# MODELLING SITUATED DESIGN OPTIMIZATION

JOHN S GERO AND UDO KANNENGIESSER
*Key Centre of Design Computing and Cognition*
*University of Sydney*

**Abstract.** This paper presents a framework for situated design optimization that expands the traditional view of design optimization. It is based on the notion of interaction providing the potential for modifications of various aspects of the optimization process: problem formulation, the optimization tool, the designer and ultimately the result. In contrast to other approaches, these modifications can drive further interactions within the same optimization process. We use parts of the situated function-behaviour-structure (FBS) framework as an ontological basis to describe the effects of intertwined interactions and modifications on the current state space of an optimization process.

## 1. Introduction

Optimization in designing is a process that aims to find the best design solution with respect to a selected set of performance criteria (Papalambros and Wilde 2000). Optimization models are often represented in terms of a set of formal mathematical expressions:

$$\begin{array}{rll} \text{Given design variables} & x \in \chi \subseteq \Re^n & (1) \\ \text{and constraints} & h(x) = 0 & (2) \\ & g(x) \leq 0 & (3) \\ \text{optimize objective} & f(x) & (4) \end{array}$$

Expression (1) defines the set of design variables as a subset of the *n*-dimensional real space $\Re^n$. The functions h(x) and g(x) specified by equations (2) and (3) represent equality and inequality constraints, respectively. They correspond to a set of specific performances that determine the feasibility or acceptability of the design. Finally, expression (4) specifies an objective function f(x) to be optimized. This function defines the set of performances that determine the optimality of the design.

Table 1 shows how these mathematical descriptions map onto a common ontological representation of design objects, namely Gero's (1990) function-behaviour-structure (FBS) ontology.

TABLE 1. Mapping between a mathematical and an ontological model in design optimization (S = structure; B = behaviour; $B^c$ = behaviour specified by constraints; $B^o$ = behaviour specified by objective function).

| Mathematical model | Ontological (FBS) model |
|---|---|
| $x \in \chi \subseteq \Re^n$ | S |
| $h(x) = 0$ | S, $B^c$ |
| $g(x) \leq 0$ | |
| $f(x)$ | $B^o$ $\quad B = B^c \cup B^o$ |

Structure (S) consists of the components of a design object as well as the relationships among these components. In most design domains, structure (S) comprises the geometrical, topological and material properties of the object. Table 1 shows that the structure state space (S) of the optimum design can be mapped on the given set of design variables. Some of the constraints specify ranges of values for these variables.

Behaviour (B) is derivable from structure (S) and specifies the performance of a design object. In the optimization model, behaviour (B) is embodied in the objective function and in some of the constraint functions. Accordingly, the behaviour state space (B) of the optimum design can be viewed as the union of the set of behaviour variables defined by the objective function ($B^o$) and the set of behaviour variables defined by some of the constraints ($B^c$).

Function (F), as the teleology of a design object, is not included in design optimization models.

The FBS ontology has been used as the basis for modelling design processes as a set of transitions between function, behaviour and structure (Gero 1990). This model is known as the function-behaviour-structure (FBS) framework. Design optimization is a specific class of design process that can be subsumed in the FBS framework, Figure 1. The FBS framework distinguishes between expected behaviour (Be) and behaviour derived from structure (Bs). Be represents a set of performance criteria used as benchmarks for the design structure (S). Bs is the set of performances that are measured or derived from structure (S). Values for Bs must be within the ranges set by Be.

While the FBS framework encompasses eight fundamental processes, the scope of optimization generally comprises only three of them, namely synthesis (labelled 2 in Figure 1), analysis (labelled 3) and evaluation (labelled 4). Synthesis in optimization generates a candidate solution as an instantiation of a point within the structure (S) state space. Analysis derives the values of the relevant behaviours of that solution using the given set of objective functions and constraints. Evaluation then compares the behaviour of different candidate solutions and decides on either continuing or stopping the search for an optimum. Most design optimization problems require iterative procedures involving large numbers of synthesis-analysis-evaluation cycles.

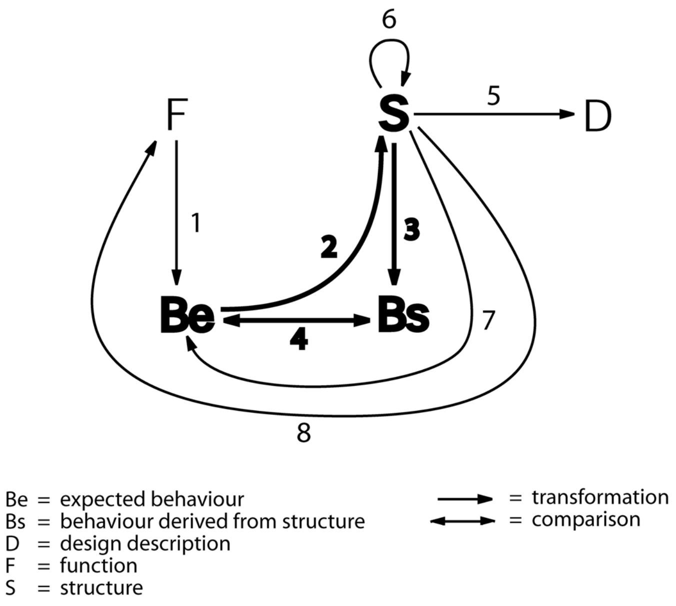

*Figure 1.* The processes involved in design optimization (highlighted) as a subset of the eight processes in Gero's (1990) framework of designing: (1) formulation, *(2) synthesis*, *(3) analysis*, *(4) evaluation*, (5) documentation, (6) reformulation type 1, (7) reformulation type 2, (8) reformulation type 3.

An extensive body of work exists in the development and application of optimization techniques in design across a large number of design disciplines (Wilde 1978; Papalambros and Wilde 2000; Pardalos and Resende 2002; Parmee and Hajela 2002). Most research in this area focuses on improving the speed of search for optimal designs and refining the quality of the optimal designs. This has resulted in various new optimization techniques. Recent advances are having only a marginal effect on the efficiency of most design optimization problems. This is mainly due to the following reasons:

- lack of transfer of earlier results as the design changes
- lack of domain knowledge in computational tools

- lack of task knowledge in computational tools
- lack of feedback into process strategies in the tool

These shortcomings are due to a view of design optimization that is too narrow, Figure 2. In this view, the designer starts optimization by selecting a computational tool with appropriate search methods for the particular optimization task and by formulating the problem in a tool-specific form. The tool then produces a result in terms of the structure and behaviour of an optimum design and optionally a set of post-optimality analyses. Based on that result, the designer may then reformulate the problem or tool/method selection and make repeated use of the tool to achieve a better result.

This view of design optimization does not explicitly include the grounds on which designers are able to gain experience in optimization and to decide on particular search spaces and methods. Individual experience and skills are key factors of successful design optimization but are hidden within a black box labelled the "designer". This traditional view of design optimization limits the development of better computational aids to support designers in finding optimum designs.

