DMP: a well-founded decision-making procedure

R.W. van der Pol
Thoughtwell
ruud@thoughtwell.nl

F.J. Wiesman
Academic Medical Center
dept. Medical Informatics

Abstract. The article introduces DMP, a new Decision-Making Procedure. It covers the entire decision process, is independent of the specific task type, and based on realistic insights on human cognition. Theoretically, it is more effective than known procedures. It is compiled from known procedures in various research areas and case-studies. Next to eight main steps, such as gathering knowledge and setting goals with evaluation criteria, DMP has a separate control step for controlling iteration through the main steps. Surprisingly, the content of each step, including the control step, contains the use of (1) trial and error, (2) knowledge and reasoning, including knowledge from old cases (i.e., from experience) and recursion to a new instance of DMP – which makes DMP deep - and (3) a combination thereof. DMP may serve as a core procedure for Systems Engineering, as well as for improved expert systems.

Introduction

In the past, various decision making procedures have been developed in different research areas. Most of these procedures have the disadvantages of

(1) being useful only for specific subject areas and problem types, and/or

(2) treating only a few steps of the whole decision-making process, and/or

(3) having wrong steps, due to being based on wrong ideas of knowledge and thinking.

The first disadvantage needs no explanation. With respect to the second, the existing procedures only cover a part of the decision making, or cover all steps superficially, not in-depth.

Regarding the third disadvantage, a procedure may for example not take into account that persons only have (subjective and time-dependant) beliefs instead of 'objective' knowledge. A belief thus may be augmented with details or may be revoked someday. Such a procedure will, among other things, have no steps for dealing with changes of beliefs.

These three disadvantages result in limited effectiveness of the known existing procedures.

Some AI researchers have asserted that it is not at all possible to provide 'a top level procedure that functionally describes human decision making'. For instance, Hofstadter mentioned that a person may 'jump to a higher level' when she gets stuck or finds herself trapped in an endless loop, and that a computer cannot make such a jump, at least not in the known, symbol manipulating computers (Hofstadter, 1980).

Nevertheless, we surmised that a top-level procedure could be formulated indeed. The notion of a general decision making procedure was based on his own experiences that, during decision making, one applies a straightforward procedure all the time, which depends – among other things - to a large degree on recursion and/or iteration.

On the basis of that idea, a procedure for decision making was created that is more comprehensive than existing procedures, in the sense of being independent of the specific task type and also covering more of the steps of decision making. Moreover, the procedure had to be based on realistic insights on human cognition – in order to make it more effective. The procedure would be for individual persons; group dynamics etc could be added at a later stage.

This paper reports the positive result of this attempt. DMP (the name is based on Decision-Making Procedure), was compiled from existing procedures in various research areas and our own case studies, and streamlined after careful analysis. Although most of the elements in the procedure are known *per se*, we did not find them assembled explicitly and they were not expressed in coherent terms – and as such not readily accessible. Some elements are new, such as the content of the steps. The latter includes the control step that has a similar outline as the procedure itself.

The practical merit of DMP is that it may be used as a guide for decision making by human beings, leading to improved decisions. Such an application is possible in, e.g., Systems Engineering, in general management, and, after formalisation, in a next generation of expert systems.

Research method

We formulated DMP on the basis of the following sources of information:

- Existing literature, in the areas of AI, economics, psychology, (industrial and systems) engineering; this was done for learning what procedures already existed,
- Scientific literature in cognitive science, communication, reasoning and formal logic; this taught us realistic insights on human thinking, and
- Case studies of human decision making; this in order to identify steps missing or mistaken in existing procedures.

We also made an analysis of said literature and cases, thereby extracting useful concepts and steps. After that, we arranged these steps orderly and obtained DMP. The case-studies were based on tasks

done by an author or a close friend, because the authors wanted to get a feeling for how realistic the methods described in the literature were, and whether these methods contained all the relevant steps and other concepts. In other words, the author wanted to know whether the known methods are compatible with the approach followed by him and his acquantainces. In analysing the cases, we observed a number of steps that were not found in the related research, and that were useful in developing DMP. We also found that several existing procedures did not take into account a realistic view of knowledge and thinking. This confirmed our expectation that an improved DM procedure could be formulated. Since we did not test DMP experimentally, the result is theoretical. Yet, its elements are obtained from practical, real cases.

2 Related research

Decision making has been studied in various research disciplines and has yielded many publications. In this section, we explore computer-based decision making (as seen in chess and AI), and methods for decision making (e.g. Systems Engineering). Economic decision theory is not discussed here because the relevant results are already incorporated in AI.

2.1 Human and Computer Problem solving

The problem-solving behavior of human beings was extensively studied by Newell and Simon (e.g., Newell and Simon, 1972; Newell, 1990). Their goal was not only to understand problem solving, but also to be able to program computers such that they could solve problems. Human problem solving, being the most powerful available type of problem solving at that time, was a source of inspiration for them.

Newell and Simon studied various subject areas and task types, starting with crypt arithmetic, logic, and chess (Newell and Simon, 1972). In these studies, they found that problem solving followed the same pattern when abstracted from the specific task type, specific subject, and specific circumstances. They formulated the pattern into an algorithm, called an 'Intelligent Problem Solver' (IPS).

The essence of the algorithm is that it starts from a *given* problem formulation, defining the initial state, a goal state, and operators (i.e., operation types) that can be applied to go from one state to another. Solving the problem means finding scenarios (i.e., sequences of operations and states) that lead from the initial state to the goal state. In chess, for example, the initial state is the starting position of the pieces on the board, the goal state is any winning state, and the operations are the moves of the pieces allowed for according to the rules of the game. The operators are being used to go from one state to another state, in order to reduce the *difference* between a state in a scenario and the goal state. With the aid of a table in which operators and differences (or changes thereof) are linked, it is possible to select an appropriate operator. What operator is considered appropriate is often selected on the basis of heuristics, i.e., rules of thumb, statistically oriented knowledge obtained from experience, in order to reduce the number of operation-sequences (i.e., scenarios).

