TOOL SUPPORT FOR PERFORMANCE MODELING AND OPTIMIZATION

Michael Syrjakow*, Elisabeth Syrjakow* and Helena Szczerbicka**

**Institute for Computer Design and Fault Tolerance*

University of Karlsruhe, 76128 Karlsruhe, Germany

{syrjakow, lisa}@informatik.uni-karlsruhe.de

***Institute for Computer Science, Fachbereich Mathematik und Informatik*

University of Hanover, 30167 Hanover, Germany

hsz@informatik.uni-hannover.de

ABSTRACT

Most of the available modeling and simulation tools for performance analysis do not support model optimization sufficiently. One reason for this unsatisfactory situation is the lack of universally applicable and adaptive optimization strategies. Another reason is that modeling and simulation tools usually have a monolithic software design, which is difficult to extend with experimentation functionality. Such functionality has gained on importance in recent years due to the capability of an automatic extraction of valuable information and knowledge out of complex models. One of the most important experimentation goals is to find model parameter settings, which produce optimal model behaviour. In this paper we elaborate on the design of a powerful optimization component and its integration into existing modeling and simulation tools. For that purpose we propose a hybrid integration approach being a combination of loose document-based and tight invocation-based integration concepts. Beside the integration concept for the optimization component we also give a detailed insight into the applied optimization strategies.

Key Words: Performance Modeling, Petri Nets, Model Optimization, Knowledge and Information Management.

1. INTRODUCTION

Complexity of computer software is constantly growing, both in the size of developed systems, and in the intricacy of its operations. This general observation particularly applies to modeling and simulation tools, which have grown enormously over the past decades. Today the most prominent approaches to master the complexities of large-scale software development are object-orientation and component technology. Component approaches being usually built up on object-orientation concentrate design efforts on defining interfaces to pieces of a system and describing an application as the collaborations that occur among those interfaces. Implementers of a component can design and build the component in any appropriate technology as long as it supports the operations of the interface and is compatible with the component execution environment. For that reason the interface is focal point for all analysis and design activities of component-based software development (Szyperski, 1999; Brown, 2000). Component technology has also deeply influenced the area of computer simulation. Here we

can distinguish two main fields of activity: component-oriented development of simulation models and component-oriented development of modeling and simulation (M&S) tools.

For a component-oriented development of distributed simulation models the US Department of Defense (DoD) Modeling and Simulation Office (DMSO) has adopted a global standard called High Level Architecture (Kuhl *et al*, 2000). In contrary to the area of component-oriented development of simulation models where a standard is available today and where a variety of research activities can be observed, the field of component-oriented development of M&S tools yet remains rather untouched. This is a very unsatisfactory situation because many M&S tools still have a monolithical software design which is difficult to maintain and to extend and which doesn't correspond any more to the modern distributed Web-centered technologies of today. In order to illustrate this unsatisfactory situation more detailed we take a look at some existing and widely used M&S tools. We focus on Petri Net tools because they are a quite suitable example to explain the disadvantages of a monolithical software design. It should be mentioned for fairness that these observations also apply to other prominent classes of M&S tools for example Queuing Network tools.

Having surveyed the software architecture of existing Petri Net tools in Chapter 2 a hybrid integration approach for legacy M&S tools based on a component architecture is presented in Chapter 3. Chapter 4 focuses on the architecture and implementation of a universally applicable optimization component. Finally, in Chapter 5 we summarize and draw some conclusions.

2. DISADVANTAGES OF CURRENT ARCHITECTURES OF PETRI NET BASED PERFORMANCE MODELING TOOLS

More than 100 different Petri Net tools are available today. A comprehensive and up-to-date database can be found at http://www.daimi.au.dk/PetriNets/tools/. Altogether these tools offer 76 different graphical Petri Net editors, 50 different token game animations, 52 different implementations for structural analysis, and 39 different implementations for performance analysis. This variety in fact is not bad because it opens many possibilities to deal with Petri Nets. The monolithical software design however makes it almost impossible to combine for example an outstanding Petri Net evaluation module from one tool with a nice graphical Petri Net editor from another tool. Beyond that, all these tools are difficult to maintain and to extend. Another significant disadvantage is the lack of interoperability. A user who has edited a Petri Net with one tool usually cannot analyse this Petri Net with another tool. The reasons for that incompatibility are the following: Every Petri Net tool uses its own proprietary file format and often supports only a specific type of Petri Nets. To overcome this unsatisfactory situation international standards are going to be established regarding

- a mathematical semantic model, an abstract mathematical syntax, and a graphical notation for High-Level Petri Nets. The standards group of the International Organization for Standardization (ISO) relevant for the Petri Nets standardization effort is called ISO/IEC JTC1/SC7/WG11. An overview of the current activities of that group is available at http://www.daimi.au.dk/PetriNets/standardisation/#sc7resources.
- a general Petri Net interchange format that supports all features of existing and forthcoming Petri Net tools. An overview of the ongoing standardization efforts of an XML-based Petri Net interchange format is given in Section 3.1.
- a component architecture for M&S tools. In addition to the two standards mentioned above appropriate component architecture for M&S tools is of great importance.

One main focus of this paper is a component architecture for performance M&S tools, which will be described in detail in the following Chapter.

3. TOWARDS A COMPONENT-ORIENTED DESIGN OF PERFORMANCE MODELING TOOLS

In contrary to the HLA, which provides with the RTI a very demanding infrastructure for a tight coupling of highly interdependent simulation components component architecture for M&S tools should support a much looser component coupling. This is justified because M&S tools usually consist of a very limited number of quite self-sufficient and coarse-grained components. From the user's point of view usually the following software parts can be identified within an M&S tool:

- Model editor: a model editor allows the modeller to edit new and to modify existing models. We can distinguish textual and graphical editors. Modern Web-based modeling tools may allow collaborative online editing of models. A model editor basically can be realized as an independent stand-alone component. Its output is a model description in a specific description format, which is characterized by the supported modeling technique.
- Model analysis/evaluation modules: these modules are used to analyse/evaluate models generated by the model editor. In case of High-Level Petri Nets (Jensen, 1991) we can distinguish between a mathematical analysis of structural properties (place-invariants, transition-invariants, boundedness, etc.) and performance evaluations (stationary or transient analyses). Performance evaluation can be computed either analytically or by (discrete-event) simulation. An evaluation module may also provide some animation features, e.g. a token game animation in case of Petri Nets.
- Experimentation modules: these modules are optional. They allow goal-driven experimentation with a model, for example to find optimal parameter settings, to determine sensitive model parameters, to perform a model validation, etc. To fulfil all these tasks usually a lot of model evaluations (experiments) are required.

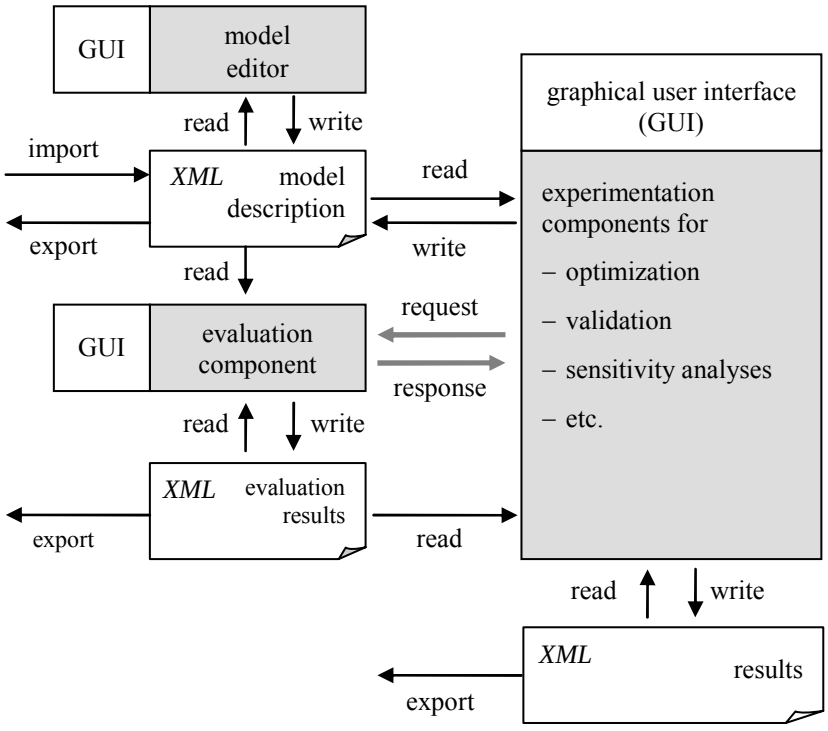


Figure 1. M&S tool components and their interdependencies

Fig. 1 shows the different M&S tool components and their interdependencies. As we have described above the collaboration of these components is based on two kinds of interactions: exchange of documents and invocation of model evaluation functionality. For that reason an obvious and pragmatic integration approach for M&S tool components is a hybrid one being a combination of loose document-based and tight invocation-based integration techniques. For remote invocations universal component 'wiring' standards like CORBA (Common Object Request Broker Architecture), RMI (Remote Method Invocation) or DCOM (Distributed Component Object Model) can be used. A specialized standard like the HLA, which focuses on the specific requirements of tightly coupled simulation models (federation management, time management, etc.) is not needed in this case. For document-based integration standardized document interchange formats are required. Today, the most promising ones are XML-based approaches.