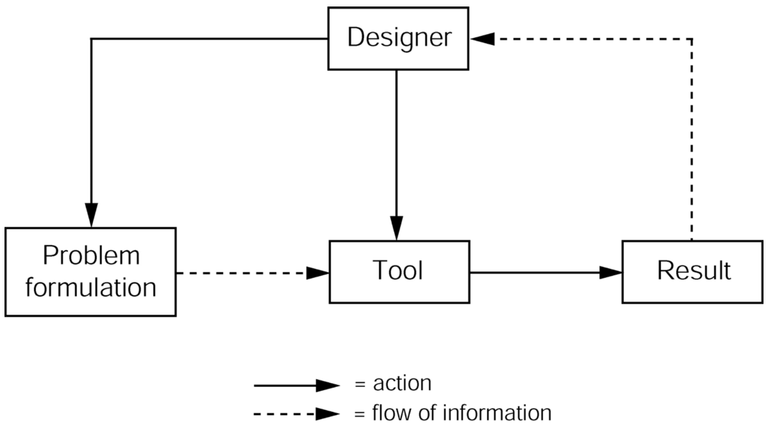

*Figure 2.* Traditional view of design optimization.

In this paper we propose an extended view of optimization that is based on the interaction of the computational tools, their users, the design problems and results. This provides a basis to guide further research that addresses the inadequacies described earlier. The foundations are drawn from work in situated cognition in design (Clancey 1997; Smith and Gero 2005). We then explore our situated view of design optimization using the FBS schema, which provides a basis for developing new optimization tools that can flexibly and efficiently reason about their interactions and adapt to their use.

## 2. Situatedness and the Notion of Interaction

Designing is an activity during which designers perform actions in order to change their environment. By observing and interpreting the results of their actions, they then decide on new actions to be executed on the environment. This means that the designer's concepts may change based on what they are "seeing", which itself is a function of what they have done. One may speak of an "interaction of making and seeing" (Schön and Wiggins 1992). This interaction between the designer and the environment strongly determines the course of designing. This idea is called situatedness, whose foundational concepts go back to the work of Dewey (1896) and Bartlett (1932).

In experimental studies of designers phenomena related to the use of sketches, which support this idea, have been reported. Schön and Wiggins (1992) found that designers use their sketches not only as an external memory, but also as a means to reinterpret what they have drawn, thus leading the design in a new direction. Suwa et al. (1999) noted, in studying designers, a correlation of unexpected discoveries in sketches with the invention of new issues or requirements during the design process. They concluded that "sketches serve as a physical setting in which design thoughts are constructed on the fly in a situated way".

An idea that fits into the notion of situatedness has been proposed by Dewey in 1896 (Clancey 1997) and is today called constructive memory. Its relevance in the area of design research has been shown by Gero (1999). Constructive memory is best exemplified by a quote from Dewey via Clancey: "Sequences of acts are composed such that subsequent experiences categorize and hence give meaning to what was experienced before". The implication of this is that memory is not laid down and fixed at the time of the original sensate experience but is a function of what comes later as well. Memories can therefore be viewed as being constructed in response to a specific demand, based on the original experience as well as the situation pertaining at the time of the demand for this memory. Therefore, everything that has happened since the original experience determines the result of memory construction. Each memory, after it has been constructed, is added to the existing knowledge (and becomes part of a new situation) and is now available to be used later, when new demands require the construction of further memories. These new memories can be viewed as new interpretations of the augmented knowledge.

The advantage of constructive memory is that the same external demand for a memory can potentially produce a different result at a later time, as newly acquired experiences may take part in the construction of that memory. Constructive memory can thus be seen as the capability to integrate new experiences by using them in constructing new memories. As a result, knowledge "wires itself up" based on the specific experiences it has had, rather than being fixed, and actions based on that knowledge can be altered in the light of new experiences.

Situated designing uses first-person knowledge grounded in the designer's interactions with the environment (Bickhard and Campbell 1996; Clancey 1997; Ziemke 1999; Smith and Gero 2005). This is in contrast to static approaches that attempt to encode all relevant design knowledge prior to its use. Evidence in support of first-person knowledge is provided by the fact that different designers are likely to produce different designs for the same set of requirements. The same designer is likely to produce different designs at different points in time even though the same requirements are presented. This is a result of the designer acquiring new knowledge while interacting with their environment.

Gero and Kannengiesser (2004) have modelled situatedness as the interaction of different worlds (or environments), including the designer's internal and external world. The internal world contains the designer's experience and goals, while the external world consists of representations of things outside of the designer. Each world can bring about changes in the other world. In addition, constructive memory provides the opportunity to modify the internal world without the need for a changed external environment. The foundation of this ability is established by the designer interacting with their memories.

The notion of interaction has been shown to be central in the concept of situatedness. Interaction may also play a role in non-situated views of the (design) world; here, however, this notion is generally interpreted in terms of a simple feedback loop to inform the actions of a well-defined system that remains itself unchanged. Changes are restricted to take place only in the external environment. In contrast, a situated view allows both entities engaged in an interaction to be affected by change. This creates the potential to emerge new situations for both external and internal environments that could not have been possible with the static, non-situated model.

An important aspect in situated interaction is the notion of interpretation. Rather than being conceived of as a simple flow of information, interpretation is understood as a form of action, originating from both the external and internal environment and resulting in changes in the internal environment. Gero and Fujii's (2000) model of interpretation as intertwined *push-pull* processes can be understood in this sense. A *push process* can be seen as an action driven by the external world aiming to change the internal world according to the data provided by the external world. A *pull process* can be seen as an action driven by the internal world aiming to change itself according to biases provided by the current expectations and goals. The notion of re-interpretation is often used to denote interpretations in which a changed internal world is the major driver of self-directed ("pull") actions leading to substantial changes in the same internal world.

## 3. An Interaction-Based View of Design Optimization

Integrating the notion of interaction into a model of design optimization addresses the shortcomings identified in Section 1, as it provides the opportunity for change – both in the internal "knowledge" of an optimization system and in the design it is operating on. This is a condition for future optimization tools to flexibly acquire task knowledge and domain knowledge that is adapted to the classes of optimization problems they are exposed to.

Figure 3 introduces the concept of interaction as the key element of a design optimization system. It establishes the relationships between the designer, the problem formulation, the tool and the result, which were connected previously by unidirectional arrows representing flows of information and action, Figure 2. This concept broadens the scope of optimization as a system in which every component has the potential to induce changes in other components as well as to modify itself. Static flows of information and action may now be viewed as being subsumed in the more general notion of interaction.

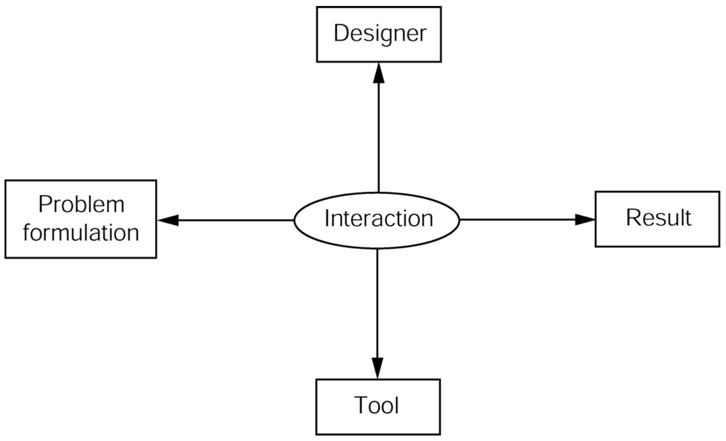

*Figure 3.* Interaction-based view of design optimization.

