Engineering with Computers (2006) 22: 33–46
DOI 10.1007/s00366-006-0028-8

ORIGINAL ARTICLE

Elina Madetoja · Kaisa Miettinen · Pasi Tarvainen

Issues related to the computer realization of a multidisciplinary and multiobjective optimization system

Received: 12 June 2005 / Accepted: 2 May 2006 / Published online: 11 July 2006

Abstract Issues and novel ideas to be considered when developing computer realizations of complex multidisciplinary and multiobjective optimization systems are introduced. The aim is to discuss computer realizations that make possible both computationally efficient multidisciplinary analysis and multiobjective optimization of real world problems. We introduce software tools that make typically very time-consuming simulation processes more effective and, thus, enable even interactive multiobjective optimization with a real decision maker. In this paper, we first define a multidisciplinary and multiobjective optimization system and after that present an implementation overview of such problems including basic components participating in the solution process. Furthermore, interfaces and data flows between the components are described. A couple of important features related to the implementation are discussed in detail, for example, the usage of automatic differentiation. Finally, the ideas presented are illustrated with an industrial multiobjective optimization problem, when we describe numerical experiments related to quality properties in paper making.

Keywords Multidisciplinary system · Multiobjective optimization · Computer realization · Efficiency · Paper making

1 Introduction

When complex real world problems are considered with computers and computational methods, studies usually concentrate on some feature or a subsystem of a large system, which can be a part of a phenomenon or dynamics of an industrial process, see for example [1]. The systems or partial ones are described by means of mathematical models, like models of partial differential equations (PDEs) type, that mimic the most important features of the systems in consideration. Since the actual target in industrial design is, however, to control system behaviour or even improve it, simulation techniques are combined with optimization to obtain so-called model-based optimization problems. In these problems, the main idea is that the simulation models produce as outputs objective function values, which optimization methods then improve. Hence, the simulation models are solved whenever objective function values are needed during the optimization processes. This makes the computations very hard because even the simulation models themselves are typically time-consuming to solve and require a lot of computer capacity.

Simulations so far have mainly described only parts of the real world systems; one could say that only sub-simulations of systems have been seriously considered, see [2]. With such simplifications, the results obtained cannot be too reliable. Nowadays, larger capacity of computers and development in mathematical modeling tools allow us to employ more demanding and accurate simulation models. We are even able to combine several submodels into a simulation model that describes now an entire system instead of a part of it. Thus, we can mimic an industrial system behaviour in more detail and obtain more precise solutions and even take into account interrelationships between the submodels. Then mathematical modeling is typically to be understood in a multidisciplinary sense, since two or more analysis disciplines may be coupled with each other for the design of the systems.

Besides multidisciplinary simulation, we are able to control system behaviour with a chosen optimization method combined with the simulation model. Hence, the related optimization problems are known as multidisciplinary optimization problems [1, 3–6]. By these means, we are able to optimize entire industrial systems. Until now, as with simulation studies, also studies related to optimization of real world systems have typically concentrated on controlling some parts of the whole sys-

E. Madetoja (✉)
Department of Physics, University of Kuopio,
P.O. Box 1627, 70211 Kuopio, Finland
E-mail: elina.madetoja@iki.fi

K. Miettinen
Helsinki School of Economics, P.O. Box 1210,
00101 Helsinki, Finland
E-mail: kaisa.miettinen@hse.fi

P. Tarvainen
Numerola Oy (Inc.), P.O. Box 126, 40101 Jyväskylä, Finland
E-mail: pasi.tarvainen@numerola.fi

tems. Then, from the point of view of the whole system, instead of optimal solutions only 'suboptimal' solutions have been obtained. The suboptimal solutions are not necessarily optimal for the whole system, because there exist system dynamics between different parts that are not taken into account in suboptimal solutions. Our intention here is to consider systems as entirenesses as much as possible and obtain optimal solutions that improve the whole system behaviour. Another shortcoming in earlier studies has been that optimization has often been defined as single objective optimization. However, most applied optimization problems have typically multiple conflicting objectives by nature and by putting all the objectives into one or by ignoring some of them, some relevant information can be lost. The optimization method and algorithm development with the increased computer capacity makes it also possible to use multiobjective optimization and even interactive multiobjective optimization methods. For an example of an interactive multiobjective optimization design strategy, see [7].

In this paper, we present a generic architecture for the implementation of multidisciplinary and multiobjective optimization systems. We define the system architecture from the point of view of optimization and requirements it sets (see also [8, 9]) such that it can be tailored for different applications. So far, this kind of studies have typically concentrated on some specific software tool and its properties for the predetermined multidisciplinary analysis of an existing system. Thus, the studies have been application-specific, see, for example [1–4]. There are also studies in which some general aspects of multidisciplinary analysis are considered, see [5, 6, 10].

Our approach differs from previous studies so that it does not fix an optimization method nor a simulation model. Thus, the architecture is feasible for both single and multiobjective optimization problems and several simulation techniques. We integrate disparate disciplinary subsystem models to develop an optimization framework so that the result is computationally efficient. In addition, we present in detail some features related to sophisticated implementation. For example, we introduce an idea of the dynamic generation of a coupled system of submodels, which allows us to easily construct different simulation models. Furthermore, it makes even possible to define different optimization problems without any changes at the code level only by changing problem input settings. Thus, the dynamic model generation is a feature that makes our architecture flexible. We concentrate mainly on consecutively coupled submodels. The dynamic model generation increases also the efficiency of computations, because simulation models generated consist only of the submodels that are necessary and not all those that are available. We can also store different (component-wise) submodels into databases. Then, during the simulation model generation, we select for computations only those submodels that participate in the production of the model outputs, which are related to the objective functions chosen to be optimized. We can also increase the efficiency of computations by using reduced submodels that are simplifications of computationally demanding submodels.

Besides introducing the system architecture, we also present basic components taking part in the solution process of a multidisciplinary optimization problem and consider data flows between these components, similarly to [11]. The basic components are related to solving the simulation model and the usage of the optimization method. The data flows between these components typically include values of the decision (optimization) variables and the objective functions. The main difference to other studies is that we include gradient information into the data flows (see, e.g. [11]). We have been studying and developing techniques that are appropriate for gradient computation in multidisciplinary optimization problems. When the objective function values are produced by a simulation model, also gradient information (if used in optimization) needs to be computed with a simulation model. We use a technique called model-based differentiation in gradient computations that is somehow similar to the so-called global sensitivity equations [12]. Using gradient information is one of those features that make optimization processes efficient and, thus, enables the optimization of the real world systems. Especially, we concentrate on interactive multiobjective optimization [13], where an expert, known as a decision maker, participates in the optimization in order to find the best compromise in the presence of conflicting objectives. Then the optimization processes need to be almost real time so that the decision maker can guide them iteratively and actively and (s)he does not need to wait too long to get new solutions. Thus, our intention is to highlight those important aspects in computer realization, which enable the decision maker's participation.

We illustrate the principles introduced for the architecture of a multidisciplinary and multiobjective optimization system and its implementation with a real industrial example. Our example is related to the paper making process, where the quality of the produced paper is optimized. The paper making processes in paper machines contain several phases in different components, so-called unit-processes, most of which can be mathematically modeled. Until now, simulation models have been developed only for unit-processes of the paper machine, as in [14–16]. Here, we combine several unit-process submodels together into a simulation model and study the whole paper making process in the paper machine, including also the finishing of the paper. Considering the paper making line as a whole gives new understanding of the phenomena involved. For more about paper making, see [17].

The outline of the paper is the following. First, we define a general architecture for a multidisciplinary and multiobjective optimization system and consider the system requirements. In addition, we give a mathematical formulation to a multidisciplinary as well as a multiobjective optimization problem in Sect. 2. Thereafter, we present a general overview of the architecture and some features of computer realization in more detail in Sect. 3. In Sect. 4, we demonstrate the approach via an industrial example, where a real decision maker solves an optimization problem that is both multidisciplinary and multiobjective. We also present associated numerical results. Finally, concluding remarks are given in Sect. 5.

