provide conceptual schema, which are similar to the ER diagrams in the traditional database design. The UML convention followed in this development is according to the convention followed in [5]. The convention is to list the class name in boldface and center the name in the box. The UML class is a box consisting of three

sections. The first section is for the class name (entity name), second for the attributes of the class and third section is for the operations that can be performed by the class.

The UML class diagram shown in Figure 4.4 represents the information model of the repository to store and organize the MDO information. MDO projects are created in the MDO framework and stored as project files. MDO projects are uniquely identified the project name and the file name. MDO project integrates analysis models and optimizers to find the optimum value for the objective. Analysis models are created using many analysis software. Analysis models which cannot be directly integrated into the MDO framework utilize wrappers for integration. Analysis models, wrappers and MDO projects are created by a person and stored in a particular location in the server. The person belongs to a particular department and is uniquely identified by name and position. MDO problems are formulated and implemented in MDO projects to find the optimum value of the objective. MDO problems formulated contain objectives, constraints, design variables and design parameters. Objectives and constraints are the associated output variables of an analysis model. Design variables and design parameters are the associated input variables of an analysis model.

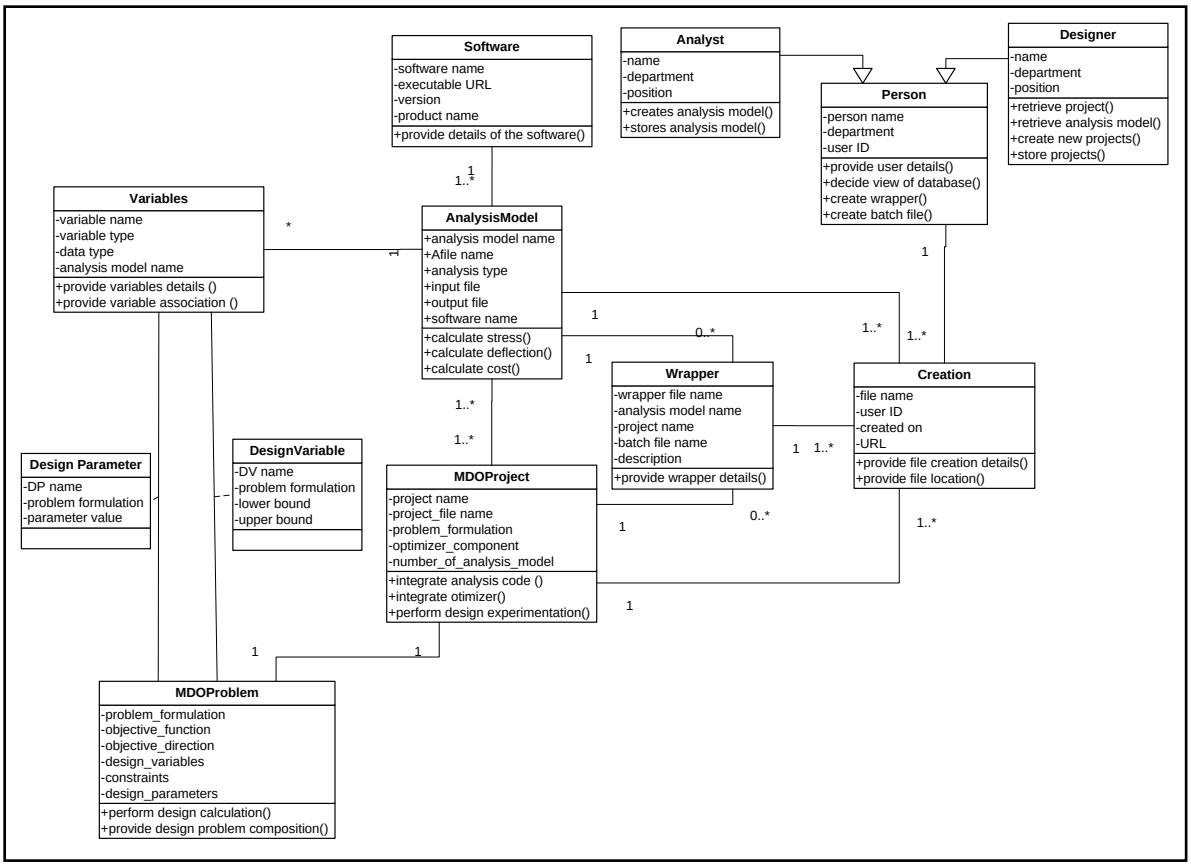


Figure 4.4: UML Class diagram

Class description: Each class in the UML diagram is described and its associated attributes are listed in Table 4.1. The class name in the UML classes is mapped to table name in the database.

Table 4.1: Class description

Class Name	Description	Attributes
Analysis Model	Describes the analysis model information. It also provides the analysis software name used to create the analysis model along with the input and output file required for the model	• File name • Software used • Analysis type • Input file • Output file
Software	Software class supports the analysis model class by storing the directory information of the executable and the latest available version of the software	• Software name • Executable URL • Version • Output file
Variables	Variable class stores the variable information corresponding to an analysis model. Input, output and data type for a particular variable is also stored	• Variable name • Analysis model name • Variable type • Data type
Wrapper	Wrapper class connects the MDO project class and the analysis model class if a wrapper is used for integrating analysis model in the project. It also provide the batch file information	• Wrapper name • Analysis model name • Project name • Batch file name
MDO Project	MDO project class stores information about the main components (optimization and Analysis model) used in a particular project file	• Project name • File name • Optimization name • Analysis model name • Number of optimizer • Number of analysis model
MDO Problem	MDO Problem describes the problem formulation details in a particular MDO project	• Project name • Objective function • Design variables • Constraints
Creation	Creation class provides the file creation and file location details	• File name • Person name • Created on • Version • URL
Person	Person class supports the creation class by providing the user information. It is a super class of Designer class and Analyst class.	• Fname, Lname • UserID • Department • Position

Attribute description: The description of the attributes and their data types are listed in Table 4.2. Attributes and data types provide the characteristics for the entity type. The data types used here are the SQL data types. The attributes in the UML classes can be mapped to column name in the database.

Table 4.2: Attribute description

Attribute name	Data type	Description
Software_name	VARCHAR(n)	Name of the analysis software
Version	FLOAT	Version of the software available
Product_name	VARCHAR(n)	Product name of the analysis software
Executable_URL	VARCHAR(n)	Directory of the analysis software executable
Fname	VARCHAR(n)	First name of the person
Lname	VARCHAR(n)	Last name of the person
UserID	VARCHAR(n)	ID of the person
Department	VARCHAR(n)	Name of the department the user belongs
Position	VARCHAR(n)	Position name of the user in a department
Analysis_model_name	VARCHAR(n)	Name of the analysis model
AFile_name	VARCHAR(n)	Name of the analysis file
Analysis_type	VARCHAR(n)	Type of analysis used in analysis model
Input_file	VARCHAR(n)	Input file for the analysis model
Output_file	VARCHAR(n)	Output file for the analysis model
Variable_name	VARCHAR(n)	Name of the variable in an analysis model
Variable_type	VARCHAR(n)	Type of a particular variable (Input/output)
Data_type	VARCHAR(n)	Data type of the variable in an analysis model
Problem_formulation	VARCHAR(n)	Name of the problem formulation
Objective_function	TEXT	Objective function definition for the design problem
Design_variable	TEXT	Name of the design variables for the design problem
Constraints	TEXT	Constraints for the design problem
Project_name	VARCHAR(n)	Name of the MDO project
PFile_name	VARCHAR(n)	Name of the MDO project file
Optimization_component	VARCHAR(n)	Name of the optimization component used in the MDO project

Wrapper_File_name	VARCHAR(n)	Name of the wrapper along with its file extension
Batch_file_name	VARCHAR(n)	Name of the batch file user in the wrapper
Description	VARCHAR(n)	Brief description of the wrapper
DV_name	VARCHAR(n)	Name of the design variable
Lower bound	FLOAT	Lower limit value for a design variable
Upper bound	FLOAT	Upper limit value for a design variable
Created_on	DATETIME	Date and time of the creation
URL	TEXT	Location of the file
CVariable_name	VARCHAR(n)	Name of the constraint variable

Relationship description: The relationship between the classes are identified and described in Table 4.3. The relationship column specifies the name of two tables of the repository whose relationship is being described. Association column provides the type of association between the tables specified in the relationship column i.e. Many to Many (M: M), One to Many (1: M) and One to One (1:1). The description column describes in detail the relationship shared between the two specified tables.

Table 4.3: Relationship description

Relationship	Associations	Description
MDO PROJECT – ANALYSIS MODEL	M:M	Many MDO projects can have many analysis models, one MDO Projects can have many analysis models and many MDO projects can have one analysis model
SOFTWARE – ANALYSIS MODEL	1:M	One software creates many analysis model
ANALYSIS MODEL – WRAPPER	1:M	An analysis model can have many wrappers
ANALYSIS MODEL – CREATION	1:M	An analysis model can have one or many creation data
ANALYSIS MODEL – VARIABLES	1:M	An analysis model can have many variables
MDO PROJECT – CREATION	1:M	An MDO project can have one or many creation data
WRAPPER – CREATION	1:M	A wrapper can have many creation
MDO PROJECT – WRAPPER	1:M	An MDO project can have many wrappers and an MDO project can have no wrappers
PERSON – CREATION	1:M	One person can create many files
MDO PROJECT - MDO PROBLEM	1:1	An MDO project can have one MDO problem

PHASE 2: The transaction design phase provides the functional characteristics of transactions carried out on the database and the information associated with it. Identifying the functional characteristics of the repository transaction is important to conceptualize the database schema. The functional characteristics and the information associated are tabulated in Table 4.4.

Table 4.4: Functional characteristics of the transactions

Functional characteristic	Associated information
Provide details of the project file	• Analysis model used • Optimization used
Retrieve analysis model information	• Analysis software used • Analysis type: structural, thermal etc. • Project the analysis model is used in • Input file for the analysis model • Output file for the analysis model
Retrieve variable information	• Input variables in the analysis model • Output variables in the analysis model
Retrieve optimization information	• Objective direction: max or min • Objective parameter: target • Constraints: stress constrain, deflection constrain etc. • Variable upper bound and lower bound
Retrieve file location information	• Analysis file name and location • Project file name and location
Retrieve the analysis software information	• Name of the analysis software • Software version • URL of the software (depending on the version)
Retrieve wrapper information	• Name of the wrapper file • Name of the analysis model it wraps • Name of the project the wrapper is used for
Update wrapper information	• Name of the wrapper file • Name of the analysis model it wraps

PHASE 3: The data model mapping involves developing a relational schema and providing the data definition language. The relationship schema provides the relationship mapping between the tables of the database along with the key constraints. The primary keys are identified and mapped as shown in Figure 4.5.

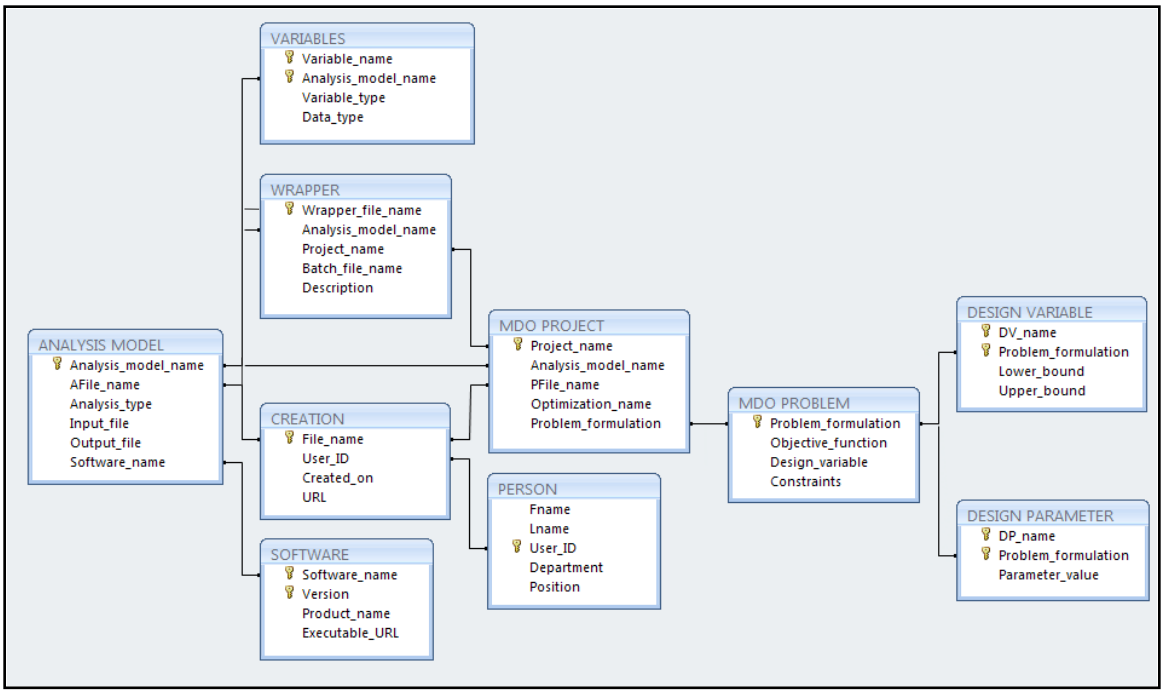


Figure 4.5: Relationship schema

The data definition language provides a set of statements to create the database. MySQL data definition statements are as shown in Table 4.5.

Table 4.5: MySQL data definition statements

```
CREATE DATABASE Structured_Repository;
USE Structured_Repository
CREATE TABLE SOFTWARE
(       Software_name           VARCHAR (45)        NOT NULL,
        Version                 FLOAT               NOT NULL,
        Product_name            VARCHAR (45),
        Executable_URL          VARCHAR (60),
PRIMARY KEY (Software_name)
);

CREATE TABLE PERSON
(       Fname                   VARCHAR (45)        NOT NULL,
        Lname                   VARCHAR (45)        NOT NULL,
        User_ID                 VARCHAR(11),
        Department              VARCHAR (45)        NOT NULL,
        Position                VARCHAR(45),
        PRIMARY KEY (User_ID)
```

```
);

CREATE TABLE ANALYSISMODEL
(       Analysis_model_name     VARCHAR (80)        NOT NULL,
        AFile_name              VARCHAR (45)        NOT NULL,
        Analysis_type           VARCHAR (45)        NOT NULL,
        Input_file              VARCHAR (45),
        Output_file             VARCHAR (45),
        Software_name           VARCHAR (45),
PRIMARY KEY (Analysis_model_name ),
FOREIGN KEY (Software_name) REFERENCES SOFTWARE (Software_name)
);

CREATE TABLE VARIABLES
(       Variable_name           VARCHAR (45)        NOT NULL,
        Variable_type           VARCHAR (45)        NOT NULL,
        Data_type               VARCHAR (45),
        Analysis_model_name     VARCHAR (45)        NOT NULL,
PRIMARY KEY (Variable_name),
FOREIGN KEY (Analysis_model_name) REFERENCES ANALYSISMODEL
(Analysis_model_name)
);

CREATE TABLE MDOPROBLEM
(       Problem_formulation     VARCHAR (80)        NOT NULL,
        Objective_function      TEXT(100),
        Design_variable         TEXT(100),
        Constraints             TEXT(100),
PRIMARY KEY (Problem_formulation)
);

CREATE TABLE MDOPROJECT
(       Project_name            VARCHAR (80)        NOT NULL,
        PFile_name              VARCHAR (80)        NOT NULL,
        Problem_formulation     VARCHAR (80)        NOT NULL,
        Analysis_model_name     VARCHAR (45),
        Optimization_component  VARCHAR (45),
PRIMARY KEY (Project_name,PFile_name),
FOREIGN KEY (Analysis_model_name) REFERENCES ANALYSISMODEL
(Analysis_model_name),
FOREIGN KEY (Problem_formulation) REFERENCES MDOPROBLEM
(Problem_formulation)
);
```

```
CREATE TABLE WRAPPER
(       Wrapper_File_name       VARCHAR (45)        NOT NULL,
        Analysis_model_name     VARCHAR (45)        NOT NULL,
        Project_name            VARCHAR (45)        NOT NULL,
        Batch_file_name         VARCHAR (45) ,
        Description             TEXT(100),
PRIMARY KEY (Wrapper_File_name),
FOREIGN KEY (Analysis_model_name) REFERENCES ANALYSISMODEL
(Analysis_model_name),
FOREIGN KEY (Project_name) REFERENCES MDOPROJECT (Project_name)
);

CREATE TABLE DESIGNVARIABLE
(       DV_name                 VARCHAR (45)        NOT NULL,
        Problem_formulation     VARCHAR (80)        NOT NULL,
        Lower_bound             FLOAT,
        Upper_bound             FLOAT,

PRIMARY KEY (DV_name, Problem_formulation),
FOREIGN KEY (Problem_formulation) REFERENCES MDOPROBLEM
(Problem_formulation)
);

CREATE TABLE CREATION
(       File_name               VARCHAR (80)        NOT NULL,
        User_ID                 VARCHAR (11),
        Created_on              DATETIME,
        URL                     TEXT(100),
PRIMARY KEY (File_name),
FOREIGN KEY (User_ID) REFERENCES PERSON (User_ID)
);

CREATE TABLE CONSTRAINT
(       CVariable_name          VARCHAR (45)        NOT NULL,
        Problem_formulation     VARCHAR (80),
        Lower_bound             FLOAT,
        Upper_bound             FLOAT,

PRIMARY KEY (CVariable_name),
FOREIGN KEY (Problem_formulation) REFERENCES MDOPROBLEM
(Problem_formulation)
);
```

PHASE 4: The database transactions phase includes details of the transactions carried out on the database. The transactions for the database along with the data modification statements are listed below.

- Transaction to get input variable and output variable details for an analysis model

```
DATABASE structured_repository
BEGIN WORK
SELECT Variable_name,Data_type
FROM variables
WHERE Analysis_model_name = "name of the analysis model"
AND Variable_type="input"

SELECT Variable_name,Data_type
FROM variables
WHERE Analysis_model_name = "name of the analysis model"
AND Variable_type=\"output\"
COMMIT WORK
```

- Transaction to get the count of number of input variable in an analysis model

```
DATABASE structured_repository
BEGIN WORK
SELECT COUNT(*) AS rowcount
FROM variables
WHERE Analysis_model_name = "name of the analysis model"
AND Variable_type= "input"
COMMIT WORK
```

- Transaction to get the count of number of output variable in an analysis model

```
DATABASE structured_repository
BEGIN WORK
SELECT COUNT(*) AS rowcount
FROM variables
WHERE Analysis_model_name = "name of the analysis model"
AND Variable_type= "output"
COMMIT WORK
```

- Transaction to get output file name generated by an analysis model

```
DATABASE structured_repository
BEGIN WORK
SELECT Output_file
FROM analysismodel
WHERE Analysis_model_name = "name of the analysis model"
COMMIT WORK
```

- Transaction to get the name of the analysis software used to create an analysis model

```
DATABASE structured_repository
BEGIN WORK
SELECT Software_name
FROM analysismodel
WHERE Analysis_model_name= "name of the analysis model"
COMMIT WORK
```

- Transaction to get the location of the analysis software executable for an analysis software

```
DATABASE structured_repository
BEGIN WORK
SELECT Executable_URL
FROM software
WHERE software_name = "name of the software"
COMMIT WORK
```

- Transaction to get the list of analysis models available

```
DATABASE structured_repository
BEGIN WORK
SELECT Analysis_model_name
FROM Analysismodels
COMMIT WORK
```

4.2 Common file system

The goal of the common file system as shown in Figure 4.6 is to store files in an organized folder structure and to provide shared access to the analysis models and the ModelCenter project files. It is to facilitate for the distribution of files over the network. An organized structure of the common file system as shown in Figure 4.5 enables for intelligent search and retrieval of the files. Files are grouped into project folders, analysis folders and file wrapper folders and are shared with the help of file information system which is a subset of the repository.