The interaction component in Figure 3 stands for three classes of interactions:

1. Interactions involving only one component: These interactions involve the designer or, alternatively, the tool as a component that interacts with itself. This includes the concept of reflective reasoning about one's own actions.

2. Interactions involving two components: Here all pairwise combinations of individual components can potentially interact. These are:
   - interactions between the designer and the problem formulation
   - interactions between the designer and the tool
   - interactions between the designer and the result
   - interactions between the problem formulation and the result
   - interactions between the tool and the problem formulation
   - interactions between the tool and the result
3. Interactions involving more than two components: These interactions can be viewed as involving composites rather than individual components. Numerous composites and interactions among them are conceivable. An important class of composites represents processes as they can be viewed as input-transformation-output triplets (Gero and Kannengiesser 2006). A typical triplet consists of the problem formulation (input), the tool (transformation) and the result (output). This triplet represents a search process involving one or more iterations, with which the designer or the tool may interact.

A number of previous models and systems of design optimization can be recast into such an interaction-based approach. Some of them map onto interactions between the problem formulation and the result. For example, Parmee's (1996) cluster-oriented genetic algorithms (GAs) produce preliminary results indicating high-performance regions within the search space from which further features such as sensitivity information are extracted. These features form the basis for concentrating search on particular areas within the search space via reformulating the problem in terms of structure (S) variables and constraints. Mackenzie and Gero (1987) have induced rules to detect certain features of Pareto optimal sets relating to curvature, sensitivity and other information. The rules use this information to reformulate the problem by reducing the behaviour (B) state space. As both components in the interaction between the problem formulation and the result are non-persistent, there is no place for new knowledge to be constructed and maintained. As a result, the execution of future optimization tasks cannot be directly improved without manual intervention.

Other research involves the tool as a persistent component to acquire and reuse new knowledge. A tool developed by Schwabacher et al. (1998) extracts characteristics of optimization results and uses them to formulate new optimization problems. These characteristics include information such as optimal design structure, mappings between structure and behaviour, infeasible behaviour and active constraints. This leads to better problem formulation in terms of reduced search spaces and improved starting points for gradient-based search. Jozwiak's (1987) system learns inactive constraints in order to predict whether or not the computation of particular constraint functions of a future optimization task may be neglected in the tool. Nath and Gero (2004) use machine learning to let a system acquire strategic knowledge as mappings between past design contexts and design decisions that led to useful results. These mappings are then available for the system to achieve solutions to similar design tasks more efficiently. Stahovich (2000) developed a tool that can extract iterative search strategies used by a designer. The tool then reuses these strategies to perform similar design tasks.

In most of this work, any change in the tool takes effect only after the optimization process is completed, i.e. an increase in efficiency can be achieved only in subsequent optimizations. This differentiates the cited work from the concept of situatedness, where changes have an effect on the same instance of interaction that they originate from. Situated design optimization has the potential to modify state spaces and strategies during the process of iterative search. Visualisation systems for process stages have been developed that can be recast to look like this view of design optimization (Ellman et al. 1998). The main driver for interactive change is still located inside the black box called the "designer".

While the notion of interaction as presented in Figure 3 can substantially shift our understanding of design optimization, it provides insufficient description to distinguish a situated view from the previous work mentioned above. Section 4 uses an ontological approach to the development of a framework for situated design optimization.

## 4. An Ontology for Situated Design Optimization

Let us have a closer look at Gero and Kannengiesser's (2004) model of situatedness, Figure 4(a). It subdivides the internal world into an interpreted and an expected world, the latter of which is a subset of the former. These two worlds are connected to each other by a process of focussing on some of the concepts located in the interpreted world and using them as goals that are then located in the expected world. The goals are subsequently used to inform actions changing the external world.

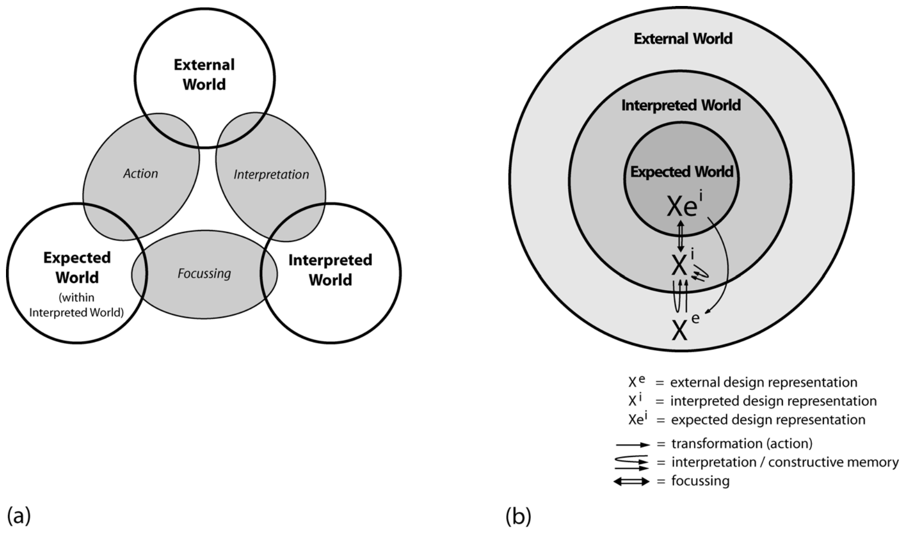
External World
Action
Interpretation
Expected World (within Interpreted World)
Focussing
Interpreted World
(a)
External World
Interpreted World
Expected World
Xe^i
X^i
X^e
X^e = external design representation
X^i = interpreted design representation
Xe^i = expected design representation
= transformation (action)
= interpretation / constructive memory
= focussing
(b)

*Figure 4.* Situatedness as the interaction of three different worlds: (a) general model, (b) specialised model for design representations (after Gero and Kannengiesser (2004)).

Figure 4(b) presents a specialised form of this view implying a designer or design agent (as the internal world) located within the external world and placing general classes of design representations into the resultant “onion” model. The set of expected design representations ($Xe^i$) corresponds to the notion of a design state space, i.e. the state space of all possible designs that satisfy the set of requirements. This state space can be modified during the process of designing by transferring new interpreted design representations ($X^i$) into the expected world and/or transferring some of the expected design representations ($Xe^i$) out of the expected world. This leads to changes in external design representations ($X^e$), which may then be used as a basis for re-interpretation changing the interpreted world. Novel interpreted design representations ($X^i$) may also be the result of constructive memory, which can be viewed as a process of interaction among design representations within the interpreted world rather than across the interpreted and the external world. Both interpretation and constructive memory are viewed as push-pull processes.