The number of operation sequences, i.e., scenarios, may become incredibly large. For instance, when a computer plays a game of chess, the number of scenarios is so large[1] that it is not feasible for even the fastest computer to enumerate them before the collapse of our solar system. In computer chess, the learning ability has been introduced, for example by means of adapting the weights in an evaluation function after each game.

Newell and Simon also took into account the physical aspects of human information processing, such as the size of human memory (limited) and the number of inferences a human can make in a second, or processing speed (also limited). For instance, it is common that a person, during the construction of solutions (i.e., scenarios), only remembers one or two previous states. She forgets the other previous states, in order to reduce the mental effort required.

The way a problem is solved differs between novices and experiences persons. This is seen in chess and medicine. Essentially, novices remember individual facts, for instance the positions of the individual pieces in chess. When they solve a problem (or, play in chess), they reasons explicitly, stepwise. In contrast, experienced persons essentially use their intuition in evaluating a chess position, and remember the positions of the individual positions of the pieces by their relationships. In medicine the same is seen: experienced doctors (Schmidt et al., 1987) and nurses Benner and Tanner (1987) intuitively 'see' patterns (as opposed to individual things). For the case of experienced nurses, the combination of intuition and analytic reasoning was observed.

Based on the abovementioned initial research on computer chess, the field of Artificial Intelligence has arisen. AI has provided us with many additional techniques for problem solving. Some of the techniques reflect human problem solving, others are only functionally equivalent to human thinking and differ from that in their implementation.

Here, we mention two relevant techniques: rule-based reasoning (RBR) and case-based reasoning (CBR). These are techniques for explicit reasoning, since these can be expressed in an explicit prescriptive procedure. Other techniques in AI are implicit techniques, such as Machine Learning, Artificial Neural Networks and Genetic Algorithms; all these techniques serve to provide scenarios, or solutions, for a given problem.

Rule-based reasoning is used in expert systems. These systems use derivation rules to derive new beliefs from given beliefs. The mechanism is that as described aboven, for Newell. In an expert system, the derivation rules serve as operators for state-transitions. The derivations may occur by using the rules by forward inferencing (i.e., deduction) as well as backward inferencing (i.e., reduction) (Bratko, 1990). Usually, the derivation mechanism implements 'resolution', a proof technique of predicate logic (Bratko, 1990; Rowe, 1988). Sometimes each inference is accompanied by the calculation

[1] A rough estimation of the number of possible different positions is 10^{120} (Breuker, 2006), which may be regarded a lower bound for the number of scenarios.

of a chance that its conclusion holds, given the chances that its premises hold (Buchanan and Shortliffe, 1984).

Two main manners of finding a path between begin state and end state (i.e. a solution or decision) are distinguished. Depth-first search is a technique that, when the space of possible solutions is drawn as a decision tree, explores the paths in the tree first to their full depth, i.e., complete solutions, before going to a next path, and starts on the left side of the tree. In contrast, in breadth-first search all partial scenarios of only one step depth are conceived of first, then those with a depth of two steps, and so on. Thus, the tree is searched over its breadth first, then over its depth.

Some expert systems use heuristics. First, the derivation rules may be heuristics, such as 'if it rains, one usually becomes wet'. More sophisticated types of heuristical rules were devised, e.g., by Clancey (1985). Second, the way of searching through the search space, i.e., control thereof, may be done by heuristics. For instance, a branch of the search space may be 'pruned' if it seems to be unpromising. In chess, such a branch may represent the giving away of one's own queen. That branch is an unpromising branch for winning the game.

The above manners of finding solutions in RBR are just the main methods; much more refined approaches exist.

In CBR, a whole case is represented as a collection of facts, typically in the shape of a frame (i.e., a template containing pairs of fields, each pair being a type/role and a slot for an instance). If a new case comes up, similar old cases are being used to fill out a slot left open; the value of the open slot is copied (or modified) from a similar old case (Aamodt and Plaza, 1994). CBR is similar to Machine Learning, but uses more explicit reasoning for determining the similarity of cases, whereas ML uses statistics. In fact, CBR may be regarded a variant of RBR, with complex rules (Van der Pol, 2000). However, it is meant for the representation of small numbers of individual cases, i.e. when it is not efficient or appropriate to formulate general rules.

Summarizing, we may use the notion of goals, (partial) scenario, position, and in particular evaluation function and learning ability and intuition/reason.

2.2 Methods for decision making

Next to the above computer-based decision-making methods, there are several methods for designing software systems and other, technical systems. These served as inspiration for our procedure, DMP. We discuss a few general methods very briefly and that of Systems Engineering into some detail. The reason for individually discussing SE is that it is the method that most resembles DMP.

2.2.1. Various methods

KADS, a method designed to create knowledge-based systems, or expert systems, was developed in the 1980s and 1990s at SWI, of the University of Amsterdam. With KADS, reuse of knowledge was made possible by explicitly distinguishing between domain (or 'product') knowledge on one hand and task (or process) knowledge on the other hand.

The waterfall model is an often-used schema for the development of products made of hardware and/or software. It consists of the stages: (1) requirements analysis; (2) system design; (3) program design; (4) coding; (5) unit & integration testing; (6) system testing (validation and verification); (7) acceptance testing; (8) operation and maintenance.