The main functionalities of the common file system are to:

- Provide structure to store files
- Allow users to share file
- Store analysis models
- Store ModelCenter projects
- Store file wrapper and batch files

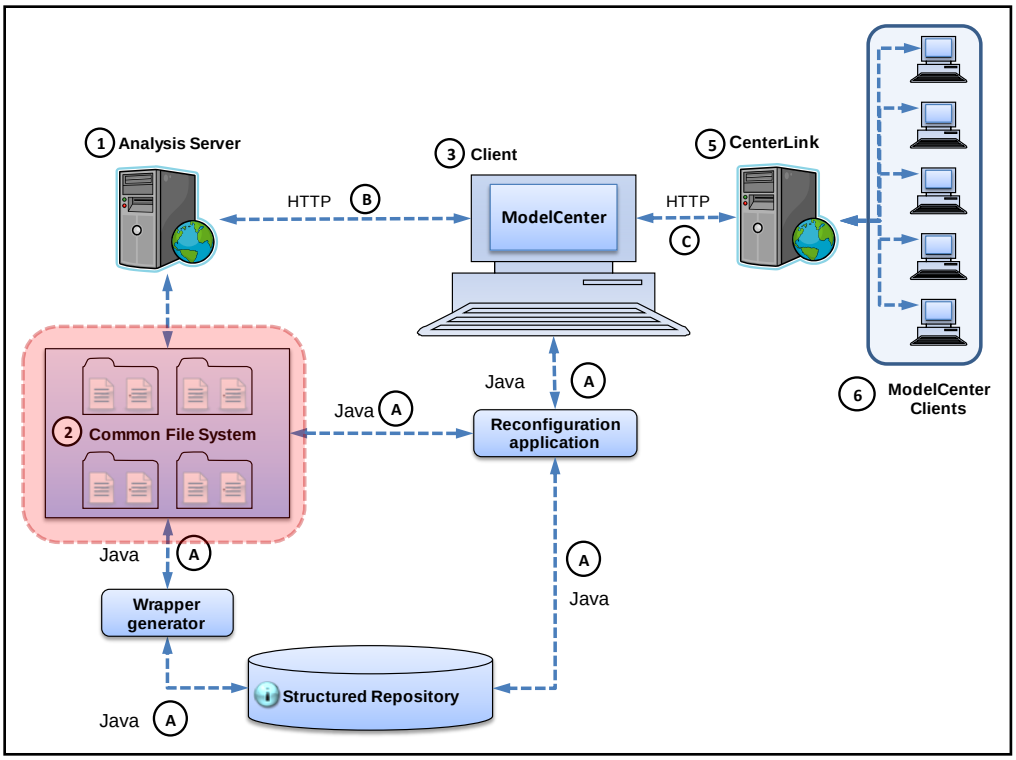


Figure 4.6: Common file system in proposed configuration

In order to enable for efficient search and retrieval of files a proper file and folder naming convention is followed in the common file system. The details of the naming convention are as follows:

- Name of the analysis model files – analysis_model_ name + file extension
- Name of the project file - project_name + file extension
- Name of the file wrapper - analysis_model_ name + wrapper file extension
- Name of the batch file analysis_model_ name + batch file extension

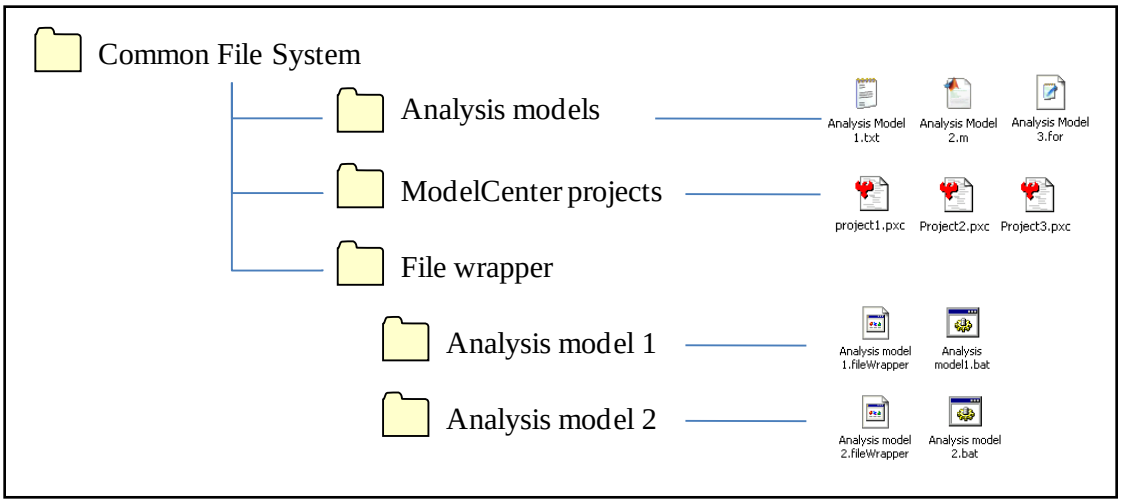


Figure 4.7: File structure in common file system

Analysis model folder: This folder contains all the analysis model files that need to be shared. The analysis models are created using several analysis software such as ANSYS, Matlab and FORTRAN are stored in this folder. Once the files are stored in this folder the location details have to be updated into the creation table and the software details are updated in the software table in the repository.

ModelCenter projects folder: This folder contains the project files created in ModelCenter. This folder stores only ModelCenter specific projects to provide shared access to the project files. Once the project files are stored in this folder the creation details have to be updated into the creation table in the repository

File wrapper folder: This folder contains the file wrapper and the corresponding batch file. The file wrapper and the batch file are stored in their respective folder. Once the file wrapper and batch files are stored in this folder the location information is updated in the creation table and file wrapper information is updated in the wrapper table in the structured repository.

4.3 Methods for accessing structured repository and common file system

Structured repository is accessed by the Java application using a MYSQL – JDBC DRIVER. This driver establishes the connection to the repository and parse the connection session to the Java application to perform queries on the repository.

Common file system is accessed by the Java application with the help of the file system URL and stored the files in the folder structure of the common file system.

4.4 Java Applications

To demonstrate the usage of the structured repository and the common file system, two Java applications are developed as shown in Figure 4.8. This Java application is compiled and built using NetBeans 5.5.1 IDE. The details of the Java applications developed are discussed in this section. The sections first describe the purpose for which the application was developed, and then the development of the application with the help of pseudo code. These Java applications enable intelligent search and retrieval of the MDO files and automate the repeatable processes such as file wrapper and batch file creation.

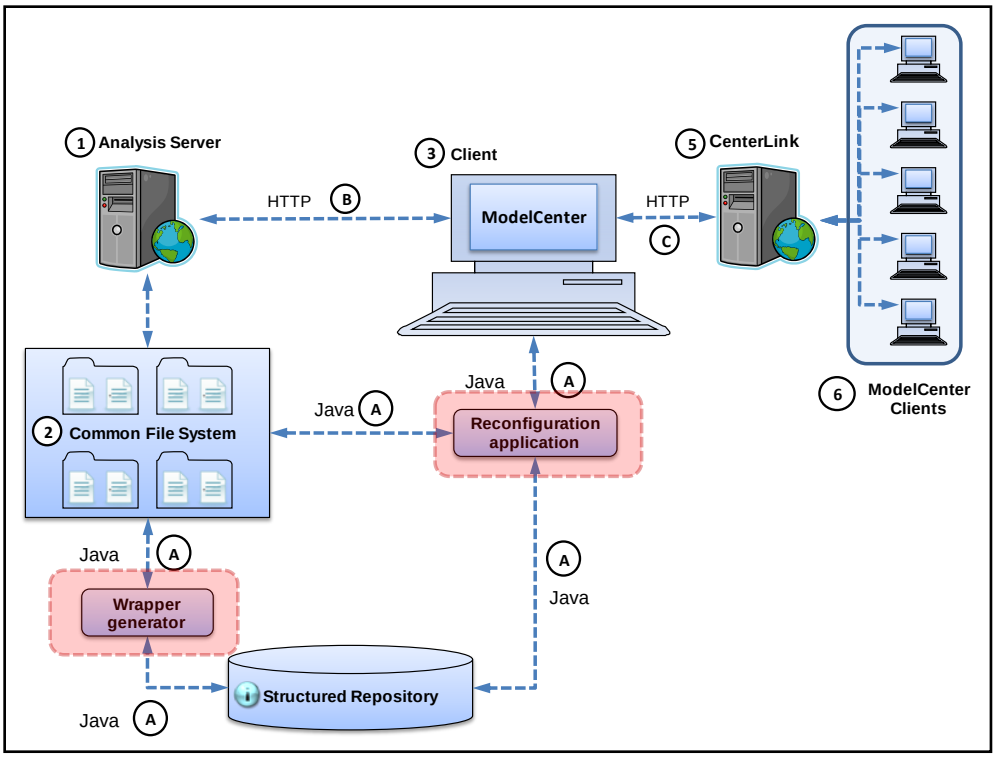


Figure 4.8 : Java applications in proposed configuration

4.6.2 Wrapper generator application: Batch file and file wrapper automation

The wrapper generator application is designed and developed to automate the repeatable and manual process of creating the file wrapper and the batch file. It also demonstrates how the information from the structured repository can be effectively utilized to reduce the user interactions in the creation, thereby save time and avoid errors which might occur during the manual generation. The details of the Wrapper generator application are discussed in the subsequent sections.

The main goal of this application is to create a file wrapper and its corresponding batch file with minimal user interaction. The main functionalities of this application are:

(1) Connect to the repository for update and retrieval of information

(2) Retrieve file wrapper and batch file specific information from the repository

(3) Create file wrapper and batch files

(4) Update file wrapper and batch file specific information into the repository

(5) Connect to the common file system to store files

(6) Store the file wrapper and batch file in the common file system structure

(7) Prevent duplication of file wrapper files and batch files

Application overview

In order to provide a design overview of the application two types of overviews are presented. The first is a black box overview as shown in Figure 4.9, which explains the user and application interaction. The second is the structural overview as shown in Figure 4.10, which explains the internal structure of the application.

Black box overview: In this overview, the application is viewed as a black box which takes the analysis model name as the input from the user and creates the file wrapper and batch files such that it can be accessed through the server browser in ModelCenter.

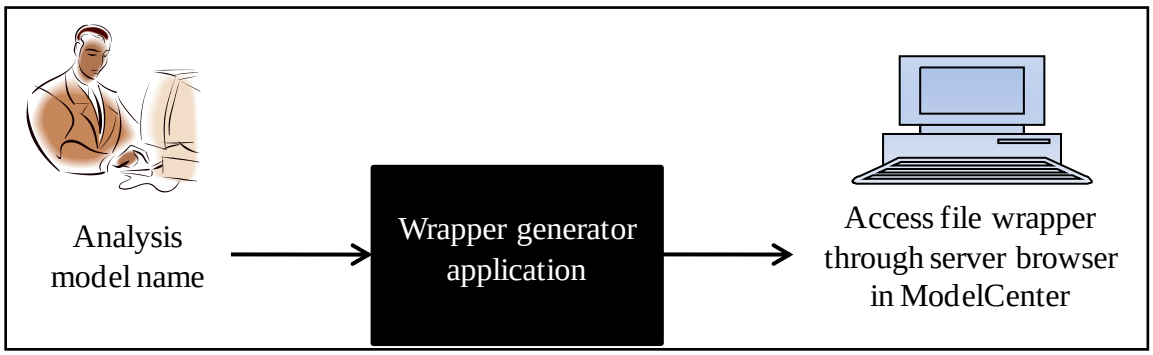


Figure 4.9: Black box overview of wrapper generator application

Structural overview: In this overview, the internal structure of the application is provided. The application has a main class which calls in two other classes i.e. a file wrapper generator class and a batch file generator class. These two classes in turn call the database connector class. In the main class, the user entered analysis model name is fed into both the file wrapper generator class and the batch file generator class. The file wrapper generator class connects to the database with the help of database connecter class, queries for the information to create a file wrapper, creates the file wrapper and updated the database with the new creation details. The batch file generator class connects to the database with the help of database connector class, retrieves the information to create a batch file, created the batch file and updates the batch file information into the database.

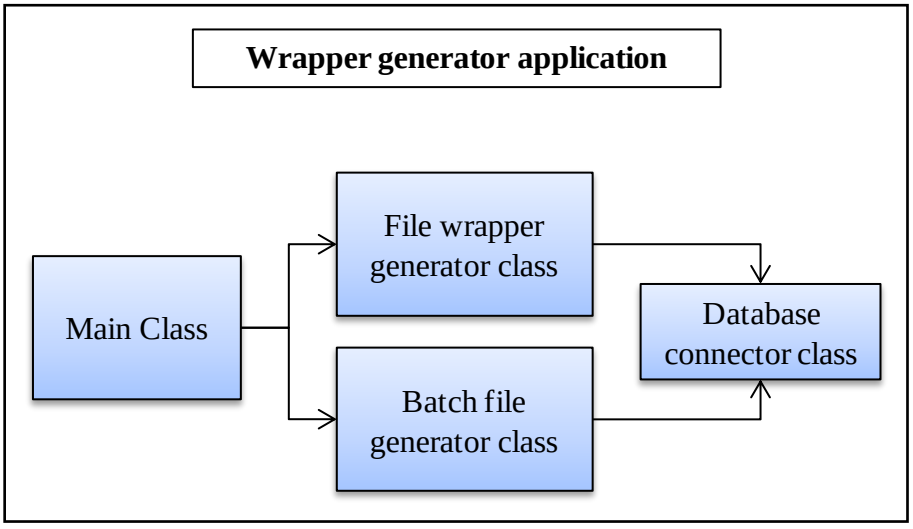


Figure 4.10: Structural overview of wrapper generator application

Application programming details

A pseudo code of the wrapper generator application is shown in Table 4.5. The pseudo code provides a detailed overview of the program to automate the file wrapper and the batch file. The complete Wrapper generator application can be seen in Appendix B. The information required for the file wrapper creation is obtained from Analysis model table; variables table and the software table in the database (see Figure 4.2). The user selects the analysis model for which a file wrapper needs to be created in the main class. The file wrapper generator class takes this analysis model name as input from the main class. Connection to the database is established and query for the wrapper file name in the wrapper table of the database is executed. The program then checks to see if the query returns any row value. If it does return a value then the program stop and prompt the user that a file wrapper already exists. If not the program continues. It creates the file wrapper in file wrapper folder of the common file system under. Finally the file wrapper generator class updates the wrapper information in the wrapper table and returns to the main program which then calls the batch file generator.

After the file wrapper generator class is executed, the main class executes the batch file generator class. The batch file generator takes the analysis model name from the main class as input and queries the database for the analysis software name, executable location and version from the software table. A batch file is created in the corresponding file wrapper folder and the command line is written into the batch file.

Table 4.6: Pseudo code for wrapper generator application

```
---------- WRAPPER GENERATOR APPLICATION MAIN CLASS ----------

// Get  user input for analysis model name
        /*User input = analysis model name */

// Generate  wrapper

        /*Call the file wrapper generator class*/

// Generate batch file

         /*Call the batch file generator class*/

END CLASS

---------- FILE WRAPPER GENERATOR CLASS ----------

//Connect to the database

        /*Call the connect to database class*/
        Query = query for wrapper file name from the selected analysis model;

        /*Check if the wrapper already exits

                IF( Query returns a row)
                        Print file wrapper already exists
                ELSE
                try

//Query using SQL to retrieve information from the database

        Query1 = query for input variable name and its data type from the selected
        analysis model;
        Query2 = query for output variable name and its data type from the selected
        analysis model;
        Query3 = query to count number of input variables in the selected analysis model;
        Query4 = query to count number of output variables in the selected analysis
        model;
        Query5 = query for output file name of the selected analysis model;
        Query6 = query to get the file name of the analysis model;
        Query7 = query to get analysis software name used to create analysis model;
        Query8 = query to get the location of the analysis software executable;

// Store all the result from the queries their respective result set
                Result Set1= statement from query 1
                Result Set2= statement from query 2
                Result Set3= statement from query 3
```

```
                Result Set4= statement from query 4
                Result Set5= statement from query 5
                Result Set6= statement from query 6
                Result Set7= statement from query 7
                Result Set8= statement from query 8

// Create a folder and file wrapper file in the common file system

                SET parent directory path = common file system URL;
                CREATE folder = name of the user selected analysis model;
                APPEND the folder name to the parent directory path;
                CREATE File Wrapper file in the APPENDED directory path;

// Write the file wrapper sections  into the created file wrapper file

                SET FilewrapperName = selected analysis model name + .fileWrapper;
                SET BatchFileName = selected analysis model name + .bat;
                SET fileGenerateName = selected analysis model name + .in;

        /*Header section commands*/

                @author: Santosh Hiriyannaiah
                @version: Trial
                @description: File wrapper

        /*File wrapper Run section commands */

                RunCommands
                {
                generate inputFile
                run = " BatchFileName"
                parse outputFile
                }

        /*File wrapper RowFieldInputFile Section commands */

                {
                templateFile: String from result set of query 6
                fileToGenerate: fileGenerateName
                markAsBeginning "Input Variables"

                WHILE (there exists next line in the result set 1)
                Variable : Variable name from result set + data type from result set + Row
                value  + 3
                 j=j+1;
                END WHILE
                }

        /*File wrapper RowFieldOutputFile Section commands */

                RowFieldOutputFile outputFile
```

```
                {
                fileToParse: String from result set of query 5
                setDelimiters "= "
                WHILE (there exists next line in the result set 2)
                keyvar: Variable name from result set + data type from result set
                END WHILE
                }
        /*Close writing to file wrapper*/
// Update database
END CLASS
```

---------- BATCH FILE GENERATOR CLASS ----------

```
//Connect to the database
        /*Call the connect to database class*/
Query = query for wrapper file name from the selected analysis model;
// Create a folder and file wrapper file in the common file system
                SET parent directory path = common file system URL;
                CREATE folder = name of the user selected analysis model;
                APPEND the folder name to the parent directory path;
        /*Check if the batch file exists*/
        IF ( it exists)
                Delete the file
        ELSE
                Try
                CREATE batch file in the APPENDED directory path;
// Write the batch file command line into the created batch file
        WRITE  <Executable_URL>+ "-b" +"-p"+ProductName+" -i <input file> + "-o"
        <OutputFile>
        Close the batchfile
        Return batch file result to the main class
// Update database
END CLASS
```

---------- CONNECT TO DATABASE CLASS ----------

```
// Register the mysql – jdbc driver
// Establish connection with the database
                Connect to the database named structuredrepository
                Log in to the database providing user and password information
                Return the connection session details
END CLASS
```

4.6.3 Reconfiguration application: Reconfiguration of ModelCenter projects

The Reconfiguration application is designed and developed to provide details of the ModelCenter project file and to demonstrate reconfiguration of ModelCenter projects. This application focuses on providing details of the project file location, details of analysis model used in the project, optimizer details and the problem formulation details. The development of the Reconfiguration application is discussed in the following sections.

The main functionalities of this application are

(1) Provide available analysis model information

(2) Provide information of the project file components

(3) Reconfigure analysis components in ModelCenter projects

Application overview

There are two types of design overview presented in this section are the black box overview and the structural overview. The black box overview explains the user and application interaction and the structural overview provides the structure of the Java application 2.

Black box overview: The black box overview is as shown in Figure 4.11.First the user first enters the project file name to the application and the application provides all the information about the project available in the repository. Then the user enters the new analysis model name to be reconfigured into the project file and the name of the new ModelCenter project for the application to reconfigure and save the new project.

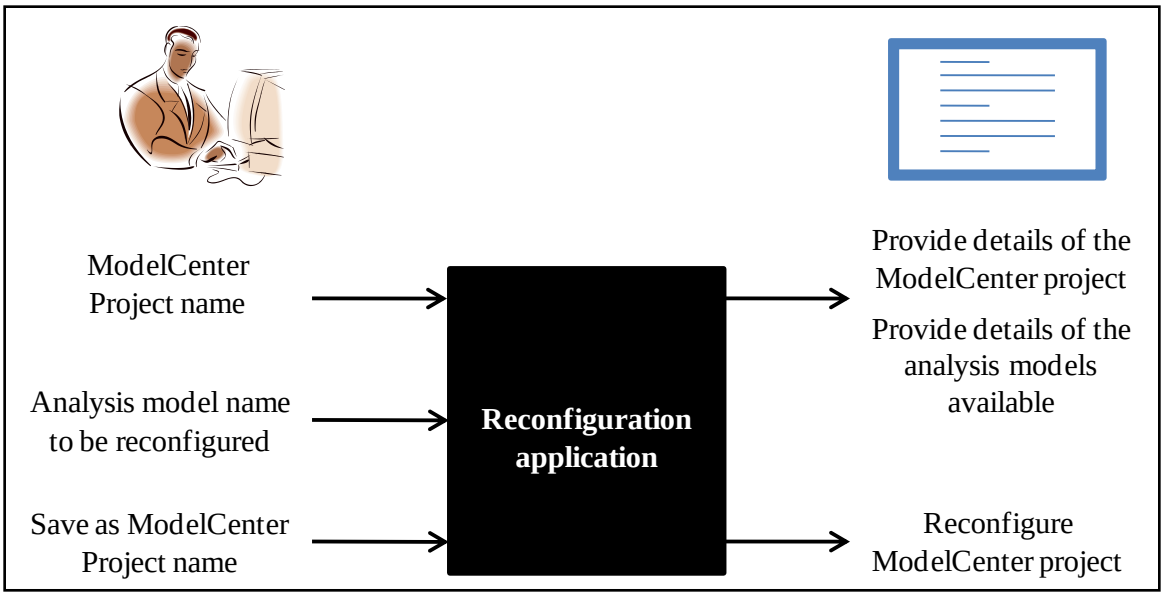


Figure 4.11: Black box overview of reconfiguration application

Structural overview: The structural overview is as shown in Figure 4.12. it provides the internal structure of the application. This application had a main class and three other classes which the main class calls. The main class first prompts the user to enter project file name for which the user requires information. taking the user input it then calls for the project information and analysis model information classes. Finally the main class once again prompts the user to enter the analysis model name and the new project name and then calls reconfiguration class to reconfigure the project.

