Tuomo Varis

INTEGRATION PLATFORM FOR SIMULATION-DRIVEN DESIGN OPTIMIZATION

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2011

TIIVISTELMÄ

Varis, Tuomo
Integration platform for simulation-driven design optimization
Jyväskylä: Jyväskylän Yliopisto, 2011, 27 p.
Tietojärjestelmätiede, Kandidaatintutkielma
Ohjaaja: Hirvonen, Pertti

Simulaatiopohjainen muodonoptimointi on iteratiivinen, monivaiheinen prosessi, joka usein vaatii useamman tietokoneavusteisen suunnittelun (CAD) ja -tekniikan (CAE) ohjelmiston saumatonta yhteistyötä. Tarvittavan työkaluketjun muodostaminen ja hallinta vaatii tietämystä kaikista prosessissa mukana olevista työkaluista. Mikäli prosessin suorittaminen nojaa tapauskohtaisesti luotuihin tilapäisratkaisuihin, tehty integraatiotyö menee hukkaan siirryttäessä käsillä olevasta optimointiongelmasta seuraavaan.

Tämä tutkielma sisältää johdannon simulaatiopohjaiseen muodonoptimointiin ja esittelee järjestelmän, jonka pyrkimyksenä on tehostaa optimointiprosessissa käytettävien ohjelmistojen välistä integraatiota. Tutkielmassa käsitellään aiempaa Jyväskylän yliopistossa aiheesta tehtyä tutkimusta ja kuvataan aiempaa tutkimusta laajentavan järjestelmän arkkitehtuuri.

Asiasanat: optimointi, simulointi, tietokoneavusteinen suunnittelu, integrointi, hajautetut järjestelmät

ABSTRACT

Varis, Tuomo
Integration platform for simulation-driven design optimization
Jyväskylä: University of Jyväskylä, 2011, 27 p.
Information Systems, Bachelor's Thesis
Supervisor(s): Hirvonen, Pertti

Simulation-driven design optimization is an iterative multi-phase process often requiring seamless co-operation of a set of different Computer Aided Design (CAD) and Computer Aided Engineering (CAE) tools. Forming and managing such tool chain requires expertise on all pieces of software involved in each phase. If the execution of the process relies on ad-hoc solutions chosen to apply only to a single optimization problem at hand the effort placed in integrating these tools is lost when engaging in a new optimization project requiring a slightly different set of tools or execution with a different set of parameters.

This thesis presents an introduction to simulation-driven design optimization and proposes a system to facilitate the integration of different software tools involved in each phase of the optimization process. Earlier research on the subject conducted at the University of Jyväskylä is examined and requirements and an architectural draft of a system extending the one presented in the original research are given.

Keywords: optimization, simulation, computer aided design, integration, distributed systems

FIGURES

Figure 1: Different tools and dataflow in a design optimization process 9
Figure 2: Data flow of the Design Optimization Platform (Salmi, 2008) 16
Figure 3: Architecture of the integration platform 24

TABLES

Table 1: Design optimization process phases and required tools with their inputs and outputs 9

TABLE OF CONTENTS

TIIVISTELMÄ 2

ABSTRACT 3

FIGURES 4

TABLES 4

TABLE OF CONTENTS 5

1 INTRODUCTION 7

2 INTRODUCTION TO SIMULATION-DRIVEN DESIGN OPTIMIZATION 8
2.1 Overview of the optimization process 8
2.2 Obtaining the geometry 10
2.3 Meshing the geometry 12
2.4 Evaluating the design 13
2.5 Generating design variable values 13

3 INTEGRATED DESIGN OPTIMIZATION PLATFORM 15
3.1 Goals and justification 15
3.2 Previous research: Design Optimization Platform 15

4 EXTENDING THE DESIGN OPTIMIZATION PLATFORM 18
4.1 Goals 18
4.2 Overview of the system 18
4.3 Components of the integration 19
4.3.1 CAD software 19
4.3.2 Mesher 19
4.3.3 Solver 19
4.3.4 Optimizer 19
4.4 Supporting components 20
4.4.1 Mesh converter 20
4.4.2 CAD model repository 20
4.4.3 Solver script repository 20
4.5 Use Cases 20
4.5.1 Add a new optimization case 21
4.5.2 Import a CAD model 21
4.5.3 Import a new solver script 21
4.5.4 Create a new script from a template or an existing script 21
4.5.5 Execute an optimization run using an optimizer connected the system 22
4.5.6 Monitor status of an optimization run 22

4.5.7 Execute an optimization run using an external client application22

5 ARCHITECTURE OF THE INTEGRATION PLATFORM 23
5.1 Overview of the architecture 23
5.2 Central server 24
5.3 CAD/CAE application server 24
5.4 Repositories 25

6 CONCLUSION 26

REFERENCES 27

1 Introduction

Simulation-driven design optimization is an iterative multi-phase process often requiring seamless co-operation of a set of different Computer Aided Design (CAD) and Computer Aided Engineering (CAE) tools. Forming and managing such tool chain requires expertise on all pieces of software involved in each phase. If the execution of the process relies on ad-hoc solutions chosen to apply only to a single optimization problem at hand the effort placed in integrating these tools is lost when engaging in a new optimization project requiring a slightly different set of tools or execution with a different set of parameters.