2 Architecture of a multidisciplinary and multiobjective optimization system

Next we define a multidisciplinary and multiobjective optimization system to be considered in this paper. We also introduce several relevant issues in constructing a general architecture for such systems and list requirements that we find to be the most important ones for the system architecture.

In what follows, we assume that we have available simulation submodels associated to different parts of a real world system considered; the system can be, for example, an industrial process or a device. The submodels may come from a variety of disciplines and because of this the submodels can also be of different type, like PDEs or statistical models based on experimental data. It is also possible to employ so-called reduced (order) models, where the idea is to simplify the physical model and, in this way, to decrease computational burden. There may also be several alternative submodels of various types modeling the same phenomenon or unit-process. We couple the submodels together in order to get a model of the whole system. The submodels can be internally or consecutively coupled and, therefore, each submodel may depend on the outputs of all the other submodels. We assume that submodels have input parameters consisting of model-wise set up input parameters and outputs of the other submodels. Typically, some of the first-mentioned input parameters are chosen to be decision variables in the optimization and the rest of them are constant input parameters to the submodels.

We present a system architecture that should allow the selection of any kind of a *feasible submodel chain* to be simulated. A feasible submodel chain means that the chain is continuous and the previous submodels produce such outputs that can be used as inputs for the following submodels. The system architecture should also allow the user to be able to choose the decision variables and the objective functions freely, that is, (s)he can define any input parameter of the chosen simulation model to be a decision variable and, similarly, any output to be an optimized quantity or an ingredient of an objective function. We want to define the system architecture on a general level so that it is applicable for different real world optimization problems. In addition, the system architecture should enable efficient solution processes. This is important, because simulation models combining several submodels from multiple disciplines are typically extremely time-consuming to solve, since they describe systems that include complex and internally coupled dynamic processes. One such feature that improves efficiency is the usage of gradient information in optimization. Finally, we specify a multidisciplinary and multiobjective problem and give a mathematical formulation for the problem. This study follows mainly the system architecture definition in [18].

2.1 System definition

Generally, a *system* is an assemblage of objectives joined in some regular interaction or interdependence [19]. A *multidisciplinary system* is a system that integrates several disciplines, that is, it is an assemblage of representatives of the disciplines. Here we call a representative of a real world system as a multidisciplinary system, when it contains several dissimilar submodels. Furthermore, a *multidisciplinary optimization system* contains also optimization routines as a part of the multidisciplinary system. To be exact, we here define a multidisciplinary optimization system such that it makes possible the simulation and optimization of different real world phenomena. When we want to emphasize that the optimization problem contains several conflicting objective functions, we call the system as a *multidisciplinary and multiobjective optimization system*. We also use terms *overall* and *partial system* to draw a line between the consideration of the whole system and some part of it, respectively.

We can state a set of requirements to the architecture of the multidisciplinary and multiobjective optimization system. Several such requirements have been presented also in [8, 9]. In general, system demands are caused by the system dynamics and, in addition, its versatility in different contexts. We present here a list of requirements emphasizing the computational efficiency of the system as well as readiness to easily solve different types of simulation model-based optimization problems. Namely, the usage of multiobjective optimization, for example, demands even more flexible structures and interfaces.

Next, we list the most important requirements that we set to the system architecture.

1. Tailorability to describe different real world systems:

(a) Simulation models consisting of possibly dissimilar submodels.
(b) Use of appropriate optimization methods.

2. Capability to integrate existing program and software modules.

3. Usage of different optimization routines, also gradient-based routines, and thus, gradient information should be available.
4. Possibility for a decision maker to participate in an interactive multiobjective solution process.
5. Efficient and accurate computations.
6. Graphical representation and visualization of the system.

Next, we concentrate on the system architecture and its implementation on a computing level and do not discuss the usability aspects. For these topics, we refer to [9], for instance.

Mathematically, we formulate a multidisciplinary optimization system as follows. Let a collection of subsystems be denoted by B_1, ..., B_{nm}, where nm is the number of subsystems. In what follows, we call subsystems as *submodels*. The interactions and data flows between the submodels are assumed to be defined. The data flows include inputs and outputs for all the submodels. Each submodel B_i has its input vector $\mathbf{p}_i$ and output vector $\mathbf{q}_i$ and by the given input the submodel produces the output. Because we want to receive gradient information in order to be able to use efficient optimization methods, each submodel should produce also the (partial) derivatives of its outputs with respect to its inputs, when needed. Such gradient information can be 'directly' obtained by using automatic differentiation. Note that the system architecture allows also the usage of gradient-free methods.

In order to make the multidisciplinary system work, we should take into account the disjointness of data flows between the submodels, when we combine existing submodels into a simulation model. These submodels can be internally coupled, when each submodel produces an output that is an input for all the other submodels, see [11]. Another possibility is that the submodels are consecutively coupled. Then the submodels constitute a model chain, where the current submodel depends on all the previous submodels in the chain. Illustrations of model couplings are given in Fig. 1.

We say that two submodels B_i and B_j, where $i,j \in \{1,..., nm\}$, are *compatible*, if some output of B_i that is an input for B_j is in a suitable form to be the input for the latter submodel and possibly vice versa. Obviously, according to the definition above, submodels may not be compatible, which means that the output of one submodel is not in the right form to be the input required for another submodel. In order to make two submodels compatible, we introduce a *transformation mapping* T_{ij} as follows (see also [11])

$$T_{ij}(\mathbf{q}_i) = \mathbf{p}_j, \qquad (1)$$

where $\mathbf{q}_i$ is the output of the submodel B_i that is in an incompatible form for the submodel B_j and, therefore, it is modified with the transformation mapping. Note that transformation mappings need to be taken into account also in the gradient calculations. Therefore, transformation mappings have to be implemented such that they are able to produce both the output and the (partial) derivatives of that output in the desired form.

In addition to the integration of different submodels together for both overall or partial system simulation, the multidisciplinary optimization system should allow the optimization of some parts or the whole real world system behaviour. For optimization, we set the following additional system requirements. First, any system output can be chosen to be optimized or used as an ingredient of an objective function to be optimized. Thus, based on knowledge of system outputs, the decision maker should be able to set an assessment criterion to any of the outputs like deviation between the output and a desired aspiration level (to be minimized), for example. Furthermore, the system needs generic component that evaluates the values of objective functions according to these assessment criteria. Secondly, we assume that all the system input parameters are available to be chosen as decision variables, that is, $\mathbf{x} \subset \{\mathbf{p}_1,...,\mathbf{p}_{nm}\}$, where the decision variables are denoted

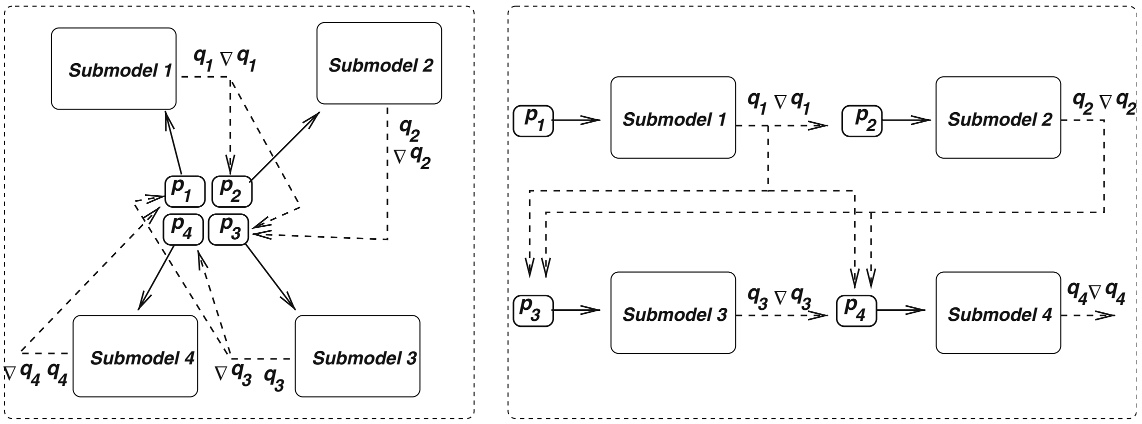

Fig. 1 Examples of internally or consecutive model couplings

by $\mathbf{x} = \{x_1,..., x_{nd}\}$. To be able to change the decision variable values the system contains for each input parameter additional data, like the maximum and minimum admissible values for the real-valued parameters (i.e., the box constraints). Those input parameters that are not chosen to be decision variables are constant input parameters and their values are determined at the beginning of the solution process. Finally, the user should be able to set optimization constraints related to model inputs or outputs.