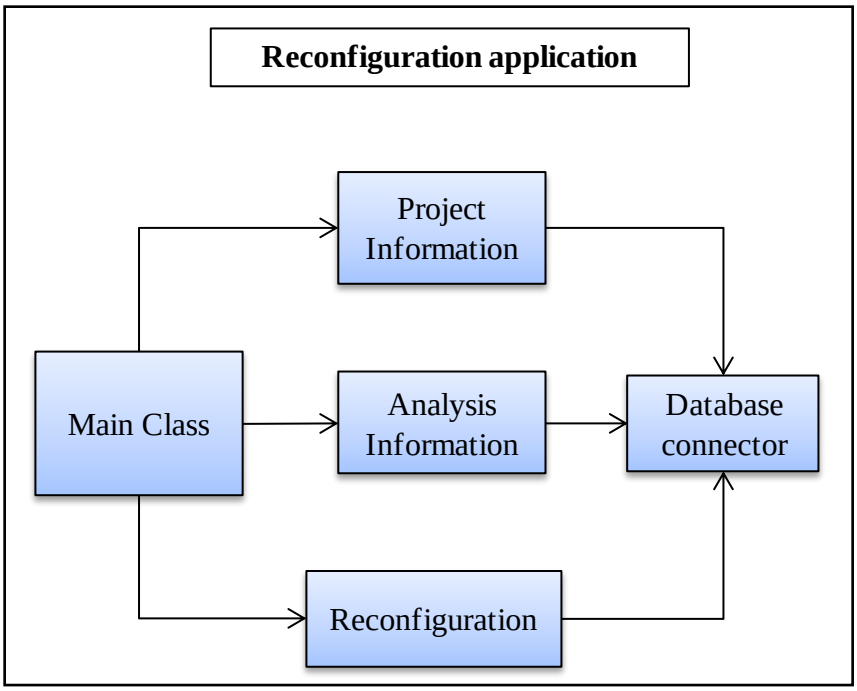


Figure 4.12: Structural overview of wrapper generator application

Application programming details

A pseudo code of the Reconfiguration application is shown in Table 4.6. The pseudo code provides a detailed overview of the program to provide project file specific information to the user and to reconfigure the MDO problem.

Table 4.7: Pseudo code for reconfiguration application

```
---------- RECONFIGURATION APPLICATION MAIN CLASS ----------
// Get  user input for project file name
        /*User input = project file name */
// Provide project specific information
        /*Call the file project information class*/
// Provide available analysis model information
        /*Call the analysis information class*/
// Get  user input for reconfiguration
```

```
        /*User input = analysis model name */
        /*User input = new project file name */
END CLASS
 ---------- PROJECT INFORMATION CLASS ----------

//Connect to the database

        /*Call the connect to database class*/

//Query using SQL to retrieve information from the database
        Query1 = query the MDO project class for project details
        Query2 = query the MDO project class for project file location
        Query3 = query the problem formulation class for formulation details
        Query4 = query the wrapper class for wrapper information

// Store all the result from the queries their respective result set
                Result Set1= statement from query 1
                Result Set2= statement from query 2
                Result Set3= statement from query 3
                Result Set4= statement from query 4

// Print out the queried information to display to the user
END CLASS

  ---------- ANALYSIS MODEL INFORMATION CLASS ----------

//Connect to the database

        /*Call the connect to database class*/

//Query using SQL to retrieve information from the database
        Query1 = query the analysis model class for available analysis model names

// Store all the result from the queries their respective result set
                Result Set1= statement from query 1

// Print out the queried information to display to the user
END CLASS
     ---------- RECONFIGURATION CLASS ----------
//Create a new model center session
        /*Load the project file selected */
        /*remove the analysis model component */
        /*Add the selected analysis model component*/
        /*Save the project file as specified by the user*/
//Close  model center session
        Return the successful creation string

// Update database
END CLASS
```

```
---------- CONNECTION CLASS ----------
// Register the mysql – jdbc driver
// Establish connection with the database
        Connect to the database named structured_repository
        Log in to the database providing user and password information
        Return the connection session details
END CLASS
```

4.5 Benefits of the proposed configuration

The structured repository, common file system and the Java application together provide an efficient information management system. It enables efficient database management, intelligent search and retrieval of MDO components and enhances reuse and reconfiguration capabilities of ModelCenter framework.

Benefit 1: Structured repository along with common file system enables intelligent search of the project files and analysis models. Structured repository along with Java applications enable efficient retrieval system. Design project and analysis models can be queried and selected using a structure repository and Java applications.

Benefit 2: The Common file system provides a centralized location for distributed design. Analysts can store analysis models and projects files into their respective folders and update the information in the repository.

Benefit 3: The Structured Repository enables queries to be performed based on problem information such as design objectives, critical design considerations and requirements, important design parameters, design constraints, optimization technique used, results from previous runs, and designer's rationale.

CHAPTER FIVE

EXAMPLE PROBLEM AND DATABASE IMPLENTATION

In order to demonstrate and test the usage of the proposed ModelCenter – Analysis Server configuration an example walkway beam structure optimization problem taken. The focus of this chapter is to provide details this example problem, extract the information and to populate the structured repository with the extracted information such that the example problem can be utilized during testing and demonstration. The implementation of the example problem in ModelCenter along with the necessary preparations is also discussed in the following sections.

5.1 Example problem: Analysis and optimization of the beam structure supporting walkways

The walkway platforms provide access between upper floors in multistoried buildings. The walkway platforms are supported by walkway structures which consist of number of beams and columns. A cross section of this beam structure with three walkway platforms one above the other is considered in this example as shown in Figure 5.1. The structure is designed to withstand the load of the concrete platform as well as the load of the people. In this example problem cantilever beams are used to support the walkway. The main objective in this example problem is to minimize the mass of the beam structure and to minimize the deflection in the walkway structure. This is an example of multi-objective analysis and optimization. While the optimization problem formulation remains the same for his example, the implementation is in two folds. First the analysis

model is generated for a rectangular cross sectional beam and is implemented in ModelCenter. Second the analysis model is generated for an I cross sectional beam. In this chapter since the focus is on the implementation of this example in database, only the details of the information extraction and implementation is discussed. The assumptions made and problem formulation is specified below.

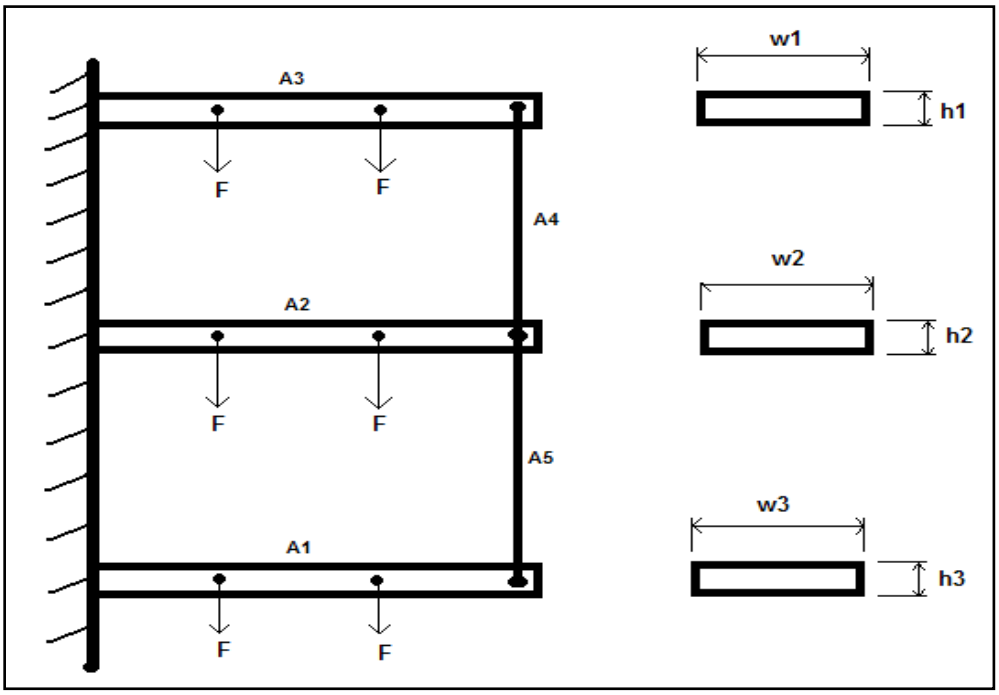


Figure 5.1 Walkway structure setup

Assumptions

- The weight of the concrete is neglected while considering the load.
- Average human weight of 85Kgs is considered and is taken as force acting on the beam by multiplying it with gravitational constant.
- Only two persons can walk on the walkway so that both of them are 1 meter from the ends.

- Length of the beam is assumed to be equal to the width of the walkway.
- Only a part of the walkway is taken into consideration for analysis
- Length of the beam and the vertical distance between the beams are assumed to be 3 meters.
- Material selected is same for all the elements in the structure.

Mathematical formulation

Minimize: (mass of the beam) and (deflection of the beam)

Subjected to: Stress constrain $\sigma \max imum < \sigma allowable$

Where,

Z = (Density * Length* Width * height) = ρ * Li* Wi * Hi

Design Parameters

ρ = Density

Li = Length of the walkway

Design variables

Wi = width of the cross-section of the beam (number of beams i = 1, 2, 3)

Hi = height of the cross-section of the beam (number of beams i = 1, 2, 3)

Rj= radius of the support cable (number of links j = 1, 2)

Bounds

Minimum width <= Wi <= Maximum width

Minimum Height<= Hi <= Maximum height

6.2.1 Preparing ANSYS analysis model for ModelCenter integration

ANSYS is the analysis software used in this example to carry out the analysis. ANSYS provides a set of APDL commands to make the analysis reusable. The set of APDL commands along with the ANSYS commands are referred to as ANSYS analysis models in this research. Since ModelCenter currently does not support direct integration of ANSYS analysis models, the model needs to be modified and prepared for the integration. As the value of the input design variables keeps changing per run in ModelCenter, the design variables and other depending variables have to be defined and named. While the optimization problem formulation remains the same for his example, the implementation is in two folds. First the analysis model is generated for a rectangular cross sectional beam and is implemented in ModelCenter. Second the analysis model is generated for an I cross sectional beam. I-beam analysis is later implemented in ModelCenter during the testing in chapter six to demonstrate reconfiguration capability of the proposed configuration. Both rectangular beam analysis and I-beam analysis models have to be prepared modified and prepared for the ModelCenter integration. The details of this modification are discussed below.

A snapshot of the modification in the analysis model is shown in Figure 5.2 and Figure 5.3. These Figures includes APDL codes up to the preprocessing stage of the analysis model and has been simplified for explanation purpose. A complete ANSYS analysis model can be seen in Appendix A. The modification done is highlighted in the box. Input variables are defined and are highlighted as box 1. These input variables are used in calculating the areas and moments and highlighted in box 2. The variable named

beam length is parsed into the command line while defining the key points and is highlighted in box 3. Finally the input variables and the calculating variables are parsed while defining the real constants. It can also be seen from the figure that before the start of the input variables it has to be commented as "! Input variables" as it is required by the file wrapper.

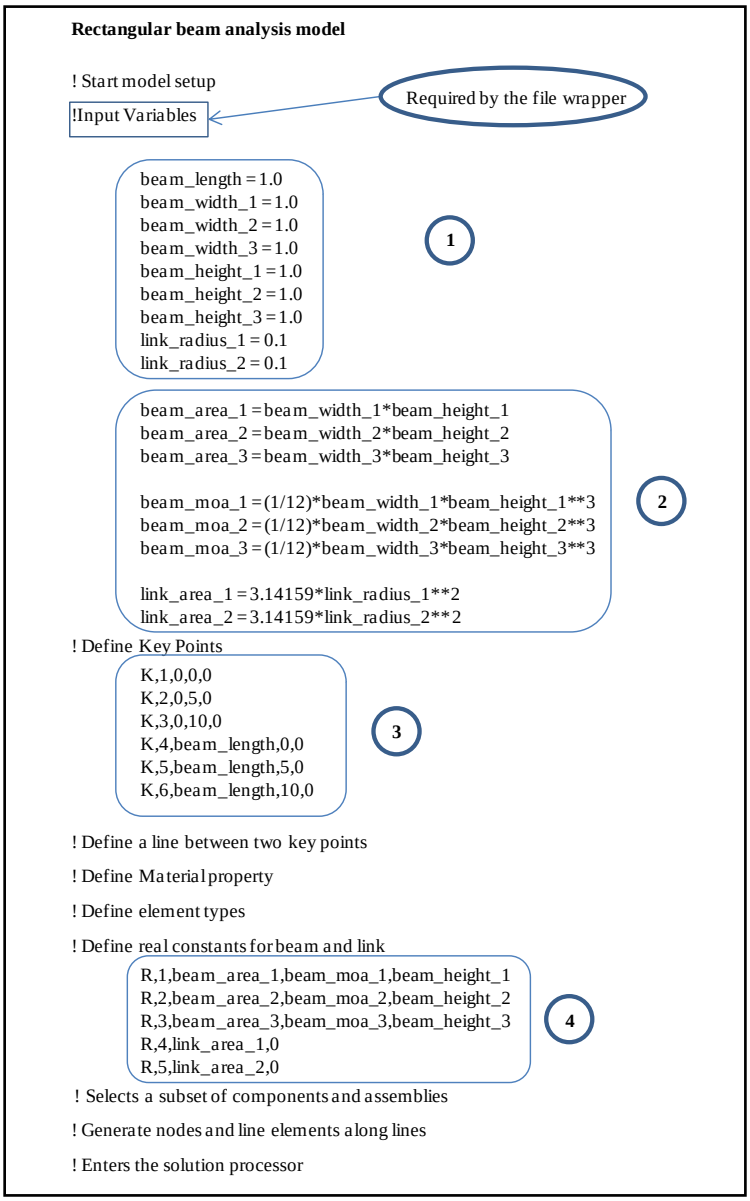


Figure 5.2: Modification in rectangular beam analysis model

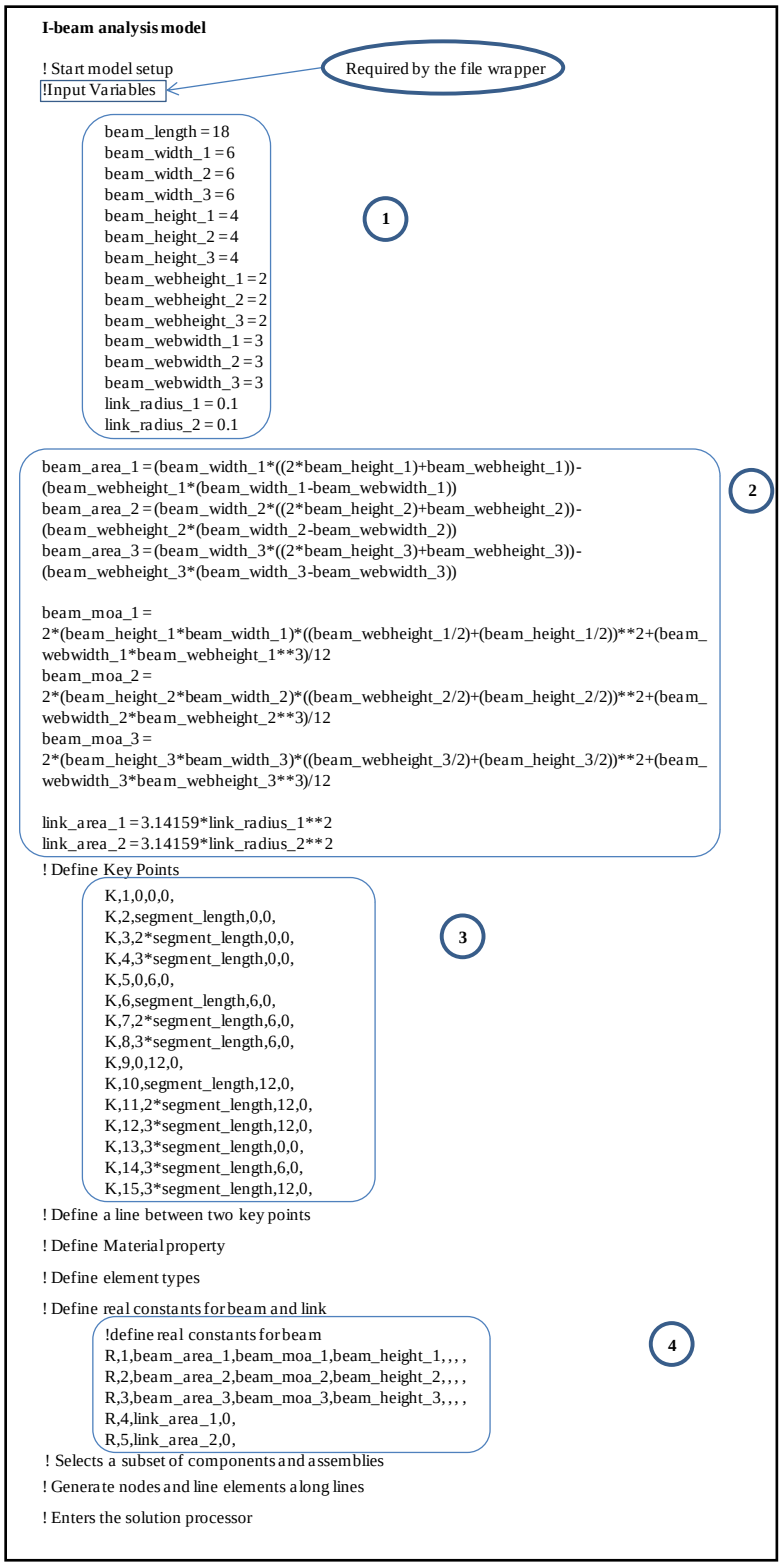


Figure 5.3 : Modification in I beam analysis model

6.2.2 Information extraction

The problem is implemented in ModelCenter using rectangular beam analysis model and the information associated with this problem is extracted with the help of the information model as shown in Figure 4.4 in chapter four. The information is then uploaded into the repository using the Data Modification Language as shown in Table 5.1. The information extracted from this example problem is recorded as follows.

ANALYSIS MODEL

Analysis model name	Analysis_of_rectangular_beam
File name	Analysis_of_rectangular_beam.txt
Analysis type	Structural
Output file	Beam_Analysis_Output.txt

VARIABLES

Variable Name	Variable Type	Data Type
beam_length	Input	Double
beam_width_1	Input	Double
beam_width_2	Input	Double
beam_width_3	Input	Double
beam_height_1	Input	Double
beam_height_2	Input	Double
beam_height_3	Input	Double
link_radius_1	Input	Double
link_radius_2	Input	Double
BeamDeflection1	Output	Double
BeamDeflection2	Output	Double
BeamDeflection3	Output	Double
BeamDeflection4	Output	Double
BeamDeflection5	Output	Double
BeamStress1	Output	Double
BeamStress2	Output	Double
BeamStress3	Output	Double
BeamStress4	Output	Double
BeamStress5	Output	Double
mass	Output	Double

SOFTWARE

Analysis software	Ansys
Version	11.0
Executable URL	C:\Program Files\Ansys Inc\V110\ANSYS\bin\intel\ansys110

MDO PROBLEM

Problem_formulation	Walkway_structure_analysis
Objective function	Z=Length*Area*Density
Design variables	Width, Height, Link Radius
Design parameters	Density, Length
Constraints	Stress constrain

CREATION

File name	Analysis_of_rectangular_beam.txt
User_ID	MDO_user_2
Created on	2008-06-03
Version number	1.0
URL	C:\Program Files\Phoenix Integration\Analysis Server 5.1\analyses\commonfilesystem\AnalysisModels

CREATION

File name	Walkway_structure_analysis_retangular_beam.pxc
User_ID	MDO_user_1
Created on	2008-06-03
Version number	1.0
URL	C:\Program Files\Phoenix Integration\Analysis Server 5.1\analyses\commonfilesystem\ModelCenter Projects

File name	Analysis_of_rectangular_beam.fileWrapper
User_ID	MDO_user_1
Created on	2008-06-03
Version number	1.0
URL	C:\Program Files\Phoenix Integration\Analysis Server 5.1\analyses\commonfilesystem\ ModelCenter Projects

MDO PROJECT

Project name	Walkway_structure_analysis_retangular_beam
File name	Walkway_structure_analysis_retangular_beam.pxc

WRAPPER

Wrapper file name	Analysis_of_rectangular_beam.fileWrapper
Batch_file_name	Analysis_of_rectangular_beam.bat
Description	This is a file wrapper component which wraps the analysis model Analysis_of_rectangular_beam.txt

Database implementation

The information extracted above is implemented in the database using the data modification language as shown in Table 5.1.