One challenge for integration of these tools is that the interfaces (inputs, outputs, file formats and parameters) of the programs vary between applications and different phases of the process require a different amount of computation power. Thus it would be reasonable for example to perform the simulation and objective function computation using a more powerful computer. Also using different types of meshers for different types of problems would be useful. Preparing a full optimization pipeline for each optimization problem is a demanding task and requires expertise of different applications from the user. In order to solve this problem an integration platform has been suggested.

The goal of this thesis is to propose a system that facilitates interoperability between different CAD and CAE tools in a distributed environment while allowing the users to adapt the tool chain in a straightforward manner according to the optimization problem at hand.

The thesis starts by an introduction to the problem domain, simulation-driven optimization. Each phase of the optimization process is described. The next section presents previous research done in the topic at our university. This is followed by requirements of the system extending the previous research expressed by non-functional requirements and use cases. As a conclusion a draft of architecture for the system is proposed.

2 Introduction to simulation-driven design optimization

2.1 Overview of the optimization process

Optimization is a mathematical method to find the best element from some set of available alternatives. In design optimization the search space contains all possible variations of a certain design and it can be used to improve the physical properties of the product, such as its aerodynamic properties or to reduce manufacturing costs via optimizing the shape of the product and reducing the amount of required material in production. On an abstract level the goal of the design optimization process is to discover the best design concerning some physical property (or properties in case of multi objective optimization) while conforming to certain constraints. This is done by creating different designs of the product and evaluating those using computational simulations.

The optimization process itself is iterative and consists of constructing a new design, preparing it for simulation, running the computational simulation and using the simulation results to determine parameters for the new design to be evaluated during the next iteration (Figure 1). First the optimizer generates initial values for the design variables. They are used to create a new design of the object under optimization using a parameterized geometry and a design tool such as CAD software or integrated design tool in a CAE software package. The geometry is then discretized by a mesher in order to obtain an element mesh. The generated mesh is used in a computational simulation of interesting properties of the object by a CAE tool such as a structural analysis or computational fluid dynamics (CFD) application. The result is the value of the objective function of the design that represents the quality of the evaluated design according to the objectives of the optimization. The value of the objective function is then used by the optimizer to determine new design variable values in order to generate a new design for evaluation. This optimization cycle is continued until a sufficient number of evaluations have been made or a certain threshold for the objective function (i.e. quality of the design) is reached. (Katsaros, Kyriazis, Dolkas & Varvarigou, 2008)

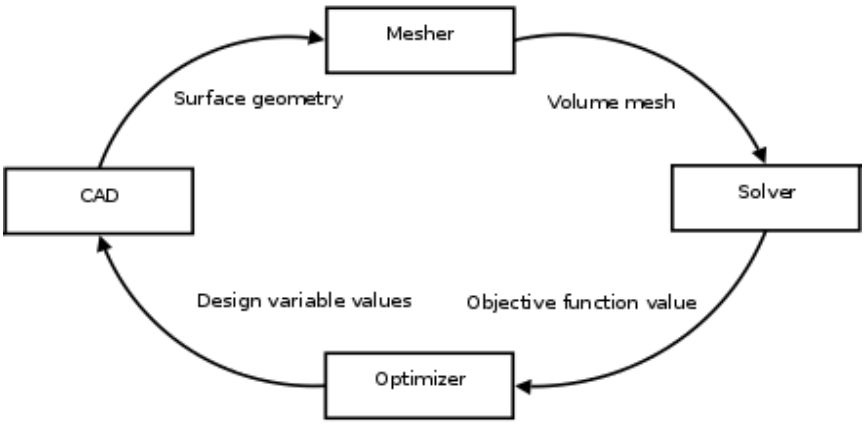

Figure 1: Different tools and dataflow in a design optimization process

There exists full-featured CAE software packages that provide functionality for all phases of the design optimization process but in industrial applications it is often required to integrate different pieces of software to form the optimization cycle. The choice of CAD software used by the organization, the complexity of the optimized geometry, mesh properties and choice of analysis software all affect the process. Table 1 lists the design optimization process phases and required tools with their inputs and outputs along with an example software product.