The explicit integration of an expected world into a model of interaction accounts for situated designing, as changes in the internal and external world provide the grounds for further changes of the current design process via reformulations of the design state space. It can now be distinguished from other approaches that can be viewed as interaction-based but inconsistent with the idea of reformulating current goals or expectations.

Gero and Kannengiesser (2004) have used this model of situatedness as a basis for a process framework of situated designing derived from Gero's (1990) original FBS framework. Figure 5 presents those parts of their process framework that are relevant in situated design optimization. The three fundamental processes in design optimization, namely synthesis, analysis and evaluation, Figure 1, can now be viewed as consisting of the following partial processes as labelled in Figure 5:

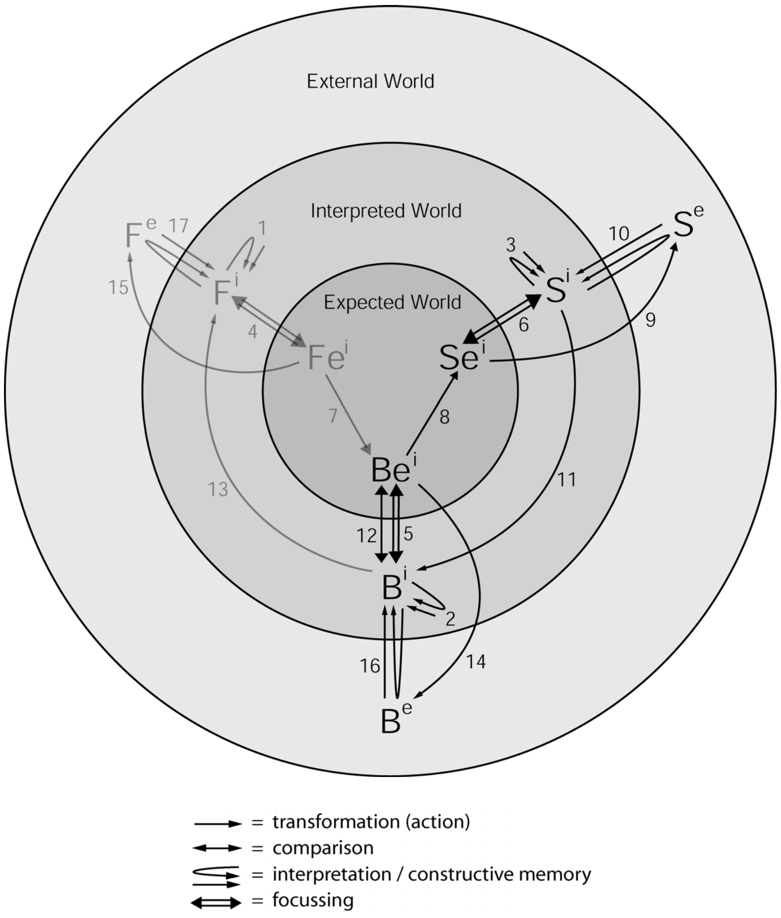

*Figure 5.* An ontological framework for situated design optimization.

- *synthesis*: the transformation of expected behaviour ($Be^i$) into expected structure ($Se^i$) (process 1 in Figure 5) and the subsequent transformation of that structure into an external structure ($S^e$) by means of action (process 2).
- *analysis*: the interpretation of the external structure ($S^e$) to produce an internal structure ($S^i$) (process 3) and the subsequent

transformation of that structure into an interpreted behaviour ($B^i$) (process 4).

- *evaluation*: the comparison of interpreted behaviours ($B^i$) and expected behaviours ($Be^i$) (process 5).

These three fundamental processes are no longer viewed as static, as they operate on design representations that are constructed on the basis of situated interaction. As a consequence, there is potential for changes in what is synthesised and in what the design is analysed and evaluated for. This potential is most significant for optimization tasks that require iterative search procedures involving extensive interaction between expected, interpreted and external structures and behaviours. Situated optimization can be viewed as a process that can modify what it searches as well as what it searches for as it proceeds through the design state space. The design state space relevant for optimization is the union of the structure state space and the behaviour state space, which are represented in Figure 5 in terms of expected structure ($Se^i$) and expected behaviour ($Be^i$).

The processes in Figure 5 representing interpretation (processes 10 and 16), constructive memory (processes 3 and 2), focussing (processes 6 and 5) and action (processes 9 and 14) can be used to view the interactions in Figure 3 in a situated way. This requires the two partners of an interaction to be associated with an internal-external world model. Such a mapping is only meaningful if at least one of the interaction partners can be viewed as an agent (as implied by the concept of internal world). The problem formulation and the result of optimization do generally not have agent qualities. We have decided in this paper not to view the tool as an agent either. We generalise the notion of the designer to include future computational aids to which some of the tasks performed by a human designer may be delegated. We therefore replace the term “designer” by the abstract term “design agent”. This leaves us with the following classes of interactions that can be modelled as situated:

1. interaction between design agent and result
2. interaction between design agent and tool
3. interaction between design agent and problem formulation
4. interaction between design agent and search process
5. interaction of the design agent with itself

Section 5 will illustrate the first four classes of interactions. The fifth class of interactions will be the topic of a forthcoming paper.

## 5. Situated Interactions in Design Optimization

This Section will first describe more thoroughly the different classes of interactions using FBS views of the different components. In particular, we will examine the ways in which action and interpretation can be performed for each of these classes. A case study will provide more detailed illustrations covering all the processes represented in our ontological framework presented in Section 4.

## 5.1. CLASSES OF INTERACTIONS

### *5.1.1. Interaction Design Agent – Result*

Interactions between the design agent and the result of an optimization process generally involve the tool as a means for the agent to act on that result. Actions are viewed here as a causal chain of modifications of the external environment, with only the first element in that chain being brought about directly by the agent. In particular, actions affect the optimization result via a modified problem formulation and/or a modified use of the tool, Figure 6. The role of the tool can be seen here as the agent's "extended effector". However, an action on the problem formulation alone is not yet an action on the result. We view actions as basic constructs whose effects have to reach an external object before it can be called an action on that object.

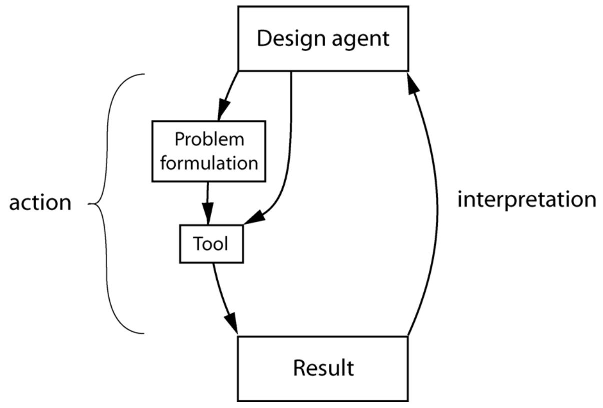

*Figure 6.* Interaction between design agent and result.