Note that the definition above for the system architecture is abstract, not application or optimization method specific. Thus, several applied simulation and optimization problems can be defined with the architecture. It is a dynamic system where the optimization method and the simulation model are chosen separately according to the problem considered during the problem setting. Then, a specific multidisciplinary problem is individualized in the problem setting such that it may contain one or several objective functions to be optimized and a collection of submodels, which are coupled to each other.

2.2 A multidisciplinary and multiobjective optimization problem

Next, we give a mathematical formulation to a multidisciplinary and multiobjective optimization problem. We assume here that all the submodels are compatible and consecutively coupled. Later, we want to study gradients of objective functions with respect to the input parameters chosen (i.e., the decision variables). Thus, we replace the previous notation of input parameters $\mathbf{p}_i$, $i = 1,...,nm$, by writing all the different input parameters individually. All the submodels have their own input vectors consisting of three types of inputs. The first group is decision variables $\mathbf{x}$. Secondly, there are submodel-wise input parameters, whose values are fixed during the problem setting when the simulation model is initialized. The third group of input variables contains state inputs, that is, a single submodel has input from the previous submodel(s), which means that the outputs of the previous submodel(s) are inputs for the next submodel. We ignore the constant submodel-specific parameters in the equations, because their values are constant through the whole solution process.

Now, we define the problem in the following form

$$\begin{aligned}
&\min\{f_1(\mathbf{x};\mathbf{q}_1,\dots,\mathbf{q}_{nm}),\dots,f_{nf}(\mathbf{x};\mathbf{q}_1,\dots,\mathbf{q}_{nm})\}\\
&\text{subject to}\begin{cases} B_1(\mathbf{x};\mathbf{q}_1)=0\\ B_2(\mathbf{x};\mathbf{q}_1,\mathbf{q}_2)=0\\ \dots\\ B_{nm}(\mathbf{x};\mathbf{q}_1,\mathbf{q}_2,\dots,\mathbf{q}_{nm})=0\end{cases}\\
&\mathbf{x}\in S,
\end{aligned}\tag{2}$$

where $\mathbf{x}$ is a vector of decision variables from the feasible set $S \subset \mathbb{R}^{nd}$, which may be formed of equality and inequality constraints and box constraints of the decision variables. We denote objective functions by $\mathbf{f} = (f_1, ..., f_{nf})$. The submodels B_1 to B_{nm} constitute a multidisciplinary simulation model. Vectors $\mathbf{q}_i \in \mathbb{R}^{k_i}$, $i = 1,\dots,nm$, are obtained as outputs of submodels. Each output can depend on the decision variables and all the previous outputs, that is, model states.

We use a short notation $\mathbf{q}_i$, $i = 1,..., nm$, defined as

$$\begin{aligned}
\mathbf{q}_1 &= \mathbf{q}_1(\mathbf{x})\\
\mathbf{q}_2 &= \mathbf{q}_2(\mathbf{x};\mathbf{q}_1)\\
&\dots\\
\mathbf{q}_{nm} &= \mathbf{q}_{nm}(\mathbf{x};\mathbf{q}_1,\dots,\mathbf{q}_{nm-1}).
\end{aligned}\tag{3}$$

Note that the problem formulation above requires that the simulation model is solved before objective functions are evaluated. In addition, consecutively coupled submodels need to be solved in the right order from B_1 to B_{nm}.

We assume here, for simplicity, that all the objective functions, submodels and their outputs are continuously differentiable functions. The objective functions are of the form

$$f_i : \mathbb{R}^{nd} \times \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_{nm}} \to \mathbb{R}, \quad i = 1,\dots,nf,$$

and the submodels are operator mappings

$$B_i : \mathbb{R}^{nd} \times \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_i} \to \mathbb{R}^{k_i}, \quad i = 1,\dots,nm.$$

If transformation mappings are needed, they are also assumed to be continuously differentiable.

When we set $nf \geq 2$ and $nm \geq 2$ in (2), we have a real multidisciplinary and multiobjective optimization problem in question. Then, our intention is to optimize several objective function at the same time. It is assumed that these objective functions are conflicting and by optimizing one function the others obtain undesired values. In multiobjective optimization, optimality is understood in the sense of Pareto optimality, which means that a vector of objective function values is optimal if we are not able to improve any of the objective function values without impairing at least the value of one objective function [13]. All the Pareto optimal solutions are mathematically equally good and they constitute a Pareto optimal set. When a single solution is desired, it can be chosen with the help of an expert, known as a decision maker, who is familiar with the problem and can express preference relations between the objective functions.

It is often helpful for the decision maker to get information about the ranges of objective functions values in the Pareto optimal set. Lower bounds of objective functions are obtained by minimizing each objective function individually with respect to the constraints. A vector in $\mathbb{R}^{nf}$ that contains the minima for each objective function is known as an ideal objective vector. Alike, a vector that contains the upper bounds in the Pareto optimal set is known as a nadir objective vector. Typically, the nadir objective vector can not be calculated precisely, but there exist ways to approximate it, for example, with a pay off table obtained while calculating the ideal objective vector [13]. If an objective function f_i is to be maximized, it is equivalent to minimize $-f_i$ in computations. However, it is better to show unaltered objective function values to the decision maker. Thus, in maximization, ideal and nadir objective vector values represent upper and lower bounds of f_i in the Pareto optimal set, respectively.

There exist lots of multiobjective optimization methods [13]. Several methods are based on expert knowledge of the optimization problem considered. In addition, there exist such methods that choose automatically one Pareto optimal solution to be the final solution without any preference information. When the decision maker participates, (s)he gives preferences concerning the objective functions and their relationships to each others. Then the final solution is chosen based on the decision maker's knowledge of the problem. The decision maker can give preference information before or after the solution process or during the solution process [13]. The last mentioned approach is used in interactive methods.

In our numerical example, we employ here an interactive multiobjective optimization method, where the decision maker participates in the solution process and gives her/his preference information iteratively through the solution process. This method is called NIMBUS [13, 20, 21]. The NIMBUS method is based on the idea of classification of objective functions. The decision maker gives preference information by classifying the objective functions according to the changes (s)he desires in their current values. The NIMBUS method is employed to obtain the best possible compromise between the conflicting objective functions.

3 Aspects of computer realization

In this section, we discuss in more detail certain aspects of an efficient computer realization of a multidisciplinary and multiobjective optimization system. We describe a generic system architecture and give a short overview of how a simulation model and an optimization method can be combined such that the requirements presented in Sect. 2 are fulfilled. First, we introduce ideas of the dynamic generation of a simulation model. After that, we present a model-based differentiation technique and discuss the usage of automatic differentiation with that technique. These two last-mentioned features increase flexibility of the system architecture and enable us to decrease computational time in the optimization process.

3.1 Overview of system architecture

Here, we introduce in more detail some general properties that are desired for a computer realization capable of solving multidisciplinary and multiobjective optimization problems. The idea is to present such properties and features that are common to different types of problems. In Fig. 2, all the basic components that take part in the solution process of a multidisciplinary and multiobjective optimization problem are presented. In addition, data flows between the components are included there.