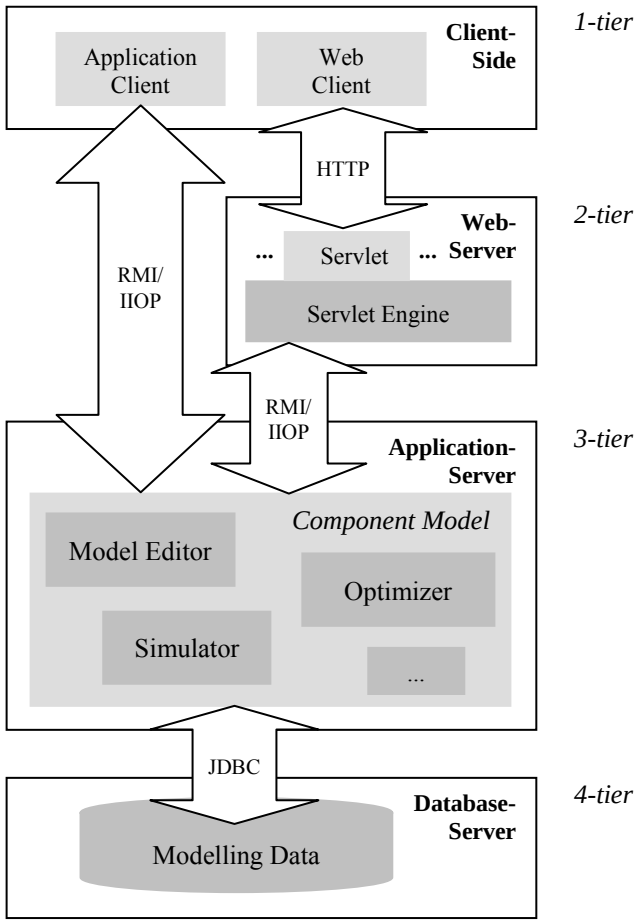


Figure 2. Example realization of a component-oriented M&S tool based on a distributed 4-tier architecture

Advantages of the hybrid integration approach described above are manifold:

1.) It enables a flexible distribution of the involved components within a computer network.
2.) It allows user access by traditional application clients or by Java-based Web clients.
3.) It enables an easy integration of existing monolithical tools as a whole by transformation of the proprietary model description format into a standardized XML-based format or partially by appropriate component wrappers.

4.) It considerably simplifies tool modifications and extensions (for example to achieve HLA compliance).
5.) It represents a good basis for agent-based approaches.
6.) Beside all these technical advantages component-orientation opens several economic and organizational advantages (software reuse, clear separation of concerns, etc.).

Fig. 2 shows an example realization of a component-oriented M&S tool based on a modern distributed 4-tier architecture. The first tier contains client components, which allow access (Web- or application-based) to server components residing on the other tiers behind. The application server contains the M&S tool components shown in Fig. 1. For their component-oriented realization several component models can be applied for example J2EE/EJB, CCM (CORBA Component Model) or (D)COM/COM+ ((Distributed) Component Object Model). Persistent modeling data are saved on a database-server representing the fourth tier of the distributed architecture.

In the following two Sections we will explain more detailed two important sub areas of our approach: 1.) An XML-based interchange format for models of a specific modeling technique (in our case High-Level Petri Nets) and 2.) experimentation components allowing the modeller to automatically extract information about the behaviour of complex simulation models. The presented methods and concepts have been already successfully used for the prototypical realization of a component-oriented Stochastic Petri Net M&S tool (Syrjakow *et al*, 2002; Syrjakow, 2003). A detailed description of Stochastic Petri Nets (SPN) being a particular type of High-Level Petri Nets can be found in (Lindemann, 1998).

3.1. An XML-Based Model Interchange Format for High-Level Petri Nets

The idea of a standardized interchange format for Petri Nets is not new. However, the attempts to define such a standard file format were not very successful in the past. The main reasons for that are the following:

1.) Each Petri Net tool usually supports a particular version of Petri Nets.
2.) As a consequence each Petri Net tool provides a specific file format, which solely satisfies the needs of the supported Petri Net type.
3.) The lack of appropriate description techniques being flexible enough to cover both the common essence of all existing Petri Net types and beyond that, the specific features of any particular Petri Net extension.

Recently however, the idea of a standardized Petri Net interchange format got a new boost due to the availability of the Extended Markup Language (XML). Today XML seems to have the power to become a major means for a homogeneous and standardized exchange of information. XML allows the specification of specialized markup languages for the convenient exchange of information in specific areas of research or business. Examples of recent markup languages based on XML are the Chemical Markup Language (CML), the Mathematical Markup Language (MathML) or the Astronomical Instrument Markup Language (AIML).

In the area of Petri Nets several research groups are currently working on an XML-based model interchange format which of course should be based on the ISO/IEC Petri Net standard. Beyond that, this format should be generic and extensible to be able to cover all existing and forthcoming Petri Net classes. A preliminary proposal for such an interchange format can be found in (Jüngel *et al*, 2000). The proposed format consists of two parts:

1.) A general part called Petri Net Markup Language (PNML), which captures the common features of all existing Petri Net versions.
2.) A specific part called Petri Net Type Definition (PNTD), which allows specifying additional features. This part is of great importance because it provides openness for future Petri Net extensions.

We have proposed a PNTD for Stochastic Petri Nets in Syrjakow and Syrjakow (2003). An overview of the ongoing standardization efforts of an XML-based Petri Net interchange format can be found at http://www.oasis-open.org/cover/xmlAndPetriNets.html.

As shown in Fig. 1 XML-based description formats for models and modeling results are an integral part of our proposed M&S tool architecture. They allow a simple document based integration of tool components, which is usually much easier to realize than invocation-based approaches. Beyond that, existing monolithically designed M&S tools can be easily integrated into our architecture without any expensive software modifications. For that purpose only appropriate file converters (C) have to be realized being able to convert the proprietary file formats of the legacy tools into the XML-based model interchange format (see Fig. 3). This has been proven to be a very simple and efficient way to achieve compatibility between several legacy M&S tools allowing the mutual use of parts (editors, evaluation components, etc.) of them.

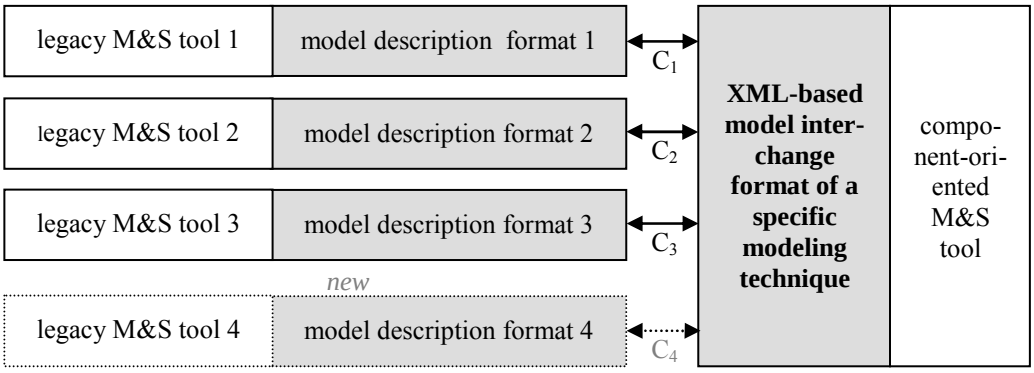


Figure 3. Necessary format conversions with an XML-based model interchange format

As shown in Fig. 3 for the integration of a new legacy tool the realization of only one additional file converter is required. Without such a standardized interchange format the number of required file converters would not increase linearly but quadratically. As indicated in Fig. 4 for n different file formats $(n^2-n)/2$ file converters would be required to achieve compatibility between the n corresponding M&S tools.

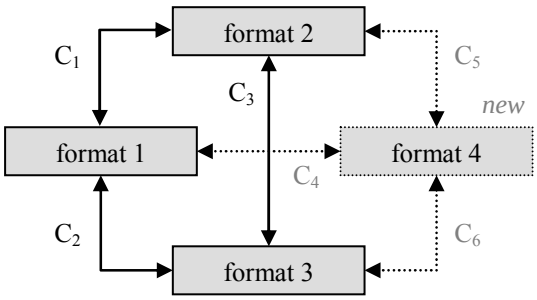


Figure 4. Necessary format conversions without an XML-based model interchange format

3.2. Experimentation Components

M&S tool developers often neglected or in the worst case just ignored experimentation components in the past. This was mainly caused by the monolithical software design of the existing M&S tools, which made a later integration of additional experimentation functionality rather intricate and expensive. With the enormous increase of model complexity however

these components have gained great importance because experimentation goals like finding optimal or sensitive model parameters cannot be reached by hand any more. Following our hybrid integration approach it is very easy to provide an existing M&S tool with additional functionality for experimentation, which can be used to automatically extract valuable information and knowledge out of complex models. In the following, we take a detailed look at a parameter optimization component, which provides efficient and universally applicable methods to optimize the behaviour of complex simulation models.

4. TOOL SUPPORT FOR MODEL OPTIMIZATION

4.1. Introduction to Model Optimization

This Section gives a brief introduction to the fascinating field of model optimization. In the following, the instance of an optimization problem is formalized as a pair (S,F). The solution space S denotes the set of all possible problem solutions. The goal function F, which has to be optimized, is a mapping defined as $F : S \rightarrow R$. In this paper we focus on parameter optimization problems where the search space is a subset of R^n ($S \subset R^n$) and the goal function is defined through a performance model, which is analysed by discrete event simulation. Such a goal function is called simulation-based goal function in the following.