Action implies the existence of an expected representation of the effect of that action. If the effect is a change in the result of optimization, then there must be an expected representation of that result in terms of values for an optimum structure and behaviour. Such expectations are often not available prior to running the optimization process. A set of (implicit) ranges of values may be constructed by the agent for routine optimization tasks that belong to well-defined classes of similar tasks. For example, if the current optimization problem differs from a previous one only by marginal variations of parameter values, the agent is justified to assume the location of the two optimum results within the same design state space. Not all of the implicit ranges of values are required to be included in the problem formulation, most notably those describing predicted ranges of performance values of an optimum design. Such predictions are only included in the expected representation of the result. We can view the state space of the result as one with the same dimensions as the state space of the problem formulation but with smaller ranges of values that represent predicted effects rather than just feasibility. Constructing this state space requires experience that is commonly referred to as domain knowledge.

### *5.1.2. Interaction Design Agent – Tool*

When expectations about (implicit ranges of) optimum values are not available or at odds with the results produced by the tool, the agent may focus on interactions with the tool, Figure 7. The tool has a software structure that is external to the agent ($S^e$). It exhibits external behaviour ($B^e$) in terms of what it produces and other performance characteristics (e.g. cost, speed, memory space needed, etc.). The agent may modify parts of the tool's software structure ($S^e$) directly by changing search methods and their parameters (e.g. step sizes and stopping conditions in gradient-based methods, cooling schedules in simulated annealing or mutation and crossover rates in genetic algorithms). The agent may also modify, indirectly, the tool's behaviour ($B^e$) (e.g. its speed) by inputting different problem formulations. The problem formulation is seen as an exogenous effect (Qian and Gero 1996), as it can affect the tool's behaviour while it is not itself part of the tool. In a similar way, interpretations of the tool can be direct or indirect. Direct interpretation includes information provided (displayed) by the tool related to its current settings ($S^e$) or to its performance ($B^e$). Indirect interpretation infers tool performance ($B^e$) via the optimization result. Similar to the notion of indirect action, indirect interpretation is viewed as basic. This prevents the mere generation of the result by the tool from being viewed as a process of interpretation.

### *5.1.3. Interaction Design Agent – Problem Formulation*

Many optimization tasks cannot be performed by concentrating one's interactions only on the result or on the tool. Interactions between the design agent and the problem formulation are often crucial for successful optimization, as they can change both what the tool searches (structure state space) and what the tool searches for (behaviour state space). External representations of the problem can be created or modified directly via action, Figure 8. Such action uses domain knowledge and requirements for the given optimization task to decide on a set of structure and behaviour variables and ranges of values representing feasibility constraints. Interpretation of the problem can be direct or indirect. Indirect interpretation uses (interpretations of) the tool or the result as a basis for inferring additional characteristics of the problem. For example, failure of the tool to produce a result can be used to interpret certain aspects of the problem formulation as incorrect or inappropriate.

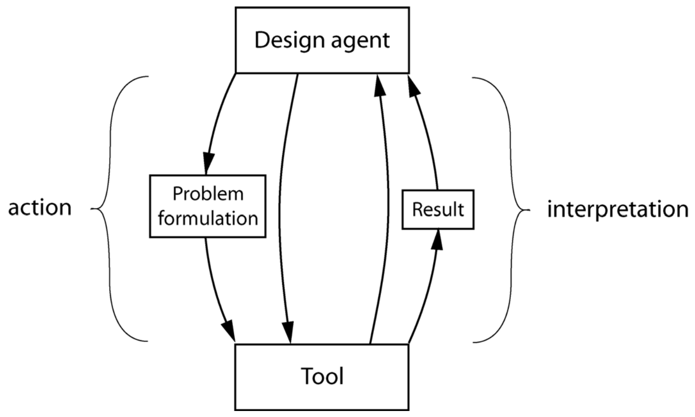

*Figure 7.* Interaction between design agent and tool.

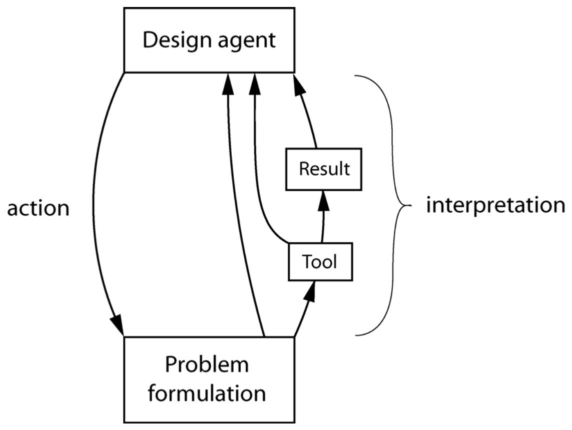

*Figure 8.* Interaction between design agent and problem formulation.

*5.1.4. Interaction Design Agent – Search Process*

It is apparent from the previous examples that interactions are sometimes hard to be viewed in isolation, as the individual components are often connected to each other. For example, an optimization problem with linear objective and constraint functions usually requires a specific class of search methods from the tool, namely linear programming methods. Another example is the case of a routine formulation of an optimization problem, which has a specific set of expectations about ranges of optimum values associated with it. Close connections among the individual components of a search process provide a reasonable basis for viewing the whole search process as one entity, Figure 9.

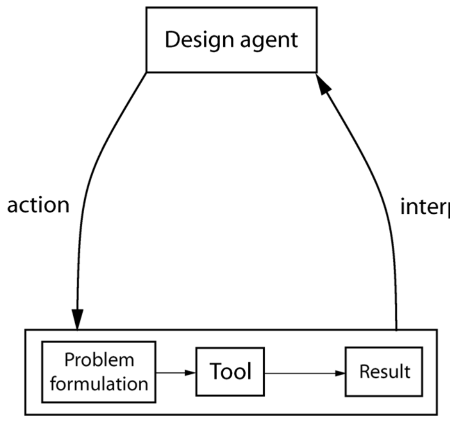

*Figure 9.* Interaction between design agent and search process.

Gero and Kannengiesser (2006) have shown that the FBS ontology can be applied to processes. In particular, the structure (S) of a process consists of its input, a transformation and its output. This maps onto the problem formulation as input, the tool as transformation and the result as output. Behaviour (B) of a process is not different to objects, but for processes typically includes properties such as speed, accuracy, rate of convergence, etc. Function (F) of a process is not different to objects. One function (F) of an optimization process is to fulfil some external (customer) demand for an optimal design. Adopting an FBS view of the search process is useful especially in the early stages of optimization, as it captures initial, high-level functional and behavioural requirements for the process as a whole.

## 5.2. CASE STUDY

To illustrate how the different classes of interactions are represented in our ontological framework, Figure 5, we use an exemplary optimization problem of supersonic aircraft design based on a case study presented by Schwabacher et al. (1998). The design problem requires finding the values for a number of structural variables (S) of the airframe that minimize the takeoff mass of the aircraft (a behavioural variable (B) commonly used in conceptual aircraft design as an estimate of life-cycle cost). The given structure variables are engine size, wing area, wing aspect ratio, fuselage taper length, effective structural thickness over chord, wing sweep over design Mach angle, wing taper ratio and fuel annulus width (Schwabacher et al. 1998).