We have extended the problem formulation introduced in [11] to contain more than two submodels or disciplines in the simulation model. In Fig. 2, we use bold arrows to describe data flows between components. All the interfaces between the different components are designed in such a way that each component produces its outputs in a suitable form for the other components. For example, submodels are implemented such that each submodel produces output, which can be used as an input to the following submodels. Similarly to [11], we use transformation mappings to convert outputs into a required fixed form, if necessary.

A general architecture for a multidisciplinary and multiobjective optimization system contains the following main components: application (main program), problem settings, simulator and optimizer. The simulator and optimizer are interfaces to the chosen problem-specific simulation model and optimization method. They are designed such that the simulator gives in a predetermined form information that can be used in the evaluation of the objective functions and their gradients. Similarly, the optimizer gives new decision variable values in such a form that they can be used in updating corresponding parameter values in the simulation model.

In Fig. 2, the solution process begins from the application or the main program. At first, the multidisciplinary and multiobjective optimization problem to be considered is initialized with a problem setting, where the following details should be determined:

- Optimization method and optimizer.
- Simulation model: submodels and their outputs.
- Objective functions and their properties.
- Constraint functions.
- Decision variables and constant parameters for the simulation model.

The problem setting begins with specifying the optimization method and the optimization setting, like decision variables and objective functions, and it is followed by the generation of a simulation model. Note that the simulation model includes only those submodels that are necessary for the production of the chosen outputs used in the optimization. The problem setting is done by the decision maker or another user according the decision maker's instructions. We want to emphasize that the problem setting should be defined independently of application and it should be done with the help of the simulator and optimizer interfaces. Thus, the problem

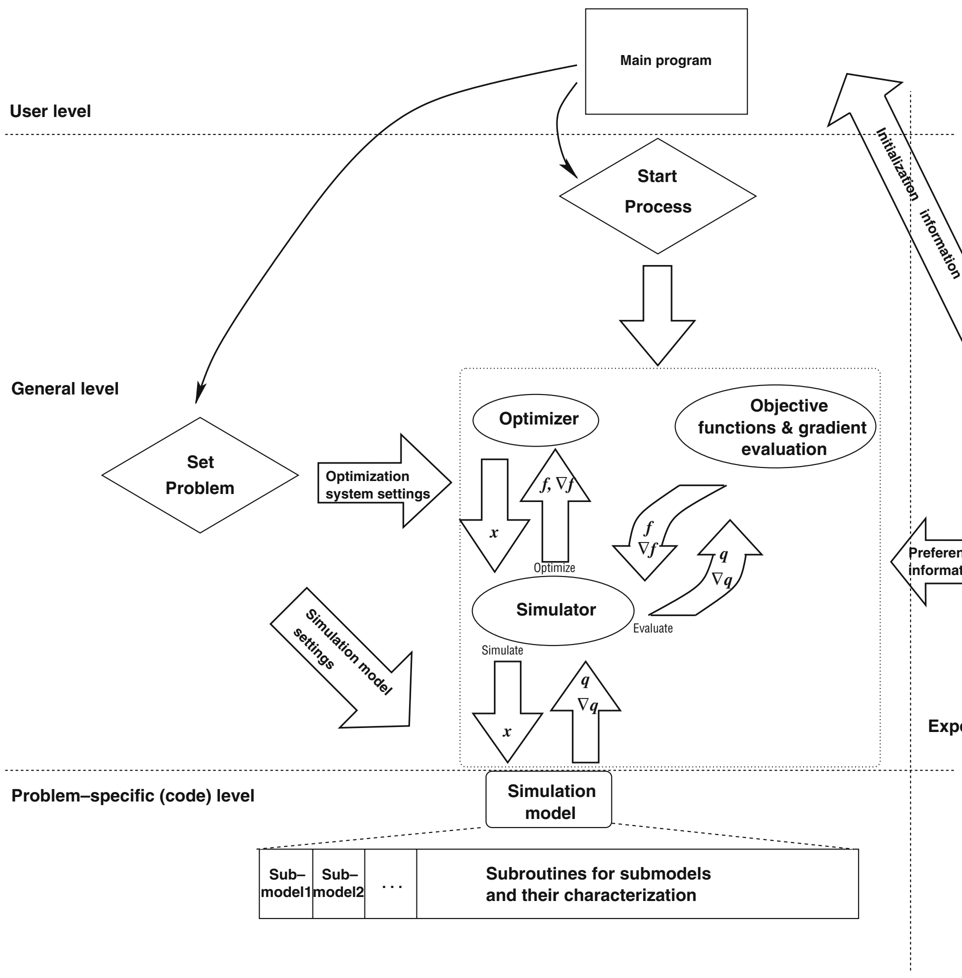

Fig. 2 General architecture and data flows between components

setting code should not contain any problem-specific input data, but the input data is given by user-defined parameter files, for example. Thus, the generic system architecture is application independent and can be tailored for different real world problems. In fact, we have designed the system architecture in such a way that the main functional parts are independent of the simulation model itself. First, the optimizer takes as the input only decision variable values, objective function and constraint function values and possibly their gradients. The simulator needs the information about the simulation model as far as it obtains certain outputs and model-based derivatives from the simulation model.

After the problem setting is done, then begins the optimization loop that is surrounded with a dotted line in Fig. 2. The optimization process is an interactive process of the simulator and optimizer, where the simulator produces chosen objective function values and possibly gradients and the optimizer improves them by changing the decision variable values. The simulator asks for a simulation service from the selected simulation model and the optimizer from the selected optimization method. In the simulation model, the chosen submodels are solved to obtain the required outputs and their derivatives if the optimizer uses derivatives. The latter means that the simulation model computes also gradient information associated to the outputs of each submodel with respect to the decision variables. This gradient information can be produced with an appropriate, but not any fixed method. For example, finite differences can be utilized, if analytical formulas are not known. However, one should note that the method used should not slow down the optimization process.

When the simulation model has been solved, the next step is to evaluate objective functions and their gradients. In the evaluation process, the objective functions and their gradients are composed of the simulation model outputs and their derivatives, respectively. Then the optimizer produces then new values for the decision variables. If an interactive multiobjective optimization method is used, the decision maker specifies preference information during the solution process. The process continues until the decision maker has found a desirable value for each objective function or none of them can be improved or some other stopping criterion is fulfilled.

3.2 Dynamic generation of simulation model

We are developing such a system architecture for multidisciplinary problems that allows the selection of any kind of a feasible submodel chain to be simulated and used in optimization. So far, we have already referred to selecting only those submodels that are really needed. In order to realize that, we suggest to use a *dynamic model generation*, which allows different choices of simulation models constructed from the existing submodels stored in databases related to the problem considered. Our idea of the dynamic generation is based on the use of submodel outputs for the identification of a simulation case. The user defines desired output(s) to be optimized and, based on that, the system fixes a chain of the currently required submodels to form the associated simulation model. The system then checks that the resulting simulation model (chain) is compatible and contains only those submodels that are necessary for the production of the desired outputs. The dependences between the submodels and the outputs of the submodels can be defined in databases, so-called model databases.

Let us illustrate the dynamic model generation shortly. At first, the user selects the outputs (s)he wants to study and optimize. Then, the corresponding optimization problem is initialized by generating a dynamic simulation model of those submodels that are necessary for the production of the chosen outputs. All the simulation model outputs are specified by means of model databases that contain dependences between outputs and models. The databases determine chains of submodels that enable to simulate some specific outputs. Another possibility for the dynamic generation of a simulation model would be to directly define the simulation model as the chain of submodels. In such a case, according to the submodel selection, the system should define and organize the related outputs using the abovementioned databases. In this way, the user would have more freedom in the generation of the simulation model, because (s)he could determine which submodels are included in the simulation model. However, it is naturally more demanding for the user, because (s)he has to know more about the submodels to be able to determine a correctly functioning simulation model.