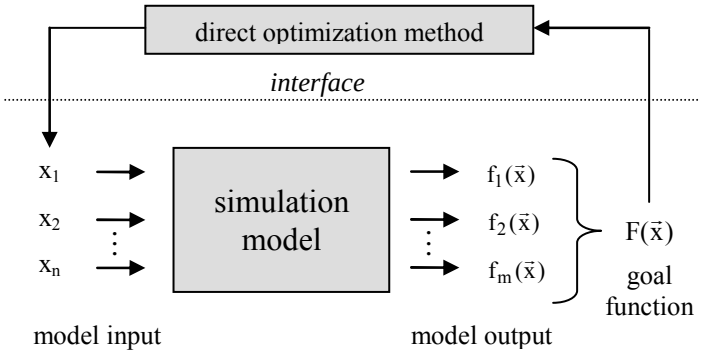


Figure 5. Parameter optimization of a simulation-based goal function

As shown in Fig. 5 a performance model maps a vector $\vec{x} = (x_1, x_2, ..., x_n), x_i \in R, i \in \{1,..., n\}$ of model input parameters onto several model outputs $f_j(\vec{x}), j \in \{1,...,m\}$. Often the relation between input and output of a performance model is so complex that it cannot be described by mathematical expressions any more. In this case the performance model represents a so-called "black-box" system.

A function $f : S \subset R^n \rightarrow R^m$ that is defined through a model is referred to as model function. In case of a performance model the model inputs can be roughly classified into system and workload parameters. The model outputs describe the system behaviour (performance, reliability, consumption of resources, etc.). As indicated in Fig. 5 the goal function F may be either one or a composition of several model outputs. The formulation of F usually is rather difficult, especially if contradictory goals are involved. Very frequently F is defined as a weighted sum of model outputs

$$F(\vec{x}) = \sum_{k=1}^{m} \omega_k \cdot f_k(\vec{x}), \ \omega_k \in R . \quad (1)$$

The overall goal of optimizing a simulation-based goal function is to find a parameter vector $\vec{x}^* \in S$ which satisfies:

$$\forall \vec{x} \in S : F(\vec{x}) \circ F(\vec{x}^*) = F^*, \text{ with } \circ \in \{\leq, \geq\} . \quad (2)$$

A solution $\vec{x}^*$ is called global optimum point. The goal function value $F(\vec{x}^*)=F^*$ is referred to as global optimum of F. Beside global optimum points there may exist local optimum points $\vec{x}^\wedge$, having the property that all neighbouring solutions have the equal or a worse goal function value. A local optimum $F^\wedge = F(\vec{x}^\wedge)$ is defined as follows:

$$\exists \varepsilon > 0, \forall \vec{x} \in S : \left\| \vec{x}-\vec{x}^\wedge \right\| < \varepsilon \Rightarrow F(\vec{x}) \circ F(\vec{x}^\wedge)=F^\wedge, \text{ with } \circ \in \{\leq, \geq\}. \quad (3)$$

Goal functions with several global and/or local optimum points are called multimodal functions. An optimization problem is either a minimization ($\circ = \geq$) or a maximization ($\circ = \leq$) problem. Minimization problems can be easily transformed into maximization problems and vice versa, because $\min\{F(\vec{x})\} = -\max\{-F(\vec{x})\}$.

Optimization of a simulation-based goal function has been proven to be a demanding task. The main challenges are the following:

- Black-box situation: Usually the relation between input and output of performance models being analysed by discrete event simulation cannot be described mathematically. For that reason classical mathematical optimization methods, which require analytical information like gradients or other problem specific knowledge, are not applicable any more.
- Expensive model evaluation process: Evaluation of a simulation model usually requires a lot of computation time which in practice may last from several minutes until many hours or even days. For that reason the optimization process should only require a very limited number of simulation runs (goal function evaluations) to reach the optimization goal.
- Inaccurate simulation results: The model outputs of probabilistic performance models being evaluated by discrete event simulation may be considerably distorted by stochastic inaccuracies. For that reason the applied optimization methods should be robust against inaccurately evaluated goal function values.
- High dimensional search space with complex parameter restrictions
- Multimodal goal function with many local and/or global optimum points

Summing up, for optimization of simulation models methods are required, which first of all are able to deal with the black-box situation. For that reason only optimization methods are applicable which solely use goal function values to guide the optimization process (blind search). Methods with this property are called direct optimization methods. As shown in Fig. 5 direct optimization methods work iteratively. A parameter vector $\vec{x}$ generated by the direct optimization process is passed on to the simulation process, where a simulation tool evaluates the optimized simulation model. Afterwards, the calculated goal function value $F(\vec{x})$ is sent back to the optimization process. Outgoing from $F(\vec{x})$, a new parameter vector is generated, which is in turn transferred to the simulation process. This way, the goal function is improved step by step until a termination condition is fulfilled. Because the evaluation of a simulation-based goal function usually requires considerable computational resources the optimization goal should be reached with a minimum number of iteration steps.

Genetic Algorithms (Goldberg, 1989; Michalewicz, 1992), Evolution Strategies (Schwefel, 1981), and Simulated Annealing (Aarts and Korst, 1989) are the most common and powerful direct methods for global optimization today. All these methods apply probabilistic search operators which imitate principles of nature. Although these operators have been proven to be well-suited for global search the required computational effort (number of goal function evaluations) and the quality of the generated optimization results still remain a big problem. In the following, an approach to further improvement of direct optimization methods is presented. Our considerations are restricted to global optimization of continuous parameter optimization

problems. In order to make direct optimization more efficient and to achieve high quality solutions, we propose a combination of existing global and local optimization methods. The structure of the resulting combined 2-phase optimization strategy is described in Section 4.2. The excellent heuristic properties of this hybrid method allow using it as the basic component of a multiple-stage optimization strategy, which is presented in Section 4.3. In Section 4.4 several methods to reduce goal function evaluations are presented. Section 4.5 describes an approach to deal with misleading problems. Section 4.6 is about the realization of our developed optimization algorithms. Finally, in Section 4.7 some optimization results achieved with our optimization strategies are presented.

4.2. Combined 2-Phase Optimization

To be able to cope with the non-trivial task of model optimization described above, we have developed a new kind of direct optimization algorithm called combined 2-phase optimization strategy. The basic idea of this hybrid method is to split the optimization process into two phases: pre-optimization with a probabilistic global optimization method and fine-optimization performed by a deterministic local Hill-Climber. The task of pre-optimization is to explore the search space in order to get into the direct neighbourhood (catchment area) of a global optimum point. The catchment area of an extreme point represents all the search points in its neighbourhood from which the extreme point can be localized by a local optimization method. Outgoing from the best solution found by pre-optimization (pre-optimization result) the task of fine-optimization is to efficiently localize the neighbouring extreme point with a user-defined accuracy. Thus, pre-optimization is predominantly responsible for optimization success, whereas fine-optimization has to ensure the quality of the optimization result.

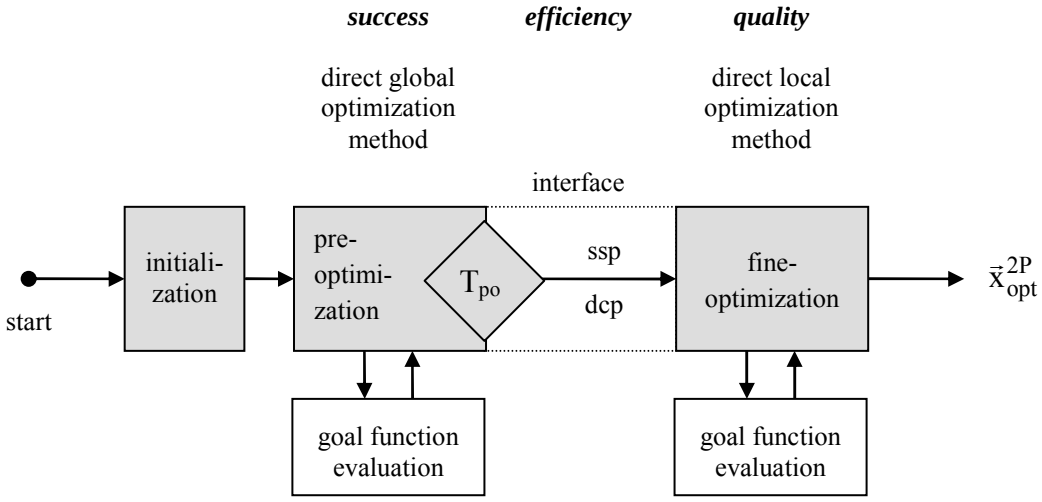


Figure 6. Basic structure of a combined 2-phase optimization strategy

Fig. 6 shows the basic structure of a combined 2-phase optimization strategy which is referred to as os_{2P} in the following. For pre-optimization os_{2P} uses a direct global optimization method. For fine-optimization a direct local optimization method is applied. The two strategies are coupled by an interface, comprising a method to select starting points (ssp) as well as a method to derive control parameter values from optimization trajectories (dcp). The result of a combined 2-phase optimization strategy as well as the required computational ef-

fort mainly depend on the specific capabilities of the employed global and local optimization method but also on the

- choice of suitable control parameter settings for the global optimization method
  For pre-optimization control parameter settings have to be used rather forcing exploration of the search space than convergence towards a search space region. This can be achieved by emphasizing the probabilistic search operators of the global optimization method.
- choice of an advantageous switch-over point from pre- to fine-optimization
  This problem affects the specification of a suitable termination condition T_{po} for the global optimization method in order to stop pre-optimization in time. This is not a trivial task because a good compromise between two contrary goals has to be found. On the one hand, a thorough exploration of the search space is required in order to avoid to get trapped into a sub-optimal region. On the other hand, the computational effort (number of goal function evaluations) for pre-optimization has to be kept as small as possible.
- choice of suitable control parameter settings for the local optimization method
  During pre-optimization the goal function is evaluated many times. The computed goal function values (optimization trajectory), representing collected knowledge about the goal function, can be used profitably to calculate suitable control parameter settings for the local optimization method. For this purpose a method to derive control parameter settings from optimization trajectories (dcp) was developed. It is based on analysing the optimization trajectory of the global optimization method by application of cluster analysis methods. From the size and form of the found clusters appropriate step sizes for the local optimization method can be derived. Well-suited initial step sizes are very important to keep the required number of goal function evaluations for local search small.
- selection of a favourable starting point $\vec{x}_{start}$ for the local optimization method (ssp)
  The simplest way to solve this problem is to choose the best solution found during pre-optimization as the starting point $\vec{x}_{start}$ for fine-optimization. A more complex approach is described in Section 4.4.