Each stage starts only after the preceding stage has been completed. This is schematically depicted by drawing diagonally from top-left to bottom-right, with an arrow going from a stage to its immediate successor. The arrows represent a 'flow' of information, a metaphor of the water in a waterfall – hence the name 'waterfall model'. It is considered a disadvantage of this way of presenting the process, that it is not treated as an iterative process, i.e., that the progressive insight is not taken into account (Pfleeger, 2001). For instance, the requirements are subject to change during the process (thus, beliefs change), since only during the later stages one becomes aware of what is feasible and useful, and what is not.

The waterfall model has many other methods as successors, providing a more flexible process. One of these methods is Extreme Programming (XP; Beck, 2004). We mention only the elements that are relevant for our purpose. This software development method has as basic assumption that insight progresses during a project, in particular the goals and requirements regarding the system. Both in the waterfall model and XP, the requirements are formulated initially. In XP, a part of the system is designed and built in only a few weeks, and then presented to at least one of its future users. This iteration loop allows for a check, adaptation, and refinement of the requirements. In this setting, the system is created piece by piece, in close collaboration with the user, who is part of the development team. An additional element of XP is that the software engineer writes the code for testing a function *before* designing and coding the function itself. In terms of DMP, one is forced to conceive of the goals and evaluation criteria first, before conceiving the solution/scenario.

The Delft Method of Industrial Design (Rozenburg and Eekels, 1995) is a step plan for developing a design. It is similar to the waterfall model, but has one interesting added concept: the distinction between a concept design and a detailed design. The concept design is iterative; the detailed design not, for practical reasons. Too many persons are involved then, so that iteration would make it more difficult to control.

2.2.2 Systems Engineering

Systems Engineering (SE) comprises a set of methods for the development of systems, in terms of complex technical products and processes as well as their accompanying infrastructure and services. The methods of SE share a common general method. For each situation, the details are filled in depending on the situation.

In the US, INCOSE (INternational Counsel On Systems Engineering, the main institute for SE professionals) has written a handbook (Defense Acquisition University Press, 2001) containing 'best practices' from its members in industry and government. The methods of SE according to the handbook take into account that large, complex technical systems are often developed by several institutions collaborating and by persons from different disciplines. It also takes into account the fact that, apart

from these stakeholders, there are other stakeholders in each project, such as the clients, users, and component suppliers. All stakeholders have to collaborate to obtain the desired system. Therefore they have to understand and accept the project goals, and they have to communicate and tune their individual contributions and interests. Additionally, for a large system, the subdivision of a system into many subsystems with accompanying interfaces, and the re-integration of the realized sub-systems into a whole, balanced realized system is essential.

In our attempt to formulate a general procedure for problem solving, we only focus on the procedure of SE, and leave out the product life cycle and stakeholders – because they are not needed in all decisions. They may be added in relevant cases.

We describe the overall method of SE on the basis of Figure 1. The figure shows three main steps:

1. Requirements analysis: writing down what requirements the system will have to satisfy, and analysing and reformulating them to reduce contradictions and to increase their feasibility.

2. Functional analysis/allocation (i.e., design): defining a functional schema, as a beginning of the system design.

3. Synthesis: defining a physical system, that performs the functions of the functional schema.

We recognize two iteration loops: (a) between steps 1 and 2, and (b) between steps 2 and 3. These are present in DMP, but the first loop is more general, extending over other steps as well.

In addition to the elements shown in the figure, SE pays attention to balancing the subsystems in the overall system. Additionally, SE recognizes the steps of verification and validation – only one verification step is shown in the figure, while in fact several verification and validation steps may be present, depending on the specific system being developed. Verification essentially is a theoretical check to determine whether the system is what the requirements asked for. Validation is essentially an experimental check whether the system is what the user needs.

As process input, information on the context is mentioned, such as needs and objectives of the customers/users, and missions (e.g., in what manner the system will be used). Also, knowledge from earlier projects is mentioned, i.e., in our terms, (part of) domain knowledge.

The realization of the design, i.e., selected scenario, is not shown in the figure above, but is, according to the handbook, part of the method. For instance, during realization, verification and validation as well as integration of subsystems play an important role.

Evaluation of a finished project, and learning from that evaluation, is not in the figure above, but is practiced – sometimes as a separate project, as observed by the authors.

The Figure of SE does not mention recursion. In our cases, reported in Section 3, it became clear that recursion is essential. Also from SE-practitioners in defense and aerospace companies, we know that recursion is applied indeed. Usually, it is named decomposition, which is one kind of recursion.