We obtain several advantages by using the dynamic model generation in the optimization system. One important advantage is that, depending on the purpose, the approach allows the use of alternative types of submodels with different levels of complexity to simulate the same output. The user can choose between a PDE-model, or its reduced variant, or a statistical model based on experimental data. Possibilities to decrease complexity are crucial when the optimization system is extended to handle interactive optimization. The dynamic model generation also allows to select several parallel submodels and compare their differences. For example, we are able to study the reliability of reduced submodels by simulating both physical and reduced submodels at the same time.

One should note that the dynamic generation of the simulation model sets new requirements to the gradient calculations, since now the submodels of a simulation model can change case-to-case. Our model-based differentiation technique, to be described next, supports the usage of dynamic model generation.

3.3 Gradient calculations

Next, we consider the gradient calculations of the multidisciplinary and multiobjective optimization problem (2). The idea is to enable using of efficient gradient-based optimizers, but the system architecture allows also the usage of derivate-free methods. We introduce here a technique for consecutively coupled submodels that is mathematically similar to the approach of global sensitivity equations [12], but, to our knowledge, computer realization in the context of a multidisciplinary optimization system described in Sect. 2 has not been reported in the literature before.

Let us consider problem (2) with $nf, nd, nm \geq 2$, and with the assumptions of Sect. 2. We also assume that the submodels are arranged in such a way that the simulation begins from the first submodel B_1 and continues further until the last submodel B_{nm}. Each model may have three types of input parameters as mentioned earlier. We also assume that each submodel can have its own decision variables, or several submodels can share decision variables. Either way, all the decision variables are collected into the vector $\mathbf{x}$ for the optimizer.

3.3.1 Model-based differentiation

Methods for calculating derivatives for multidisciplinary systems are introduced in [12], see also [22]. We employ an approach that is based on so-called global sensitivity equations, but discuss it here primarily from the point of view of computer implementation as a part of an optimization system. Namely, we next introduce a model-based differentiation technique for multidisciplinary optimization problems, where the submodels are consecutively coupled. However, it can be used also in the case of internally coupled submodels. The method is here formally derived using implicit differentiation and the chain rule and its ideas were introduced in [18, 23]. We present here the technique for one objective function. It can be generalized for multiple objective functions in a straightforward way.

Let us first consider partial derivatives of an objective function f_i with respect to a decision variable x_j. By using the implicit differentiation and the chain rule, we obtain

$$\begin{aligned}\frac{\partial f_i(\mathbf{x};\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_{nm})}{\partial x_j} &= \frac{\partial f_i(\mathbf{x})}{\partial x_j} + \frac{\partial f_i(\mathbf{q}_1)}{\partial \mathbf{q}_1}\frac{\partial \mathbf{q}_1}{\partial x_j} + \cdots \\ &\quad + \frac{\partial f_i(\mathbf{q}_{nm})}{\partial \mathbf{q}_{nm}}\frac{\partial \mathbf{q}_{nm}}{\partial x_j} \\ &= \frac{\partial f_i(\mathbf{x})}{\partial x_j} + \sum_{k=1}^{nm}\frac{\partial f_i(\mathbf{q}_k)}{\partial \mathbf{q}_k}\frac{\partial \mathbf{q}_k}{\partial x_j}.\end{aligned} \tag{4}$$

Assume that $\partial f_i(\mathbf{x})/\partial x_j$ and $\partial f_i(\mathbf{q}_k)/\partial \mathbf{q}_k$ can be calculated for all $i = 1,\ldots,nf$, $j = 1,\ldots, nd$, and $k = 1,\ldots, nm$, and the problem is to determine the unknown partial derivatives $\partial \mathbf{q}_k/\partial x_j$, for all $j = 1,\ldots,nd$ and $k = 1,\ldots,nm$ of (4).

We calculate the so-called total derivatives of each submodel, which means that we study both implicit and explicit dependences on the decision variables. Then, by using the implicit differentiation, we formulate the equations for the total derivative of the output vectors.

Let us consider the first submodel in (2) and the unknown derivatives $\partial \mathbf{q}_1/\partial x_j$, for all $j = 1,\ldots,nd$. By employing the implicit differentiation, we obtain

$$\frac{\partial B_1}{\partial x_j} + \frac{\partial B_1}{\partial \mathbf{q}_1}\frac{\partial \mathbf{q}_1}{\partial x_j} = 0, \quad j = 1,\ldots,nd, \tag{5}$$

and further,

$$\frac{\partial \mathbf{q}_1}{\partial x_j} = -\frac{\partial B_1}{\partial \mathbf{q}_1}^{-1}\frac{\partial B_1}{\partial x_j}, \quad j = 1,\ldots,nd. \tag{6}$$

For the second submodel, we employ both the chain rule and the implicit differentiation in order to obtain the unknown derivatives $\partial \mathbf{q}_2/\partial x_j$, for all $j = 1,\ldots,nd$ as follows:

$$\begin{aligned}&\frac{\partial B_2}{\partial x_j} + \frac{\partial B_2}{\partial \mathbf{q}_1}\frac{\partial \mathbf{q}_1}{\partial x_j} + \frac{\partial B_2}{\partial \mathbf{q}_2}\frac{\partial \mathbf{q}_2}{\partial x_j} = 0, \quad j = 1,\ldots,nd, \\ &\frac{\partial \mathbf{q}_2}{\partial x_j} = -\frac{\partial B_2}{\partial \mathbf{q}_2}^{-1}\left(\frac{\partial B_2}{\partial x_j} + \frac{\partial B_2}{\partial \mathbf{q}_1}\frac{\partial \mathbf{q}_1}{\partial x_j}\right), \quad j = 1,\ldots,nd.\end{aligned} \tag{7}$$

In the general case, we obtain the following equation for the unknown derivatives $\partial \mathbf{q}_k/\partial x_j$, for all $j = 1,\ldots,nd$ and $k = 1,\ldots, nm$ by employing several times the chain rule and once the implicit differentiation

$$\begin{aligned}&\frac{\partial B_k}{\partial x_j} + \sum_{l=1}^{k}\frac{\partial B_k}{\partial \mathbf{q}_l}\frac{\partial \mathbf{q}_l}{\partial x_j} = 0, \quad j = 1,\ldots,nd, \\ &\frac{\partial \mathbf{q}_k}{\partial x_j} = -\frac{\partial B_k}{\partial \mathbf{q}_k}^{-1}\left(\frac{\partial B_k}{\partial x_j} + \sum_{l=1}^{k-1}\frac{\partial B_k}{\partial \mathbf{q}_l}\frac{\partial \mathbf{q}_l}{\partial x_j}\right), \quad j = 1,\ldots,nd.\end{aligned} \tag{8}$$

By using formula (8) for each submodel in (2), we are able to compute all the unknown derivatives. When we assemble them into formula (4), we obtain the partial derivatives of the objective function with respect to the decision variables. We can also utilize a similar approach for functions that are not necessarily differentiable in a classical sense, see [24].

Note that transformation mappings are taken into account in the model-based differentiation. They are treated as submodels in gradient computations.

3.3.2 *Automatic differentiation*

By using model-based differentiation, we can (almost) automatically produce the associated gradient information simultaneously when the simulation model is solved. Namely, we combine the model-based differentiation technique with automatic differentiation (AD) [25, 26]. Our idea is to use AD both in the submodel and the objective function gradient calculations and then collect the total derivatives of the outputs automatically. This approach makes it also possible, in many cases, to hide the differentiation from the user.

In the multidisciplinary optimization system, the gradient calculations should be implemented dynamically, that is, designed so that any of the input parameters can be chosen to be a decision variable and defined as an AD variable. Then, the simulation model is solved and for each output that depends on that input parameter (implicitly or explicitly) a certain partial derivative with respect to the chosen parameter is computed automatically. One should also note that the system implementation allows the usage of 'implicit' simulation models, like PDE-based models. Their usage requires that the corresponding submodel implementations produce both the state and gradient information.