As already mentioned the specification of an appropriate termination condition T_{po} is decisive for the efficiency of a combined 2-phase strategy. On principal T_{po} may be based on the following criteria:

- The number of generated search points: This criterion allows specifying the maximum number of goal function evaluations which should be spent for pre-optimization.
- Search point constellations: Specific search point constellations (regional accumulations of search points in the search space) indicate convergence of the global optimization method towards a search space region. Applying standard cluster analysis methods, this property can be exploited profitably to compute switch-over points of good quality.
- The improvement of the goal function: This criterion has been proven to be the most powerful one. Pre-optimization is stopped, if the goal function could not be improved p%, $p \in R^+$ over a specified number of iteration steps.
- A priori knowledge about the goal function: A priori knowledge about the goal function usually provides advantageous hints to improve efficiency. Hence, if available, it should be exploited in any case.

Of course, it is also possible to specify termination conditions which combine several of the criteria listed above.

Table 1. Powerful direct optimization methods for global and local search

	global	local
direct optimization methods	Genetic Algorithm (GA) Simulated Annealing (SA)	Pattern Search (PS) of Hooke and Jeeves (Hooke and Jeeves, 1961)

Table 1 shows some powerful direct optimization methods for global and local search which are well-suited for realization of a combined 2-phase optimization strategy. Outgoing from the optimization methods presented in Table 1 there exist the following realization possibilities:

- Genetic Algorithm + Pattern Search (GA+PS)
- Simulated Annealing + Pattern Search (SA+PS)

Both realization alternatives have been already implemented and thoroughly examined (Syrjakow, 1997; Syrjakow and Szczerbicka, 1997; Syrjakow and Szczerbicka, 1999). Some quantitative optimization results achieved with GA+PS are presented in Section 4.7.

4.3. Multiple-Stage Optimization

All optimization strategies considered so far localize only one optimum point when executed. Outgoing from these so-called single-stage optimization strategies, we want to present an optimization algorithm which is able to detect several optimum points of a given (multimodal) optimization problem. The basic structure of such a multiple-stage optimization algorithm which is referred to as os_{ms} in the following is shown in Fig. 7.

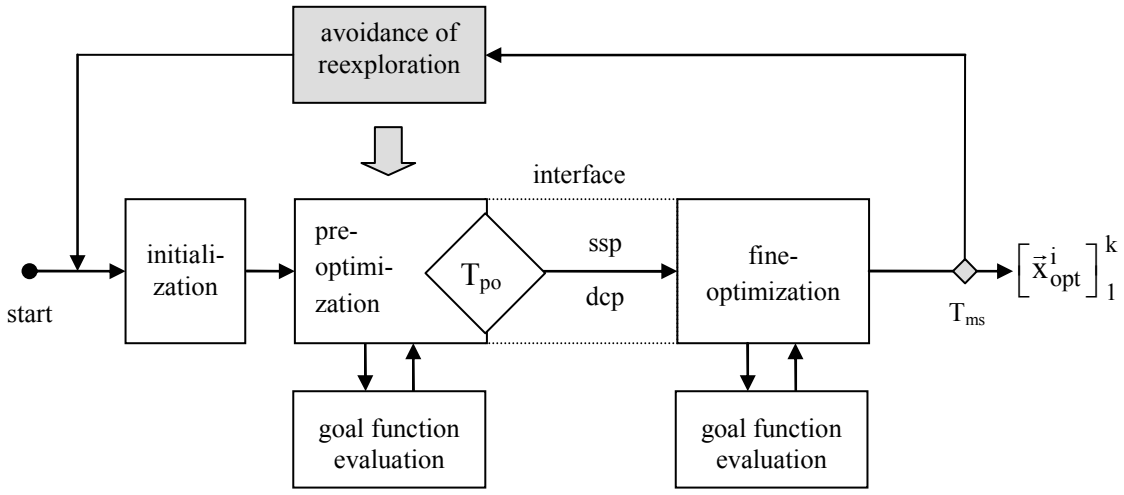


Figure 7. Basic structure of a multiple-stage optimization strategy

The main component of a multiple-stage optimization strategy is a combined 2-phase strategy os_{2P}. os_{2P} is embedded in an exterior iteration process, which generates step-by-step a sequence of optimum points $\vec{x}^{\,1}_{opt}, \vec{x}^{\,2}_{opt}, ..., \vec{x}^{\,k}_{opt}$, $k \in N$. An iteration step of a multiple-stage

optimization strategy is called optimization stage. os_{ms} stops, if the termination condition T_{ms} is fulfilled. A good termination criterion has been proven to be: stop, if a new optimum point could not be located over a specified number of optimization stages. If T_{ms} is not fulfilled, a method called avoidance of reexploration (AR) is applied. The task of AR is to avoid, that previously found optimum points are located again in subsequent optimization stages. This is done by making already explored regions of the search space unattractive for the global optimization method used for pre-optimization. For that purpose attractiveness values are introduced and related to each search point of the search space. Attractiveness values are computed by means of an attractiveness function

$$av(\vec{x})=\prod_{i=1}^{k}\left[1-(1+\alpha\cdot d_i)^{-\beta}\right], \qquad (4)$$

with $d_i=\sqrt{(\vec{x}-\vec{x}^{\,i}_{opt})^2}$; α,β: scaling factors; k: number of already found optimum points.

Multiple-stage optimization can be viewed as a substantial improvement compared to conventional optimization methods because not only one optimal solution is localized but a sequence of the most prominent extreme points of the given optimization problem. This enables the modeller to get a comprehensive overview of the behaviour of the optimized system. For a further description of multiple-stage optimization we refer to (Syrjakow and Szczerbicka, 1994; Syrjakow, 1997).

4.4. Methods for Reduction of Goal Function Evaluations

As already mentioned simulation-based goal functions may require a lot of time for evaluation. Thus, for direct optimization of these functions additional methods to reduce goal function evaluations are of great importance. A very simple and obvious way to save goal function evaluations is to avoid reevaluations of search points, which are generated several times during the optimization process. This can be done very easily by search of the optimization trajectory, which comprises all generated search points together with their corresponding goal function values.

Within a combined 2-phase strategy the pre-optimization phase offers an additional possibility to save goal function evaluations. This is due to the primary goal of pre-optimization, which is not to exactly localize a globally optimal solution but only to get into its catchment area. This property as well as the robustness of probabilistic global optimization strategies against inaccurately evaluated goal function values makes it possible to also use goal function approximations. The goal function value of a search point can be approximated if there are several search points in its direct neighbourhood, which have been already evaluated. Through goal function approximation a lot of possibly very expensive goal function evaluations can be saved without a substantial loss of optimization success. Especially multiple-stage optimization makes the application of a goal function approximator very advantageous. In this case with each optimization stage some more information about the goal function is gathered which in turn can be exploited in subsequent optimization stages for approximation. Fig. 8 shows the multiple-stage optimization strategy of Fig. 7 extended by a goal function approximator, which is embedded between the pre-optimization process and the process of goal function evaluation. For approximation we use a simple grid-based technique as well as a special kind of neural networks called Rectangular Basis Function Networks (Berthold and Huber, 1995). A more detailed description of our approach to accelerate pre-optimization by goal function approximation can be found in (Syrjakow *et al*, 1996).

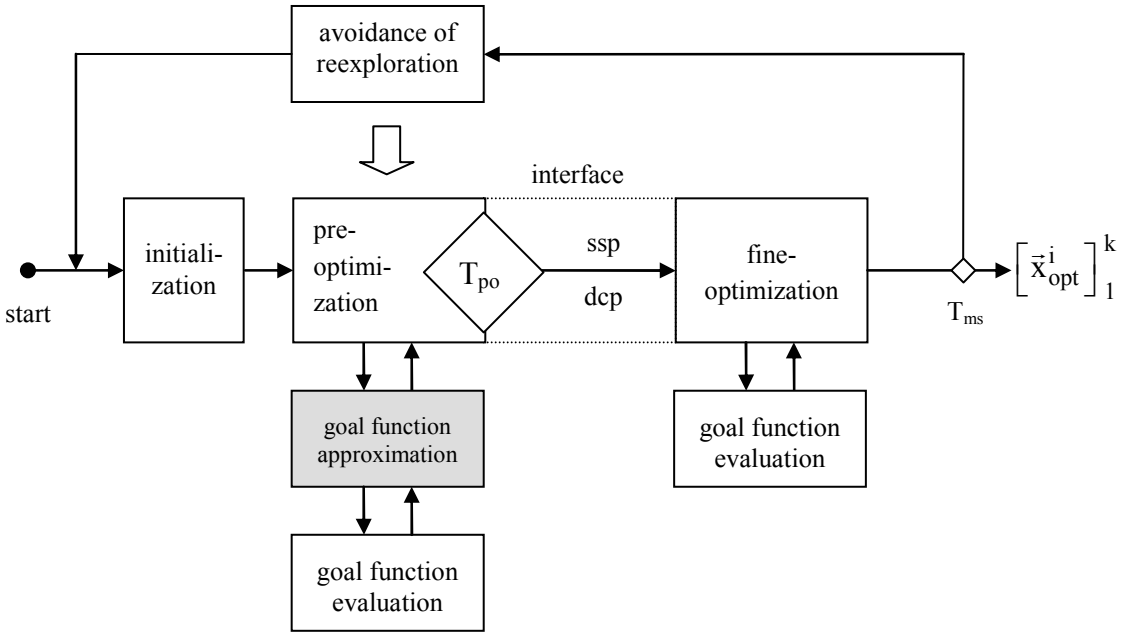


Figure 8. Acceleration of pre-optimization through goal function approximation

Another possibility to save goal function evaluations is to start fine-optimization not only once after a pre-optimization run but several times. This repeated start of fine-optimization which is shown in Fig. 9 has been proven to be very successful, especially in case of multi-modal goal functions with many global and/or local extreme points. Then, the probability is rather high that during pre-optimization several similar good solutions are found being located in the catchment areas of different extreme points. These extreme points can be obtained with only one pre-optimization run through the repeated start of fine-optimization.