Table 1: Design optimization process phases and required tools with their inputs and outputs

Optimization process phase	**Required tool**	**Inputs**	**Outputs**	**Method of invocation**	**Software example**
Remodeling	CAD system	Vector of design variable values	Boundary representation of the geometry of the new design (Multiple file formats)	Macros in a batch mode or an application programming interface (API)	CATIA
Meshing	Separate mesher or a mesher in a CAD/CAE toolkit	Boundary representation of a geometry (Multiple file formats)	Element mesh corresponding to the geometry (Multiple file formats)	Batch mode	TetGen

Computing objective function value	FEM/FEA toolkit	Computation model and an element mesh (Multiple file formats)	Objective function value	Batch mode execution of a macro or a script	FreeFem++
Optimization	Optimizer	Objective function value	Vector of design variable values	Batch mode or a graphical user interface	ModeFRONTIER

2.2 Obtaining the geometry

CAD software is the primary tool for designing products for manufacturing. Unfortunately the CAD models intended for manufacturing have very different requirements compared to models intended for computational simulation (Lee, 2005). Often the models used in manufacturing include far too many details for simulations. The details may be too small resulting in overly complex computations with only minor effects on the actual results or altogether irrelevant (such as the interior of a car in CFD analysis on its aerodynamic properties). The model intended for manufacturing might also contain small gaps or other abnormalities that make using it in simulations difficult. Also a method to conveniently create alternative versions (designs) of the same product is needed. This means the CAD model of the product must be prepared carefully in order to be useful in optimization. The problem of details and gaps is solved by suppressing features of the original design and healing the geometry. This is largely manual work of the engineer while some automatic tools to speed up the process do exist.

In CAD modeling the geometry of the product is represented as a set of features and the resulting boundary representation (B-rep) (Beall, Walsh & Shephard, 2004). Such a model can be parameterized thus enabling automatic generation of differing designs of the product. These parameters can be continuous (e.g. the height or length of a feature) or discrete (e.g. thickness of metal plating varying in certain steps). The choice of parameterization is crucial to the success of the optimization. According to Pierret (2005) an ideal parameterization is characterized so that it must:

- Be able to generate a large variety of physically realistic shapes with as few design variables as possible

- Be robust meaning that a random perturbation of the design variables should still provide a realistic design
- Be able to import any existing geometries from CAD files with very few engineering time, few computational resources and to an arbitrary accuracy specified by the designer
- Be generic to be applied to a large variety of shape optimization problems and able to be integrated or coupled with any existing CAD system
- Provide design variables that can easily be handled by an engineer in order to define design variable bounds
- Provide an easy optimization problem by minimizing the skewness and improving the conditioning of the design space

While some of above characteristics are highly dependent on the CAD and CAE software used, the qualities concerning the variety and validity of the designs while maintaining the optimization problem small enough to be solved is much in the hands of the engineer making the choice of parameterization.

When using a parameterized CAD model in design optimization a method for assigning new values for the parameters (design variables) and extracting the modified geometry without user interaction is needed. In a paper by Beall, Walsh & Shephard (2003) four techniques for CAD geometry access are covered, each having their own benefits and drawbacks:

- Translation & Healing
- Discrete Representations
- Direct Geometry Access
- Unified Topology Accessing Geometry Directly

Translation and Healing is the simplest and historically most commonly used method to access a CAD geometry for simulations. In the context of design optimization this can be done by launching the software in batch mode and executing a macro that performs the loading of the model, altering the parameters according to design variable values and exporting the geometry to a file. This approach, while historically the most commonly used, has unfortunate drawbacks. Because the internal, continuous, representation of the model in the CAD software is dependent on the features of the software itself, such as proximity tolerances set on surfaces the software considers to be connected, the exporting of the geometry in a standard file format such as STEP or IGES and loading in a different application can result in dirty geometry like loss of features, addition of unwanted features and malformations in topology of the object. There are automatic tools designed for healing of dirty geometry like CADFix (TranscenData, 2011).

Discrete representations approach attempts to solve the issue of dirty geometry by generating and exporting a discrete, faceted representation of the

model inside the CAD system. This however causes loss of feature information and makes further modifications of the model later in the pipeline very difficult.

Direct geometry access allows for accessing the internal representation of the model in the CAD system through an application programming interface (API) provided by the CAD system itself or a third party toolkit like CAPRI (CADNexus, 2011). Accessing the geometry in a native format through the own geometry kernel of the CAD system guarantees that no intended features are lost due to mismatch between the way different applications interpret geometry information. In order to be useful in simulations this method requires the usage of a CAD model specifically de-featured to be suitable in simulations.

The unified topology method extends direct geometry access by providing ability maintain a relationship between the original CAD model used for manufacturing and the simplified model used for simulations and optimization. One of the benefits of this approach is that de-featuring done in order to successfully perform simulations does not affect the original CAD model but the changes in the design by optimization are directly applied to the original model as well.

2.3 Meshing the geometry

Due to the nature of numerical methods used in computational simulation, the geometry obtained from a CAD model needs to be discretized. This process is commonly referred to as "meshing". It transforms the continuous representation of shapes into a discrete approximation of the geometry using a set of small interconnected elements. The result is called a "mesh" or a "grid".