The purpose of this optimization is to provide decision support in the conceptual design stage. Takeoff mass is numerically computed by a simulator based on selected structure values and on external parameters describing a particular flight mission, Table 2. These parameters vary depending on the phase in the mission. The simulator is used by an optimization tool to analyse a set of design candidates. A number of constraints can be specified to ensure feasibility of the design and to handle possible exceptions (mathematical errors) caused by the simulator.

TABLE 2. Mission specification for a supersonic aircraft (capacity: 70 passengers) (after Schwabacher et al. (1998))

| Phase | Velocity | Altitude | | Duration | Comment |
|---|---|---|---|---|---|
| | (Mach) | (m) | (ft) | (min) | |
| 1 | 0.227 | 0 | 0 | 5 | "takeoff" |
| 2 | 0.85 | 12192 | 40000 | 50 | subsonic cruise (over land) |
| 3 | 2.0 | 18288 | 60000 | 225 | supersonic cruise (over ocean) |

We have developed a possible scenario of how the (human or artificial) design agent proceeds in this optimization task to demonstrate how that agent interacts with the result, the tool, the problem formulation and the search process. The scenario and the corresponding processes in the FBS framework for each interaction are summarised in Tables 3 and 4. The steps of the scenario are only an instance of an optimization process; other instances may use different steps or different execution orders. In the following, we will discuss more thoroughly the FBS processes representing each of the 25 steps of the scenario.

*Step 1*:

The optimization task commences when the design agent interprets the external problem statement usually given in a natural language description such as presented above. Interpretation of the problem formulation is modelled by processes 10 (interpretation of $S^e$) and 16 (interpretation of $B^e$). Process numbers are from the numbering in Figure 5. The above problem statement for aircraft optimization explicitly specifies part of the tool's structure ($S^e$) in terms of its input variables. This aspect of the tool is interpreted via process 10 and produces $S^i$. In our exemplary problem statement, there is also an external representation of a function ($F^e$) of the search process, namely "to provide decision support in the conceptual

TABLE 3. FBS processes involved in different interactions for an optimization scenario. The numbers refer to the processes in Figure 5.

| Steps of an optimization scenario | Design agent – Result | Design agent – Tool | Design agent – Problem formulation | Design agent – Search process |
|---|---|---|---|---|
| 1. Design agent interprets problem statement | - | 10 | 10, 16 | 17 |
| 2. Design agent adds implicit constraints | 3, 2 | 3, 2, 1 | 3, 2 | 3, 2, 1 |
| 3. Design agent forms initial design state space | 6, 5 | 6, 5, 4, 7 | 6, 5 | 6, 5, 4, 7 |
| 4. Design agent formulates the external problem representation | - | - | 9, 14 | - |
| 5. Design agent selects a gradient-based search method and a starting point | - | 9 | - | - |
| 6. Design agent makes tool produce a result | 9, 14 | 14 | - | 9 |
| 7. Design agent interprets result | 10, 16 | 16 | 10, 16 | 10, 11 |
| 8. Design agent evaluates design performance as not satisfactory | 12 | 12 | 12 | 12 |
| 9. Design agent relaxes implicit constraints on design variables and changes search strategy | 6 | 6 | - | 6 |
| 10. Design agent selects multiple starting points | - | 9 | - | - |
| 11. Design agent makes tool produce a number of results | 9, 14 | 14 | - | 9 |
| 12. Design agent interprets these results | 10, 16 | 16 | 10, 16 | 10, 11 |
| 13. Design agent evaluates design performance as not satisfactory and speed of tool as too slow | 12 | 12 | 12 | 12 |
| 14. Design agent identifies design variables to be eliminated | - | - | 6, 5 | 6 |

TABLE 4. FBS processes involved in different interactions for an optimization scenario. The numbers refer to the processes in Figure 5.

| Steps of an optimization scenario | Design agent – Result | Design agent – Tool | Design agent – Problem formulation | Design agent – Search process |
|---|---|---|---|---|
| 15. Design agent reformulates the external problem representation | - | - | 9, 14 | - |
| 16. Design agent makes tool produce a number of results | 9, 14 | 14 | - | 9 |
| 17. Design agent interprets these results | 10, 16 | 16 | 10, 16 | 10, 11 |
| 18. Design agent evaluates design performance as not satisfactory | 12 | 12 | 12 | 12 |
| 19. Design agent decides on using a stochastic search method | - | 6, 5 | - | 6, 5 |
| 20. Design agent modifies the search method to genetic algorithms | - | 9 | - | - |
| 21. Design agent changes the external problem representation to be compatible with the new search method | - | - | 9, 14 | - |
| 22. Design agent makes tool produce a result | 9, 14 | 14 | - | 9 |
| 23. Design agent interprets result | 10, 16 | 16 | 10, 16 | 10, 11 |
| 24. Design agent evaluates design performance as satisfactory but speed of tool as too slow | 12 | 12 | 12 | 12 |
| 25. Design agent accepts results | - | - | - | 5 |

design stage", which is interpreted via process 17 and produces $F^i$. Depending on how the external representations are represented and depending on the design agent's prior knowledge, different interpreted representations can be constructed.

*Step 2*:

The interpreted representations of the problem, the tool and the search process resulting from step 1 are used as cues for the construction of further, implicit requirements (processes 3 and 2). This is important as the explicit problem statement leaves implicit many assumptions that make the optimization model feasible from a domain view. For example, it is implicitly assumed that the aircraft must not stall (Gelsey et al. 1996). This additional behaviour must be generated via constructive memory and may be expressed by introducing the new behaviour variable "lift force" with an associated maximum value. This new requirement is then likely to be used to further constrain the structure state space by imposing an upper bound for values of the structure variable "wing area".

Processes 3 and 2 are also used to construct predictions of a design state space in which the optimization result is located. This includes ranges of values predicted for the location of the optimum design in the structure state space and ranges of values predicted for the takeoff mass, both of which are generated from the agent's experience from interacting with previous aircraft designs for similar flight missions.

Explicit information about the tool and the search process interpreted in step 1 are similarly used as cues for gathering more implicit assumptions relating to their function, behaviour and structure. Important implicit assumptions include a number of appropriate search methods and their settings ($S^i$ of tool) and required speed and accuracy ($B^i$ of tool and search process).

*Step 3*:

After having gathered all the relevant knowledge about the problem, the tool, the result and the search process, the design agent produces an initial state space of the relevant structure (via process 6) and behaviour (via process 5) of all four components. In addition, the design agent constructs expectations of function ($Fe^i$) related to the tool and the search process (via process 4). The decision about which interpreted representations to include in the expected world in each case is driven by the design agent's individual experience. For example, the design agent might select Sequential Quadratic Programming (SQP) instead of GAs as a result of its confidence that the optimum solution can be found fairly quickly by gradient-based search with an appropriate starting point based on the predicted optimum location. The agent may or may not decide to constrain parts of the problem formulation according to its predictions about the result. Such decisions are sensitive to the agent's confidence that these predictions are confirmed by using the tool.

*Step 4*:

The design agent produces an external representation of structure and behaviour of the problem formulation (processes 9 and 14). At this stage, the design agent's action does not have a direct impact on the tool, the result and the search process. The external representation produced uses a specific set of symbols that can be understood by the specific tool. This is a consequence of previous experience of the design agent about using that tool.