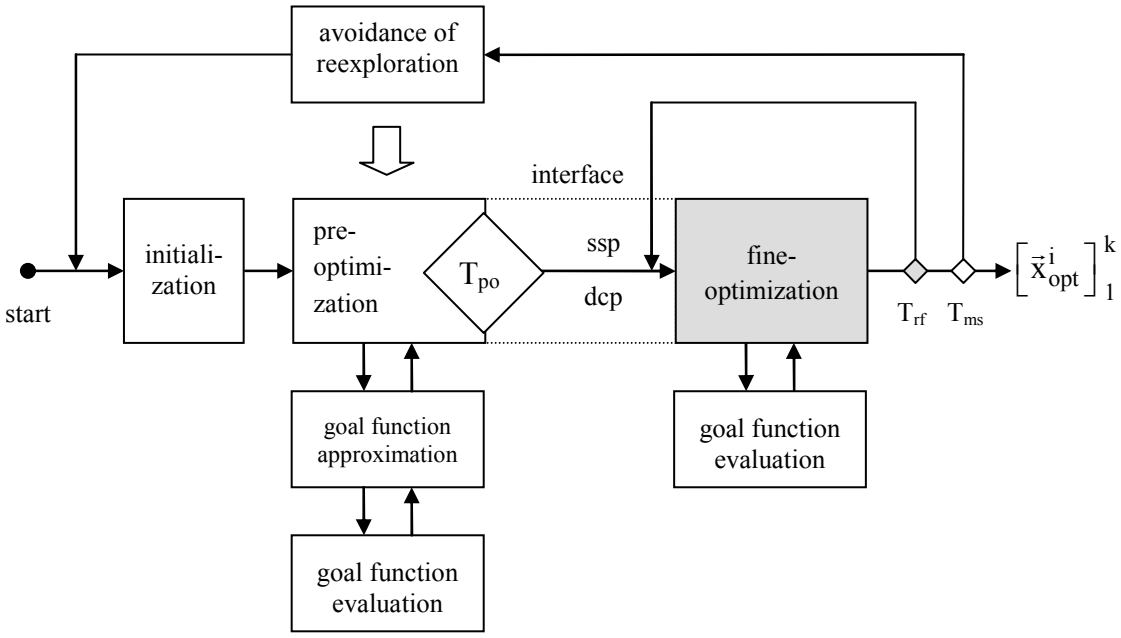


Figure 9. Repeated start of fine-optimization

4.5. Methods to Deal with Misleading Problems

Global optimization strategies which are based on principles of nature like Evolutionary Algorithms or Simulated Annealing have been proven to be very suitable for pre-optimization in many cases. However, there also exist problems which are very difficult to solve with these strategies. "Very difficult" for a particular optimization strategy means that pure Monte Carlo search works better on average. Problems with this property are called misleading for the considered optimization strategy in the following. An example of a misleading problem for Evolutionary Algorithms is presented in Fig. 10.

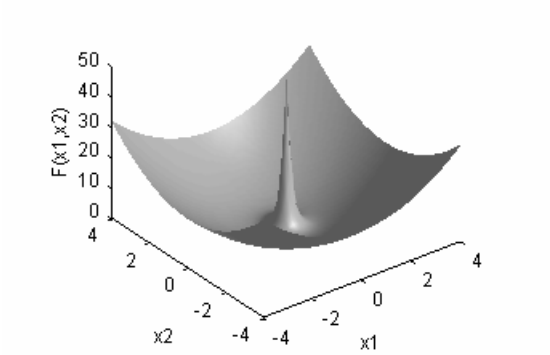


Figure 10. Misleading problem for Evolutionary Algorithms

This problem is difficult to solve because the catchment area of the global optimum point, which is located exactly in the middle of the search space is much smaller than the catchment areas of the surrounding four local optimum points. To solve this problem with an Evolutionary Algorithm the catchment area of the globally optimal solution has to be found very early in the optimization process. Otherwise the Evolutionary Algorithm converges quickly towards one of the four locally optimal solutions at the edge of the search space. The more the Evolutionary Algorithm converges towards one of the locally optimal solutions the less the probability will be to get back to the catchment area of the globally optimal solution. This is not the case when pure Monte Carlo search is applied because here the generated search points are equally distributed all over the search space.

An obvious possibility to improve the optimization success of a multiple-stage optimization strategy in case of misleading problems is to cyclically apply several pre-optimization methods. This possibility is depicted in Fig. 11 where a Genetic Algorithm (GA), Simulated Annealing (SA), Evolution Strategies (ES) and pure Monte Carlo (MC) search are applied cyclically one after the other. This collection of pre-optimization strategies has been proven to be very suitable because several different search principles are applied. That way the overall performance of multiple-stage optimization might slightly decrease in case of good-natured problems. However, the risk is considerably lowered to work on a particular problem with a completely unsuited optimization strategy.

As already mentioned the misleading problem presented in Fig. 10 can be solved on average more successfully with pure Monte Carlo search than with Evolutionary Algorithms. However, pure Monte Carlo search certainly is also not a very good choice because the observable performance difference is far away from being substantial and more and more shrinking with increasing problem dimension (see Section 4.7). A much more promising approach has been proven to be a slight modification of the multiple-stage strategy presented in Fig. 11. The modification is that the search objective (minimization or maximization) of pre-optimization remains not always the same but changes from time to time. Pre-optimization with the inverse search objective is called inverse pre-optimization in the following. When maximizing the goal function depicted in Fig. 10 inverse pre-optimization means that the applied pre-optimization algorithm is not looking for the global maximum but for the global

minimum. Because misleading problems usually have a function surface where globally minimal solutions are located close to catchment areas of globally maximal solutions and vice versa the probability is rather high to get into such a catchment area through inverse pre-optimization. For that reason inverse pre-optimization is an elegant and very effective possibility to avoid that misleading problems remain undiscovered.

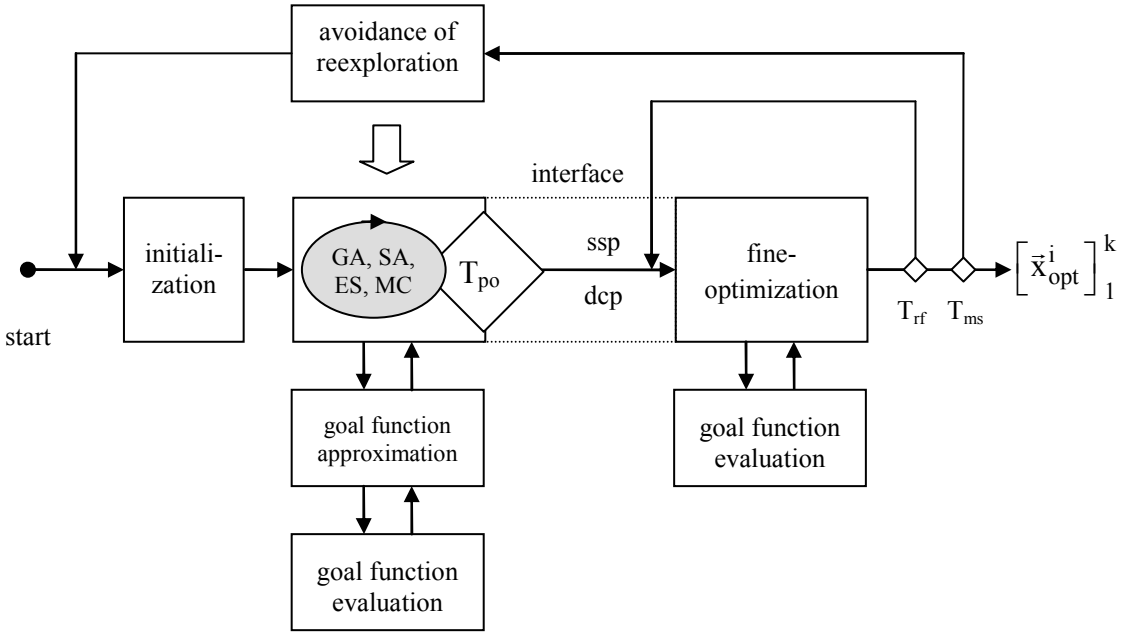


Figure 11. Application of different pre-optimization strategies in cyclical change

In case of the misleading problem shown in Fig. 10 the globally minimal solutions surround the catchment area of the globally maximal solution and the probability of getting into it through pre-minimization is about 0.5 (see Section 4.7). This probability can be raised very close to one by means of a short maximization phase in a limited area around the solution found by pre-minimization. For that maximization phase some dozens of goal function evaluations usually are sufficient to ensure not to narrowly miss the catchment area of the globally maximal solution.