Typically meshes are divided in three different categories: structured, unstructured and hybrid (Frey & George, 2000). In a structured mesh, each element has the same number of neighbors and the elements are typically rectangular or parallelepiped (Salmi, 2008). This allows referencing of each element by using three indexes: i, j and k. Due to the possibility of using a structured mesh in a control-volume method computation the elements are sometimes called "volumes" or "cells". The counterpart to structured meshes, unstructured meshes, are suitable to be used in Finite element method (FEM) computations. Traditionally they consist of triangular or tetrahedron shaped elements but also other geometric shapes can be used. While the usage of an unstructured triangular mesh requires a high resolution to correctly represent physical phenomena in for example computational fluid dynamics, they are efficient in approximating complex geometries. To mitigate the computational time requirements caused by a dense, high resolution, mesh, a hybrid mesh can be used. A hybrid mesh is a mesh that combines properties of both structured and unstructured meshes by using rectangular or hexahedron shaped elements close to rigid surfaces and triangles or tetrahedrons on the boundaries. (Siikonen, 2010)

The quality of results obtained from simulation is highly dependent on the type of mesh used. It is necessary that the mesh can approximate the original geometry with sufficient accuracy. Also, the choice of the mesh type must be

done accordingly to the computational model and method used. Some methods work only on structured meshes while an ill-formed non-structured mesh might fail to capture essential properties of the physical phenomena the computational model aims to simulate. On the other hand, using an unnecessarily detailed mesh increases both CPU and memory requirements of the simulation.

2.4 Evaluating the design

To discover how the object under optimization behaves, computational simulation is used. This is done by modeling the physical phenomena on the geometry using a CAE tool. Based on a mesh with material properties and boundary, initial and loading conditions applied it formulates a set of discrete simultaneous system equations using a numerical method such as Finite Element Method (FEM), Finite Volume Method (FVM) or Finite Difference Method (FDM) and solves them. In general FEM is used for structural simulation of solids and the latter two are used for simulation involving fluids, while applying FEM or FDM to both kinds of problems is possible. A detailed description on the mathematical basis of computational simulation is too broad for this thesis but interested readers can refer to Liu & Quek (2003) and Hämäläinen & Järvinen (2006) for further information. However, it is necessary to note that simulation itself involves different types of methods and depending on the optimization problem at hand a suitable one must be chosen.

Results of the simulation are used to compute a value for the objective function. Objective function is the mathematical representation of the goal of the optimization that can be used to measure the "goodness" of a design and compare two designs quantitatively. It is traditionally formulated in such a way that designs closer to the optimum have a lower objective function value. (Alonso, 2004) The tools involved in this phase of the process are referred to as "solvers" in this thesis.

2.5 Generating design variable values

Using computational simulations to test product designs is a method successfully used in industry to validate product designs and cut down the costs of experimenting with physical prototypes (Aberdeen Group, 2006). In order to use the information obtained from simulations to actually improve the designs a method for determining new, preferably better, designs based on simulated experiments is needed. Historically this process has been a manual process relying on engineer experience. However, the mathematical method of optimization can be applied to the problem to automate the generation of new designs. On a general level an optimization problem can be defined as a task to find a value to minimize a function, subject to a set of constraints. In design optimization the

function to minimize is the objective function, the value is a vector of design variable values and the constraints constrain the search to the set of viable designs.

As an area of applied mathematics, optimization research has developed a wide range of different algorithms and methods depending on the type of the optimization problem. The suitability of each method depends on the type and complexity of the objective function, the size of the search space (increased by the range and number of design variables) and whether the design variables are continuous or discrete. In general, finding a global minimum of the objective function - the absolutely best possible design - is very difficult in many cases and the performance of different algorithms varies greatly depending on circumstances. Different algorithms also require a varying amount of design evaluations to converge to a solution. As design evaluation is the most computationally expensive step of the design optimization process, it is crucial to choose an appropriate optimization algorithm for each problem. (Alonso, 2004)

3 Integrated design optimization platform

3.1 Goals and justification

The previous section shows that while simulation driven design optimization consists of multiple phases that remain conceptually similar from problem to problem - geometry needs to be prepared and meshed, simulation needs to be performed and new designs need to be generated - the methods used vary greatly depending on the type of the problem and physical phenomena simulated. Thus it is reasonable to propose an integrated software suite where the main workflow of the process remains static, yet the parts performing each individual task can be changed or tuned according to the requirements posed by the problem. Previously two research projects at the University of Jyväskylä, DESIGN (FiDiPro, 2007) and GENEDES (TEKES, 2009) have aimed to address this need.

3.2 Previous research: Design Optimization Platform

DESIGN (Advanced Simulation Methods for Integrated Multi Disciplinary Industrial Design) was a University of Jyväskylä project led by fiDiPro professor Jacques Periaux. One of the objectives of the project was to create and deploy a platform for design optimization integrating various CAD and CAE software and facilitating collaboration by sharing of information and computational resources (Periaux, 2007). The first step to the creation of such a platform was the Design Optimization Platform created by Santtu Salmi and described in his master's thesis. (Salmi, 2008)

The original Design Optimization Platform consists of following components:

- The CAD system and CAPRI

- Mesh generation
- Solver
- ModeFRONTIER

The CAD system used in the Design Optimization Platform is CATIA V5 (Dassault Systèmes, 2011). Choice of CATIA as the primary target CAD system for the Design Optimization Platform was natural, because it is one of the most popular CAD systems on the market and also widely used among industrial partners of the DESIGN and GENEDES projects.