*Step 5*:

The design agent initialises the tool to use the SQP method and also specifies step size, stopping conditions and a starting point. Setting these parameters requires adopting tool-specific conventions and formats that the agent must have learned previously. The agent's action on the tool is represented by process 9.

*Step 6*:

Upon initiation of the optimization run, the tool produces a result in terms of the external structure ($S^e$) and performance ($B^e$) of a specific design solution. If this activity is viewed as part of the interaction between the design agent and the result, we can represent it by processes 9 and 14. We can also view it as an external representation of the behaviour ($B^e$) of the tool, produced via process 14. Finally, it completes the external representation of the structure ($S^e$) of the search process, modelled by process 9.

*Step 7*:

In this step, interpretation of the result is represented by processes 10 (interpretation of result $S^e$) and 16 (interpretation of result $B^e$). Interpretation of the problem formulation is modelled by the same processes. Interpretation of the tool is represented by process 16 (interpretation of tool $B^e$). The search process is interpreted via process 10 (interpretation of $S^e$) and process 11 that derives performance ($B^i$) from $S^i$ representing speed of the process.

Differences in interpreted representations of the problem formulation are often based on differences in the external representation of the result. For example, the tool may display additional behaviours ($B^e$) of the result (e.g. the drag force of the aircraft computed by the simulator) that were neither specified in the explicit problem statement nor constructed previously by the design agent. Such behaviours, if found to be relevant based on the agent's previous experiences, may be the driver for reformulating the problem.

*Step 8*:

Evaluation includes a comparison of expected and interpreted behaviour (process 12). This applies to all $Be^i$ and $B^i$ constructed so far, of all entities interacting with the design agent. In our scenario, only the result is evaluated as not satisfactory, as the design performance (takeoff mass) is not within the expected ranges of values. For example, let us assume that the takeoff mass found is 161,000 kg, while the expected lower and upper bounds are 130,000 kg and 140,000 kg, respectively. The outcome of evaluation is clearly dependent on such previous expectations, which have been constructed using first-person knowledge.

*Step 9*:

The deviation of the interpreted from the expected result has made the design agent realise that the current optimization task differs from previous tasks in a more significant way than initially presumed. Maintaining the previously expected ranges of values for minimum takeoff mass, the design agent now relaxes some of the implicit constraints for the location of the optimum in the structure state space. This eliminates part of the previously expected structure ($Se^i$) from the design state space via process 6, which also affects the search strategy to be changed from using a single start to using random multiple starts. The change in strategy concerns the expected structure ($Se^i$) of the tool and of the search process, represented by process 6. This change is based on the agent's confidence that the optimization resembles previous experiences to an extent that allows applying the same fundamental class of search methods, namely gradient-based search, to the current problem.

*Step 10*:

The changes in the expected world have consequences in the external world by changing the settings of the tool to implement a random multi-start, gradient-based search strategy. This is represented by process 9. Tool-specific experience needs to be considered in devising this action.

*Step 11*:

Running the optimization tool now produces a set of results. As in step 6, this activity is represented by processes 9 and 14 for the result, process 14 for the tool and process 9 for the search process.

*Step 12*:

Every result is now interpreted. The same processes are used here as described in step 7: Processes 10 and 16 to model interpretation of result and problem formulation, process 16 to model interpretation of the tool and processes 10 and 11 to model interpretation of the search process. For example, the agent may chunk the multiple results of the multiple search runs into just one result, namely that of a multi-start strategy. This ability is a consequence of the push-pull character of interpretation that structures its world according to current goals and knowledge.

*Step 13*:

All evaluation processes of all entities are represented by process 12. In the example, the performance of all results produced using the multi-start strategy is evaluated as still not satisfactory. In addition, the observed speed ($B^i$) of both the tool and the search process is evaluated as too slow in comparison to the speed that is usually expected from them ($Be^i$). These evaluations are based on the agent's subjective experience.

*Step 14*:

The poor results achieved so far make the design agent question the appropriateness of the problem formulation. Specifically, the agent assumes that discontinuities in the search space have prevented the tool from finding the true optimum. This assumption is constructed from the agent's dissatisfaction with the result and from the agent's individual experience captured as strategic knowledge. Monotonicity analysis could be used by the agent to identify inequality constraints that can be incorporated into the search space (Papalambros and Wilde 2000; Schwabacher et al. 1998). This reduces the search space and potentially removes the critical discontinuity. The modification of the design state space represented by processes 6 (reformulation of $Se^i$) and 5 (reformulation of $Be^i$).

*Step 15*:

The design agent modifies the external representation of structure ($S^e$) and behaviour ($B^e$) of the problem formulation according to the new design state space (processes 9 and 14). The same tool-specific experience is required as previously.

*Step 16*:

A new set of results is produced by running the optimization tool on the reformulated problem. This modifies the external structure and behaviour of the result (processes 9 and 14), the external behaviour of the tool (process 14) and the external structure of the search process (process 9).

*Step 17*:

Every result is interpreted using the same processes as described for steps 7 and 12: Processes 10 and 16 to model interpretation of result and problem formulation, process 16 to model interpretation of the tool and processes 10 and 11 to model interpretation of the search process. The multiple results may again be interpreted as just one.

*Step 18*:

All evaluation processes of all entities are represented by process 12. The design performance of the result is still evaluated as unsatisfactory based on the design agent's expectations and experience.

*Step 19*:

The design agent now decides on abandoning the gradient-based search method in favour of a stochastic one, namely genetic algorithms. This modifies expected structure and behaviour of the tool and the search process (processes 6 and 5). This change of search method is a consequence of the agent's dissatisfaction with the gradient-based search class for the current problem and the agent's strategic knowledge.

*Step 20*:

Changing the search method to GAs and choosing values for GA parameters such as mutation rates and crossover rates is represented by process 9 (action on external tool structure ($S^e$)). Tool-specific experience and general experience with setting GA parameters need to be considered in devising this action.

*Step 21*:

The design agent modifies the external representation of structure and behaviour of the problem formulation (processes 9 and 14) to become compatible with the genetic algorithms used by the tool. Specifically, a representation of the problem in terms of genotypes and phenotypes is required to be understood by the tool. This re-representation is based on the agent's experience of using GAs and of the specific input format of the tool.

*Step 22*:

A new set of results is produced by running the GAs. This modifies the external structure and behaviour of the result (processes 9 and 14), the external behaviour of the tool (process 14) and the external structure of the search process (process 9).

*Step 23*:

This step involves process 10 modelling the interpretation of external structure ($S^e$) of the problem, the result and the search process and process 16 modelling interpretation of external behaviour ($B^e$) of the problem, the result and the tool. In addition, process 11 is used to describe the interpretation of the speed (interpreted behaviour ($B^i$)) of the search process, derived from the time difference between its input (the problem) and its output (the result) that are components of its interpreted structure ($S^i$). This requires time-based representations of the structure ($S^i$) of the search process that are constructed as a consequence of the agent's interest in speed.

*Step 24*:

All evaluation processes of all entities are represented by process 12. Here the design performance of the result is evaluated as satisfactory; however, the speed of the search process is still evaluated as too slow based on the design agent's expectations and experience.