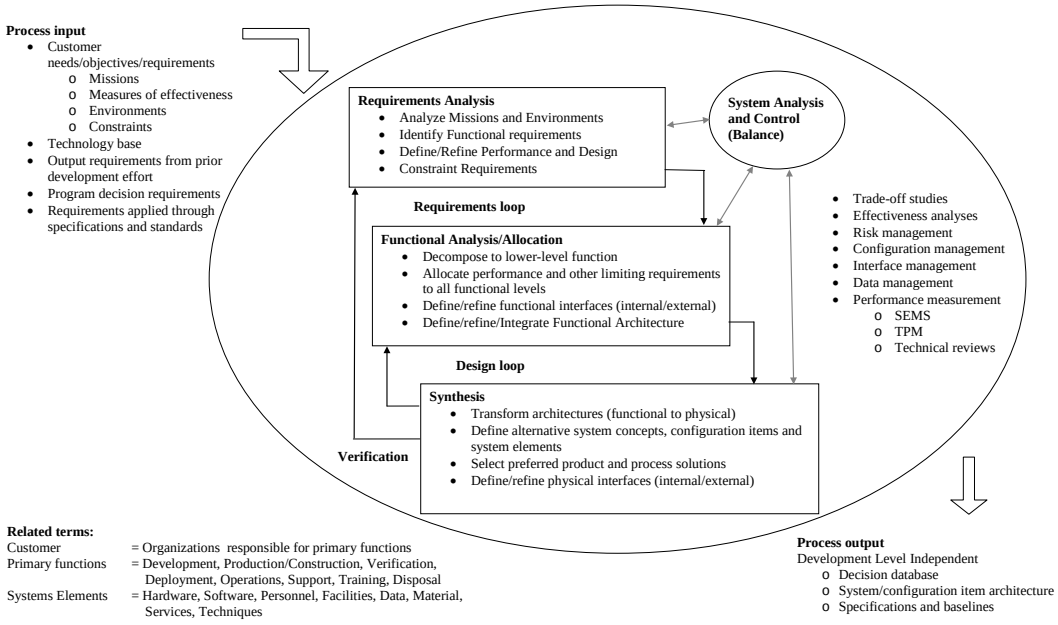


Figure 1: The Systems Engineering process, after (Dept. Of Defense, 2001).

2.3 Cognition

2.3.1 Intuitive and rational thinking

Cognitive neuroscience distinguishes two mechanisms for problem solving, a rational and an intuitive mechanism (see, e.g., Gazzaniga, Ivry, and Mangun, 1998). We already mentioned these mechanisms above. They are closely related to conscious respectively unconscious thinking.

Rational thinking is characterized as thinking whereby one uses – consciously - all his knowledge in the best known manner, according to the best accepted ideas about effective conscious thinking, without mistakes and without being led or distracted by reasons or goals other than the original goal of the thought process at hand. What is rational thinking therefore is ultimately dependent on the opinions of many.

Intuitive thinking is thinking whereby an unconsciously active mechanism is involved, leading to thoughts, usually accompanied with a feeling, together giving an impression about which decision or solution is good and which not. These feelings may be based on training or experience, and may be very effective. Intuitive thinking may take several factors in account, many more than a human is able to deal with consciously, in rational thinking (typically four).

Dijksterhuis et al. (2006) add to the former, that intuitive decisions are better (i.e., more effective) than rational decisions, if all relevant information is gathered and consumed beforehand. He argues that a mixture of rational and intuitive thinking seems to be very effective: first gather relevant information, consciously; then do something else that takes one's attention. After at least 10 minutes, decide on the basis of intuition. In experiments, this has proven to be more effective than decisions without such a time-off in the process, at least for problems having some complexity, such as buying a car; very complex problems, such as the reorganization of a multinational company, have not been tested.

We add that intuition may also be wrong, as is clearly visible in examples given by Dijksterhuis himself. We incorporate the above concepts and insights into DMP.

2.3.2 Characteristics of brain activity

The following are relevant ideas on brain activity and control thereof, taken from cognitive science.

* The brain is a neural network, in which a signaling activity occurs.

* The aforementioned activity of the brains, of the natural neural network, constitutes the controlling of a person's human body and mental tasks. In daily life, the brain activity is perceived as distinct activities, such as observation, pattern recognition, perception, recalling from memory, pronouncing a word or sentence. Each activity (block) may take place in many or few neurons, and a large or small area in the brain, and two or more activities may use one or more of the same neurons. Thus, the distinction of tasks is only at a functional level.

* This succession of distinct mental activities takes place both in serial and in parallel order.

* Usually, a person is aware (conscious) of only one mental activity at a time, sometimes also zero. This is known from what persons report only; we have no means yet to study consciousness in an objective manner. Some persons report to be aware of a few things/thoughts simultaneously. Still, a lot of mental activity seems not to reach consciousness, for example most of the activity for controlling functions of the body.

* Figure 2 shows the above in a schematic form. The grey boxes represent activities of which the person is aware.

* Two or more activities may become, by learning, so strongly connected that if one activity is executed, the other will also be executed. This means that the two previously distinct activities become one integral activity (or, procedure).

* An activity of a specific type is not necessarily always the same; in each time it may be executed a little bit different. This represents that somewhat different neurons are involved, depending on the other activity in the brain, the strengths of the connections involved at that time and the hormonal levels at that time. (This corresponds to (1) execution of a procedure such as in a computer, with differing parameters each time, and (2) changes in that procedure over time, e.g., due to learning or forgetting).

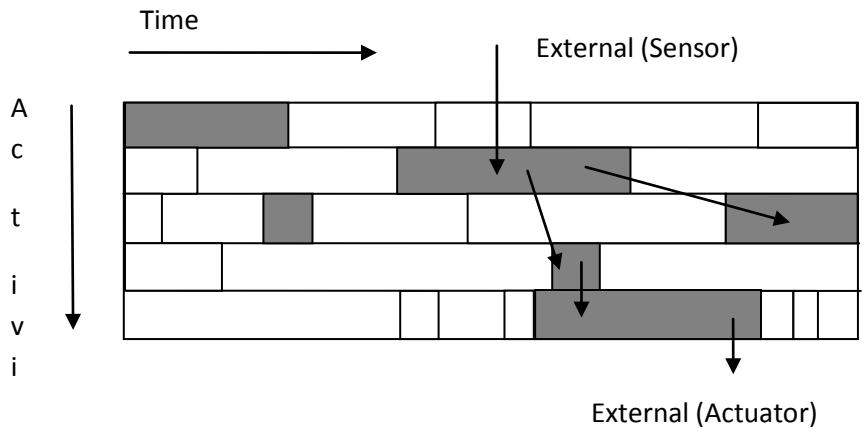


Figure 2: Parallel activities in the human brains.

* The content, or subject, of an activity, may be reflective. In other words, the activity may have another activity as subject and evaluate that subject.