Design Optimization Platform uses the method of direct geometry access and the CAPRI Application Programming Interface (CADNexus, 2011) to modify CAD geometries. It is used to access the features of the CAD system in order to load the initial design of the object under optimization, remodel the design via design parameters and discretize the geometry. The discretized geometry is then transferred to the mesher. In Design Optimization Platform's case an open source meshing software called TetGen was used to create the element mesh used for computations. For simulation two pieces of software were tested: Elmer, an open source finite element method software package, and Comsol, a commercial multiphysics software package. Finally, the objective function value is obtained from the solver and fed back to ModeFRONTIER, a commercial CAD/CAE integration and optimization software in order to obtain new parameter values for the geometry and launch a new iteration of the process. Figure 2 displays the data flow of the system.

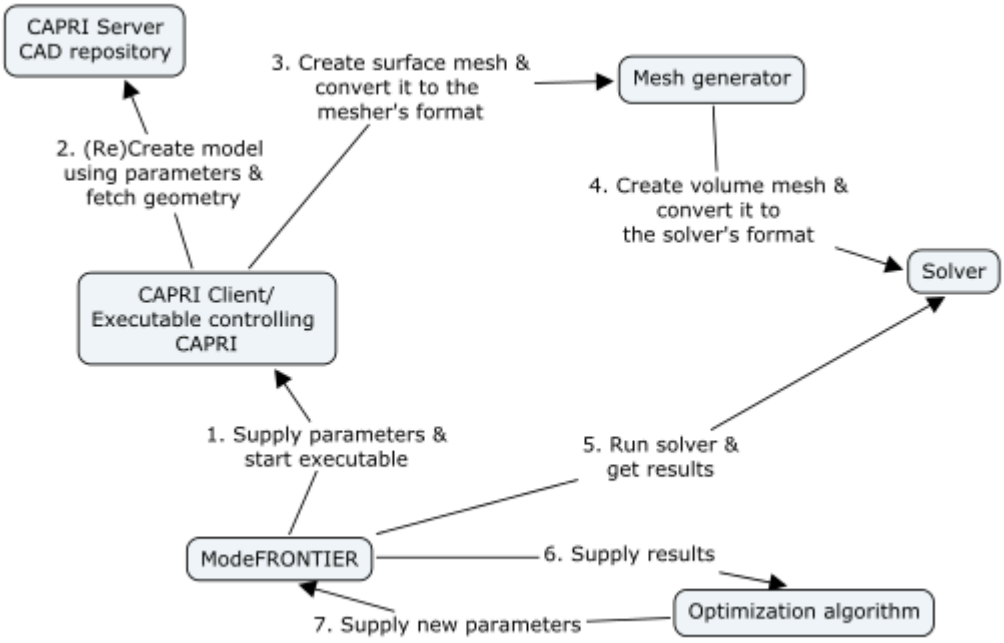

Figure 2: Data flow of the Design Optimization Platform (Salmi, 2008)

In order to distribute the different steps of the process to different computers, the Design Optimization Platform requires a Secure Shell (SSH) server to be installed on each workstation used. SSH is a secure protocol that can be used to transfer files and launch processes over a TCP/IP network using an encrypted connection (Ylönen, 2006). In Design Optimization Platform specific batch scripts were created to launch each application and specific naming conventions to determine input and output files for each application were used.

While Salmi's test runs using simple optimization test cases were successful, some design choices present in the Design Optimization Platform makes usage and extending the system difficult. Reliance on custom batch scripts and naming conventions for execution of each application enforces tight coupling not only between different components executing the process but also between the scripts and the optimization problem at hand. The process of changing a component (e.g. switching from TetGen to another mesher) in the system is made difficult. Preparing the system for a new optimization problem would require altering (or a complete re-write of) the scripts at all workstations used. Another disadvantage of reliance on naming conventions for input and output files without separating different tasks risks the chance of concurrent users of the system overwriting each other's work.

4 Extending the Design Optimization Platform

4.1 Goals

In this section I present the requirements of a system that extends Salmi's ideas and process model by adding features concerning malleability of the system, especially by loosening the coupling of applications. The requirements aim directly towards a multiple user system where concurrent tasks do not interfere with each other. The requirements were elicited during multiple meetings involving members from the DESIGN and GENEDES projects by personal communications with Jacques Periaux, Pekka Neittaanmäki, Tero Tuovinen, and steering group members of DESIGN and GENEDES projects.