Table 5.1: Data modification language

```
INSERT INTO SOFTWARE (Software_name, Version, Product_name,
Executable_URL)
VALUES ("ANSYS", 10.0, "aa_t_me", "C:\Program Files\Ansys
Inc\V100\ANSYS\bin\intel\ansys100");

INSERT INTO PERSON (Fname, Lname, User_ID, Department, Position )
VALUES ("Santosh", "Hiriyannaiah", "MDO_user_1", "Mechanical Engineering",
"Designer");

INSERT INTO PERSON (Fname, Lname, User_ID, Department, Position )
VALUES ("S", "Hiriya", "MDO_user_2", "Mechanical Engineering", "Analyst");

INSERT INTO ANALYSISMODEL ( Analysis_model_name, AFile_name,
Analysis_type ,Input_file, Output_file, Software_name )
VALUES ("Analysis_of_rectangular_beam", "Analysis_of_rectangular_beam.txt",
"Structural", "null", "rec_beam_fea_output.txt", "ANSYS");

INSERT INTO VARIABLES (Variable_name, Variable_type , Data_type,
Analysis_model_name )
VALUES ("beam_length", "Input", "Double", "Analysis_of_rectangular_beam");

INSERT INTO MDOPROBLEM (Problem_formulation, Objective_function,
Design_variable, Constraints)
VALUES ("walkway_structure_analysis", "Z=Length*Area*Density",
"Width,Height,LinkRadius", "Stess constraint");

INSERT INTO MDOPROJECT (Project_name, PFile_name, Problem_formulation,
```

Analysis_model_name, Optimization_component)
VALUES (“walkway_structure_analysis_rectangular_beam”,
“walkway_structure_analysis_rectangular_beam.pxc”, “walkway_structure_analysis”,
“Analysis_of_rectangular_beam”, “GradientOptimizer”);

INSERT INTO WRAPPER (Wrapper_File_name, Analysis_model_name,
Project_name, Batch_file_name, Description)
VALUES (“Analysis_of_rectangular_beam.fileWrapper”,
“Analysis_of_rectangular_beam”, “walkway_structure_analysis_rectangular_beam”,
“Analysis_of_rectangular_beam.bat”, “This is a file wrapper component which wraps
the analysis model Analysis_of_rectangular_beam. txt”);
INSERT INTO DESIGNVARIABLE (DV_name, Problem_formulation, Project_name,
Lower_bound, Upper_bound)
VALUES (“beam_width_1”, “walkway_structure_analysis”,
“walkway_structure_analysis_rectangular_beam”, null ,null);

INSERT INTO CREATION (File_name, User_ID, Created_on, URL)
VALUES (“Analysis_of_rectangular_beam.txt”, “MDO_user_2”, “2008-06-03”,
“C:\Program Files\Phoenix Integration\Analysis Server
5.1\analyses\commonfilesystem\AnalysisModels”);

INSERT INTO CONSTRAINT
(CVariable_name,Problem_formulation,Lower_bound,Upper_bound) VALUES
(“BeamStress1”, “walkway_structure_analysis”, null, null);

CHAPTER SIX

TESTING AND INFERENCES

Testing is a process of exercising a software component using a selected set of test cases with intent of revealing defect and evaluating quality [6]. In this chapter testing is referred to as a process of exercising the components of the proposed ModelCenter – Analysis Server configuration under a scenario, with intent of evaluating the quality and the performance. The quality is measured in terms of adherence to the scenario specific requirements and performance is measured by the consistency in the output and ease in usage of the components of the proposed configuration. Three tests are conducted and a functional based testing approach is utilized to see how ModelCenter – Analysis Server configuration performs under a particular scenario. For these tests the example of analysis and optimization of walkway beam structure described in chapter five is considered.

Testing is monitored, executed and recorded with the help of test documents which are adopted from the documentation standards specified in IEEE standard for software test documentation [14].The test documents for testing include test plan, test case specification, test procedure specification, test log and a test summary report.

6.1 Overview of the testing

Each test is designed and developed to test and demonstrate specific functionality of the proposed configuration. An overview of the functionality demonstrated in the tests is as shown in Table 6.1

Table 6.1 : Overview of the functions tested

Functions offered by proposed configuration	Test 1	Test 2	Test 3
Automate batch file creation	✓	✓	
Automate of file wrapper creation	✓		
Update information	✓	✓	
Retrieve information	✓		
Reconfigure MDO problem			✓
Efficient search of files		✓	✓
Reuse MDO components			✓
Update batch file		✓	
Store files in a structure	✓	✓	✓

6.2 TEST 1: File wrapper and batch file creation

In this test a scenario is presented in which a new file wrapper and a batch file needs to be created for a new analysis model such that it can be integrated into ModelCenter. The overall objective of this test is to evaluate the quality and performance of the wrapper generator, structured repository and common file system under the test scenario. The quality is measured in terms of adherence to the task specific requirements while creating the file wrapper and performance is measured by the consistency in the output and ease in creating the file wrapper and its corresponding batch file.

6.2.3 Test plan

Introduction

In this test structured repository, common file system and wrapper generator application are tested. Structured repository is tested for retrieval of information for file wrapper and batch file creation and for allowing the update of the wrapper file information. Common file system is tested for storage of these files in a structure and Wrapper generator application is tested for automation of file wrapper and batch file creation.

Scenario

When a new ANSYS analysis model is introduced into the design, a file wrapper and a corresponding batch file needs to be created. In a design environment the main focus of the designer is to find alternatives for the design through design exploration techniques and trade off studies provided by ModelCenter. At the same time it is also important to take care that the file wrapper and batch files are properly created and stored in Analysis Server. This ensures proper integration and execution of the analysis model in ModelCenter.

Scenario specific requirements

1. File wrapper format should adhere to the format specified in the documentation
2. Batch file should have the same name as specified in file wrapper
3. Batch file should run the analysis model for the user specified inputs

   (The input file should be the copy of the analysis file generated by the file wrapper)
4. Analysis model, batch file and file wrapper should be in the same folder

Test items

The item or components of the proposed configuration tested are

- Wrapper generator application
- Common file system
- Structured repository

Functions to be tested

- Automate batch file creation
- Automate of file wrapper creation
- Storage of file wrapper, batch file and other files
- Update of the file information.
- Retrieve information for file wrapper

Approach

- Test case 1 and test case 2 are executed to test update and retrieval function of the structured repository.
- Test case 3 and test case 4 are executed to test storage function of the common file system to store file wrapper, batch file and analysis model file
- Test case 5, test case 6 and test case 7 are executed to test the function of the wrapper generator application to automate of file wrapper creation
- Test case 8 and test case 9 are executed to test function of the wrapper generator application to automate batch file creation

(Refer test case specification for more details)

Item pass/fail criteria

PASS: When actual output agrees with the expected output

FAIL: When actual output does not agrees with the expected output

Testing tasks

Task 1: Execute test case 1 and test case 2

- Check if information of analysis model for file wrapper creation is retrieved
- Check if information of analysis model for file wrapper creation is updated

Task 2: Execute test case 3 and test case 4

- Check for the creation of file wrapper folder in Common File System
- Check if a copy of analysis file is placed in the file wrapper folder
- Check if analysis file, batch file and file wrapper in the same folder

Task 3: Execute test case 5, test case 6 and test case 7

- Check the format of the automated file wrapper to match the prescribed format

Task 4: Execute test case 8 and test case 9

- Check if the format of the automated batch file matches to the prescribed format
- Check the name of the batch file to be same as specified in file wrapper
- Check if the input file name in batch file is same as file to generate name in file wrapper

Environmental needs

Before the execution of the test some test preparation steps have to be carried out. The software tools required to run the test are ModelCenter 7.1, Analysis Server 5.1 and NetBeans5.5.1 IDE. Mentioned below are the steps involved to set up a test environment

1. Load wrapper generator application into the NetBeans IDE and is set for the run
2. Provide print statements in the application to view queried results in the output window of NetBeans IDE
3. Log into Analysis Server 5.1to access common file system under the path "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/"
4. Open ModelCenter 7.1 to view the server browser

6.2.4 Test case specification

Purpose

Details of each of the test cases are specified. The test cases test certain functions of structured repository, common file system and wrapper generator application. This section is referred by Section 6.2.3 for the details of the test cases. These test cases are executed sequentially by the tasks.

TEST CASE 1	
Test items	Wrapper generator application and Structured Repository
Function tested	Retrieve information for file wrapper
Inputs	"Analysis_of_rectangular_beam" into wrapper generator
Outputs	Queried output from Wrapper generator o Name of input variables o Name of output variables o Number of input and output variables o Analysis software name o Analysis model file name o Analysis model generated output file name
pass/fail criteria	• **Pass**: If the actual queried output displayed in the output window matches with the expected output • **Fail**: If the actual queried output displayed in the output window does not matches with the expected output

TEST CASE 2	
Test items	Wrapper generator application and structured repository
Function tested	Update file wrapper and batch file information
Inputs	**Submit query:** SELECT Wrapper_file_name,Project_name, Batch_file_name FROM wrapper WHERE Analysis_model_name='Analysis_of_rectangular_beam';
Outputs	Should return updated information about file wrapper and batch file from wrapper table of the repository
pass/fail criteria	• **Pass**: If test query returns file wrapper and batch file information • **Fail**: If test query does not returns file wrapper and batch file information

TEST CASE 3	
Test items	Wrapper generator application and common file system
Function tested	Store file wrapper and batch file
Inputs	"Analysis_of_rectangular_beam" into wrapper generator application
Outputs	• Folder created in Common file system under file wrapper folder (Folder name = Analysis_of_rectangular_beam) • File wrapper and batch file stored in the file wrapper folder
pass/fail criteria	• **Pass**: If folder name equals "Analysis of a rectangular beam " and file wrapper and batch file exists in that folder • **Fail**: If folder name is not equals "Analysis of a rectangular beam" and file wrapper and batch file does not exists in that folder

TEST CASE 4	
Test items	wrapper generator application and Common file system
Function tested	Store a copy of analysis model file into file wrapper folder
Inputs	"Analysis_of_rectangular_beam" into Wrapper generator application
Outputs	"Analysis_of_rectangular_beam.txt" stored in file wrapper folder
pass/fail criteria	• **Pass**: If "Analysis of a rectangular beam.txt" exists in the file wrapper folder • **Fail**: If "Analysis of a rectangular beam.txt" does not exists in the file wrapper folder

TEST CASE 5	
Test items	Wrapper generator application and Common file system
Function tested	Automate file wrapper creation
Inputs	"Analysis_of_rectangular_beam" to wrapper generator application
Outputs	Access "Analysis_of_rectangular_beam.fileWrapper" in server browser in ModelCenter
pass/fail criteria	• **Pass**: If "Analysis_of_rectangular_beam.fileWrapper" can be accessed through server browser in ModelCenter • **Fail**: If "Analysis_of_rectangular_beam"fileWrapper" cannot be accessed through server browser in ModelCenter

TEST CASE 6	
Test items	Wrapper generator application
Function tested	Automate file wrapper creation
Inputs	"Analysis_of_rectangular_beam" to wrapper generator application
Outputs	"Analysis_of_rectangular_beam.fileWrapper" file with commands adhering to the prescribed syntax
pass/fail criteria	• **Pass:** If the format of the Analysis_of_rectangular_beam.fileWrapper command match the prescribed format • **Fail**: If the format of the Analysis_of_rectangular_beam.fileWrapper command does not match the prescribed format

TEST CASE 7	
Test items	Wrapper generator application
Function tested	Automate of batch file creation
Inputs	"Analysis_of_rectangular_beam" to wrapper generator application
Outputs	Batch file name "Analysis of a rectangular beam.bat"= Batch file name specified in Analysis of a rectangular beam.fileWrapper
pass/fail criteria	• **Pass**: If the batch file name specified in the automated file wrapper is same as the automated batch file name • **Fail**: If the batch file name specified in the automated file wrapper is not same as the automated batch file name

TEST CASE 8	
Test items	Wrapper generator application
Function tested	Automate batch file creation
Inputs	Analysis model name to wrapper generator application
Outputs	"Analysis_of_ rectangular_beam.bat" file with commands adhering to the prescribed syntax
pass/fail criteria	• **Pass**: If the format of the Analysis_of_ rectangular_beam.bat command match the prescribed format • **Fail**: If the format of the Analysis_of_ rectangular_beam.bat command match the prescribed format

TEST CASE 9	
Test items	Wrapper generator application
Function tested	Automate batch file creation
Inputs	Analysis model name to wrapper generator application
Outputs	Automated "Analysis_of_ rectangular_beam.fileWrapper" and "Analysis of a rectangular beam.bat "with input file name in the command line of .bat file = file name in File to generate section of .file wrapper file
pass/fail criteria	• **Pass**: If input file name in.bat file equals file name in File to generate section of .file wrapper file • **Fail**: If input file name in.bat file not equals file name in File to generate section of .file wrapper file

6.2.5 Test procedure specification

This section provides details of the steps required to execute the task specific test cases. This procedure details is referred by testing task section in the test plan. Each task is executed sequentially as described to ensure the prerequisite for a particular task execution is met in the previous task.

Test procedure for task 1

Purpose

This procedure describes the steps required for the execution of test case 1 and test case 2 to check for the retrieval of analysis model information and to check if the newly created file wrapper information is updated in the repository

Procedure steps

1. Run Wrapper generator application
2. Enter the analysis model name as "Analysis_of_rectangular_beam" when prompted by the wrapper generator application
3. View the queried output from wrapper generator application
4. Query the repository to check if the file wrapper information is updated
5. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 1and test case 2
6. Log the result along with the pass/fail criteria in the Test log document

Test procedure for task 2

Purpose

This procedure describes the steps required for the execution of test case 3, test case 4 to check if the file wrapper folder is created in the common file system and to check if a copy of analysis file , batch file and file wrapper are in the in the same folder

Procedure steps

1. Ensure test procedure 1is executed
2. Go to the path "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ Analysis_of_ rectangular_beam /" in Analysis Server
3. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 3and test case 4
4. Log the result along with the pass/fail criteria in the Test log document

Test procedure for task 3

Purpose

This procedure describes the steps required for the execution of test case 5, test case 6 and test case 7 to check if the format of the automated file wrapper matches to the prescribed format

Procedure steps

1. Ensure test procedure 1is executed
2. Go to the path "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ Analysis_of_ rectangular_beam /" in Analysis Server

3. Open Analysis_of_ rectangular_beam.fileWrapper
4. Compare it to the prescribed file wrapper format to the one specified in ModelCenter documentation
5. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 5 test case 6and test case 7
6. Log the result along with the pass/fail criteria in the Test log document

Test procedure for task 4

Purpose

This procedure describes the steps required for the execution of test case 8, test case 9 to check if the format of the automated batch files matches to the prescribed format and to ensure if the input file name in batch file is same as file to generate name in file wrapper

Procedure steps

1. Ensure test procedure 1is executed
2. Go to the path "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ Analysis_of_ rectangular_beam /" in Analysis Server
3. Open Analysis of a rectangular beam.bat
4. Compare it to the prescribed batch file format
5. Open Analysis of a rectangular beam.fileWrapper
6. Compare the input file name in batch file to the file to generate name in file wrapper

7. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 8and test case 9
8. Log the result along with the pass/fail criteria in the Test log document

6.2.6 Test log

Description

The items tested in this test are wrapper generator application, structured repository and common file system. For each task the expected procedure result is described and the pass and fail decision is based on the criteria mentioned in their respective test case specification (refer test case specification section 6.2.3). The test is executed task wise and the results are recorded below.

TASK 1 Execution

***Expected procedure results*:** Test case 1

- *Input variables:* beam_length, beam_width_1, beam_width_2, beam_width_3, beam_height_1, beam_height_2, beam_height_3, link_radius_1, link_radius_2
- *Output variables:* BeamDeflection1, BeamDeflection2, BeamDeflection3, BeamDeflection4, BeamDeflection5 , BeamStress1, BeamStress2, BeamStress3, BeamStress4, BeamStress5, mass
- *Number of input variables:* 9
- *Number of output variables:* 11
- *Analysis software name:* Ansys
- *Analysis model file name:* Analysis_of_ rectangular_beam.txt

- *Analysis model generated output file name:* Beam_Analysis_Output.txt

***Expected procedure results*:** Test case 2

- *Analysis model name* = Analysis_of_ rectangular_beam
- *Wrapper file name* Analysis_of_ rectangular_beam.fileWrapper
- *Batch file name* = Analysis_of_ rectangular_beam.bat

Actual procedure results: Test case 1

The snapshots of the actual result are obtained from the output window of NetBeans 5.5.1.

- *Input variables*

```
Input Variable: beam_height_1
Input Variable: beam_height_2
Input Variable: beam_height_3
Input Variable: beam_length
Input Variable: beam_width_1
Input Variable: beam_width_2
Input Variable: beam_width_3
Input Variable: link_radius_1
Input Variable: link_radius_2
```

- *Output variable*

```
Output Variable: BeamDeflection1
Output Variable: BeamDeflection2
Output Variable: BeamDeflection3
Output Variable: BeamDeflection4
Output Variable: BeamDeflection5
Output Variable: BeamStress1
Output Variable: BeamStress2
Output Variable: BeamStress3
Output Variable: BeamStress4
Output Variable: BeamStress5
Output Variable: mass
```

- *Number of input variables*

```
Number of input variables =  9
```

- *Number of output variables*

```
Number of output variables =  11
```

- *Analysis software name*

```
Analysis Software name =  ANSYS
```

- *Analysis model file name*

```
Analysis model file name =  Analysis_of_rectangular_beam.txt
```

- *Analysis model generated output file name*

```
Analysis model generated output file name =  Beam_Analysis_Output.txt
```

Actual procedure results: Test case 2

```
mysql> SELECT Analysis_model_name, Wrapper_file_name,Batch_file_name
    -> FROM wrapper
    -> WHERE Analysis_model_name='Analysis_of_rectangular_beam';
+-----------------------------+-----------------------------------------+-------------------------------+
| Analysis_model_name         | Wrapper_file_name                       | Batch_file_name               |
+-----------------------------+-----------------------------------------+-------------------------------+
| Analysis_of_rectangular_beam | Analysis_of_rectangular_beam.fileWrapper | Analysis_of_rectangular_beam.bat |
+-----------------------------+-----------------------------------------+-------------------------------+
1 row in set (0.00 sec)
```

Pass / Fail decision

- *Test case 1: Pass*

  The actual queried output displayed in the output window matches with the expected output

- *Test case 2: Pass*

  The test query returns file wrapper and batch file information

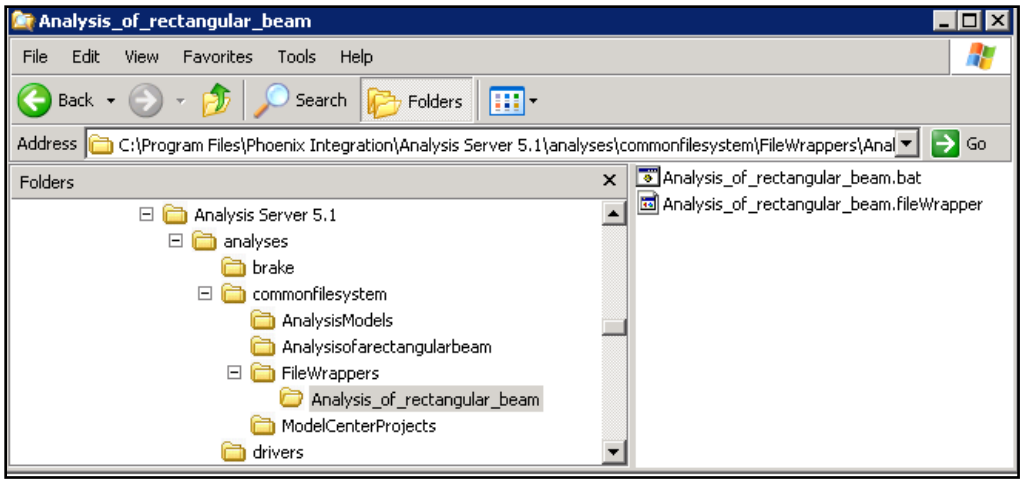
TASK 2 Execution

Expected procedure results

- *Folder name in file wrapper folder = Analysis_of_rectangular_beam*
- *File wrapper and batch file stored in the file wrapper folder*

Actual procedure results

- *Folder name = Analysis_of_rectangular_beam*
- *File wrapper and batch file stored in the file wrapper folder*



Pass / Fail decision

- *Test case 3:* Pass

  The folder name equals "Analysis of a rectangular beam" and file wrapper and batch file exists in that folder

- *Test case 4:* Pass

  The file "Analysis of a rectangular beam.txt" exists in the Analysis of a rectangular beam file wrapper folder