4.6. Realization

All the optimization strategies described above have been already implemented and integrated into the parameter optimization component shown in Fig. 12. Such a parameter optimization component can be viewed as a special kind of experimentation component. As already depicted in Fig. 1 the extension of an M&S tool by an experimentation component requires in addition to the exchange of standardized documents also a more tight invocation-based integration concept. This is unavoidable because the two alternating processes of experimentation (generation of model input parameters) and model evaluation have to be coupled with each other allowing data exchange as well as process synchronisation.

Fig. 12 shows the interactions of a direct parameter optimization component with a model evaluation component. For specification of the optimization problem the optimization component has access to two files: the model description and the evaluation results. The model description comprises all existing model parameters allowing the user to select the parameters which have to be optimized. To define the goal function the user has to select one or to combine several model outputs which can be found in the evaluation results file. In each iteration

step of the optimization process the direct search strategy generates a vector of parameter values which are entered into the model description. Subsequently, the optimization component sends a request to the evaluation component containing several evaluation parameters. In case of a discrete-event simulation component for example the simulation run length, kind of confidence interval method etc. has to be defined. After model evaluation the evaluation component sends a response message to the optimization component to indicate that the required model outputs have been calculated and are now available in the evaluation results file. Outgoing from these outputs the optimization strategy generates a new parameter vector. This alternating process continues until a termination criterion is fulfilled.

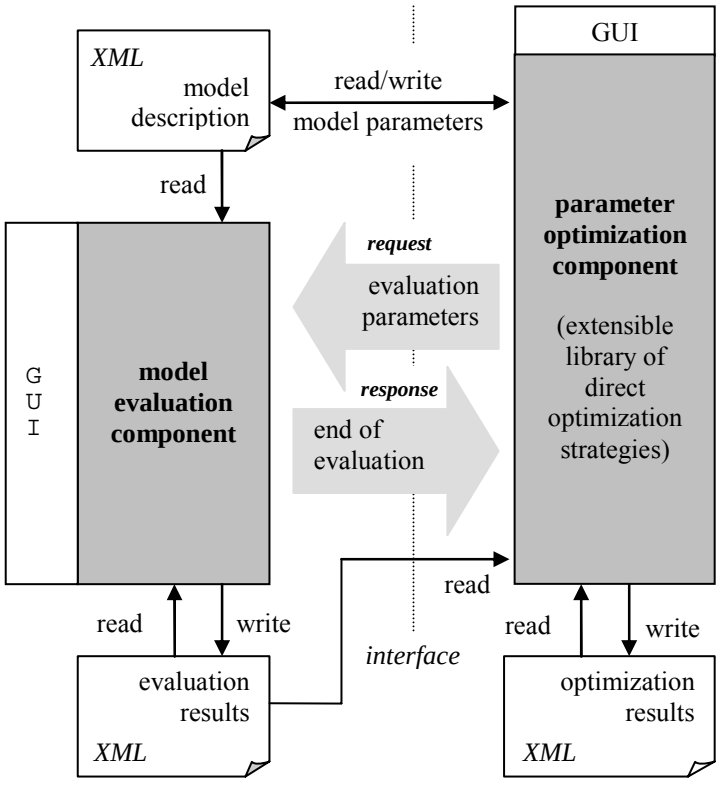


Figure 12. Interactions between the components for model evaluation and optimization

For realization of the required invocations of the model evaluation component universal middleware standards like CORBA, RMI or DCOM can be used. In our case we have used CORBA because it provides the following main advantages:

- Programming-language independent interface
- Interfaces between clients and servers are defined in a standardized Interface Definition Language (IDL)
- Using IDL, programmers can encapsulate existing applications in wrappers and use them as objects on the ORB (Object Request Broker)
- Rich set of distributed object services and facilities

To allow a flexible and platform independent usage the parameter optimization component was implemented in Java. Beside the optimization algorithms our optimization component offers a powerful graphical user interface (GUI). As depicted in Fig. 13 the GUI is divided into three parts. Each part is designed for a particular kind of user group to ensure a comfortable dealing with the implemented optimization algorithms.

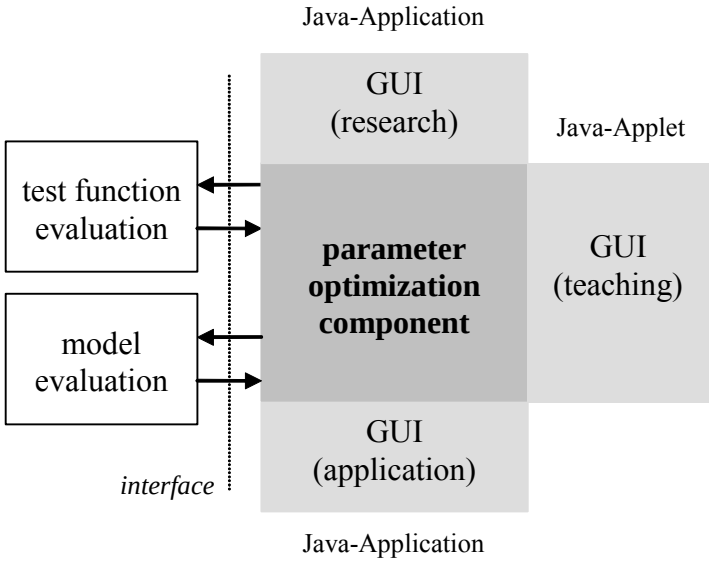


Figure 13. Graphical user interface (GUI) of the optimization component

Altogether the following kinds of use are supported:

- Getting advanced knowledge about complexity, behaviour, and performance of the implemented optimization strategies. For that purpose a lot of mathematical test functions are offered. Compared to simulation-based goal functions mathematical test functions have the great advantage that they can be evaluated very quickly. This property allows making a large number of optimization experiments in a short time, which is a basic prerequisite for statistically-sound performance analyses of probabilistic optimization strategies.
- Getting familiar with the implemented optimization strategies. For that purpose the complex search processes of several direct optimization methods are visualized. To ease the access to this version we have implemented it as a Java-Applet. It is available on the Web at http://goethe.ira.uka.de/~syrjakow/anim_env3/start_environment.html.
- Application of the implemented optimization algorithms to optimization problems from practice. For this purpose the GUI allows the specification of model optimization problems. Afterwards a personal assistant (wizard) supports the user to choose an appropriate optimization strategy. Finally, the user has the possibility to observe the ongoing optimization process and to look at the computed optimization results.

4.7. Evaluation

Some very important theoretical results regarding direct optimization strategies are summarized in the so-called "No Free Lunch" theorems for optimization (Wolpert and Macready, 1997) which can be viewed as a framework to explain the connection between effective direct optimization algorithms and the problems they solve. These theorems, loosely speaking, say that all algorithms that search for an extremum of a goal function perform exactly the same, when averaged over all possible goal functions. In other words no direct optimization algorithm, when averaged across all possible goal functions, is able to outperform pure Monte Carlo search. This in turn means that without any structural assumptions on an optimization problem it doesn't make any difference what kind of direct optimization algorithm is chosen.

At first sight this looks unpleasant because pure Monte Carlo search gets the same rating as much more sophisticated nature-analogous optimization methods like Genetic Algorithms or

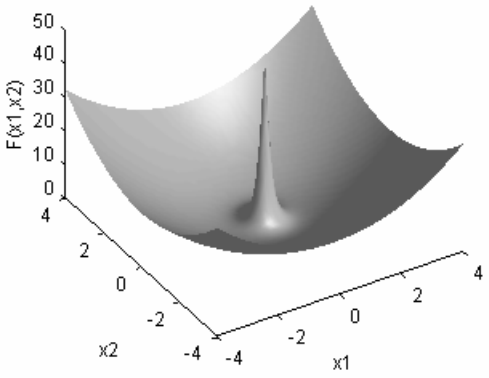
Simulated Annealing, which don't use chance completely arbitrarily but in a goal-driven way. Fortunately however, many simulation-based goal functions from practice are structured that way that nature-analogous optimization methods perform better than Monte Carlo search (Droste *et al*, 1999). Such good-natured optimization problems can be characterized as follows:

1.) The search space comprises only a limited number of extreme points.
2.) Each extreme point has an extensive catchment area.
3.) The goal function surface above the catchment area of an extreme point is not a thin peak.

The properties listed above usually are fulfilled if the system which has to be optimized is a technical system. This is not surprising because in this case a well-defined (non-chaotic) system behaviour can be assumed.

As already mentioned our proposed optimization methods usually perform well on good-natured problems. However, there also exist some exceptions. In the following, we consider a mathematical test problem, which in the 2- and 3-dimensional case fulfils the simplifying assumptions above. Even so, it isn't easy to optimize. This problem which was already investigated in Section 4.5 is specified in Table 2. The problem specification comprises the definition of the solution space S and the goal function F. Beyond that, the function surface for the 2-dimensional case (S^2, F^2) is presented. For all problem dimensions $n \in N$ test problem (S^n, F^n) has exactly one globally maximal solution at $\vec{x}^* = (0,0,...,0)$ which is surrounded by four locally maximal solutions. Although the function surface fairly well fulfils the simplifying assumptions it can be shown that Evolutionary Algorithms solve this problem less successfully than pure Monte Carlo search.