*Step 25*:

The design agent accepts the optimum result achieved by the GAs and satisfices the search process by relaxing the constraint (expectation) on its speed ($Be^i$) (process 5). This is a consequence of the agent's realisation that, based on the outcomes of the previous search processes and based on limitations of knowledge and technologies, a better result in terms of design and process performance is unlikely to be achieved.

## 6. Conclusion

We have proposed a model of situated design optimization that represents a departure from previous models. It is based on the notion of interaction as the fundamental driver of how optimization iterates through a search space and across different search spaces. Most previous models were focused on representing optimization as a process that traverses a fixed search space. Although there have been some models that assume search spaces to be flexible, they fail to adequately capture the experience needed to modify them. Our interaction-based view represents a major step beyond the "black-box" view inherent in these models. It is distinguished from other approaches by its ability to describe various changes in how optimization proceeds as a result of its own trajectory through current and previous search spaces. We claim that this provides a basis for better understanding optimization as a situated activity whose outcomes are determined by the individual experiences derived from the design agent's interactions.

Our model of situated design optimization can be used as a foundation for developing a new generation of optimization tools. Such tools would be conceived of as computational agents that gain experience by interaction. This would provide the potential for a broader scope for automated optimization support, which could include the formulation and reformulation of optimization problems and the selection of search methods. As a result, the performance of optimized designs and the efficiency to generate them is likely to be significantly enhanced.

The FBS ontology has been shown to be a foundation for representing design knowledge in a general and uniform way (Gero 1990). This is beneficial for implementing situated design optimization systems that require learning and adaptation to novel design situations. We can represent both knowledge about objects and knowledge about processes using the FBS ontology (Gero and Kannengiesser 2006). This allows agents to learn from a variety of interactions, including interactions with the problem formulation, the tool, the result and the search process.

One of the most important classes of interactions in optimization involves the problem formulation, as it determines the bounds of search in terms of structure and behaviour state spaces. Learning from such interactions usually results in generative, procedural problem representations rather than in specialised representations for every individual optimization case. This turns the original object-centred view of problem formulation into a process-centred view. Process-centred representations of problem formulation are generally termed strategies, as they precede and constrain the search process carried out by the tool. The uniform representation of all processes provided by the FBS schema allows an agent to treat problem formulation as a form of meta-search – the search for a state space in which another search is to be carried out. This enables learning of strategic optimization knowledge that can itself be modified as the agent uses this knowledge for interacting with its own strategies.

## Acknowledgements

This research is supported by a grant from the Australian Research Council. Parts of this paper were presented at the 8th International Conference on Design and Decision Support Systems 2006 in Eindhoven, The Netherlands.

## References

Bartlett F.C. (1932 reprinted in 1977) *Remembering: A Study in Experimental and Social Psychology*, Cambridge University Press, Cambridge.

Bickhard M.H. and Campbell R.L. (1996) Topologies of learning, *New Ideas in Psychology* **14**(2): 111-156.

Clancey W.J. (1997) *Situated Cognition: On Human Knowledge and Computer Representations*, Cambridge University Press, Cambridge.

Dewey J. (1896 reprinted in 1981) The reflex arc concept in psychology, *Psychological Review* **3**: 357-370.

Ellman T., Keane J., Banerjee A. and Armhold G. (1998) A transformation system for interactive reformulation of design optimization strategies, *Research in Engineering Design* **10**(1): 30-61.

Gero J.S. (1990) Design prototypes: A knowledge representation schema for design, *AI Magazine* **11**(4): 26-36.

Gero J.S. (1998) Conceptual designing as a sequence of situated acts, *in* I. Smith (ed.) *Artificial Intelligence in Structural Engineering*, Springer-Verlag, Berlin, pp. 165-177.

Gero J.S. (1999) Constructive memory in design thinking, *in* G. Goldschmidt and W. Porter (eds) *Design Thinking Research Symposium: Design Representation,* MIT, Cambridge, MA, pp. 29-35.

Gero J.S. and Fujii H. (2000) A computational framework for concept formation for a situated design agent, *Knowledge-Based Systems* **13**(6): 361-368.

Gero J.S. and Kannengiesser U. (2004) The situated function-behaviour-structure framework, *Design Studies* **25**(4): 373-391.

Gero J.S. and Kannengiesser U. (2006) A function-behaviour-structure ontology of processes, *Design Computing and Cognition'06*, Springer-Verlag, Berlin, to appear

Gelsey A., Schwabacher M. and Smith D. (1996) Using modeling knowledge to guide design space search, *in* J.S. Gero and F. Sudweeks (eds) *Artificial Intelligence in Design'96*, Kluwer, Dordrecht, pp. 367-385.

Jozwiak S.F. (1987) Improving structural optimization programs using artificial intelligence concepts, *Engineering Optimization* **12**: 155-162.

Mackenzie C.A. and Gero J.S. (1987) Learning design rules from decisions and performances, *Artificial Intelligence in Engineering* **2**(1): 2-10.

Nath G. and Gero J.S. (2004) Learning while designing, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **18**(4): 315-341.

Papalambros P. and Wilde D.J. (2000) *Principles of Optimal Design: Modeling and Computation*, Cambridge University Press, Cambridge.

Pardalos P.M. and Resende M.G.C. (eds) (2002) *Handbook of Applied Optimization*, Oxford University Press, New York.

Parmee I.C. (1996) Towards an optimal engineering design process using appropriate adaptive search strategies, *Journal of Engineering Design* **7**(4): 341-362.

Parmee I.C. and Hajela P. (eds) (2002) *Optimization in Industry*, Springer-Verlag, London.

Qian L. and Gero J.S. (1996) Function-behaviour-structure paths and their role in analogy-based design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **10**: 289-312.

Schön D.A. (1983) *The Reflective Practitioner*, Harper Collins, New York.

Schön D.A. and Wiggins G. (1992) Kinds of seeing and their functions in designing, *Design Studies* **13**(2): 135-156.

Schwabacher M., Ellman T. and Hirsh H. (1998) Learning to set up numerical optimizations of engineering designs, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **12**: 173-192.

Smith G.J. and Gero J.S. (2005) What does an artificial design agent mean by being 'situated'?, *Design Studies* **26**(5): 535-561.

Stahovich T.F. (2000) LearnIT: An instance-based approach to learning and reusing design strategies, *Journal of Mechanical Design* **122**(3): 249-256.

Suwa M., Gero J.S. and Purcell T. (1999) Unexpected discoveries and s-inventions of design requirements: A key to creative designs, *in* J.S. Gero and M.L. Maher (eds) *Computational Models of Creative Design IV,* Key Centre of Design Computing and Cognition, University of Sydney, Sydney, Australia, pp. 297-320.

Wilde D.J. (1978) *Globally Optimal Design*, Wiley, New York.

Ziemke T. (1999) Rethinking grounding, *in* A. Riegler, M. Peschl and A. von Stein (eds) *Understanding Representation in the Cognitive Sciences: Does Representation Need Reality?*, Plenum Press, New York, pp. 177-190.