4.2 Overview of the system

The core functionality of the system is to provide a common central interface for the applications used in the optimization process. The system does not require expertise of specific details of different pieces of software from the user but provides a component based abstraction for defining and launching optimization workflows. The system must also provide methods for easily creating, adding and removing components from the system without modifications to the core of the system. The component interface must be platform independent so that encapsulating software running on different operating systems is possible.

Each piece of software used in the same phase of the optimization process should be interchangeable. Thus each software component of the integration must be encapsulated in such a way they can be accessed through a similar well-defined interface. The user must be able to combine these components in order to create optimization workflows.

The integration platform is a distributed system. Each of the components must be able to reside physically on a different computer. This makes it possible

to run the most computationally expensive tasks on the most powerful computer. Existing network infrastructure (such as TCP/IP network) must be used for communications between different components.

The system must support concurrent usage by different users performing different tasks in different roles. Tasks performed by different users must not interfere with each other.

The system must provide a graphical user interface to the users that is accessible through a web browser. For application clients, a remote call interface along with a detailed interface description must be available.

4.3 Components of the integration

4.3.1 CAD software

CAD software is used to remodel CAD models according to new design variable values and extract the surface geometry of the new design. In practice the system should provide a simple interface for manipulating a parameterized CAD model instead of exposing all of the program's functionality.

4.3.2 Mesher

Automatic mesh generation software uses the surface geometry extracted from the CAD model to form a volume mesh. The algorithms and file formats used by different applications differ vastly. The integration platform should provide the user with ability to adjust the parameters of mesh generation (e.g. mesh type, mesh accuracy, different types of constraints).

4.3.3 Solver

The task of the solver is to compute the objective function value (i.e. the fitness of the design). In order to do this it requires the volume mesh generated by the mesher and a script or a macro defining the computational model provided by the user responsible for defining the optimization workflow. This part of the process is usually computationally the most expensive one.

4.3.4 Optimizer

The optimizer is a piece of software that is used to find the optimal design variable values to minimize the value of the objective function. On each of its iterations it invokes a workflow involving CAD software, mesher and solver and uses the resulting objective function value to generate new parameters for the next iteration. It continues generating and experimenting new designs for a cer-

tain amount of iterations or until a pre-determined threshold for the objective function value is reached. In the integration platform the optimizer should be able to act either as a client application to the system or as a process executed within the system.

4.4 Supporting components

The following components act as a part of the system as supporting components for the optimization process. While they are not crucial for the optimization process itself, they provide functionality for centralized storage and file format conversions.

4.4.1 Mesh converter

Different types of solvers require a different type of mesh and there is currently no single universal file format supported by all software. Thus the integration platform needs to provide a conversion mechanism to convert meshes between different formats. This conversion should be explicitly chosen by the engineer defining the optimization workflow.

4.4.2 CAD model repository

The CAD model repository stores parameterized CAD models used in the optimization process. The repository provides an interface for fetching CAD models and their metadata and uploading and storing new models.

4.4.3 Solver script repository

The solver script repository stores definitions of computational models used by solvers. The definitions are scripts and script templates in text format. The repository provides methods for the users to download scripts and script templates or upload their own.

4.5 Use Cases

Purpose of this section is to give the reader an overview of the proposed system from a user's point of view. These use cases cover the core functionality of the system and can be extended to functional requirements in a future design phase.

4.5.1 Add a new optimization case

1. User launches the user interface for administering optimization cases (workflows).
2. User chooses the creation of a new workflow.
3. User chooses the CAD model for optimization from the repository or imports one to the system (4.5.2).
4. User chooses the model parameters to be optimized.
5. User chooses the mesher to be used, the parameters for the program and possibly mesh converter to be used.
6. User chooses the solver to be used.
7. User chooses the solver script for computing the desired objective function.
8. User saves the new workflow.

4.5.2 Import a CAD model

1. User launches the UI for administering optimization workflows.
2. User chooses CAD model repository management.
3. User chooses adding a new CAD model.
4. User uploads the new model.
5. The UI shows a summary of the model's features (e.g. model part hierarchy, parameters) and user chooses to accept or reject the action.

4.5.3 Import a new solver script

1. User launches the UI for administering optimization workflows.
2. User chooses script repository management.
3. User chooses adding a new script.
4. User uploads the new script.

4.5.4 Create a new script from a template or an existing script

1. User launches the UI for administering optimization workflows.
2. User chooses script repository management.
3. User chooses browsing scripts and templates.
4. The user is presented with a list of existing scripts and templates sorted by solver and type.
5. User chooses to download a script or a template.
6. User opens the script or template to a text editor and creates a new script.
7. User imports the new solver script to the system (4.5.3).

4.5.5 Execute an optimization run using an optimizer connected the system

This use case covers performing optimizations using an optimizer component integrated to the system. For performing optimizations using external client applications see 4.5.7.

1. User launches the UI for executing optimization runs.
2. User chooses the optimization case defined (4.5.1) in the system.
3. User chooses an optimizer connected to the system.
4. User chooses the parameters (number of iterations, optionally objective function threshold) for the optimizer.
5. User starts the optimization.