TASK 3 Execution

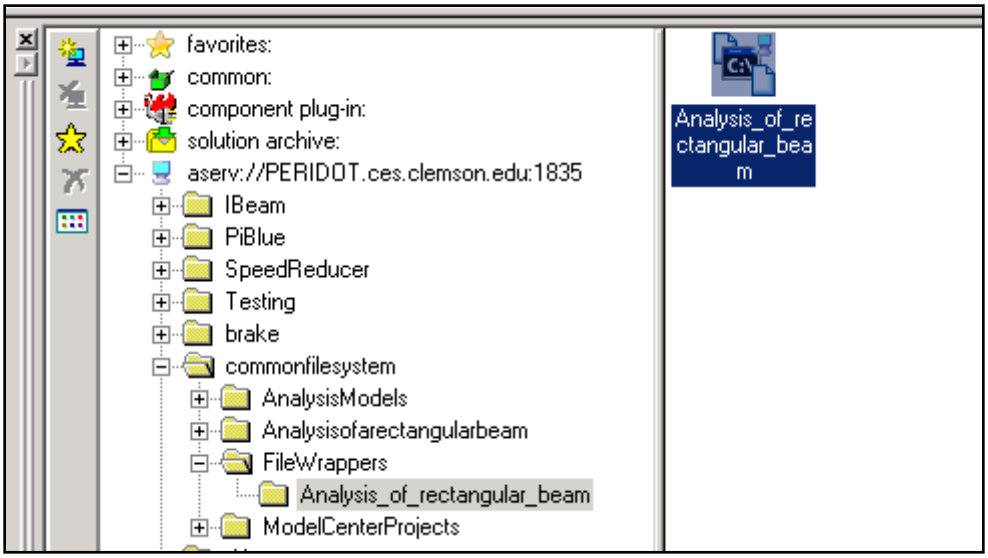
Expected procedure results

- Access "Analysis_of_rectangular_beam.fileWrapper" in server browser in ModelCenter
- "Analysis_of_rectangular_beam.fileWrapper" file with commands adhering to the prescribed syntax. Should include header,Run, RowFieldInputfile and RowFieldOutnputfile

- Batch file name “Analysis_of_rectangular_beam.bat”= Batch file name specified in the run section of Analysis_of_rectangular_beam.file Wrapper

Actual procedure results

- “Analysis_of_rectangular_beam.fileWrapper” in server browser in ModelCenter



- The sections in “Analysis_of_rectangular_beam.fileWrapper” adhering to the prescribed syntax. It includes all the four sections

Analysis_of_rectangular_beam.file...

```
# @author: Santosh Hiriyannaiah
# @version: Trial
# @description: File Wrapper

RunCommands
{
generate inputFile
run "Analysis_of_rectangular_beam.bat"
parse outputFile
}

RowFieldInputFile inputFile
{
templateFile:Analysis_of_rectangular_beam.txt
fileToGenerate:Analysis_of_rectangular_beam.in

markAsBeginning "Input Variables"

variable: beam_height_1      Double      2                3
variable: beam_height_2      Double      3                3
variable: beam_height_3      Double      4                3
variable: beam_length        Double      5                3
variable: beam_width_1       Double      6                3
variable: beam_width_2       Double      7                3
variable: beam_width_3       Double      8                3
variable: link_radius2       Double      9                3
variable: link_radius_1      Double      10               3
}

RowFieldOutputFile outputFile
{
fileToParse:Beam_Analysis_Output.txt
setDelimiters " ="

keyVar: BeamDeflection1      Double "BeamDeflection1"
keyVar: BeamDeflection2      Double "BeamDeflection2"
keyVar: BeamDeflection3      Double "BeamDeflection3"
keyVar: BeamDeflection4      Double "BeamDeflection4"
keyVar: BeamDeflection5      Double "BeamDeflection5"
keyVar: BeamStress1      Double "BeamStress1"
keyVar: BeamStress2      Double "BeamStress2"
keyVar: BeamStress3      Double "BeamStress3"
keyVar: BeamStress4      Double "BeamStress4"
keyVar: BeamStress5      Double "BeamStress5"
keyVar: mass             Double "mass"
}
```

- Batch file name “Analysis_of_rectangular_beam.bat” and Batch file name specified in Analysis_of_rectangular_beam.file Wrapperare same

```
RunCommands
{
generate inputFile
run "Analysis_of_rectangular_beam.bat"
parse outputFile
}
```

Pass / Fail decision

- *Test case 5:* Pass

  “Analysis_of_rectangular_beam.fileWrapper” can be accessed through server browser in ModelCenter

- *Test case 6:* Pass

  Format of the Analysis_of_rectangular_beam.fileWrapper command match the prescribed format

- *Test case7:* Pass

  The batch file name specified in the automated file wrapper is same as the automated batch file name

TASK 4 Execution

Expected procedure results

- "Analysis_of_rectangular_beam.bat" file with commands adhering to the syntax specified below

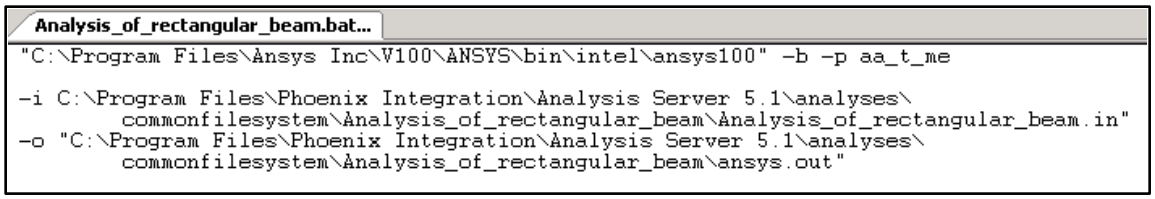
"<drive>:\Program Files\Ansys Inc\V100\ANSYS\bin\intel\ansys100" –b -i inputname -o outputname

- Automated "Analysis_of_rectangular_beam.fileWrapper" and "Analysis of a rectangular beam.bat "with input file name in the command line of .bat file = file name in File to generate section of .file wrapper file

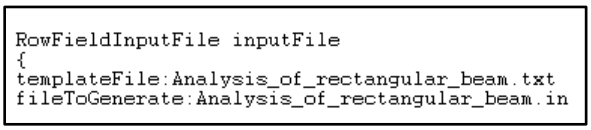
```
RowFieldInputFile inputFile
{
templateFile:Analysis_of_rectangular_beam.txt
fileToGenerate:Analysis_of_rectangular_beam.in

markAsBeginning "Input Variables"
```

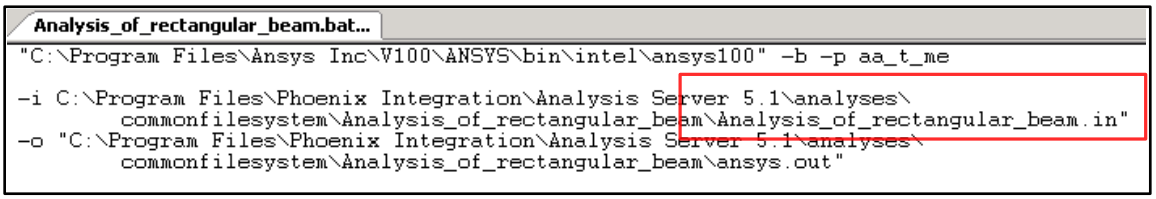
Actual procedure results

- The command line in "Analysis_of_rectangular_beam.bat" file adhering to the prescribed syntax



- Input file name in the command line of Analysis_of_rectangular_beam .bat file = file name in FiletoGenerate section of .file wrapper file





Pass / Fail decision

- *Test case8:* Pass

  The format of the Analysis_of_rectangular_beam.bat command match the prescribed format

- *Test case 9:* Pass

  Input file name in.bat file equals file name in File to generate section of .file wrapper file

6.2.7 Test summary report

The test results are tabulated in Table 6.2. The pass result from test case 1 and test case 2 indicates that the structured repository allows the queries to be performed to retrieve and update information. The pass result from test case 3 and test case 4 indicates

that the common file system stores the file wrapper, batch file and the analysis model in a folder structure. The pass result from test case 5, test case 6, test case 7, test case 8 and test case 9 indicates that the wrapper generator Java application successfully generates the file wrapper and batch file with minimal user interaction.

Table 6.2: Test 1 result summary

Test cases	Result
Test case 1 ; Test case 2	Pass
Test case 3 ; Test case 4	Pass
Test case 5 ; Test case 6; Test case 7	Pass
Test case 8 ; Test case 9	Pass

The features/items of the proposed configuration meet the task specific requirements while creating the file wrapper. Thus the wrapper generator creates file wrapper and batch files with good quality.

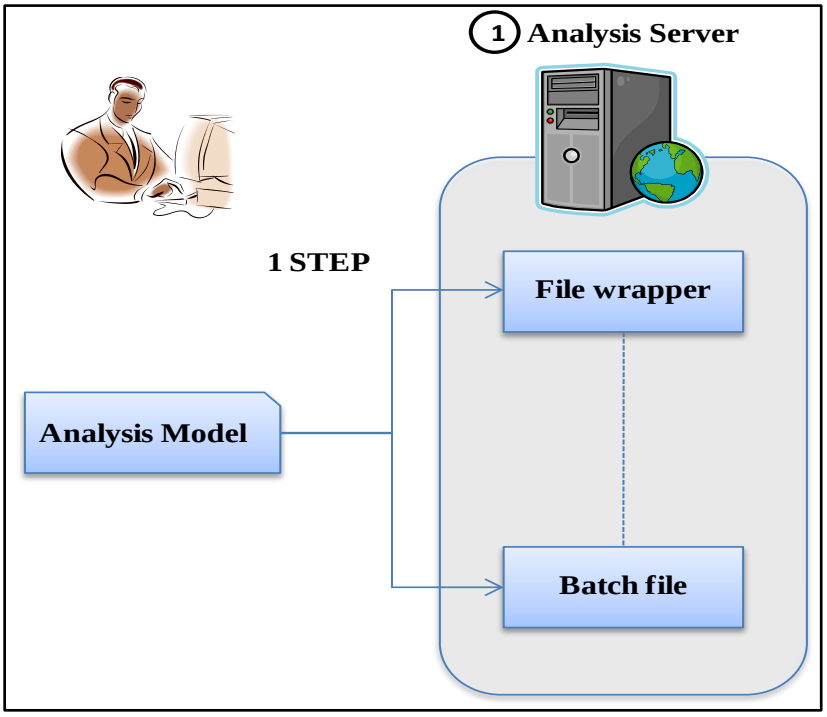


Figure 6. 1: Steps in wrapper creation using wrapper generator application

With the help of the information from the structured repository and the wrapper generator the file wrapper and batch file creation is reduced to 1 step process (see Figure 6.1) from 18 step process. This demonstrated the utilization the structured repository and the wrapper generator for better performance.

6.3 TEST 2: Migration of analysis software

In this test a scenario is presented in which there is a migration of analysis software version from ANSYS 10.0 to ANSYS 11.0. The overall objective of this test is to evaluate under a scenario the quality in terms of adherence to task specific requirements and performance of the wrapper generator in terms of ease in updating the batch file. The test demonstrates ease in updating a batch file when the migration occurs.

6.3.1 Test plan

Introduction

In this test structured repository and wrapper generator application are tested. Structured repository is tested for retrieval of analysis software executable location information to update the batch file. Wrapper generator application is tested for function in which the batch file is updated.

Scenario

In spite of setting up the problem and creating file wrappers for the ANSYS analysis models, designers might face issue such as migration of analysis software version. This migration of analysis software from across version requires change in the batch files for the proper execution of the corresponding file wrapper. The change to be

made is in the command line of the batch file. The ANSYS 10.0 software executable location needs to be updated to ANSYS 11.0 software executable location. It is also required to ensure proper command line syntax in the batch file and the storage of the batch file in the corresponding file wrapper folder.

Scenario specific requirements

1. The batch file should run the latest version of analysis software
2. The batch file should be stored in the corresponding file wrapper folder

Test items

The item or components of the proposed configuration tested are

- Wrapper generator application
- Structured repository

Functions to be tested

- Automate batch file creation
- Retrieve information for batch file creation
- Store the batch file in the file wrapper folder
- Update batch file

Approach

- Test case 10 is executed to test if the retrieval function of the repository
- Test case 3 is executed to test the storage function of the wrapper application
- Test case 8 is executed to test the automate batch file function
- Test case 11 is executed to test the update batch file function of the wrapper generator

(Refer Section 6.3.2 Test case specification for more details)

Item pass/fail criteria

PASS: When actual output agrees with the expected output

FAIL: When actual output does not agrees with the expected output

Testing tasks

Task 1: Execute test case 10

- Check the queried output for batch file information

Task 2: Test case 3

- Check the batch file storage in the corresponding file wrapper folder

Task 2: Test case 8

- Check the batch file creation for proper syntax

Task 2: Test case 11

- Check the batch file is updated with latest version

Environmental needs

Before the execution of the test some test preparation steps have to be carried out. The software tools required to run the test are ModelCenter 7.1, Analysis Server 5.1 and NetBeans5.5.1 IDE. Mentioned below are the steps involved to set up a test environment

1. Load Wrapper generator application into the NetBeans IDE
2. Provide print statements in the application to view queried results in the output window of NetBeans IDE
3. Log into Analysis Server 5.1to access common file system under the path

"C: /Program Files/Phoenix Integration/Analysis Server

5.1/analyses/commonfilesystem/"

4. Update the repository with the new analysis software version information

6.3.2 Test case specification

Purpose

Details of each of the test cases are specified. The test cases test certain functions of structured repository, common file system and wrapper generator application. This section is referred by Section 6.3.1 test plan document for the details of the test cases. Test case 3 and test case 8 from reused from test 1.

TEST CASE 10	
Test items	Wrapper generator application and structured repository
Function tested	Retrieve information for batch file
Inputs	"Analysis_of_rectangular_beam" to wrapper generator application
Outputs	Queried output from Batch File Generator class o Executable_URL o Product name
pass/fail criteria	• **Pass**: If the actual queried output displayed in the output window matches with the expected output • **Fail**: If the actual queried output displayed in the output window does not matches with the expected output

TEST CASE 3	
Test items	Wrapper generator application and common file system
Function tested	Store the batch file in the file wrapper folder
Inputs	"Analysis_of_rectangular_beam" to wrapper generator application
Outputs	• Folder created in Common file system under file wrapper folder (Folder name = Analysis_of_rectangular_beam) • File wrapper and batch file stored in the file wrapper folder
pass/fail criteria	• **Pass**: If folder name equals "Analysis of a rectangular beam " and file wrapper and batch file exists in that folder • **Fail**: If folder name is not equals "Analysis of a rectangular beam" and file wrapper and batch file does not exists in that folder

TEST CASE 8	
Test items	Wrapper generator application
Function tested	Automate batch file creation
Inputs	"Analysis model name" to wrapper generator application
Outputs	"Analysis_of_ rectangular_beam.bat" file with commands adhering to the prescribed syntax
pass/fail criteria	• **Pass**: If the format of the Analysis_of_ rectangular_beam.bat command match the prescribed format • **Fail**: If the format of the Analysis_of_ rectangular_beam.bat command match the prescribed format

TEST CASE 11	
Test items	Wrapper generator application
Function tested	Automate batch file creation
Inputs	"Analysis_of_rectangular_beam" into Wrapper generator application
Outputs	Batch file command line includes ANSYS 11.0 version executable location
pass/fail criteria	• **Pass**: If input file name in.bat file equals file name in File to generate section of .file wrapper file • **Fail**: If input file name in.bat file not equals file name in File to generate section of .file wrapper file

6.3.3 Test procedure specification

This section provides details of the steps required to execute the task specific test cases. This procedure details is referred by testing task Section in the test plan. Each task is executed sequentially as described to ensure the prerequisite for a particular task execution is met in the previous task.

Test procedure for task 1

Purpose

This procedure describes the steps required for the execution of test case 10 to check the queried output for batch file information

Procedure steps

1. Run Wrapper generator application
2. Enter the analysis model name as Analysis_of_rectangular_beam when prompted by the wrapper generator application
3. View the queried batch file output from wrapper generator application
4. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 1and test case 10
5. Log the result along with the pass/fail criteria in the Test log document

Test procedure for task 2

Purpose

This procedure describes the steps required for the execution of test case 3 to check the batch file storage in the corresponding file wrapper folder

Procedure steps

1. Go to the path "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ Analysis_of_ rectangular_beam /" in Analysis Server
2. Look in the folder for "Analysis_of_rectangular_beam.bat"
3. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 3
4. Log the result along with the pass/fail criteria in the Test log document

Test procedure for task 3

Purpose

This procedure describes the steps required for the execution of test case 8 to check the batch file creation for proper syntax

Procedure steps

1. Ensure test procedure 1is executed
2. Go to the path "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ Analysis_of_ rectangular_beam /" in Analysis Server
3. Open Analysis_of_rectangular_beam.bat
4. Compare it to the prescribed batch file format
5. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 8and test case 8
6. Log the result along with the pass/fail criteria in the Test log document

Test procedure for task 4

Purpose

This procedure describes the steps required for the execution of test case 11 to check the batch file is updated with the new analysis software executable location

Procedure steps

1. Go to the path "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ Analysis_of_ rectangular_beam /" in Analysis Server
2. Open Analysis_of_rectangular_beam.bat
3. Check the executable location details in the command line
4. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 11
5. Log the result along with the pass/fail criteria in the Test log document

6.3.4 Test log

Description

The items tested in this test are wrapper generator application and the structured repository. For each task the expected procedure result is described and the pass and fail decision is based on the criteria mentioned in their respective test case specification. The test is executed task wise and the results are recorded below.

TASK 1 Execution

***Expected procedure results*:** Test case 10

- Analysis software name = ANSYS
- Executable URL = C:\ Program Files\Ansys Inc\V110\ANSYS\bin\intel\ansys110
- Product name = aa_t_me

Actual procedure results: The snapshots of the actual result are obtained from the output window of NetBeans 5.5.1.

- *Analysis software name*

```
Analysis Software name =  ANSYS
```

- Executable URL

```
Executable_URL =  C:\Program Files\Ansys Inc\V110\ANSYS\bin\intel\ansys110
```

- Product name

```
ProductName =  aa_t_me
```

Pass / Fail decision

- *Test case 10: Pass*

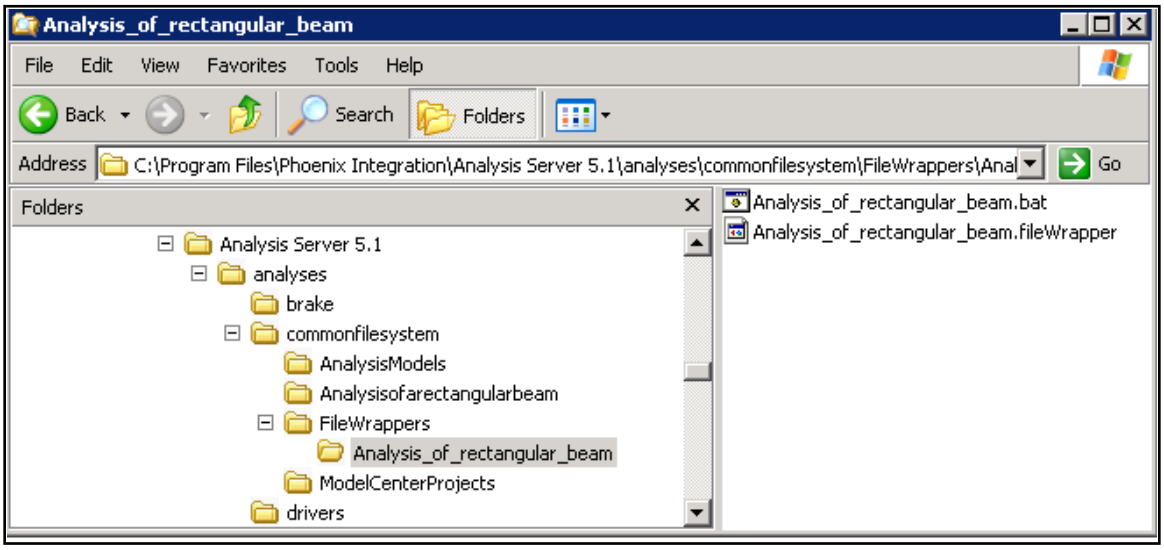
  The actual queried output displayed in the output window matches with the expected output

TASK 2 Execution

Expected procedure results: Test case 3

- "Analysis_of_rectangular_beam.bat" stored in Analysis_of_rectangular_beam folder under file wrapper folder

Actual procedure results



Pass / Fail decision

- *Test case 3: Pass*

  The file "Analysis_of_rectangular_beam.bat" exists in the

  Analysis_of_rectangular_beam file wrapper folder

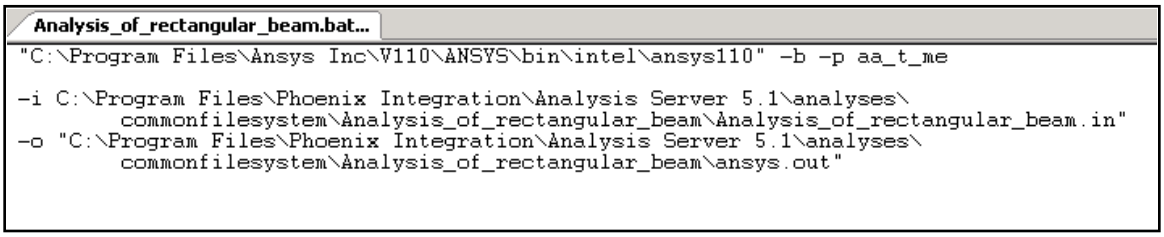
TASK 3 Execution

Expected procedure results: Test case 8

- *"Analysis_of_rectangular_beam.bat" file with commands adhering to the syntax specified below*

"<drive>:\Program Files\Ansys Inc\V110\ANSYS\bin\intel\ansys110" -b -i inputname -o outputname

Actual procedure results



Pass / Fail decision

- *Test case 8: Pass*

  The format of the Analysis_of_rectangular_beam.bat command match the prescribed format

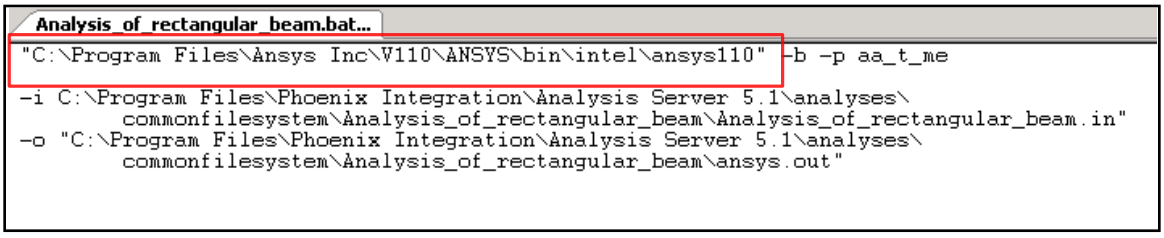
TASK 4 Execution

Expected procedure results: Test case 11

- Batch file command line includes ANSYS 11.0 version executable location

  "C:\ Program Files\Ansys Inc\V110\ANSYS\bin\intel\ansys110"

Actual procedure results (snapshot of the file wrapper folder in common file system)



Pass / Fail decision

- *Test case 11: Pass*

  Batch file command line includes ANSYS 11.0 version executable location

6.3.5 Test summary report

The test results are tabulated in Table 6.3. The pass result from test case 10 indicates that the structured repository allows the queries to be performed to retrieve information. The pass result from test case 3 indicates that the common file system stores batch file in the folder structure. The pass result from test case 8 indicates that the wrapper generator Java application successfully generates the batch file with minimal user interaction and adheres to the prescribed syntax. The pass result from test case 11 indicates that the wrapper generator Java application generates batch file with the updated version of the analysis software version.