Table 2. Test problem (S^n, F^n)

Specification of test problem (S^n, F^n)	Function surface of (S^2, F^2)
Solution space: $S^n = \left\{ \vec{x} \in R^n \,\middle|\, -a_i \le x_i \le a_i;\ i \in \{1,...,n\} \right\}$ Goal function: $F^n(\vec{x}) = \sum_{i=1}^{n} x_i^2 + \frac{1}{0.02 + \sum_{i=1}^{n} x_i^2}$	

In the following, this is shown empirically by means of a comprehensive optimization experiment where we study not only the 2-dimensional but also the 3- and 4-dimensional case. The values of the constants a_i, $i \in \{1,...,n\}$, which restrict the search space and determine the goal function values of the four locally maximal solutions of (S^n, F^n), $n \in \{2,3,4\}$ are presented in Table 3.

Table 3. Constants a_i, $i \in \{1,...,n\}$ of test problem (S^n, F^n), $n \in \{2,3,4\}$

problem dimension n	2	3	4
constants a_i	4	2.5	2

To solve the test problem specified in Table 2 a multiple-stage optimization strategy os_{ms} was applied. Within the multiple-stage strategy a combined 2-phase optimization algorithm os_{2P} was used consisting of a Genetic Algorithm (being a special kind of Evolutionary Algorithms) for pre-maximization and Pattern Search for fine-maximization. The demanded accuracy of the Pattern Search algorithm was set to 0.01 for each coordinate. In order to get a representative overview of the performance behaviour of os_{ms} 200 independent optimization runs were carried out for each considered problem dimension $n \in \{2,3,4\}$. In each of these 200 multiple-stage runs a sequence of 20 maximum points was generated with os_{ms}.

Diagram 1 summarizes the results of this experiment which is referred to as E_1 in the following. The x-axis of Diagram 1 shows the problem dimension $n \in \{2,3,4\}$. The bars represent the average number of goal function evaluations required in one optimization stage (optimization effort oe). The white bars show the optimization effort of the GA, the grey ones represent the optimization effort of PS. The lines show the optimization success os(s) achieved after $s \in \{1,2,...,5\}$ optimization stages, which is defined as follows:

$$os(s)=\sum_{i=1}^{s} p^{i}_{\vec{x}^*,\varepsilon} \,. \tag{5}$$

In the expression above s denotes the optimization stage and $p^{i}_{\vec{x}^*,\varepsilon}$ is the empirical probability of finding the global maximum point $\vec{x}^*$ for the first time in optimization stage i with accuracy ε:

$$p^{i}_{\vec{x}^*,\varepsilon} = \frac{\text{number of first } (\vec{x}^*,\varepsilon)\text{-hits in stage i}}{\text{total number of stage i optimization runs}} \,. \tag{6}$$

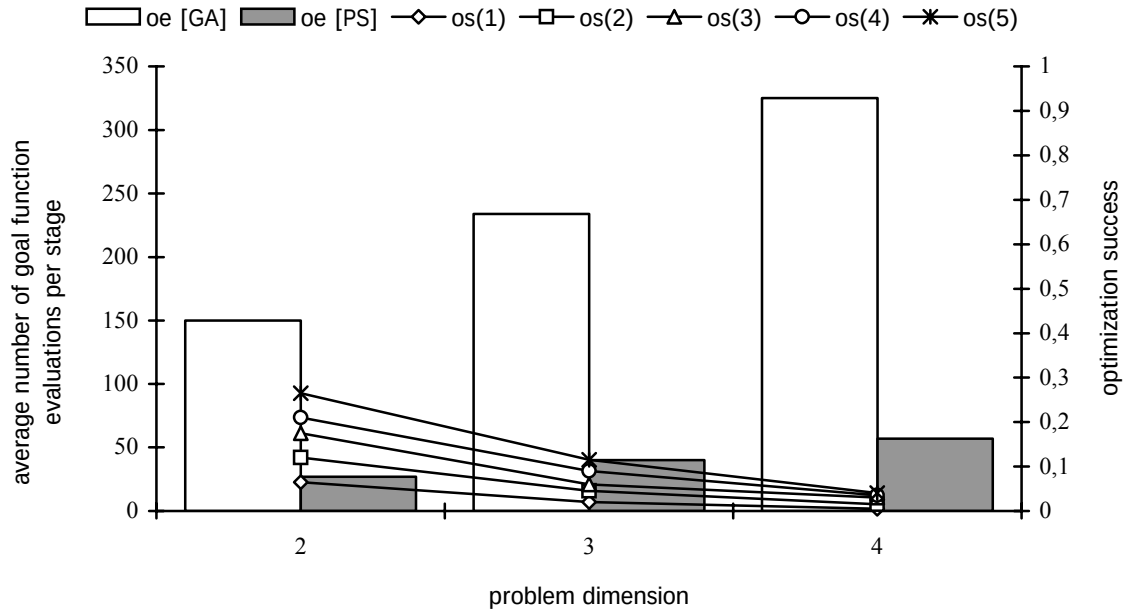


Diagram 1. Results of optimization experiment E_1

The results presented in Diagram 1 impressively show that it is very difficult for the Genetic Algorithm to get into the catchment area of the globally maximal solution. Already for the 2-dimensional problem (S^2, F^2) the optimization success after the first optimization stage is very low and cannot be increased considerably with additional optimization stages. With increasing problem dimension the success curves os(s), $s \in \{1,2,...,5\}$ quickly drop towards zero.

To get more evidence that (S^n, F^n) is really a misleading problem for Genetic Algorithms another comprehensive optimization experiment was made. In this experiment which is referred to as E_2 in the following the same multiple-stage optimization strategy as in E_1 was used except that for pre-optimization the Genetic Algorithm was replaced with pure Monte Carlo search. The results of E_2 are summarized in Diagram 2. They confirm that (S^n, F^n) actually is a misleading problem for Genetic Algorithms because with exactly the same pre-optimization effort pure Monte Carlo search works better on average than Genetic Algorithms. However, the difference between the success curves shown in Diagram 1 and 2 is rather small and drops with increasing problem dimension. For that reason pure Monte Carlo search cannot be considered as a really good substitute for Genetic Algorithms.

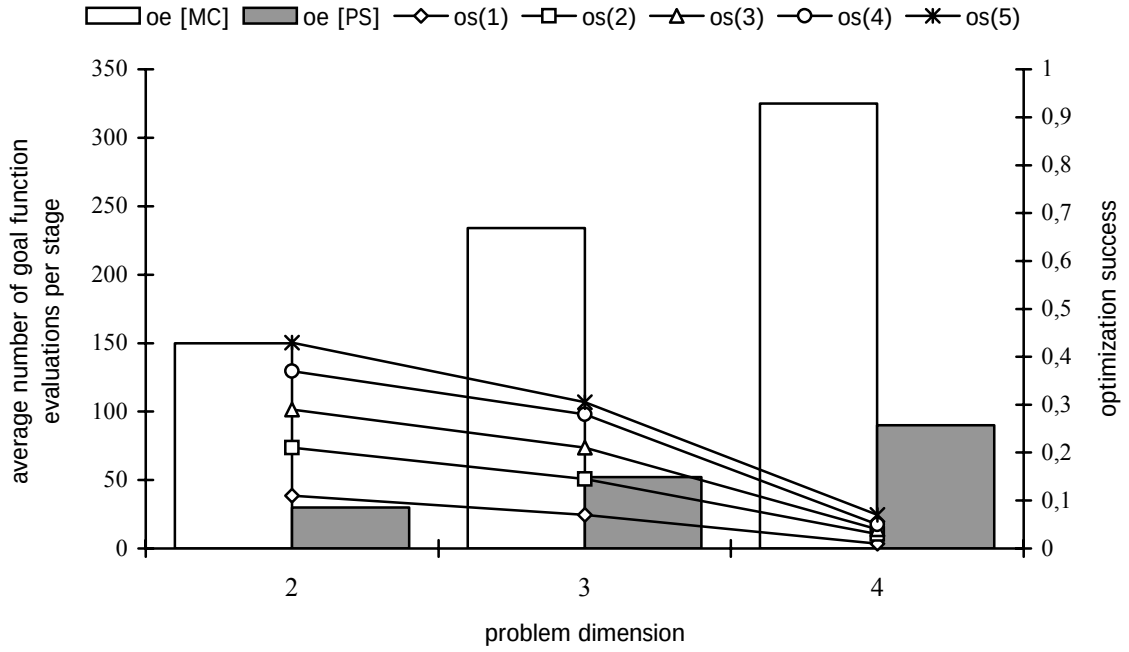


Diagram 2. Results of optimization experiment E_2

The results of the next optimization experiment E_3 show that it is quite possible to solve test problem (S^n, F^n) both, successfully, and efficiently. The optimization strategy used in E_3 is the same as in E_1 apart from the fact that for pre-optimization the Genetic Algorithm is used inversely, i.e. the Genetic Algorithm is used for minimization instead of maximization. The results of E_3 are summarized in Diagram 3. With exactly the same control parameter settings as in E_1 the Genetic Algorithm gets into the catchment area of the globally maximal solution in the first optimization stage with a probability of almost 0.5 independently of the problem dimension. The success curves os(i), $i \in \{2,3,...,5\}$ show that with each additional optimization stage the optimization success can be increased considerably. After five optimization stages the optimization success has almost reached the maximal value of 1 for all considered problem dimensions. The success curve os(1) presented in Diagram 3 can be further improved if

after pre-minimization a short maximization phase is carried out in a limited area around the solution found by pre-minimization. That way it is possible to raise os(1) almost to 1 by spending only some dozens of goal function evaluations.