* What controls the flow of activities? In other words, what triggers an activity? First, the sources: (1) Influences from the outside, via the senses, (2) One or more other activities. Second, again from introspection, the subjective experience of persons, it is believed that, in general, a person is able to maintain a line of thought, i.e., a succession of mental (and physical) activities that are coherently working towards a single shared goal.

It is also known from experience that such maintaining a line of thoughts, i.e., such focus or concentration, can be maintained for a while, ranging from a few minutes to several hours. After that, the attention drops, no matter how hard the person tries to maintain it.

Causes for the decline of attention are

- weariness: a person gets tired or exhausted. This has biochemical causes – some of the nutricial substances for the brain are depleted,

- the decision is finished, or

- other activities become predominant, by means of the aforementioned external stimuli via the senses or by the other mental activities. For example, a person may be stung by a mosquito.

Yet another fact known from daily life is that the capacity to maintain attention differs among persons. Moreover, that capacity can be trained, with as result that the capacity within a single person increases (and may decrease again when training is stopped). The training may also result in an increased capacity to ignore distracting stimuli.

Endless loops

As a final remark, we discuss a special issue in control: the issue of getting stuck in problem, for example by getting stuck in an endless loop. We mentioned this 'problem' in the introduction. As we mentioned, this problem was posed by Hofstadter (1980). The question is how a person does not get stuck in reasoning and rapidly finds her way towards conclusions, where, in contrast, getting stuck or getting lost may and will happen in proposition logic and predicate logic.

There is at least one mechanism known to help in jumping out of being stuck and out of an endless loop: reflection. From the above, it becomes clear that each person gets detracted after some time. Afterwards the person can take up the task and then (by habit, and/or by training) will also perform reflection on his own thinking: "What am I doing? I'm not at all making progress. I'm in a loop", and then becomes capable of 'jumping' out of the loop, in a simple manner: choose to do something else, or tackle, approach, the problem in a different way. Differences among persons with respect to the power of jumping out of the loop or out of the situation of being stuck are present. These differences are caused by a person's nature or are learned (explicitly or experience) (i.e., they are obtained by nurture).

Reflection is not only possible after diversion but determines also simply part of a stream of thoughts. Therefore, a person does not per se have to wait for diversion. The other way round: reflection is not necessary for jumping out of the loop; one may just get distracted and start doing something else. The person may not even notice being stuck, and may also end up in the same stuck situation again at a later moment.

Conclusion about control

Concluding, the flow of a person's thoughts, as flow of activities, is (consciously) controllable by that person to some degree, but not fully (just like free will does not exist). In the end, the control activity (of thinking and other activities) itself is also one of the activities (i.e., physical processes) taking place. We mentioned already that a person does make choices and these choices – although the result of an autonomous physical process – may be regarded a kind of (limited) 'free will' or 'control'.

The lessons on control are incorporated in DMP.

3 Description of cases

In addition to related research we have studied decision-making cases as applied in practise. In analysing these cases, we observed a number of steps that we did not find in the related research, and that were useful in developing DMP.

The tasks in the cases were performed by three persons (one author, and two other persons), to the best of their knowledge. The tasks were done before the idea of a study arose; they were real tasks, done in the real world, not in a laboratory. The tasks were done effectively and not-too-inefficiently, to the judgement of the authors and the performers of the tasks.

In the case studies, we used observation and introspection. We recognise that introspection is not always reliable; it is nevertheless the best means we have for accessing the human brain on the level we need. The effectiveness of the procedures found in the cases was confirmed by the test persons, and some confidence exists that they are also efficient. The fact that there is no hard evidence for their optimality (in the sense of effectiveness and efficiency), is not considered a problem, since they served mainly as inspiration for finding procedures. In the next step, the analysis, the procedures found were scrutinised and mistakes were removed.

Each case[2] is formed by the execution of a specific task in a given subject area. The tasks are varied, including: (1) Going from one place to another, by foot, bicycle, car and aeroplane; (2) Finding a house to live; (3) Creating a definition; (4) Writing a piece of text; (5) Performing information retrieval, i.e., finding documents in a (digital) library by means of formulating queries; (6) Writing patent application texts, i.e., texts that describe state of the art, problems therewith, and an inventive solution to the problems; (7) Designing and planning in various projects; (8) Designing of an airconditioning control system for a passenger car; (9) Developing of an improved air conditioning for a passenger car, with non-uniform cooling from the ceiling, or cooling of the car 'envelope'; (10) Selecting and buying a car; (11) Investing money on the stock market; (12) Designing climate control systems for utility buildings, in a large engineering firm.

In the cases, we recognise many features of the related research as reported in Section 2. In addition, our analysis of the cases yields: (1) A similar pattern within the steps, (2) Control as a problem-solving process itself, and (3) Intuition. All the above elements are embedded in our procedure, to be introduced in the following section.

4 Top-level procedure for problem solving

This section specifies our procedure, DMP. First, an overview is provided in Section 4.1. Then, its elements are treated into more detail, in Sections 4.2 through 4.12.

4.1 Overall Procedure

Figure 3 shows a diagram of DMP, our procedure for decision-making: its steps and the information and knowledge used and produced in the steps. Although each step uses the results of steps shown above it, the order shown is not necessarily the order of execution of the steps. In fact, the order of execution will almost always differ from the order shown. Moreover, the steps are often intertwined. For instance, iteration will usually occur of the setting of goals and the conception of scenarios. As another example, one may partly execute a scenario, and only then complete the plan for the remaining, unexecuted part of that scenario. The shown order of the steps plays a role nevertheless. Before a step can be completed, the preceding steps have to be completed far enough to produce the required information, knowledge, or data. As such, we regard the steps in Figure 4 to have a 'logical order'. The basic steps of

[2] The cases are reported on www.thoughtwell.nl/cases_DMP.

the process are explained in Sections 4.2-4.9, the control of the process is introduced in Section 4.10, and recursion in Section 4.11. The balance in reason and intuition is treated in Section 4.12.