Table 6. 3: Test 2 result summary

Test cases	Result
Test case 10	Pass
Test case 3	Pass
Test case 8	Pass
Test case 11	Pass

The features/items of the proposed configuration meet the task specific requirements while updating the batch file and adhere to the prescribed syntax. Thus the wrapper generator creates file batch files with good quality and with the updated version of the analysis software version.

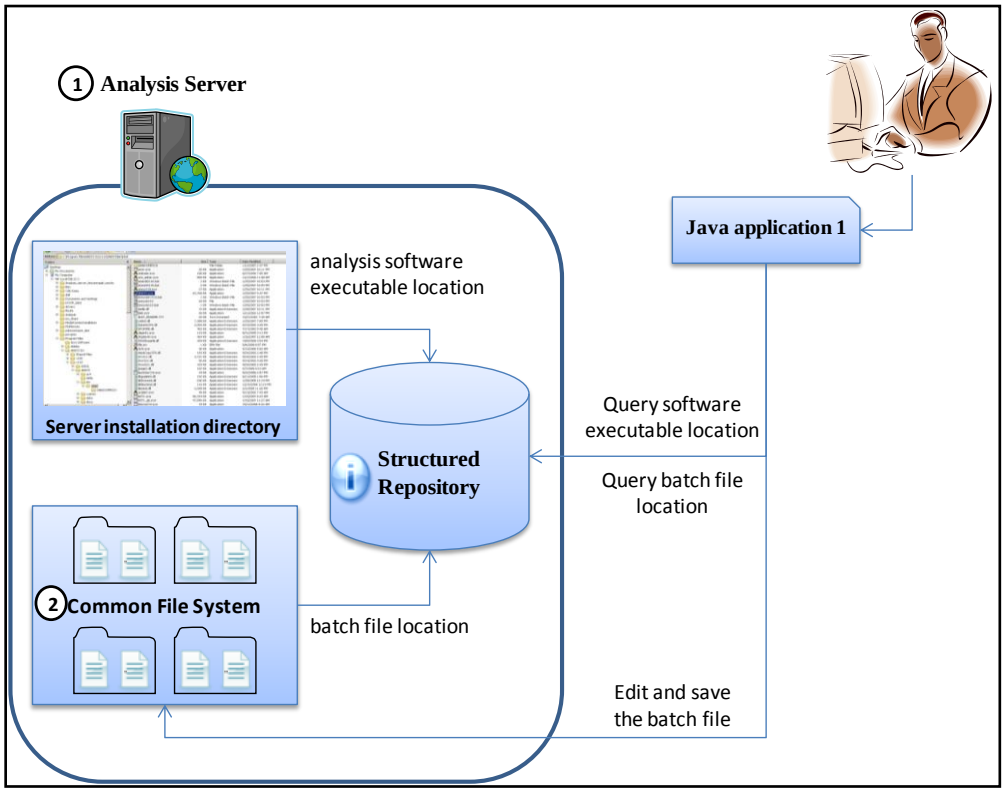


Figure 6.2: Steps involved in the update of batch file

With the help of the information from the structured repository and the wrapper generator updating the batch file creation is reduced to 1 step process (see Figure 6.2) where the user inputs the analysis model name for which the latest version of batch file needs to be generated. This demonstrated the utilization the structured repository and the wrapper generator for better performance while updating the batch file thereby reducing errors during manual generation.

6.4 TEST 3: Reconfiguration of MDO problem

In this test a scenario is presented where the design changes from analysis of a rectangular beam to analysis of an I-beam. The overall objective of this test is to evaluate the performance reconfiguring an existing problem under the test scenario. This test also demonstrates the reconfiguration capability of the proposed configuration.

6.4.1 Test plan

Introduction

In this test structured repository and reconfiguration applications are tested. Structured repository is tested for retrieval of project and analysis model information to provide users the details of the project file. Common file system is tested for storage of these files and Wrapper generator application is tested for automation of file wrapper and batch file creation.

Scenario

In the walkway beam structure analysis there is a requirement to change the analysis from rectangular beam to an I-beam. This required the use of I-beam analysis model instead of a rectangular beam analysis model. The mathematical formulation remains the same hence the existing walkway beam structure project should be reconfigured.

Scenario specific requirements

1. Should use an existing I-beam analysis model
2. Project file should be stored in the respective folder structure

Test items

The item or components of the proposed configuration tested are

- Reconfiguration application
- Common file system
- Structured repository

Functions to be tested

- Retrieve information
- Reuse MDO component
- Reconfigure MDO project
- Store the project file in the project folder

Approach

- Test case 12 is executed to test the retrieval function of the repository
- Test case 13 is executed to test the storage of MDO projects
- Test case 14 is executed to test reconfiguration application

(Refer Section 6.4.2 Test case specification for more details)

Item pass/fail criteria

PASS: When actual output agrees with the expected output

FAIL: When actual output does not agrees with the expected output

Testing tasks

Task 1: Execute test case 12

- Check the queried output for project file and available analysis model information

Task 2: Execute test case 13

- Check the storage of the project files in the projects folder of the common file system

Task 3: Execute test case 14

- Check the reconfiguration capability of reconfiguration application

Environmental needs

Before the execution of the test some test preparation steps have to be carried out. The software tools required to run the test are ModelCenter 7.1, Analysis Server 5.1 and NetBeans5.5.1 IDE. Mentioned below are the steps involved to set up a test environment

1. Load reconfiguration application into the NetBeans IDE
2. Provide print statements in the application to view queried results in the output window of NetBeans IDE
3. Log into Analysis Server 5.1to access common file system under the path

   "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ModelCenterProjects/"

6.4.2 Test case specification

Purpose

Details of each of the test cases are specified. The test cases test certain functions of structured repository, common file system and Java application. This section is referred by section Test plan document for the details of the test cases. These test cases are executed sequentially by the tasks.

TEST CASE 12	
Test items	Structured repository
Function tested	Retrieve information
Inputs	"walkway_structure_analysis_rectangular_beam" into reconfiguration application
Outputs	Queried output from ProjectInfo class of the application o Project details o Problem formulation details o MDO component Details
pass/fail criteria	• **Pass**: If the actual queried output displayed in the output window matches with the expected output • **Fail:** If the actual queried output displayed in the output window does not matches with the expected output

TEST CASE 13	
Test items	Reconfiguration application and common file system
Function tested	Store project files
Inputs	walkway_structure_analysis_I_beam into reconfiguration application
Outputs	walkway_structure_analysis_I_beam.pxc saved in ModelCenter projects folder
pass/fail criteria	• **Pass**: If the project file exists in the ModelCenter projects folder • **Fail:** If the project file does not exist in the ModelCenter projects folder

TEST CASE 14	
Test items	Reconfiguration application
Function tested	Reconfigure MDO problem
Inputs	"Analysis_of_I_beam" into reconfiguration application
Outputs	"Analysis_of_I_beam" model used in the walkway_structure_analysis_I_beam project
pass/fail criteria	• **Pass:** If the Analysis_of_I_beam is used in the project • **Fail:** If the Analysis_of_I_beam is not used in the project

6.4.3 Test procedure specification

This section provides details of the steps required to execute the task specific test cases. This procedure details is referred by testing task section in the test plan. Each task

is executed sequentially as described to ensure the prerequisite for a particular task execution is met in the previous task.

Test procedure for task 1

Purpose

This procedure describes the steps required for the execution of test case 12 to check for the retrieval of project information and analysis model information

Procedure steps

1. Run reconfiguration application
2. Enter the project name as "walkway_structure_analysis_rectangular_beam" when prompted by the reconfiguration application
3. View the queried output from reconfiguration application
4. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 12
5. Log the result along with the pass/fail criteria in the Test log document

Test procedure for task 2

Purpose

This procedure describes the steps required for the execution of test case 13 to check if the new reconfigured project file is stored in the ModelCenter projects folder.

Procedure steps

1. Run reconfiguration application
2. Enter the project name as "walkway_structure_analysis_I_beam" when prompted by the reconfiguration application

3. View the project file in the path

   "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ModelCenterProjects/"
4. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 13
5. Log the result along with the pass/fail criteria in the Test log document

Test procedure for task 3

Purpose

This procedure describes the steps required for the execution of test case 14 to check if the new reconfigured project file uses the "Analysis_of_I_beam "analysis model.

Procedure steps

1. Ensure task 1 and task 2 are executed
2. Open the project file walkway_structure_analysis_I_beam.pxc in the path

   "C: /Program Files/Phoenix Integration/Analysis Server 5.1/analyses/commonfilesystem/ModelCenterProjects/"
3. View the project file in ModelCenter and check if "Analysis_of_I_beam" is used
4. State weather the test is a pass or fail based on the criteria specified in the test case specification of test case 14
5. Log the result along with the pass/fail criteria in the Test log document

6.4.4 Test log

Description

The items tested in this test are reconfiguration application, the structured repository and the common file system. For each task the expected procedure result is described and the pass and fail decision is based on the criteria mentioned in their respective test case specification.

TASK 1 Execution

Expected procedure results: Test case 12

- Project details
- Problem formulation details
- MDO component Details

Actual procedure results

```
_____Project Details_____
Project name                :: walkway_structure_analysis_rectangular_beam
Project file name           :: walkway_structure_analysis_rectangular_beam.pxc
Project file location       :: C:\Program Files\Phoenix Integration\Analysis Server
                               5.1\analyses\commonfilesystem\ModelCenterProjects
-------------------------------0--------------------------
_____Problem Formulation Details_____
Problem formulation used    :: walkway_structure_analysis
Objective function          :: Z=Length*Area*Density
Design Variables            :: Width,Height,LinkRadius
Constraints                 :: Stess constraint
----------------------0----------------------
_____MDO component Details_____
Optimizer used              :: GradientOptimizer
Analysis Model used         :: Analysis_of_rectangular_beam
Wrapper name                :: Analysis_of_rectangular_beam.fileWrapper
Batch file name             :: Analysis_of_rectangular_beam.bat
Description                 :: This is a file wrapper component which wraps the analysis
                               model Analysis_of_rectangular_beam. txt
-----------------0-----------------
```

Pass / Fail decision

- *Test case 12: Pass*

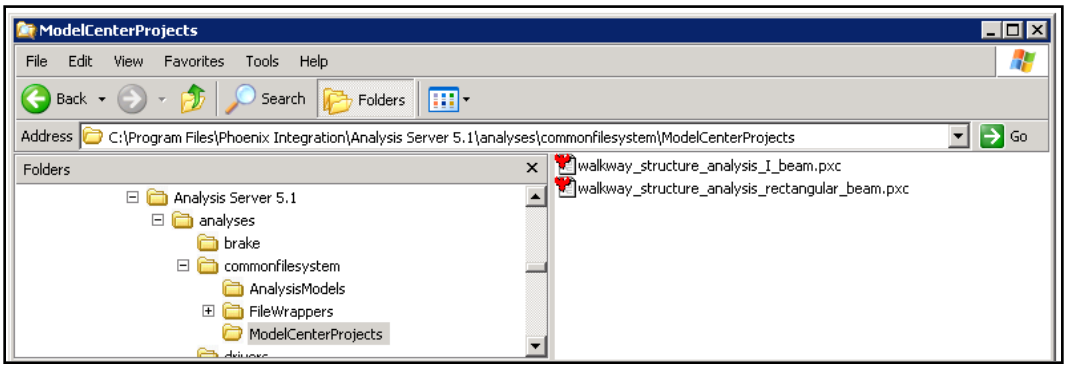
  The actual queried output displayed in the output window matches with the expected output

TASK 2 Execution

Expected procedure results: Test case 13

- "walkway_structure_analysis_I_beam.pxc" is saved in ModelCenter projects folder

Actual procedure results



Pass / Fail decision

- *Test case13 : Pass*

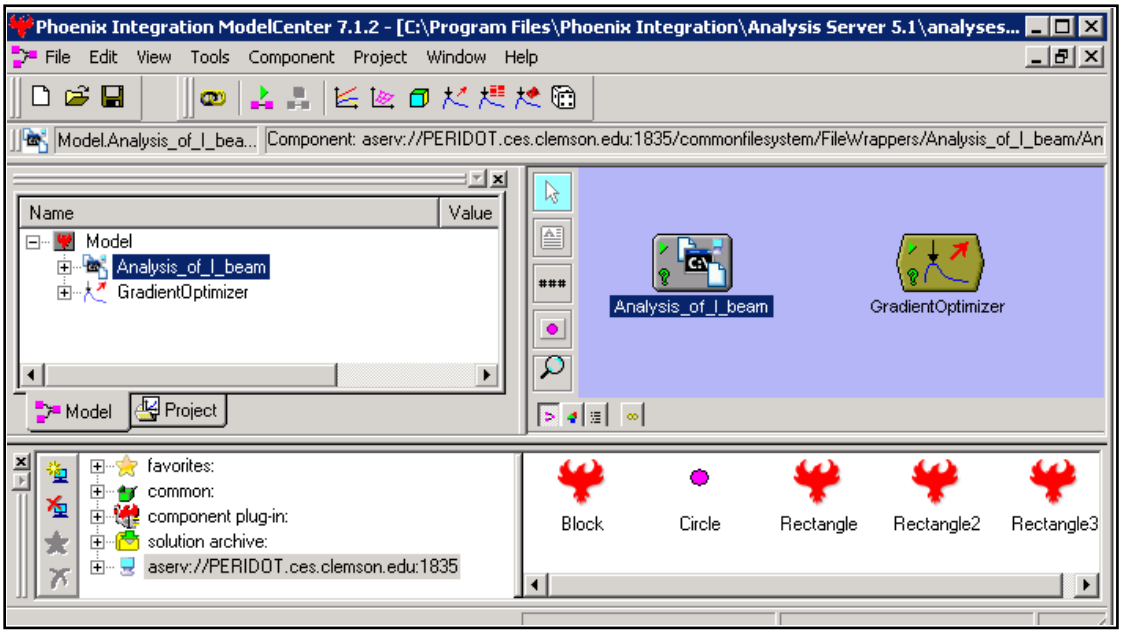
  The project file exists in the ModelCenter projects folder and the Analysis_of_I_beam is used in the project

TASK 3 Execution

Expected procedure results: Test case 14

- “Analysis_of_I_beam” model used in the walkway_structure_analysis_I_beam project

Actual procedure results



Pass / Fail decision

- *Test case14 : Pass*

  The project file exists in the ModelCenter projects folder and the Analysis_of_I_beam is used in the project

6.4.5 Test summary report

The test results are tabulated in Table 6.4. The pass result from test case 12 indicates that the structured repository allows the queries to be performed to retrieve information.

Table 6.4: Test 3 result summary

Test cases	Result
Test case 12	Pass
Test case 13	Pass
Test case 14	Pass

The pass result from test case 13 indicates that the common file system stores project file in the folder structure. The pass result from test case 14 indicates that the reconfiguration application successfully reconfigures the project file

CHAPTER SEVEN

CONCLUSION

From the testing, it is observed that storing the MDO related information in a structured repository benefits the designers with the prior content information required to reuse and reconfigure the existing MDO problems. The stored information can be efficiently utilized to automate the manual and repeatable processes such as file wrapper and batch file creation, thus reducing time taken in MDO problem set up in the MDO frameworks. The common file system enables the MDO components and projects to be stored in a structure and shared. The file information system which is the subset of the structured repository stores the information about the location of the file in the common file system. This information enhances the intelligent search and retrieval of the components in the MDO frameworks.

		Information management requirements				
		Database Management		Modularity	Reconfiguration	Intelligent search and retrieval
		Provide MDO information	Utilize MDO information	Reusable components	Incorporate changes	
Functions offered by proposed configuration	Automate batch file creation	✓	✓			
	Automate file wrapper creation	✓	✓			
	Update information		✓			✓
	Retrieve information		✓		✓	✓
	Reconfigure MDO problem			✓	✓	
	Efficient search of files			✓		✓
	Reuse MDO components		✓	✓		✓
	Store files in a structure	✓	✓		✓	✓

Figure 7.1: Functions addressing information management requirements

The functions offered by the features of the proposed configuration meet the extended information management requirements as shown in Figure 7.1. It can be seen that automating the file wrapper and batch file creation address the database management requirement. Storing the files in a structure facilitates reuse along with the information provided by the repository. The reconfiguration application utilizes the information from the repository and enhances the reconfiguration capability of the MDO framework. Thus the structured repository, common file system and the software applications enhance the information management capability of ModelCenter framework as seen in Figure 7.1.

Table 7.1: ModelCenter meets requirement 1

Requirements	Model Center
(1) Database Management	●
(2) Modularity in problem formulation	●
(3) Reconfiguration capability	●
(4) Intelligent search and retrieval	●
Fully met - ●	

7.1 Addressing Research Questions

The Research Questions formulated in Chapter 1 have been addressed at different stages in the research and in different chapters of this thesis.

Research Question 1: What are the information management requirements of MDO framework to support reuse and reconfiguration?

This Research Question focused on the identification of requirements for managing MDO related information is addressed in Chapter three. The Research Question is addressed first by identifying the general requirements for MDO framework development, correlating these requirements with current MDO frameworks to identify

the gaps in development. The gaps include retrieval and reconfiguration of existing MDO problems; capture and storage of information for the integration of disciplinary analysis models; representing constraints and requirements in formulating MDO problems. Based on these gaps the information management requirements such as database management, modularity, reconfiguration capability and intelligent search and retrieval are extended. These requirements also help in emphasizing the reuse and reconfiguration in MDO frameworks.

Research Question 2: What are the features that need to be integrated into the MDO framework to enhance information management capabilities?

This Research Question focused on developing features to enhance information management capabilities in MDO framework is addressed in Chapter three. ModelCenter, modeFRONTIER, and iSIGHT FD frameworks are evaluated against the information management requirements extended while addressing Research Question 1. Based on the evaluation ModelCenter framework is selected as the suitable framework for extending the information management capabilities. The drawbacks of the ModelCenter framework configuration are identified and structured repository and common file system are the features proposed to help address these drawbacks and enhance the information management capabilities

Research Question 3: What is the structure of the information model to enable efficient reuse and reconfiguration in MDO problems?

This Research Question focused on providing the information model is addressed in Chapter four. The information model of the repository stores MDO information and

allows the user to update and retrieve the information. The information stored provides the designer the prior knowledge and facilitates for reuse and reconfiguration.

Research Question 4: How will the repository be interfaced/ integrated with an MDO framework in general and ModelCenter/Analysis server specifically?

This Research Question focused on the integrating proposed structured repository and the common file system in ModelCenter/Analysis Server configuration is addressed in Chapter four. Java interface provides the methods to access the repository and the common file system. Structured repository is accessed using MYSQL-JDBC driver and the common file system is accessed using the file system URL.

7.2 Future work

The immediate future work is to refine and extend the structure of the information model to capture information related to MDO formulations and to enable content based search. The information model currently stores information related to MDO project files. The structure of the information model enables the information to be searched and queried based on the name of the project files and name of the analysis models.