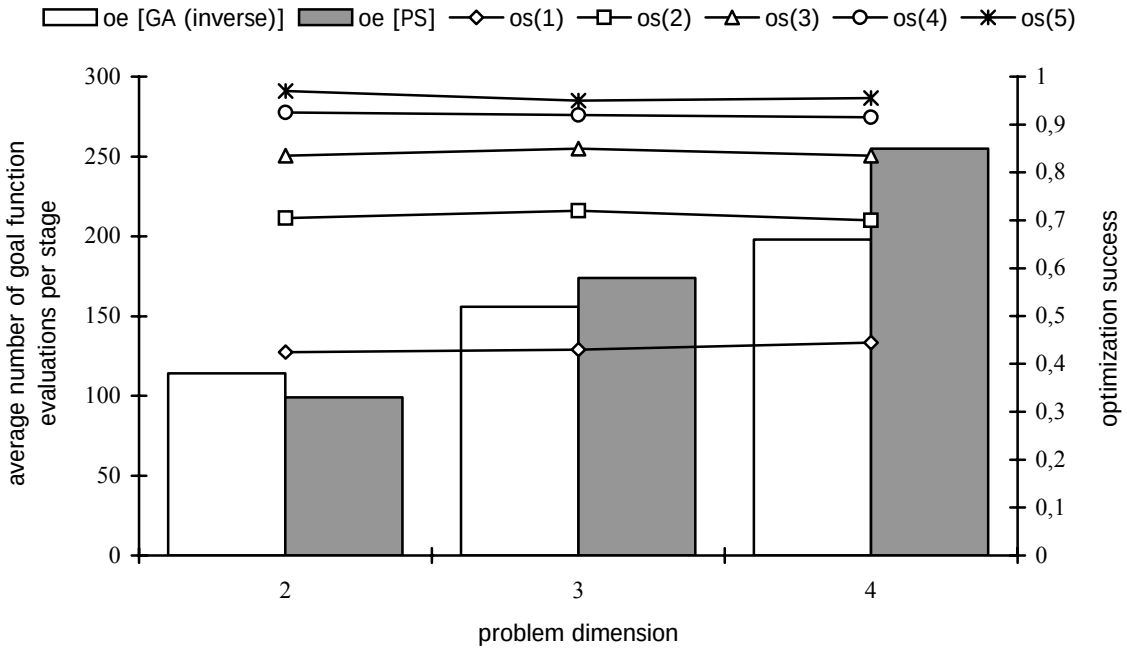


Diagram 3. Results of optimization experiment E_3

Finally, it should be mentioned that the optimization results presented in this Section could be further improved through application of the methods to reduce goal function evaluations described in Section 4.4. In the experiments described above only the very simple method to avoid reevaluations of previously evaluated search points was applied in order to save goal function evaluations. Through application of goal function approximation and repeated fine-optimization it is possible to further reduce the optimization effort of os_{ms} considerably. Quantitative results regarding this can be found in (Syrjakow *et al*, 1996; Syrjakow, 1997).

5. CONCLUSION

In this paper we presented a hybrid integration approach for M&S tool components being a combination of loose document-based and tight invocation-based integration concepts. The core of our approach is an XML-based model interchange format, which allows a homogeneous and standardized information exchange between tool components. For the tight coupling of tool components universal component 'wiring' standards are used. Our integration concept has been proven to be very flexible and applicable to all kinds of M&S tools. For its validation we have applied it to realize a component-oriented SPN-based M&S tool. A great advantage of M&S tools with a component-oriented software design is their openness for all kinds of extensions. As a result tool developers can fully concentrate on the development of such extensions and are not any longer needlessly stressed with their integration.

Today, especially experimentation components are of great importance because they allow to automatically extract valuable information and knowledge about the behaviour of complex simulation models which isn't possible by hand any more. In this paper an experimentation component was presented in detail, which provides efficient and universally applicable methods to optimize the behaviour of complex simulation models. Beside common direct strategies for global and local search our optimization component offers combined 2-phase and

multiple-stage optimization being a substantial improvement compared to existing nature-analogous optimization methods like Genetic Algorithms, Simulated Annealing, and Hill-Climbing. Combined 2-phase strategies are combinations of global and local search methods trying to exploit their advantages. The excellent heuristic properties of combined 2-phase optimization are an important prerequisite for multiple-stage optimization allowing to efficiently localize not only one but a sequence of prominent extreme points of a given goal function. Altogether our optimization component offers a powerful modular assembly system of direct optimization strategies which can be flexibly adapted to a broad range of optimization problems. The achieved optimization results show that our developed optimization methods work very well on a variety of good-natured problems. Even misleading problems can be solved efficiently through inverse pre-optimization. In our future work we intend to further improve our optimization algorithms. At the moment we are looking for local fine-optimization strategies which could replace the deterministic Pattern Search algorithm. A promising candidate seems to be the probabilistic SPSA (Simultaneous Perturbation Stochastic Approximation) method (Spall, 1998). Beyond that, we will develop other kinds of experimentation components e.g. for sensitivity analysis or model validation. And finally, we will further apply our hybrid integration concept to build powerful and innovative M&S tools.

ACKNOWLEDGEMENTS

We want to thank Prof. D. Schmid for his encouragement and support of our work. We also thank our students, especially S. Schillinger, F. Schmidt, H. Renfranz, T. Sommer, D. Haag, C. Bentz, J. Gramlich, and A. Kehl for their engagement and contributions.

REFERENCES

Aarts, E., Korst, J. (1989). Simulated Annealing and Boltzmann Machines. John Wiley & Sons.

Berthold, M.R., Huber, K.-P. (1995). Extraction of Soft Rules from RecBF Networks. Advances in Intelligent Data Analysis. Vol. 1, pp. 11-15, IIAS.

Brown, A.W. (2000). Large-Scale, Component-Based Development. Prentice Hall.

Droste, S., Jansen, T., Wegener, I. (1999). Perhaps Not a Free Lunch But At Least a Free Appetizer. Proceedings of the Genetic and Evolutionary Computation Conference. Orlando, Florida, USA, July 13-17, pp. 833-839.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

Hooke, R.A., Jeeves, T.A. (1961). Direct Search Solution for Numerical and Statistical Problems. Journal ACM, 8, pp. 212-221.

Jensen, K. (1991). High-Level Petri Nets: Theory and Application. Springer.

Jüngel, M., Kindler, E., Weber, M. (2000). The Petri Net Markup Language. Proceedings of the AWPN-Workshop. Koblenz, Germany.

Kuhl, F., Weatherly, R., Dahmann, J. (2000). Creating Computer Simulation Systems - An Introduction to the High Level Architecture. Prentice Hall.

Lindemann, C. (1998). Performance Modelling with Deterministic and Stochastic Petri Nets. John Wiley & Sons.

Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag.

Schwefel, H.-P. (1981). Numerical Optimization of Computer Models. John Wiley & Sons.

Spall, J.C. (1998). An Overview of the Simultaneous Perturbation Method for Efficient Optimization. Johns Hopkins APL Technical Digest. Vol. 19, pp. 482–492.

Syrjakow, M., Szczerbicka, H. (1994). Optimization of Simulation Models with REMO. Proceedings of the European Simulation Multiconference. Barcelona, Spain, June 1-3, 1994, pp. 274-281.
Syrjakow, M., Szczerbicka, H., Berthold, M.R., Huber, K.-P. (1996). Acceleration of Direct Model Optimization Methods by Function Approximation. Proceedings of the 8th European Simulation Symposium. Genoa, Italy, October 24-26, 1996, Volume II, pp. 181-186.
Syrjakow, M. (1997). Verfahren zur effizienten Parameteroptimierung von Simulationsmodellen. Dissertation at the Institute for Computer Design and Fault Tolerance at the University of Karlsruhe. Shaker-Verlag, Aachen.
Syrjakow, M., Szczerbicka, H. (1997). Efficient Methods for Parameter Optimization of Simulation Models. Proceedings of the 1st World Congress on Systems Simulation. Singapore, Republic of Singapore, September 1-3, 1997, pp. 54-59.
Syrjakow, M., Szczerbicka, H. (1999). Efficient Parameter Optimization based on Combination of Direct Global and Local Search Methods. In L. D. Davis, K. De Jong, M. D. Vose, L. D. Whitley (eds) Evolutionary Algorithms (Ima Volumes in Mathematics and Its Applications Vol. 111). Springer Verlag, New York, pp. 227-249.
Syrjakow, M., Syrjakow, E., Szczerbicka, H. (2002). Towards a Component-Oriented Design of Modeling and Simulation Tools. Proceedings of the International Conference on AI, Simulation and Planning in High Autonomy Systems. Lisbon, Portugal, April 7-10.
Syrjakow, M. (2003). Web- und Komponenten-Technologien in der Modellierung und Simulation. Habilitation at the Faculty of Informatics at the University of Karlsruhe.
Syrjakow, E., Syrjakow, M. (2003). XML for Data Representation in Modeling and Simulation Environments. Proceedings of the IASTED International Conference on Modelling, Simulation, and Optimization. Banff, Alberta, Canada, July 2-4, pp. 100-107.
Szyperski, C. (1999). Component Software - Beyond Object-Oriented Programming. Addison-Wesley.
Wolpert, D.H., Macready, W.G. (1997). No Free Lunch Theorems for Optimization. In IEEE Transactions on Evolutionary Computation. Vol.1, No.1, April, pp. 67-82.