The use of automatic differentiation allows us to change the optimization problem without making changes at the code level of the simulation model. This makes the system architecture flexible to define and solve different problems. A proper implementation also offers a possibility to easily change submodels in the simulation model or replace the whole simulation model with a different model without code changes. The model-based differentiation technique itself does not require automatic differentiation. Therefore, it allows to employ also existing (code) modules as a part of the simulation model, because the model-based differentiation does not care how the gradient information is produced. Thus, the underlying differentiation technique in some submodel can be, for example, the finite difference method in cases when the associated submodel implementation does not enable to use automatic differentiation.

Next, we want to illustrate some benefits of using our model-based differentiation technique. In our example, we optimize one objective function and the simulation model is generated of two submodels and the associated six outputs. The example is described in more detail in [18].

We employ model-based differentiation with automatic differentiation and compare it with an approach, where gradient information of the objective function is obtained by the finite difference method. We use the forward finite differences

$$\frac{\partial f(\mathbf{x};\mathbf{q}_1,\ldots,\mathbf{q}_{nm})}{\partial x_j} \approx \frac{f(\mathbf{x}+h\mathbf{e}^j;\mathbf{q}_1,\ldots,\mathbf{q}_{nm}) - f(\mathbf{x};\mathbf{q}_1,\ldots,\mathbf{q}_{nm})}{h}, \quad h > 0.$$

In this example, we study the CPU-time that is needed for a single gradient evaluation. The computations were carried out on a personal computer containing AMD Athlon(tm) XP processor 2600+ (2,133 MHz) processor and 1 GB memory. All the times listed in Table 1 are CPU-times in seconds and the columns describe the numbers of decision variables.

As one can see, the model-based differentiation technique with automatic differentiation does not depend on the number of decision variables like the finite difference method that depends linearly on the number of decision variables. Moreover, the finite difference method typically suffers from numerical inaccuracies near the optimal solution. Hence, the solution process requires more objective function evaluations and, still, the desired accuracy can not always be reached. Typically, automatic differentiation is able to produce numerically accurate gradients, and numerical inaccuracy problems are rare.

4 Industrial example

In this section, we present an industrial example of a multidisciplinary and multiobjective optimization problem, where we demonstrate how the issues raised can be realized efficiently. In other words, we use the dynamic generation of the simulation model and the model-based differentiation. We consider a multiobjective optimization problem related to paper quality, where the simulation model contains several submodels. A paper machine expert participates in the interactive solution process as a decision maker. First, we describe a paper making process shortly, and then introduce its modeling aspects. Thereafter, we present an interactive solution process of the optimization problem.

The operational and numerical results presented below do not correspond to any existing machine construction or setup. We are not here allowed to express all the detailed information about the simulation model used in computations.

4.1 Paper making process

The paper making process in a paper machine begins from a headbox, where about one percent of substance is dry solids and the rest is water. After several phases of pressing and evaporation, paper is ready to be wound onto a roll, where it is taken to the finishing part for improving the paper surface properties. An example of a paper machine is given in Fig. 3.

The paper making process consists of four main parts in the paper machine and then chosen finishing components. In the figure, paper making begins from the upper right corner from the *headbox*, where the stream of fibre suspension is lead to a *wire section* (also known as a forming section), where the suspension forms a paper web via dewatering. The construction of the wire section differs widely depending on what kind of paper we are manufacturing. However, in all the machines the dominant mechanism in the web forming is filtration. After the filtration, the paper web has a porous structure, where the pores are filled with water. In this stage, a fifth of the paper web consists of fibres. A part of water located in the pores is removed by squeezing the paper web between two pressing rolls. This stage is just after the wire section and it is called a *press section*. The remaining water is removed in a *drying section* by evaporation. This is done by contacting the paper web with a series of steam-heated cylinders. The drying section is typically quite large part as seen in the figure. After the drying section the production continues in a *finishing part*, where the paper is treated to obtain a smoother and glossier surface. The finishing part can be located next to the drying section, so-called on-line finishing or paper can be first wound on rolls and then taken to off-line finishing as presented in Fig. 3. Examples of finishing methods are coating and calendering. For more about paper making, see, for example, [17].

4.2 Modeling and optimizing of paper making line

We have several different types of submodels to describe different phases of paper making in the paper machine. The submodels are stored in a model database, where we

Table 1 The CPU-time comparison

Method	Number of variables			
	10	20	50	100
Model-based differentiation with automatic differentiation (AD)	5.160	5.230	5.230	5.240
Finite difference method	45.060	90.423	228.310	460.190

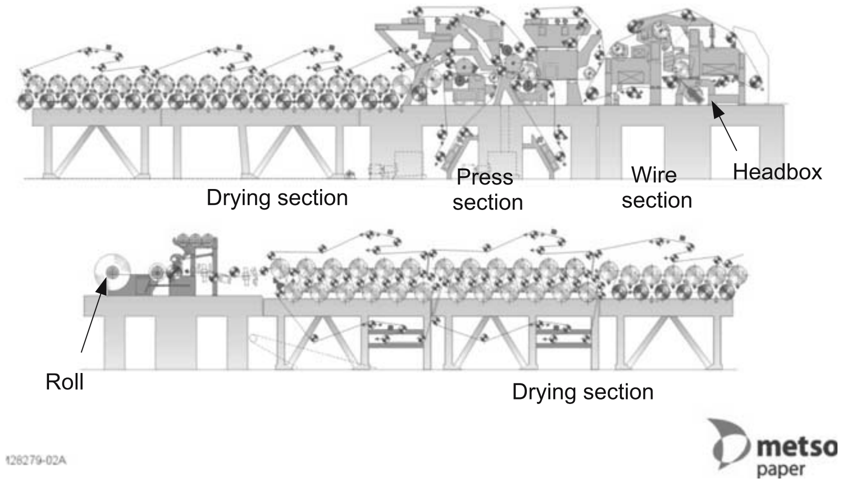

Fig. 3 Example of paper machine (Courtesy of Metso Paper Inc.)

can choose for each problem to be considered a chain of suitable submodels to form a simulation model. When we combine these submodels together, we obtain a multidisciplinary model for the whole machine. We employ submodels with partial differential equations describing physical phenomena in some machine components, the so-called unit-process models. For example, for the fluid flows in the headbox, we employ Navier–Stokes type equations [14, 23]. Furthermore, we have statistical models that are based on experimental data from the paper making process describing both process and quality properties. Examples of statistical submodels are qualitative properties that are determined during the paper making process. As submodels we also have model surrogates that simplify computationally demanding submodels. An example of a model surrogate is a polynomial submodel simplifying a physical wet pressing model [16]. In addition, the database contains parallel submodels describing the same phenomenon, of which different simulation model constructions can be assembled, according to the actions of the user. Most of the submodels have been implemented with Fortran 95, but some unit-process models have been implemented with Fortran 77. In addition, such programming languages as C, C++ and Python have been applied in the system implementation.

As the optimization software, we employ the interactive multiobjective optimization system NIMBUS [13, 20, 21, 27] that is available for academic purposes on the Internet (http://nimbus.it.jyu.fi). The NIMBUS method is based on an idea of classification of objective functions into five classes. At each iteration the decision maker directs the solution process according to her/his preferences in the set of Pareto optimal solutions. (S)he is shown the current solution and the approximated ranges of the Pareto optimal set, and asked about the desirable changes to the objective function values by classifying them. The first class contains functions to be improved from their current level, the second class functions to be improved until a given desirable level, the third class functions that are satisfactory at the moment, the fourth class functions that are allowed to get worse values, but not exceeding a given bound and the last (fifth) class contains functions that are allowed to change freely. Because of the definition of Pareto optimality, we cannot improve all the objective functions simultaneously and, thus, some of the objective functions must be allowed to impair in the classification. Based on preference information, NIMBUS produces single objective optimization problems, so-called scalarized subproblems, which can be solved with a chosen single objective optimizer. There are four different scalarized subproblems in NIMBUS at the moment and therefore, NIMBUS can produce one to four new Pareto optimal solutions according to the classification information [27]. This means that the decision maker can ask for 1–4 new solutions. In addition, the decision maker is able to produce one or several intermediate solutions between any two different solutions with NIMBUS. In this way the decision maker can get to know the problem and concentrate on such Pareto optimal solutions (s)he is interested in.