4.5.6 Monitor status of an optimization run

1. User launches the UI for executing optimization runs.
2. The UI shows the user a summary of the user's own running optimization processes.

4.5.7 Execute an optimization run using an external client application

This use case describes how an external application interacts with the integration platform. "User credentials" refers to the user's ID and password or a unique key assigned by the system (the identification process and security policies are yet to be defined).

1. The client application requests a list of available optimization workflows from the central server identifying itself with the user's credentials.
2. The client application requests to begin a new optimization session and sends the desired workflow's unique identifier along with the user's credentials.
3. The client application sends new design variable values to the system along with the session identifier and identifies itself with the user's credentials.
4. The client receives the objective function value as a response to previous request, determines new design variable values. If exit conditions are not met the client application continues from 3.

5 Architecture of the integration platform

5.1 Overview of the architecture

The integration platform consists of a central server that is responsible for providing a central access point to the system, application servers containing the encapsulated applications and repositories used for central storages of CAD models and solver scripts. Figure 3 displays the components and their interfaces.

The design draws some ideas from the field of Service Oriented Architecture (SOA) and web services (Papazoglou & Georgakopoulos, 2003), for example the idea of independent loosely coupled applications with public strictly defined interfaces as building blocks for workflows and a central registry for different components. Such techniques have previously been a topic of research in projects involving CAD/CAE integration and further integrating simulation as a part of product lifecycle management (Park, Lee, Bang & Shin, 2006) (Zhu, Li & Yuan, 2007) (Wang, Liu, Han & Zhang, 2008). It however does not imply a specific SOA or Web Service related technique is required for implementation.

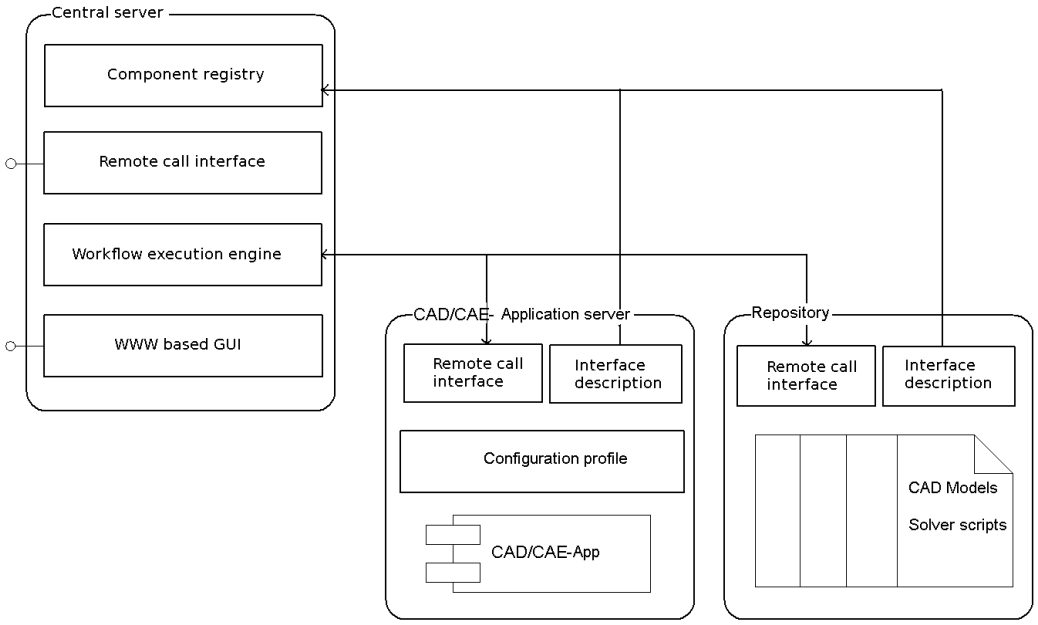

Figure 3: Architecture of the integration platform

5.2 Central server

The central server maintains a registry of all components (application servers) connected to the system along with interface descriptions required to request computation services from them. It is also responsible for providing a graphical user interface to the users in form of a web application and a remote call interface to client applications.

When execution of an optimization workflow is requested, the central server works as an orchestrator by forwarding a sequence of requests of the optimization workflow to corresponding application servers. Therefore the central server acts as a primary client for the application servers when performing optimization.

5.3 CAD/CAE application server

The application server is a three layer application. It publishes an interface description containing all the details (parameters, inputs and outputs) required to execute the encapsulated application to the component registry in the central server. The key idea is that all application servers offering computation services of a certain kind (remodeling, meshing, file format conversion etc.) comply to the same interface while the capabilities and parameters might vary. This enables interchangeability of components in an optimization workflow and convenient experimentation with different parameters of an application.

The application server uses a configuration profile specific to the used application to map the request matching to a generic interface description to the actual invocation of the application. This may be done by altering the command-line options, batch scripts or macros used in executing the application.