4.2 Collect relevant knowledge

The first step comprises collecting relevant knowledge. By 'knowledge', we refer to beliefs, opinions, established facts, theories, etc. We also refer to constituting 'parts' of knowledge, such as concepts and relations among concepts. By 'relevant knowledge', we mean both domain knowledge (i.e., knowledge on the subject area of the problem) and process knowledge (i.e., knowledge on how to solve a problem in the subject area). Sources of knowledge are: (1) memory (one's own, or that of others via communication), (2) reasoning, i.e. theory (including recursion to DMP and use of knowledge), (3) observation, and (4) definition (creating and naming concepts, etc.), or a combination of the above.

4.3 Conceive of goals and evaluation criteria

Goals

At least one goal is set. A goal, in its simplest shape, is defined as a state in the world (or a world) that the problem solver wants to realise or wants to have realised. A goal may also comprise a series of states to be realized in the world, possibly at different moments.

Each goal contains at least two components:

goal :: <degree of desirability, goal content>

By 'degree of desirability', we refer to some indication of how much the problem solver desires the goal to be realised, e.g., 'must' or 'may' or a relative weight as a number between 0 and 1. The goal content is what the goal is about. It usually includes *when* the state has to hold for the given subject, i.e., at least one point in time or a time interval. Thus, goal content is written, or can be re-written, as a triplet:

goal content:: <subject, state, time>

We distinguish two kinds of goal content (1) functional and (2) physical[3].. The term constraint is also often used, and indicates, in our opinion, a negative formulation of a goal. For example, a constraint is that the weight of an object is to be *no more than* than 1 kg. This corresponds with the goal: the system is to weigh *less than or equal to* 1 kg.

Usually, dependencies exist between the goals, in the sense that one goal may influence the chance of success of another goal. If possible, this dependency should be understood, in order to be able to set attainable goals.

At least one of the goals should always be related to the *process* of problem solving itself. This goal specifies the amount of resources available for the process, in terms of time and possibly also in

[3] Other kinds can be mapped onto these two.

terms of thinking-power, cost price, or other resources. If such a goal is missing, the problem solver can be quite sure that she made a mistake; in practical cases, there is such a limitation to the process. Another goal should deal with the risk of failure, of no longer meeting the other requirements.

Examples of goal content are:

- To be at a specific address at a specific moment, e.g., at the Eiffel Tower, in Paris, France, at Friday evening at 8 PM, in order to socialise and enjoy the view,
- To support your children to be healthy and happy, nearly all the time,
- To design an improved air conditioning for a car, within a year in at most 1000 manhours,
- To finish the decision-making process before 5 PM, with a risk of failure < 2%.