We present here one example construction of a simulation model that is used to mimic SC-paper (Super-Calendered paper) production. In this example, we utilized all four scalarized subproblems in each classification. We used with NIMBUS a subgradient based single objective optimizer [28] and produced gradient information with model-based differentiation and automatic differentiation, where the latter implementation is based on the Numerrin 2.0 software package [29, 30].

All the computations were carried out on the server computer containing dual-processor system AMD Athlon (tm) XP processor 1900+ (1,600 MHz) processor and 2 GB memory.

4.3 Multiobjective optimization of paper making line

In this example, we follow the system architecture described earlier. The problem contains five objective functions chosen by the decision maker and in order to optimize the chosen objectives the corresponding simulation model is generated of 21 submodels and one transformation mapping. There are over 30 submodels available in the model databases, where the simulation model was generated. The simulation model is constructed according to our system architecture with the help of databases that contain information about submodels related to paper making and properties of paper. The obtained simulation model is able to mimic the chosen qualitative properties of the produced paper by a virtual paper making line.

The chosen objective functions describe qualitative properties of produced paper at the finishing part of the paper machine. However, to be able to simulate these properties, we need also information of paper properties before the finishing part. Therefore, the simulation model is a model for the whole paper machine including a headbox, a wire section, a press section, a drying section and a finishing component called a supercalender. In this example, the following five qualitative properties are studied. The first and the second objective functions describe a so-called PPS 10 property on both top and bottom sides of paper, and their values are to be minimized. PPS 10 describes roughness of paper surface. The third and the fourth objective functions represent gloss of paper surface also on both top and bottom sides of paper, and their values are in this example maximized. The fifth function describes the final moisture in the paper and its value is maximized. There are altogether 22 decision variables, which are typical controls of the paper machine components that have an effect on the chosen objective functions. The problem contains box constraints for the decision variables, but it does not have any inequality or equality constraints. The decision maker was able to get four solutions for every classification, but we do not show them all here (for saving space), but only those solutions the decision maker found the most desirable.

Table 2 Initial values and the approximated ranges of the Pareto optimal set

Name	Minimized/ maximized	Lowest values	Initial solution	Highest values
Top PPS 10	Minimized	0.92	1.20	2.00
Bottom PPS 10	Minimized	1.03	1.29	2.02
Top gloss	Maximized	0.55	1.09	1.11
Bottom gloss	Maximized	0.51	0.99	1.15
Final moisture	Maximized	0.08	1.88	2.52

In this optimization problem, the interactive solution process was guided by preference information of a paper machine expert, who was acting as a decision maker. In Table 2, we collect information about the names of the objective functions, whether they were to be minimized or maximized and give approximated information about their ranges in the Pareto optimal set. At the beginning, objective functions had the (normalized) initial values presented also in Table 2. The initial solution was produced with typical values of the decision variables and it was projected on the Pareto optimal set by NIMBUS. In addition, the ranges of the Pareto optimal set were produced by NIMBUS and their values were updated during the solution process, whenever smaller or higher Pareto optimal function values were obtained during the solution process.

Throughout the optimization process, the decision maker had the following aims. He wanted to obtain such a solution in which both PPS 10 and gloss would be as equal as possible on the top and bottom sides of paper. In addition, he found the minimization of PPS 10 very important.

First classification In the first classification, the decision maker considered the initial solution and wanted to improve both top and bottom values of PPS 10 and equalize differences between the values of the gloss on top and bottom sides of paper. Therefore, the bottom gloss was set to be improved until a desired level 1.0 and to obtain similar values for both the glosses, he let the top gloss to get worse values and gave it a lower bound 1.0. Furthermore, he let the final moisture change freely. The decision maker obtained new solutions and decided to produce new intermediate solutions between the original initial solution and one of the obtained solution. Unfortunately, he did not obtain improvements he wanted and, therefore, the decision maker chose to go back to the initial solution and classify the objective functions again.

Second classification Based on knowledge obtained from the first classification, the decision maker decided to make a new classification to improve the PPS 10 properties and equalize the gloss on top and bottom. Differing from the first classification, he gave both the PPS 10 values desired levels, which were 1.15 for both. As in the first classification, the bottom gloss was set to be improved until a desired level 1.0 and the top gloss was given a lower bound 1.0. The final moisture was still allowed to change freely. In this way, objective functions

Table 3 Intermediate solutions

Name	Five intermediate solutions						
Top PPS 10	1.20	1.11	1.13	**0.94**	0.86	0.79	0.82
Bottom PPS 10	1.29	1.23	1.20	**1.15**	1.12	1.07	1.03
Top gloss	1.09	1.09	1.09	**1.09**	1.09	1.09	1.09
Bottom gloss	0.99	1.01	1.04	**1.06**	1.09	1.11	1.14
Final moisture	1.88	1.55	1.21	**0.89**	0.56	0.23	0.10

had more freedom to change. In consequence, the decision maker obtained new solutions that were almost satisfactory. He decided to produce five intermediate solutions between the initial solution and one of the obtained solutions. The new solutions are given in Table 3, where after the name of the objective function, we list the starting point of the classification and in the right, the chosen solution after the classification. Between these two, we give the five intermediate alternatives.

As seen in the table the lowest value of the top PPS 10 was updated with value of 0.79.

Then, he obtained desired improvements and chose the third intermediate solution (that is expressed in bold face in the table) as the starting point of a new classification.

Third classification The decision maker wanted to equalize differences between top and bottom side of paper for the both PPS 10 properties and glosses. Therefore, he let the top PPS 10 and top gloss values to get worse by given them an upper and a lower bound, respectively. The given bounds were 1.0 for both. Then, the bottom PPS 10 value was to be improved as much as possible, the bottom gloss value was found satisfactory at the moment and the final moisture was allowed to change freely. With this classification, even though both the PPS 10 and gloss values were impaired, the decision maker found that the solution obtained was satisfactory for PPS 10 from the point of view of equality on bottom and top sides. The objective vector had the following values (1.24, 1.27, 1.05, 0.95, 1.93) after the third classification. However, the objective function values were still too large and needed improvement.

Fourth classification This time, the decision maker wanted to improve both PPS 10 properties so that difference between the paper sides would not be increased. In addition, he wanted the bottom gloss value to obtain a larger value to equalize the difference between paper sides. The final moisture was given an upper bound 1.0. From the fourth classification the decision maker obtained a solution that was satisfactory to be the final solution. The solution was (1.01, 1.04, 1.07, 1.09, 1.19) and it is also presented in Table 4 together with the solutions selected after each classification. Thus, we can see how the solution process progressed and how the final solution fulfills the decision maker's requirements concerning the equality of paper quality on top and bottom sides of paper.

The above-described optimization process did not only improve the process and quality parameters, but can also give new knowledge and insight of the paper making process. The interactive multiobjective optimization made possible for the decision maker to learn about the conflicting qualitative properties and their interrelationships. The implementation (utilizing the issues discussed in previous sections) gave a completely new perspective to paper making when the paper making process could be considered as a whole. In addition, the example gives us information about the models used and feedback whether they represent the real processes well enough or whether some further development is needed. This numerical example clearly proves that the system architecture presented is useful and that it is able to solve complex real world problems.