5.4 Repositories

The repository is essentially a remote storage service. Its task is to store and retrieve CAD models (binary data), solver scripts and templates (text data) and to maintain appropriate meta-data connected to the stored objects.

6 Conclusion

The goal of this thesis was to propose an integration platform to support simulation-driven design optimization activities. The reader was given an introduction to the multi-phased simulation-driven design optimization process. It was realized that while the process itself remains conceptually similar from a problem to problem, the nature of the optimization problem has large implications on the methods used in each phase. It was also noted that if the execution of the process relies on ad-hoc solutions chosen to apply only to a single optimization problem at hand the effort placed in integrating these tools is lost when engaging in a new optimization project requiring a slightly different set of tools or execution with a different set of parameters.

The third section presented previous research aiming to resolve the problem of software integration in design optimization. The last two sections drew on previous research by presenting requirements and an architectural draft of a system that extends the original research by adding features concerning malleability of the system, especially by loosening the coupling of applications.

REFERENCES

Aberdeen Group. (2006). *Simulation Driven Design Benchmark Report - Getting It Right the first Time.* Aberdeen Group.

Alonso. (2004). *AA222: Introduction to Multidisciplinary Design Optimization lecture material.* Stanford University. Fetched from http://adg.stanford.edu/aa222/

Beall, M.J., Walsh, J., Shephard, M.S. (2004). A comparison of techniques for geometry access related to mesh generation. *Engineering with Computers* (20), 210-221.

Beall, M.J., Walsh, J., Shephard, M.S. (2003). Accessing CAD geometry for mesh generation. *12th International Meshing Roundtable.* Sandia National Laboratories.

CADNexus. (2011). *CAPRI Overview*. Fetched from CADNexus Web site: http://www.cadnexus.com/index.php/products

Dassault Systèmes. (2011). *About CATIA*. Fetched from Dassault Systèmes Web site: http://www.3ds.com/products/catia/

FiDiPro. (2007). *Jacques Periaux researcher profile.* Fetched from FidiPro - Finland Distinguished Professor Programme - FiDiPro Professors and Fellows site: http://www.fidipro.fi/templates/fidipro_common/pdf_all.php?l=en

Frey, P.J., George, P.L. (2000). *Mesh Generation, Application to finite Elements. p. 97.* Hermes Science Publishing.

Hämäläinen, J., Järvinen J. (2006). *Elementtimenetelmä virtauslaskennassa.* Tieteen tietotekniikan keskus CSC.

Katsaros, G., Kyriazis, D., Dolkas, K., Varvarigou, T. (2008). *CFD Automatic Optimization in Grid Environments*. 3rd International Conference From Scientific Computing to Computational Engineering (IC-SCCE), Athens, Greece, 2008.

Lee, S.H. (2005). A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques. *Computer-Aided Design* (37), 941-955.

Liu, G.R., Quek, S.S. (2003). *The finite Element Method: a Practical Course.* Butterworth Heinemann.

Papazoglou, M.P., Georgakopoulos, D. (2003). Service Oriented Computing. *Communications of the ACM , 46* (10).

Park, S.W.,Lee, J.K.,Bang, J.S., Shin, B.C. (2006). Development of an e-Engineering Framework for Automotive Module Design. in *Computer Supported Cooperative Work in Design II* (ss. 264-273). Springer Berlin / Heidelberg.

Pierret, S. (2005). Multi-objective and Multi-Disciplinary Optimization of Three-dimensional Turbomachinery Blades. *Proceedings of the 6th World Congresses of Structural and Multidisciplinary Optimization.*

Salmi, S. (2008). *Multidisciplinary Design Optimization in an Integrated CAD/FEM Environment.* Master's Thesis, University of Jyväskylä, Department of Mathematical Information Technology.

Siikonen. (2010). *Virtaussimulointi.* Lecture Material, Helsinki University of Technology, Department of Applied Mechanics. Fetched from http://cfdthermo.tkk.fi/Teaching/Ene-39.4054/

TranscenData. (2011). *CAD Translation - CADFix*. Fetched From TranscenData Web Site: http://www.transcendata.com/products/cadfix/

TEKES. (2009). *Generic Product Design Optimization Environment with Knowledge Transfer Aspect*. Fetched from DTP - Projektit: http://www.tekes.fi/ohjelmat/DTP/Projektit?id=9773617

Wang, H., Liu, G., Han, B., Zhang, J. (2008). Collaborative Simulation Environment Framework Based on SOA. *CSCWD'08 12th International Conference on Computer Supported Cooperative Work in Design.*

Ylönen, T. (January 2006). *RFC-4251: The Secure Shell (SSH) Protocol Architecture.* Fetched from IETF Documents: http://www.ietf.org/rfc/rfc4251.txt

Zhu, H., Li, G., Yuan, L. (2007). WSHLA: Web Services-Based HLA Collaborative Simulation Framework. *CDVE'07 Proceedings of the 4th International Conference on Cooperative Design, Visualization, and Engineering.*