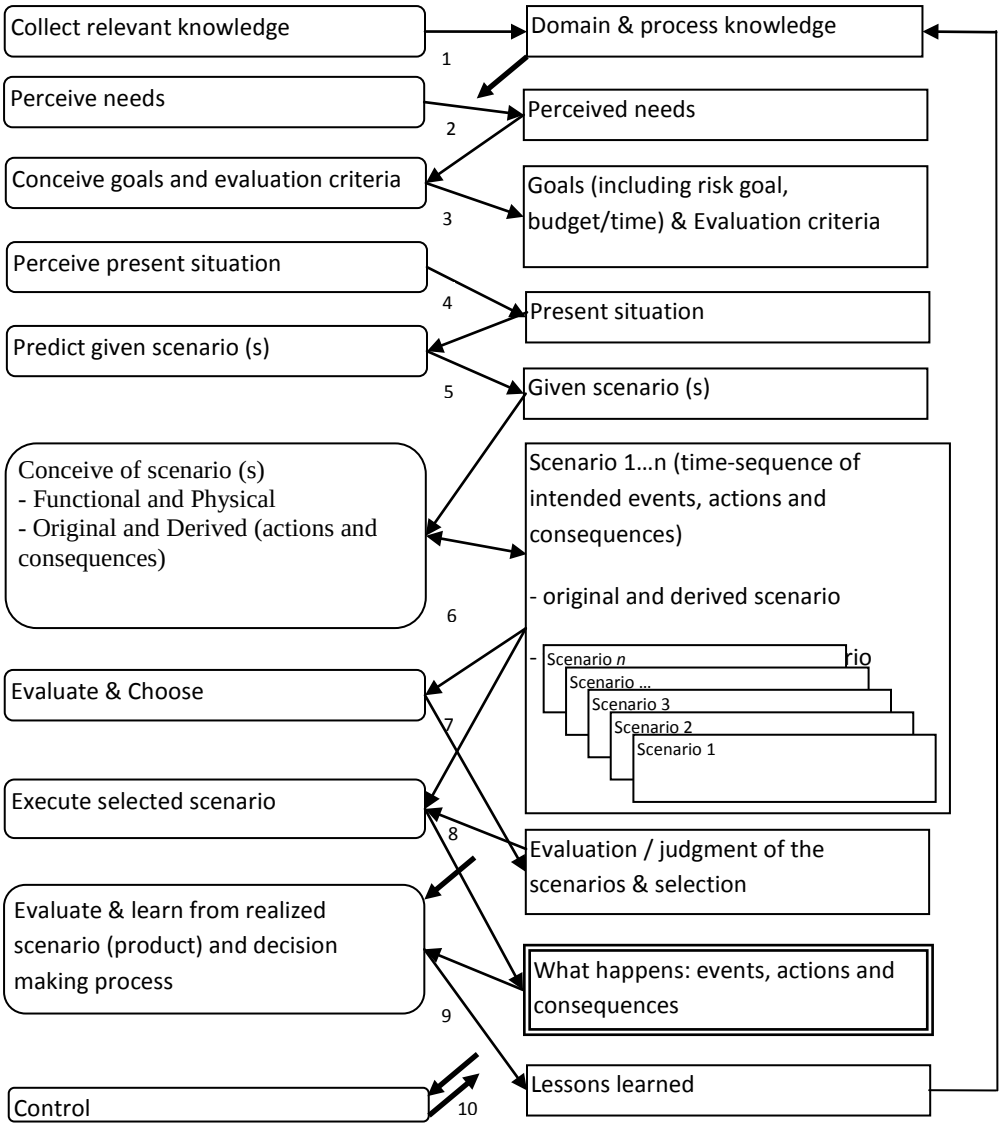
Evaluation criteria

The evaluation criteria are a means for measuring to what degree the goals have been realized. They may take several shapes. For instance, it is common practice to just formulate individual goals and criteria that tell when each goal is met, without calculating an overall result as a number. The overall result is only in terms of 'met' and 'not met', or, possibly, also in-between-answers, such as 'partially met'.

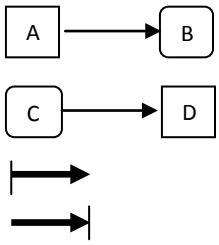
It is also quite common, though not necessary, to formulate a function in which all goals are taken into account (possibly weighted), and in which a single number is calculated, expressing the quality of each scenario, i.e., expressing to what degree the goals are met by each scenario or expressing a value of the scenario, as a measure of 'goodness'. For instance, this is done in computer chess programs. In economics, this function is usually called a 'utility function'. We will call it a scenario evaluation procedure (SEP), yielding a scenario value (SV).

4.4 Perceive present situation

Where the goal defines the end state (usually of a series of states to be realised) the starting point for reaching the end state will always be the current situation, or current state. This state influences which other states can be realised, and in what manner.



Legend:



Step B uses information, knowledge, or data mentioned in box A

Step C produces information, knowledge, or data mentioned in box D

Information, knowledge, or data in box used by several steps

Information, knowledge, or data in step is taken from several boxes

Within each step (not 'execute'):
Conscious:
- trial and error
- knowledge and reasoning, including
- knowledge from an old case (experience)
- recursion to a new instance of DMP
- a combination thereof
  (within these also decomposition/composition

Unconscious:
- intuition

Figure 3: Process outline of 'DMP' procedure

4.5 Predict given scenarios

By the step of predicting given scenarios, we mean determining the scenarios that we regard as given, i.e., that we do not actively want to change or establish[4]. Thus, it is a choice. Strictly, this would imply determining what will happen to the whole world. Of course, that will take too much effort. Therefore, only a part of what will happen to the world is to be determined, viz. that what seems relevant for the problem at hand. A given scenario (or a part of it) is relevant, if it influences the scenarios that we conceive of in order to realise the goals.

4.6 Conceive of scenarios for realising the goals

In order to realise a goal, at least one scenario must be conceived. Each conceived scenario is then evaluated, i.e., an estimation is made of to what degree the scenario will meet the goal. The evalution and choice of scenarios are discussed in Section 4.7, the conception is discussed in the present section. The order of execution of the three steps (conception, evaluation, and choice) may differ per task, and they may also be executed in turns, iteratively. For the sake of clarity, it is mentioned that the set of actions in a scenario does not have to be linearly ordered; it may also be branched, with conditional branches. In fact, it may be any sequence of actions.

Functional and physical scenarios

Each scenario should consist of two counterparts: (1) a functional part and (2) a physical part. The functional scenario defines *what* happens, i.e., which actions, and the physical scenario defines *how* the functional scenario is realised, i.e., by what means. For instance, a scenario may comprise the functional part 'move person from city A to city B' and the physical part may comprise 'a car' (and traffic infrastructure, etc.).

[4] In terms of SE this is similar, but not identical, to choosing the system boundaries.

Ideally, the functional scenario is a copy of the functional goals. In practise, it is somewhat modified, in particular extended. Usually, the design process, or conception of scenarios, is an iterative process, going back and forth between functional and physical design steps.

Original and derived scenarios

In addition to the distinction of functional and physical scenarios, each scenario (functional or physical) should be constituted of an *original* scenario and a *derived* scenario. The original scenario is that part of the scenario that requires deliberate action to be executed, whereas the derived scenario is that part of the scenario that follows from the deliberate actions (i.e., the original scenario), the current situation, and the given scenario. In other words, the original scenario consists of actions, the derived scenario consists of the consequences thereof.

Conceiving of the consequences of a scenario is also often a starting point for a re-evaluation and even reformulation of the goals. The reason for this is that only when a specific consequence becomes clear, it also becomes clear that this is an unwanted consequence. The next step then consists of revising the goals in such a manner that the unwanted consequence should not occur.

In the real world, the derived scenario can be much larger than one is able to oversee. Therefore, one should predict only the consequences that are relevant to realising the goals (i.e., contributing to or inhibiting). Moreover, often the consequences are only predicted and evaluated for the near future.

How scenarios are conceived of

The very first step of conceiving of new scenarios may be to decompose the goals (problem), to conceive scenarios-elements to the decomposed goals, and to compose the individual scenarios (solutions) into an integrated scenario.

The first scenario to be conceived is a scenario that may easily be overlooked: a no-change-scenario, i.e., leave things as they are evolving (at least for a while). The original scenario thus is empty, and it may well be the best solution to some problems.