5 Concluding remarks

In this paper, we have collected issues necessary in developing a multidisciplinary and multiobjective optimization system. We have introduced an approach for a generic optimization model architecture, which fixes neither the optimization method nor the application field. Moreover, the system architecture allows the use of different types of submodels, like PDEs and statistical models based on experimental data. The main emphasis was given to the consecutively coupled submodels.

We discussed a couple of important aspects of the system architecture and their computer realization. These are features that enable efficient computations and, for example, make possible to use interactive multiobjective optimization. We introduced a model-based differentiation technique and dynamic generation of the simulation model. The first-mentioned allows us to employ gradient-based optimizers in multidisciplinary optimization, and by combining it with automatic differentiation, we are able to realize accurate and efficient gradient computations. We illustrated this with a comparison, where the results obtained showed that the model-based differentiation together with automatic differentiation was highly more efficient and accurate than the corresponding finite difference approach. Dynamic generation of the simulation model is a novel way to organize and manage unit-process submodels associated to a real world system. It also permits more efficient simulations and optimizations, since the system

Table 4 The results of optimization process

Name	Minimized/ maximized	Initial solution	Second class solution	Intermediate solution	Third class solution	Final solution
Top PPS 10	Minimized	1.20	0.82	0.94	1.24	1.01
Bottom PPS 10	Minimized	1.29	1.03	1.15	1.27	1.04
Top gloss	Maximized	1.09	1.09	1.09	1.05	1.07
Bottom gloss	Maximized	0.99	1.14	1.06	0.95	1.09
Final moisture	Maximized	1.88	0.10	0.89	1.93	1.19

uses only those submodels that are required to produce desired simulation outputs or the optimized objectives.

Finally, we illustrated the ideas and the architecture with an industrial example, a complicated multidisciplinary and multiobjective optimization problem. The example was related to paper making technology. In this context, we presented an optimization process with the interactive NIMBUS method, where the paper machine expert solved a multidisciplinary and multiobjective problem. The solutions obtained showed that the decision maker was able to find a satisfactory compromise between the conflicting objective functions. Besides that, the decision maker was able to learn about the problem during the optimization process. The example showed that the system architecture presented in this paper can be used in solving complex real world problems. We can say that the new system covering the whole paper machine gives revolutionary possibilities to control the processes as a whole when different interdependences can be taken into account. In addition, it gives possibilities to develop optimization-based decision support systems for paper making professionals. Similar benefits can be expected to be obtained in a variety of application areas where only parts of the complex systems have been studied so far.

Acknowledgements This research was financially supported by the National Technology Agency of Finland, project: NIMBUS—multiobjective optimization in product development. The authors wish to thank Doctor Heikki Kettunen from Metso Paper, Inc., Professor Jari P. Hämäläinen from the University of Kuopio and Docent Marko M. Mäkelä from the University of Jyväskylä.

References

1. Shubin GR (1995) Application of alternative multidisciplinary optimization formulations to a model problem for static aeroelasticity. J Comput Phys 118:73–85
2. Alexandrov NM, Hussaini MY (eds) (1997) Multidisciplinary design optimization. In: State of the Art. SIAM Publications, Philadelphia
3. Giunta AA (1999) Sensitivity analysis method for aeroelastic aircraft models. Aircraft Des 2:207–230
4. Olds J (1994) System sensitivity analysis applied to the conceptual design of a dual-fuel rocket SSTO. In: 5th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization, AIAA paper 94-4339, Panama City
5. Walsh JL, Townsend JC, Salas AO, Samareh JA, Mukhopadhyay V, Barthelemy J-F (2000) Multidisciplinary high-fidelity analysis and optimization of aerospace vehicles part 1: formulation. AIAA paper 2000-0418, NASA Langley Research Center
6. Walsh JL, Weston RP, a Samareh JA, Mason BH, Green LL, Biedron RT (2000) Multidisciplinary high-fidelity analysis and optimization of aerospace vehicles part 2: preliminary results. In: 38th aerospace sciences meeting and exhibit, AIAA paper 2000-0419, Reno, Nevada
7. Tappeta RV, Renaud JE, Rodríguez JF (2002) An interactive multiobjective optimization design strategy for decision based multidisciplinary design. Eng Optim 34(5):523–544
8. Kodiyalam S, Sobieszczanski-Sobieski J (2001) Multidisciplinary design optimization—some formal methods, framework requirements, and application to vehicle design. Int J Vehicle Des 25:3–22
9. Salas AO, Townsend JC (1998) Framework requirements for MDO application development. In: 7th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization, AIAA paper 98-4740, St. Louis
10. Sobiesczanski-Sobieski J, Haftka RT (1997) Multidisciplinary aerospace design optimization: survey of recent developments. Struct Optim 14(1):1–23
11. Cramer EJ, Dennis JE Jr, Frank PD, Lewis RM, Shubin GR (1994) Problem formulation for multidisciplinary optimization. SIAM J Optim 4(4):754–776
12. Sobieszczanski-Sobieski J (1990) Sensitivity of complex, internally coupled systems. AIAA J 28(1):153–160
13. Miettinen K (1999) Nonlinear Multiobjective Optimization. Kluwer, Boston
14. Hämäläinen J (1993) Mathematical modeling and simulation of fluid flows in the headbox of paper machines. PhD Thesis, University of Jyväskylä
15. Hämäläinen JP, Miettinen K, Tarvainen P, Toivanen J (2003) Interactive solution approach to a multiobjective optimization problem in a paper machine headbox design. J Optim Theory Appl 116(2):265–281
16. Hiltunen K (1995) Mathematical and numerical modeling of consolidation processes in paper machines. PhD Thesis, University of Jyväskylä
17. Gavelin G (1998) Paper machine design and operation: descriptions and explanations. Angus Wilde Publications, Vancouver
18. Madetoja E, Tarvainen P (2003) A computer realization of multidisciplinary optimization system. Reports of the Department of Mathematical Information Technology, series B. Scientific Computing, vol 14, University of Jyväskylä
19. Giordano FR, Weir MD, Fox W (1997) A first course in mathematical modeling, 2nd edn. Brooks/Cole Publishing Company, California
20. Miettinen K, Mäkelä MM (1995) Interactive bundle-based method for nondifferentiable multiobjecive optimization: NIMBUS. Optimization 34:231–246
21. Miettinen K, Mäkelä MM (2000) Interactive multiobjective optimization system WWW–NIMBUS on the Internet. Comput Oper Res 27(7–8):709–723
22. Martins JRRA, Alonso JJ, Reuther JJ (2005) A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. Optim Eng 6(1):33–62
23. Madetoja E (2003) On interactive multiobjective optimization related to paper quality. Licentiate Thesis, University of Jyväskylä
24. Madetoja E, Mäkelä MM (2006) On sensitivity analysis of nonsmooth multidisciplinary optimization problems in engineering process line applications. Struct Multidisciplinary Optim 31(5):355–362
25. Barthelemy J-FM, Hall LE (1995) Automatic differentiation as a tool in engineering design. Struct Des 9:76–82
26. Haslinger J, Mäkinen RAE (2003) Introduction to shape optimization: theory, approximation, and computation. SIAM, Philadelphia
27. Miettinen K, Mäkelä MM (2006) Synchronous approach in interactive multiobjective optimization. Eur J Oper Res 170(3):909–922
28. Mäkelä MM, Neittaanmäki P (1992) Nonsmooth optimization: analysis and algorithms with applications to optimal control. World Scientific, Singapore
29. Hiltunen K, Laitinen M, Niemistö A, Tarvainen P (2003) Using mathematical concepts in software design of computational mechanics. In: Råback P, Santaoja K, Stenberg R (eds) VIII Finnish Mechanics Days, Espoo. Helsinki University of Technology, Laboratory for Mechanics of Materials
30. Numerrin 2.0 (2003) Simulation software based on finite element method (in Finnish), 2003. Numerola Oy