The compatibility of the information model with other MDO frameworks was not considered in this research. Framework specific versions of the repository need to be developed such that the repository can be incorporated into other MDO frameworks.

The reconfiguration applications developed in this research reconfigures the project file by replacing the analysis model. The linking between the variables of the analysis models and the optimizer is not achieved with this application. Therefore the Java API's to link the variables need to used in the application.

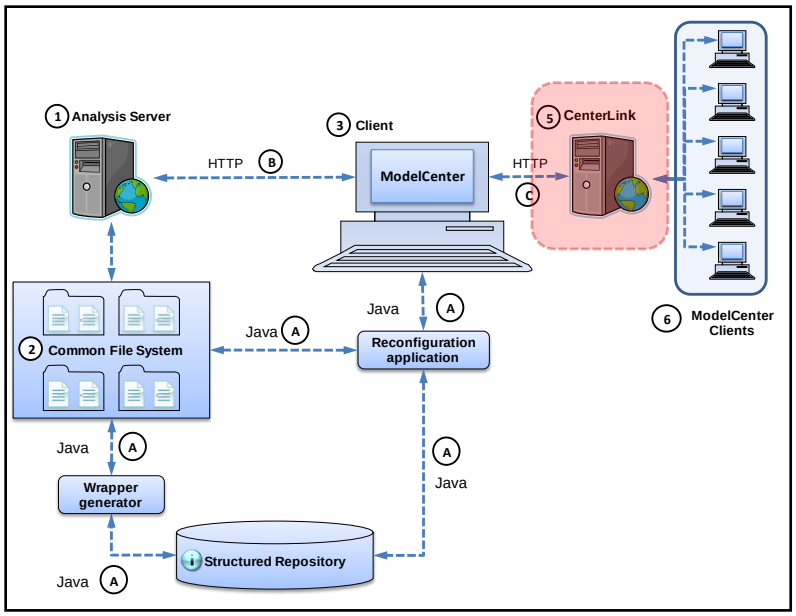


Figure 7.2: CenterLink in ModelCenter configuration

The proposed structured repository can be extended to include information model to store and manage information in the grid computing module called CenterLink available in the current ModelCenter configuration (see Figure 7.2). The Analysis library in CenterLink can be coupled with a repository to enable intelligent search of analysis models uploaded into the library.

APPENDICES

Appendix A

APDL code for analysis of a rectangular beam

```
/COM,Preferences for GUI filtering
have been set to display:
/COM,  Structural
/PREP7
!Input Variables
beam_length = 1.0
beam_width_1 = 1.0
beam_width_2 = 1.0
beam_width_3 = 1.0
beam_height_1 = 1.0
beam_height_2 = 1.0
beam_height_3 = 1.0
link_radius_1 = 0.1
link_radius_2 = 0.1

beam_area_1 =
beam_width_1*beam_height_1
beam_area_2 =
beam_width_2*beam_height_2
beam_area_3 =
beam_width_3*beam_height_3

beam_moa_1 =
(1/12)*beam_width_1*beam_height_1*
*3
beam_moa_2 =
(1/12)*beam_width_2*beam_height_2*
*3
beam_moa_3 =
(1/12)*beam_width_3*beam_height_3*
*3

link_area_1 =
3.14159*link_radius_1**2
link_area_2 =
3.14159*link_radius_2**2

mat_density = 7826

!define Key Points
k,1,0,0,0
k,2,0,5,0
k,3,0,10,0
k,4,beam_length,0,0
k,5,beam_length,5,0
k,6,beam_length,10,0

!Defines a line between two keypoints
l,1,4
l,2,5
l,3,6
l,4,5
l,5,6

mptemp, , , , , , , ,
mptemp,1,0
mpdata,ex,1, ,200e9
mpdata,prxy,1,,0.33
mpdata,dens,1,,mat_density

!define element types
ET,1,BEAM3
ET,2,LINK1

!define real constants for beam and link
R,1,beam_area_1,beam_moa_1,beam_he
ight_1
R,2,beam_area_2,beam_moa_2,beam_he
ight_2
R,3,beam_area_3,beam_moa_3,beam_he
ight_3
R,4,link_area_1,0
R,5,link_area_2,0
```

```
CMSEL,S,,LINE,1
LATT,1,1,1, , , ,
lmesh,1
cmsel,all

CMSEL,S,,LINE,2
LATT,1,2,1, , , ,
lmesh,2
cmsel,all

CMSEL,S,,LINE,3
LATT,1,3,1, , , ,
lmesh,3
cmsel,all

CMSEL,S,,LINE,4
LATT,1,4,2, , , ,
lmesh,4
cmsel,all

CMSEL,S,,LINE,5
LATT,1,5,2, , , ,
lmesh,5
cmsel,all

FINISH
/SOLU

cmsel,all
cmsel,s,,KP,2
fk,4,FY,-100

cmsel,all
dk,1,all,0
dk,2,all,0
dk,3,all,0

SOLVE  !solve the system
FINISH !finish the solution for post
processing

/POST1
/shrink,0
/eshape,1.0
/efacet,1
/ratio,1,1,1
/cformat32,0
/replot

lsel,s,,,1
nsll,r,1
nsort,u,y,0,1
*GET, defymax1,sort,0,max

ALLSEL, ALL
lsel,all
lsel,s,,,2
nsll,r,1
nsort,u,y,0,1
*GET, defymax2,sort,0,max

ALLSEL,ALL
lsel,all
lsel,s,,,3
nsll,r,1
nsort,u,y,0,1
*GET, defymax3,sort,0,max

ALLSEL,ALL
lsel,all
lsel,s,,,4
nsll,r,1
nsort,u,y,0,1
*GET, defymax4,sort,0,max

ALLSEL,ALL
lsel,all
lsel,s,,,5
nsll,r,1
nsort,u,y,0,1
*GET, defymax5,sort,0,max

ALLSEL,ALL
lsel,s,,,5
esll,r
ETABLE,SAXL,LS, 1
! Axial Stress
ESORT,ETAB,SAXL,0,0,,
```

```
*GET,smax5,sort,,max

ALLSEL,ALL
lsel,s,,,4
esll,r
ETABLE,SAXL,LS, 1
 ! Axial Stress
ESORT,ETAB,SAXL,0,0,,
*GET,smax4,sort,,max

ALLSEL,ALL
lsel,s,,,1
esll,r
ETABLE,SMAX, NMISC, 1
 ! Axial Stress
ESORT,ETAB,SMAX,0,0,,
*GET,smax1,sort,,max

ALLSEL,ALL

lsel,s,,,2
esll,r
ETABLE,SMAX, NMISC, 1
 ! Axial Stress
ESORT,ETAB,SMAX,0,0,,
*GET,smax2,sort,,max

ALLSEL,ALL
lsel,s,,,3
esll,r
ETABLE,SMAX, NMISC, 1
 ! Axial Stress
ESORT,ETAB,SMAX,0,0,,
*GET,smax3,sort,,max

ALLSEL,ALL
```

```
*CFOPEN,'BeamAnalysisOutput',txt,,
*VWRITE,defymax1
('BeamDeflection1 =',f20.10)

*VWRITE,defymax2
('BeamDeflection2 =',f20.10)

*VWRITE,defymax3
('BeamDeflection3 =',f20.10)

*VWRITE,defymax4
('BeamDeflection4 =',f20.10)

*VWRITE,defymax5
('BeamDeflection5 =',f20.10)

*VWRITE,smax1
('BeamStress1 =',f20.10)

*VWRITE,smax2
('BeamStress2 =',f20.10)

*VWRITE,smax3
('BeamStress3 =',f20.10)

*VWRITE,smax4
('BeamStress4 =',f20.10)

*VWRITE,smax5
('BeamStress5 =',f20.10)

*VWRITE,(beam_area_1+beam_area_2
+beam_area_3)*beam_length*0.1+(link
_area_1+link_area_2)*5*0.1
('Mass =',G17.11)
*CFCLOS
```

APDL code for analysis of a I beam

```
/COM,
/COM,Preferences for GUI filtering
have been set to display:
/COM,  Structural
!*
/REPLOT,RESIZE
/PREP7
!Input Variables
beam_length = 18
segment_length = beam_length/3
beam_width_1 = 6
beam_width_2 = 6
beam_width_3 = 6
beam_height_1 = 4
beam_height_2 = 4
beam_height_3 = 4
beam_webheight_1 = 2
beam_webheight_2 = 2
beam_webheight_3 = 2
beam_webwidth_1 = 3
beam_webwidth_2 = 3
beam_webwidth_3 = 3
link_radius_1 = 0.1
link_radius_2 = 0.1

beam_area_1 =
(beam_width_1*((2*beam_height_1)+be
am_webheight_1))-
(beam_webheight_1*(beam_width_1-
beam_webwidth_1))
beam_area_2 =
(beam_width_2*((2*beam_height_2)+be
am_webheight_2))-
(beam_webheight_2*(beam_width_2-
beam_webwidth_2))
beam_area_3 =
(beam_width_3*((2*beam_height_3)+be
am_webheight_3))-
(beam_webheight_3*(beam_width_3-
beam_webwidth_3))
link_area_1 =
3.14159*link_radius_1**2
link_area_2 =
3.14159*link_radius_2**2

beam_moa_1 =
2*(beam_height_1*beam_width_1)*((be
am_webheight_1/2)+(beam_height_1/2)
)**2+(beam_webwidth_1*beam_webhei
ght_1**3)/12
beam_moa_2 =
2*(beam_height_2*beam_width_2)*((be
am_webheight_2/2)+(beam_height_2/2)
)**2+(beam_webwidth_2*beam_webhei
ght_2**3)/12
beam_moa_3 =
2*(beam_height_3*beam_width_3)*((be
am_webheight_3/2)+(beam_height_3/2)
)**2+(beam_webwidth_3*beam_webhei
ght_3**3)/12

total_beam_height_1=
(2*beam_height_1)+beam_webheight_1
total_beam_height_2=
(2*beam_height_2)+beam_webheight_2
total_beam_height_3=
(2*beam_height_3)+beam_webheight_3

mat_density = 7826

!define element types
ET,1,BEAM3
ET,2,LINK1

!* Section Type Information
SECTYPE,   1, BEAM, I, , 0
SECOFFSET, CENT
SECDATA,beam_width_1,beam_width_
1,total_beam_height_1,beam_height_1,b
eam_height_1,beam_webheight_1,0,0,0,
0

SECTYPE,   2, BEAM, I, , 0
SECOFFSET, CENT
```

```
SECDATA,beam_width_2,beam_width_
2,total_beam_height_2,beam_height_2,b
eam_height_2,beam_webheight_2,0,0,0,
0

SECTYPE,   3, BEAM, I, , 0
SECOFFSET, CENT
SECDATA,beam_width_3,beam_width_
3,total_beam_height_3,beam_height_3,b
eam_height_3,beam_webheight_3,0,0,0,
0

!* Define Material Properties
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,200e9
! Young's modulus for material
MPDATA,PRXY,1,,0.33
! Poisson's ratios  for material
MPDATA,dens,1,,mat_density
! Density for material

!define real constants for beam
R,1,beam_area_1,beam_moa_1,beam_he
ight_1, , , ,
R,2,beam_area_2,beam_moa_2,beam_he
ight_2, , , ,
R,3,beam_area_3,beam_moa_3,beam_he
ight_3, , , ,

!define real constants for link
R,4,link_area_1,0,
R,5,link_area_2,0,

!define Key points
K,1,0,0,0,
K,2,segment_length,0,0,
K,3,2*segment_length,0,0,
K,4,3*segment_length,0,0,
K,5,0,6,0,
K,6,segment_length,6,0,
K,7,2*segment_length,6,0,
K,8,3*segment_length,6,0,
K,9,0,12,0,
K,10,segment_length,12,0,
K,11,2*segment_length,12,0,
K,12,3*segment_length,12,0,

K,13,3*segment_length,0,0,
K,14,3*segment_length,6,0,
K,15,3*segment_length,12,0,

!define line between two key points
L,      1,      4
L,      5,      8
L,      9,     12
L,     13,     14
L,     14,     15

!MESHING

CMSEL,S,,LINE,1
LATT,1,1,1, , , ,1
lmesh,1
cmsel,all

CMSEL,S,,LINE,2
LATT,1,2,1, , , ,2
lmesh,2
cmsel,all

CMSEL,S,,LINE,3
LATT,1,3,1, , , ,3
lmesh,3
cmsel,all

CMSEL,S,,LINE,4
LATT,1,4,2, , , ,
lmesh,4
cmsel,all

CMSEL,S,,LINE,5
LATT,1,5,2, , , ,
lmesh,5
cmsel,all

!Defines coupled degrees of freedom at
an interface
```

```
CPINTF,ux,1e-4
CPINTF,uy,1e-4

FINISH
/SOLU

! Apply Loads
CMSEL,all
CMSEL,S,,KP,2
!F,3,FY,-833
FK,4,FY,-8833
!F,7,FY,-833
!F,8,FY,-833
!F,11,FY,-833
!F,12,FY,-833

!give displacements
cmsel,all
DK,1, , , ,0,ALL, , , , , ,
DK,5, , , ,0,ALL, , , , , ,
DK,9, , , ,0,ALL, , , , , ,

!DK,1,ALL,0
!DK,5,ALL,0
!DK,9,ALL,0

SOLVE
FINISH

/POST1
/shrink,0
/eshape,1.0
/efacet,1
/ratio,1,1,1
/cformat32,0
/replot

!ALLSEL, ALL
lsel,s,,,1
nsll,r,1
nsort,U,Y,0,1
*GET, defymax1,sort,0,max

ALLSEL, ALL
lsel,all
lsel,s,,,2
nsll,r,1
nsort,u,y,0,1
*GET, defymax2,sort,0,max

ALLSEL,ALL
lsel,all
lsel,s,,,3
nsll,r,1
nsort,u,y,0,1
*GET, defymax3,sort,0,max

ALLSEL,ALL
lsel,all
lsel,s,,,4
nsll,r,1
nsort,u,y,0,1
*GET, defymax4,sort,0,max

ALLSEL,ALL
lsel,all
lsel,s,,,5
nsll,r,1
nsort,u,y,0,1
*GET, defymax5,sort,0,max

ALLSEL,ALL
lsel,s,,,5
esll,r
ETABLE,SAXL,LS, 1        ! Axial
Stress
ESORT,ETAB,SAXL,0,0,,
*GET,smax5,sort,,max

ALLSEL,ALL
lsel,s,,,4
esll,r
ETABLE,SAXL,LS, 1        ! Axial
Stress
ESORT,ETAB,SAXL,0,0,,
*GET,smax4,sort,,max

ALLSEL,ALL
```

```
lsel,s,,,1
esll,r
ETABLE,SMAX, NMISC, 1        !
Axial Stress
ESORT,ETAB,SMAX,0,0,,
*GET,smax1,sort,,max

ALLSEL,ALL
lsel,s,,,2
esll,r
ETABLE,SMAX, NMISC, 1        !
Axial Stress
ESORT,ETAB,SMAX,0,0,,
*GET,smax2,sort,,max

ALLSEL,ALL
lsel,s,,,3
esll,r
ETABLE,SMAX, NMISC, 1        !
Axial Stress
ESORT,ETAB,SMAX,0,0,,
*GET,smax3,sort,,max

*CFOPEN,'Beam_Analysis_Output.txt',t
xt,,
*VWRITE, defymax1
('BeamDeflection1 =',f20.10)

*VWRITE,defymax2
('BeamDeflection2 =',f20.10)

*VWRITE,defymax3
('BeamDeflection3 =',f20.10)

*VWRITE,defymax4
('BeamDeflection4 =',f20.10)

*VWRITE,defymax5
('BeamDeflection5 =',f20.10)

*VWRITE,smax1
('BeamStress1 =',f20.10)

*VWRITE,smax2
('BeamStress2 =',f20.10)

*VWRITE,smax3
('BeamStress3 =',f20.10)

*VWRITE,smax4
('BeamStress4 =',f20.10)

*VWRITE,smax5
('BeamStress5 =',f20.10)

*VWRITE,(beam_area_1+beam_area_2
+beam_area_3)*beam_length*0.1+(link
_area_1+link_area_2)*5*0.1
('Mass =',G17.11)
*CFCLOS
```

Appendix B

Wrapper Generator Application

```
//Main class
package wrapperapplication;
import java.io.*;
import java.sql.*;
import javax.sql.*;
import java.util.*;
import java.lang.*;
import javax.swing.table.*;
import javax.swing.JOptionPane;

public class Main {

   public Main() {
   }
   public static void main(String[] args) {

    try {

//FOR USER INPUT  ----------------------------------------------------------------*/
   String getter =JOptionPane.showInputDialog("Enter analysis model name:");

//Call Filewrapper generator  ----------------------------------------------------*/
    FileWrapperGenerator newobject1 = new FileWrapperGenerator();
    String FileWrapperresult=  newobject1.main(getter);
    System.out.print("FILEWRAPPER RESULT
"+FileWrapperresult+System.getProperty("line.separator"));

//Call Filewrapper generator  ----------------------------------------------------*/
    BatchFileGenerator newobject2 = new BatchFileGenerator();
    String batchfileresult=  newobject2.main(getter);
    System.out.print("BATCHFILE RESULT
"+batchfileresult+System.getProperty("line.separator"));
    }catch (Exception e){
    }
   }
}

//FileWrapperGenerator Class
package wrapperapplication;
import java.io.*;
```

```
import java.sql.*;
import wrapperapplication.*;
import javax.sql.*;
import java.util.*;
import java.lang.*;
import javax.swing.table.*;

public class FileWrapperGenerator {
       public FileWrapperGenerator() {
   }
    public static String main(String getter)
{
      String          FileWrapperresult = null;
      String          S;
      Statement       stmt = null;
      Statement       stmt1 = null;
      Statement       stmt2 = null;
      Statement       stmt3 = null;
      Statement       stmt4 = null;
      Statement       stmt5 = null;
      Statement       stmt6 = null;
      Statement       stmt7 = null;
      Statement       stmt8 = null;
      Statement       stmt9 = null;
      Statement       stmt10 = null;
      ResultSet       res = null;
      ResultSet       res1 = null;
      ResultSet       res2 = null;
      ResultSet       res3 = null;
      ResultSet       res4 = null;
      ResultSet       res5 = null;
      ResultSet       res6 = null;
      ResultSet       res7 = null;
      ResultSet       res8 = null;
      ResultSet       res9 = null;
      ResultSet       res10 = null;
      String          query = null;
      String          query1 = null;
      String          query2 = null;
      String          query3 = null;
      String          query4 = null;
      String          query5 = null;
      String          query6 = null;
      String          query7 = null;
```

```
        String          query8 = null;
        String          query9 = null;
        String          query10 = null;
        String          resultvalue = null;
        String          Variable = null;
        String          DataType = null ;
        String          string = null;
        String          string2 = null;
        String          n =System.getProperty("line.separator");
        int i=0;
        int j=2;

ConnectToDatabase connect1 = new ConnectToDatabase();
Connection connect=  connect1.main();

//QUERYING WITH SQL TO READ INFORMATION IN THE DATABASE

try {
            stmt =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
            stmt1 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
            stmt2 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
            stmt3 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
            stmt4 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
            stmt5 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
            stmt6 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
            stmt7 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
```

```
        stmt8 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
        stmt9 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
        stmt10 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);

          query="SELECT Wrapper_file_name FROM wrapper where
          Analysis_model_name=\""+getter+"\";";
          res = stmt.executeQuery(query);
          res.next();
          int row= res.getFetchSize();
          System.out.print("Entry in the database"+row+n+n);

       if (row != 0)//If there is an entry in the database
            {
              System.out.print("filewrapper exists ");
              FileWrapperresult = ("filewrapper exists");
              return FileWrapperresult;
            }
       else{

try{
              query1 = "SELECT Variable_name,Data_type FROM variables WHERE
Analysis_model_name=\"" +getter+"\" and Variable_type=\"Input\" ;";
              res1 = stmt1.executeQuery(query1);

              query2 = "SELECT Variable_name,Data_type FROM variables WHERE
Analysis_model_name=\"" +getter+"\" and Variable_type=\"Output\" ;" ;
              res2 = stmt2.executeQuery(query2);

              query3 = "SELECT COUNT(*) AS rowcount FROM variables WHERE
Analysis_model_name=\"" +getter+"\" and Variable_type=\"Input\" ;";
              res3 = stmt3.executeQuery(query3);
              res3.next();
              int iCount = res3.getInt("rowcount");
 System.out.println("Number of input variables =  " + iCount+n);//FOR TESTING
              res3.close();

              query4 = "SELECT COUNT(*) AS rowcount FROM variables WHERE
Analysis_model_name=\"" +getter+"\" and Variable_type=\"Output\" ;";
```

```
            res4 = stmt4.executeQuery(query4);
            res4.next();
            int iCount2 = res4.getInt("rowcount");
System.out.println("Number of output variables =  " + iCount2+n);//FOR TESTING
            res4.close();

            query5 = "SELECT Output_file FROM analysismodel WHERE
Analysis_model_name=\"" +getter+"\" ;";
            res5 = stmt5.executeQuery(query5);
            res5.next();
System.out.println(n+n+"Analysis model generated output file name =  " +
res5.getString(1)+n+n);//FOR TESTING

            query6 = "SELECT AFile_name FROM analysismodel WHERE
Analysis_model_name= \"" +getter+"\";";
            res6 = stmt6.executeQuery(query6);
            res6.beforeFirst();
            res6.next();
System.out.println(n+n+"Analysis model file name =  " + res6.getString(1)+n+n);//FOR
TESTING

            query7 = "SELECT Software_name FROM analysismodel where
Analysis_model_name=\""+getter+"\";";
            res7 = stmt7.executeQuery(query7);
            res7.next();
            String softwarename= res7.getString(1);
System.out.print(n+n+"Analysis Software name =  "+softwarename+n+n);//FOR
TESTING

            query8="SELECT Executable_URL FROM software where
software_name=\""+softwarename+"\";";
            res8 = stmt8.executeQuery(query8);

            res1.beforeFirst();
          } catch (SQLException sqe2){
            System.out.println("Caught SQL Exception: " + sqe2);
          } catch (Exception e){
            System.err.println ("Error writing to file");
          }
          try{
// CREATE A FOLDER AND THE FILE WRAPPER THE COMMON FILE SYSTEM
            String FilewrapperName =getter+".fileWrapper";
            String BatchFileName =getter+".bat";
            String fileGenerateName = (getter+".in");
```

```
            StringBuffer parentDirPath = new
StringBuffer("//Peridot.ces.clemson.edu/c$/Program Files/Phoenix Integration/Analysis
Server 5.1/analyses/commonfilesystem/FileWrappers/");
            String folder = new String(getter);
            parentDirPath.append(folder);
            File parentDir = new File( parentDirPath.toString());
            parentDir.mkdir();
            String Ssd = parentDirPath.toString();
            System.out.println("Caught SQL Exception: " + Ssd);
            File file = new File(parentDir, FilewrapperName);
            parentDir.mkdirs();
            file.createNewFile();
            BufferedWriter out2 = new BufferedWriter(new PrintWriter(new
FileWriter(file)));

// COPY AND PASTE ANALYSIS FILE FROM ANALYSIS MODEL FOLDER TO
THE NEW FOLDER CREATED

// WRITE THE FILE WRAPPER COMMANDS INTO THE CREATED FILE
WRAPPER FILE
// Header section
            out2.write("# @author: Santosh Hiriyannaiah"+n);
            out2.write("# @version: Trial "+n);
            out2.write("# @description: File Wrapper "+n+n);

// File wrapper Run section
            out2.write("RunCommands "+n+"{"+ n);
            out2.write("generate inputFile"+n+"run \""+BatchFileName+"\""+ n );
            out2.write("parse outputFile"+ n+"}"+ n);

// File wrapper RowFieldInputFile Section
            out2.write("RowFieldInputFile inputFile"+ n+"{"+ n);
            out2.write("templateFile:"+ res6.getString(1)+ n) ;
            out2.write("fileToGenerate:"+fileGenerateName+ n+ n);
            out2.write("markAsBeginning \"Input Variables\""+ n+n);

            while(res1.next())
            {
               System.out.println(n+"Input Variable: " + res1.getString(1) );//FOR
TESTING
               try {
                  out2.write("variable: " + res1.getString(1) + "      "+ res1.getString(2) +"
"+j+"          3  "+System.getProperty("line.separator"));
               }catch (IOException e) {
```

```
                }
                j=j+1;
            }//end res1 while
            out2.write("}"+ n+ n);

//File wrapper RowFieldOutputFile Section
            out2.write("RowFieldOutputFile outputFile"+ n+"{"+ n);
            out2.write("fileToParse:"+res5.getString(1)+ n);
            out2.write("setDelimiters \" =\" "+ n+n);
            while(res2.next())
            {
System.out.println(n+"Output Variable: " + res2.getString(1) );//FOR TESTING
                try {
                    out2.write("keyVar: " + res2.getString(1) + "      "+ res2.getString(2) +"
\""+res2.getString(1)+"\""+ System.getProperty("line.separator"));
                }catch (IOException e) {
                }
            }//end res2 while
            out2.write("}"+ n+ n);
            out2.close();
            FileWrapperresult = ("FileWrapperresult successfully created");
// Update database
            query10=("INSERT INTO wrapper (Wrapper_file_name,
Analysis_model_name, Batch_file_name)
VALUES(\""+FilewrapperName+"\""+","+"\""+getter+"\""+","+"\""+BatchFileName+"\""+
");") ;
            stmt10.executeUpdate(query10);
            System.out.println ("Updated in database");
          } catch (Exception e){
            System.err.println ("Error writing to file");
          }
       }
     } catch (SQLException sqe2){
        System.out.println("Caught SQL Exception: " + sqe2);
     } catch (Exception e){
        System.err.println ("Error writing to file");
     }
      return FileWrapperresult;
   }
}

// BatchFileGenerator
package wrapperapplication;
import java.io.*;
```

```
import java.sql.*;
import javax.sql.*;
import java.util.*;
import java.lang.*;
import javax.swing.table.*;

public class BatchFileGenerator {

   public BatchFileGenerator() {
   }
      public static String main(String getter)   {
      String          batchfileresult = null;
      Statement        stmt1 = null;
      Statement        stmt2 = null;
      Statement        stmt3 = null;
      Statement        stmt4 = null;
      Statement        stmt5 = null;
      ResultSet        res1 = null;
      ResultSet        res2 = null;
      ResultSet        res3 = null;
      ResultSet        res4 = null;
      ResultSet        res5 = null;
      String          query1 = null;
      String          query2 = null;
      String          query3 = null;
      String          query4 = null;
      String          query5 = null;
      String n =System.getProperty("line.separator");

      ConnectToDatabase connect1 = new ConnectToDatabase();
      Connection connect=  connect1.main();

try{
         stmt1 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
         stmt2 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
         stmt3 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
```

```
        stmt4 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);
        stmt5 =
connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_READ_ONLY);

        query1="SELECT Software_name FROM analysismodel where
Analysis_model_name=\""+getter+"\";";
        res1 = stmt1.executeQuery(query1);
        res1.next();
        String softwarename= res1.getString(1);
System.out.print(n+n+"Analysis Software name =  "+softwarename+n+n);//FOR
TESTING
        res1.close();

        query4="SELECT Version FROM software where
software_name=\""+softwarename+"\";";
        res4 = stmt4.executeQuery(query4);
        res4.next();
        int softwareVersion= res4.getInt(1);
        System.out.print("Version of the software used =="+softwareVersion+n);
        res4.last();
        int softwareLatestVersion= res4.getInt(1);
System.out.print("Latest Version of the software used =="+softwareLatestVersion+n);

        String BatchFileName =getter+".bat";
        StringBuffer parentDirPath = new
StringBuffer("//Peridot.ces.clemson.edu/c$/Program Files/Phoenix Integration/Analysis
Server 5.1/analyses/commonfilesystem/FileWrappers/");
        String folder = new String(getter);
        parentDirPath.append(folder);  //creates analysis model name folder in file
wrapper dir
        File parentDir = new File( parentDirPath.toString());
        parentDir.mkdir();
        parentDir.mkdirs();

        File file2 = new File(parentDir, BatchFileName);
        if(file2.exists())
        {
           file2.delete();
        }
        else
        {
```

```
            file2.createNewFile();
         }

         BufferedWriter out3 = new BufferedWriter(new PrintWriter(new
FileWriter(file2)));
         String attach = (getter+".in");
         String softwareLatestVersionstring = Integer.toString(softwareLatestVersion);

         query2="SELECT Executable_URL FROM software where
Software_name=\""+softwarename+"\" AND Version
=\""+softwareLatestVersionstring+"\";";
         res2 = stmt2.executeQuery(query2);
         res2.next();
         String Executable_URL  = res2.getString(1);
         System.out.print(n+n+"Executable_URL =  "+Executable_URL+n+n);//FOR
TESTING
         res2.close();

         query3="SELECT Product_name FROM software where
software_name=\""+softwarename+"\" AND Version
=\""+softwareLatestVersionstring+"\";";
         res3 = stmt3.executeQuery(query3);
         res3.next();
         String ProductName  = res3.getString(1);
System.out.print(n+n+"ProductName =  "+ProductName+n+n+n+n);//FOR TESTING
         res3.close();

         String Dir = ("C:"+"\\"+"Program Files"+"\\"+"Phoenix
Integration"+"\\"+"Analysis Server
5.1"+"\\"+"analyses"+"\\"+"commonfilesystem"+"\\"+"");
         String OutputFile = ("ansys.out");

         out3.write("\""+ Executable_URL+ "\""+ " -b "+"-p "+ProductName+" -i
"+Dir+getter+"\\"+attach+"\""+"-o "+"\""+Dir+getter+"\\"+OutputFile+"\"");
         out3.close();

         batchfileresult = ("batch file successfully created");

         }catch (Exception e){
            System.err.println ("Error ");
         }
      return batchfileresult;
      }
}
```

```
package wrapperapplication;
import java.io.*;
import java.sql.*;
import javax.sql.*;
import java.util.*;
import java.lang.*;
import javax.swing.table.*;

public class ConnectToDatabase {
   public ConnectToDatabase() {
   }    public static Connection main()   {
      Connection        connect = null;
      try{
         Class.forName("com.mysql.jdbc.Driver").newInstance();
         System.out.println("Driver Registration Successful.");
      }
      catch (InstantiationException ie){
         System.out.println("Class Instantiation Exception: " + ie);
      } catch (ClassNotFoundException cnf){
         System.out.println("Class Not Found Exception: " + cnf);
        } catch (IllegalAccessException iae){
           System.out.println("Illegal Access Exception: " + iae);
        }
// Establish connection with the Database
      try {
         connect =
DriverManager.getConnection("jdbc:mysql://peridot.ces.clemson.edu/structured_reposito
ry?user=root&password=peridot");
         System.out.println("Connection to MySQL Database
Successful"+System.getProperty("line.separator"));
      } catch (SQLException sqe1){
         System.out.println("Caught SQL Exception: " + sqe1);
      }
      return connect;
   }
}
```

Appendix C

Reconfiguration Application

```
package reconfigurationapplication;
import java.io.*;
import com.phoenix_int.ModelCenter.* ;
import com.phoenix_int.ModelCenter.util.*;
import com.phoenix_int.aserver.* ;
import java.sql.*;
import javax.sql.*;
import java.util.*;
import java.lang.*;
import javax.swing.table.*;
import javax.swing.JOptionPane;

public class Main {

   public Main() {
   }
   public static void main(String[] args) {

      String n =System.getProperty("line.separator");

try {

String Project_name =JOptionPane.showInputDialog("Enter Project name  :");

//PROVIDE PROJECT FILE INFORMATION
            ProjectInfo info = new ProjectInfo();
            String Formulation_used = info.main(Project_name);

//PROVIDE ANAYSIS AVAILABLE INFORMATION
            AnalysisAvailable models = new AnalysisAvailable();
            models.main();
// ------------------------------------------------------------------------------------

String Analysis_name =JOptionPane.showInputDialog("Enter the Analysis model to
replace existing  :");
String Project_nameAS =JOptionPane.showInputDialog("Project to be saved as :");
Reconfiguration reconfigure = new Reconfiguration();
String ReconfigurationResult= reconfigure.main(Analysis_name, Project_nameAS,
Project_name);
System.out.print("result  : "+ReconfigurationResult);
```

```
            } catch (Exception e){
              System.err.println ("Error writing to file");
        }
    }
}
```

// ProjectInformation Class

```
package reconfigurationapplication;
import java.io.*;
import com.phoenix_int.ModelCenter.* ;
import com.phoenix_int.ModelCenter.util.*;
import java.sql.*;
import javax.sql.*;
import java.util.*;
import java.lang.*;
import javax.swing.table.*;

public class ProjectInfo {
    public ProjectInfo() {
    }
          public static String main(String Project_name) {
          String Formulation_used=null;
          Statement       stmt1 = null;
          Statement       stmt2 = null;
          Statement       stmt3 = null;
          Statement       stmt4 = null;
          Statement       stmt5 = null;
          Statement       stmt6 = null;
          ResultSet       res1 = null;
          ResultSet       res2 = null;
          ResultSet       res3 = null;
          ResultSet       res4 = null;
          ResultSet       res5 = null;
          ResultSet       res6 = null;
          String          query1 = null;
          String          query2 = null;
          String          query3 = null;
          String          query4 = null;
          String          query5 = null;
          String          query6 = null;
          String n =System.getProperty("line.separator");
try
          {
```

```
Connecttodatabase connect1 = new Connecttodatabase();
Connection connect=  connect1.main();

   stmt1 = connect.createStatement();
   query1="SELECT PFile_name,Problem_formulation,
   Analysis_model_name,Optimization_component FROM mdoproject
   where Project_name="+"\""+Project_name+"\";";
   res1 = stmt1.executeQuery(query1);

    res1.next();
   String ProjectFile_name= res1.getString(1);
   String Formulation= res1.getString(2);
   String Analysis_model_name= res1.getString(3);
   String Optimization_component= res1.getString(4);
   res1.close();

    stmt2 = connect.createStatement();
   query2="SELECT URL FROM creation where
   File_name=\""+ProjectFile_name+"\";";
    res2 = stmt2.executeQuery(query2);
   res2.next();
   String Location= res2.getString(1);
   res2.close();

   stmt3 = connect.createStatement();
   query3="SELECT Objective_function,Design_variable,Constraints
   FROM mdoproblem where Problem_formulation =\""+Formulation+"\";";
   res3 = stmt3.executeQuery(query3);
   res3.next();
   String Objective_function= res3.getString(1);
   String Design_variable= res3.getString(2);
   String Constraints= res3.getString(3);
   res3.close();

   stmt4 = connect.createStatement();
   query4="SELECT Wrapper_File_name,Batch_file_name,Description
   FROM wrapper where Project_name =\""+Project_name+"\";";
   res4 = stmt4.executeQuery(query4);
   res4.beforeFirst();
   res4.next();
   String Wrapper_File_name= res4.getString(1);
   String Batch_file_name= res4.getString(2);
   String Description= res4.getString(3);
```

```
            res4.close();

            System.out.print(n+n+"_________Project Details_________  "+n);
            System.out.print("Project name            ::  "+Project_name+n);
            System.out.print("Project file name       ::  "+ProjectFile_name+n);
            System.out.print("Project file location   ::  "+Location+n);
            System.out.print("----------------0----------------  "+n);

            System.out.print("_________Problem Formulation Details_________  "+n);
            System.out.print("Problem formulation used ::  "+Formulation+n);
            System.out.print("Objective function      ::  "+Objective_function+n);
            System.out.print("Design Variables        ::  "+Design_variable+n);
            System.out.print("Constraints             ::  "+Constraints+n);
            System.out.print("----------------------0----------------------  "+n);

            System.out.print("_________MDO component Details_________  "+n);
            System.out.print("Optimizer used          ::  "+Optimization_component+n);
            System.out.print("Analysis Model used     ::  "+Analysis_model_name+n);
            System.out.print("Wrapper name            ::  "+Wrapper_File_name+n);
            System.out.print("Batch file name         ::  "+Batch_file_name+n);
            System.out.print("Description             ::  "+Description+n);
            System.out.print("-----------------0-----------------  "+n+n+n);
            Formulation_used = Formulation;

        }catch (SQLException sqe2){
           System.out.println("Caught SQL Exception: " + sqe2);
        }
     return Formulation_used;
      }
}
```

// Analysis Information class

```
package reconfigurationapplication;
import java.io.*;
import java.sql.*;
import javax.sql.*;
import java.util.*;
import java.lang.*;
import javax.swing.table.*;

public class AnalysisAvailable {
      public AnalysisAvailable() {
   }
```

```
  public static void main() {
        Statement       stmt1 = null;
        Statement       stmt2 = null;
        ResultSet       res1 = null;
        ResultSet       res2 = null;
        String          query1 = null;
        String          query2 = null;
        String n =System.getProperty("line.separator");
        int m=1;

        try
        {
           Connecttodatabase connect1 = new Connecttodatabase();
           Connection connect=  connect1.main();

           stmt1 = connect.createStatement();
           query1="SELECT Analysis_model_name FROM analysismodel;";
           res1 = stmt1.executeQuery(query1);
           System.out.println(n+"-----------------------------" ) ;
           System.out.println("  ANALYSIS MODELS AVAILABLE  " ) ;
           System.out.println("-----------------------------"+n ) ;

           while(res1.next())
           {
              System.out.println("(#)"+res1.getString(1)+n ) ;
           }

        }catch (Exception e){
           System.err.println ("Error ");
        }
     }
}
```

//Connecttodatabase Class

```
package reconfigurationapplication;
import java.io.*;
import java.sql.*;
import javax.sql.*;
import java.util.*;
import java.lang.*;
import javax.swing.table.*;

public class Connecttodatabase {
```

```
   public Connecttodatabase() {
   }
   public static Connection main()   {

   Connection        connect = null;

       try{
         Class.forName("com.mysql.jdbc.Driver").newInstance();
         System.out.println("Driver Registration Successful.");
         }
           catch (InstantiationException ie){
               System.out.println("Class Instantiation Exception: " + ie);
         } catch (ClassNotFoundException cnf){
               System.out.println("Class Not Found Exception: " + cnf);
         } catch (IllegalAccessException iae){
              System.out.println("Illegal Access Exception: " + iae);
         }

// Establish connection with the Database
       try {
         connect =
DriverManager.getConnection("jdbc:mysql://peridot.ces.clemson.edu/structured_reposito
ry?user=root&password=peridot");
         System.out.println("Connection to MySQL Database Successful"+connect);

       } catch (SQLException sqe1){
           System.out.println("Caught SQL Exception: " + sqe1);
           }
   return connect;
   }
}
```

LIST OF REFERENCES

1. AIAA Technical Committee on Multidisciplinary Design Optimization (MDO). *White paper on current state of art* 1991: American Institute of Aeronautics and Astronautics

2. Alexandrov, N.M. and R.M. Lewis. *Reconfigurability in MDO Problem Synthesis, Part 1; Paper No. AIAA-2004-4307*. in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2004. Albany, New York USA.

3. Alexandrov, N.M. and R.M. Lewis. *Reconfigurability in MDO Problem Synthesis, Part 2; Paper No. AIAA-2004-4308*. in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2004. Albany, New York USA.

4. Amitay I , S.K., Mujumdar P M, *Design and Development of MDO Framework*, in *MSO-DMES* 2003.

5. Blaha, M.R., J., ed. *Object-Oriented Modeling and Design with UML*. 2005, Prentice Hall: Upper Saddle River, NJ.

6. Brunstein, I., *Practical Software Testing*. 2003: Springer-Verlag New York, Inc.

7. Engineous Software Inc., *iSIGHT-FD, version 2.5 Development Guide*. 2006.

8. Engineous Software Inc., *iSIGHT-FD, version 2.5 Getting Started Guide*. 2006.

9. Engineous Software Inc., *iSIGHT-FD, version 2.5 Runtime Gateway Guide*. 2006.

10. Engineous Software Inc., *iSIGHT-FD, version 2.5 User's Guide*. 2006.

11. ESTECO, *modeFRONTIER 3 User Manual*. 2003.

12. Grosse, I.R., J.M. Milton–Benoit, and J.C. Wileden, *Ontologies for Supporting Engineering Analysis Models.* Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2005. **19**(1): p. 1-18.

13. Hazelrigg, G., *Systems Engineering: An Approach to Information-Based Design*. 1996, Upper Saddle River, NJ: Prentice Hall.

14. IEEE Std 829-1998, *IEEE Standard for Software Test Documentation.*

15. Isaacs, A., K. Sudhakar, and P.M. Mujumdar. *Design and Development of MDO Framework, Paper No. 78*. in *International Conference on Modeling, Simulation, Optimization for Design of Multi-disciplinary Engineering Systems (MSO-DMES)*. 2003. Goa, India.

16. Mocko, G., R. Malak, C. Paredis, and R.S. Peak. *A Knowledge Repository For Behavioral Models in Engineering Design*. in *24th ASME Computers and Information in Engineering Conference (CIE)*. 2004. Salt Lake City, Utah USA.

17. Padula, S.L. and R.E. Gillian, *Multidisciplinary Environments: A History of Engineering Framework Development*, in *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2006: Portsmouth, Virginia USA.

18. Padula, S.L., J.J. Korte, H.J. Dunn, and A.O. Salas, *Multidisciplinary Optimization Branch Experience Using iSIGHT Software*. 1999, NASA Langley Technical Report Server.

19. Phoenix Integration Inc., *AnalysisServer Help version 5.1*. 2007.

20. Phoenix Integration Inc., *ModelCenter Help, version 7.1*. 2007.

21. Salas, A.O. and J.C. Townsend. *Framework Requirements for MDO Application Development, Paper No. AIAA-98-4740*. in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. 1998. St. Louis, Missouri USA.