In Section 2.2, we introduced several manners of conceiving scenarios. These manners have the following features: they rely on experience, on trial and error (e.g. by enumerating all possible solutions/scenarios, or by modifying a scenario ad random), or are created in a goal-directed manner, by invoking a new decision-making process. In other words, by recursion.

A mixture of copying old scenarios, trial-and-error, and recursion is in practise often applied. For example, if a problem is very similar to an old problem for which a scenario is known that has proven to be effective, the old scenario is copied, and modified by changing a few of its elements. The changes can be made either by just trying a new element (trial-and-error), or by deliberately conceiving of a suitable

new element (recursion). As such, a high efficiency (by using an old, often proven scenario) may be combined with a high effectiveness. An important example of mixing old scenario's with new steps is the use of the product life cycle. That cycle has proven to be useful and is often incorporated in the scenario's for new products. It is not useful in all scenario's however.

Thus, scenarios can be conceived of as shown in the lower box of Figure 1.

4.7 Evaluation and choice

The evaluation and choice of scenarios may occur in different manners. Usually, the idea is to evaluate all scenarios conceived of (and, if possible, assign an SEP), and to select the best scenario, i.e, the scenario that meets the evaluation criteria to the highest degree. That scenario is to be executed.

In problems with many goals, and thus many evaluation criteria, the evaluation may be too demanding, e.g., for the brain, and may be simplified by looking at the most important criterion first or only. The subset of scenarios that meet the main criterion is much smaller than the original set. Its members may be further evaluated and compared using one or more of the other criteria.

The step of creating a scenario and the step of evaluating it against other scenarios are often intertwined, i.e., done iteratively. In such cases, it is common to add one element to a scenario, viz. the most promising element, thus the element that yields the highest SEV. This approach thus is a Hill Climbing technique.

The step of evaluating takes time and effort.

The structure of the evaluation step, and the choice, can be obtained according to the lower box in Figure 1.

4.8 Execute scenario

The step of executing the preferred, selected scenario comprises, unsurprisingly, doing the steps of the scenario in the given order in that scenario.

4.9 Observe, evaluate and learn from product(s) and process

This step consists of observing (parts of) the process and product, whereby the concept of 'process' refers to all steps discussed above. It does not refer to the step of learning itself; that step should be evaluated in a separate decision-making process. The concept of 'product' refers to the *outcome* of the realised scenario.

Both the process and product may be evaluated against the needs and against the conceived goals: were they realised, and to what degree?

4.10 Control process

By 'control process', we refer to the process that controls the order of execution of the (other) steps in the procedure in order to obtain an effective and efficient process. In other words, the order of the

steps discussed above. Hereafter, such steps will be referred to as the steps of the basic level, as opposed to the control level.

Goals of control

The goals of the control process always include, just like any other decision making process, an allowed maximum period of time, and allowed other means. Moreover, there is usually a maximum allowed risk of failure. These goals should be derived from the original entire decision making process, since they serve the overall process.

For the control procedure, we search for an efficient procedure. To this end, we are looking for a procedure that matches that natural manner of control of people - those cost relatively little effort for persons. For very large tasks, at which several persons work, that reasoning holds possibly not for the complete task, but for the work items of it, when these are carried out by a single person.

As a natural manner of control for people, we take the descriptions of Section 2.3 as a starting point. We repeat that the activities may represent various types of mental activities, from small to large. Moreover, several such activities may be combined into activities of DMP, for example ‘gather domain knowledge’.

The decision maker should try to maintain a line of thought that does/performs one of the steps of Figure 2, as long as is useful. It should end when the person becomes tired, or has priorities beyond the decision process at hand, or when the process itself requires jumping to another step.

Order of execution

As already mentioned, the order of a decision making process is seldom according to the order presented in the procedure. Nevertheless, the idea is that, once the decision making process is finished, all steps have been completed. In practice, they are nearly always executed iteratively: a part of one step is done, then a part of another step, then still another step, then e.g. again a part of the first step, etc.

For instance, a problem solver may get an idea (i.e., conceives of a scenario, or a part thereof). Unconsciously, she will also have a goal in mind that is served by the idea and have used some domain knowledge in getting the idea. Next, she may study the idea, and define the goals more precisely. Then, she extends her domain knowledge and refines and extends the scenario conceived, and creates alternative scenarios. This was observed in several of the cases studied.

Sometimes not all the steps are taken, or a step is taken much later. For instance, the evaluation and learning step may be done years later. By not taking a step, the risk of a bad, i.e., ineffective scenario increases - although the efficiency may increase.

The (scenario of the) control process sometimes contains a monitoring loop that monitors the progress in the basic level and takes action when the progress is insufficient. This loop may be apart from another part of the control process, which determines merely which step of the basic level is next. Then, only when the loop detects abnormalities, the other part of the control process is interrupted, overruled. Such a loop is ideal for a parallel implementation that continuously monitors the rest of the activities.

Although hopping from one step to another step is often useful, De Bono (1985) argues that it is best to do each step long enough to obtain the right mindset (attitude and concentration) for doing the step and to make serious progress. One should end only when little or no progress is being made for a longer period of time. Switching too often may prevent one from obtaining good results, since one may not get the right mindset[5].

The best scenario of the control process depends on the issue at hand, i.e., it is somewhat task-dependant. Thus, DMP is not entirely task-independent, but the dependency *is* maintained within a single step of the procedure.

Description of control process

The control process is in itself a decision-making process and has the basic level steps as subject. We have found that the control process has, in human beings, essentially the same structure as the basic level of DMP. This implies that it has the same steps, and is also controlled. This implies that there exists 'control of control'. This structure can be implemented by, human and computer, by alternating between steps of the basic level and steps of the control level. These steps/activities may be done serially or parallel.

As a consequence of control being a decision-making process itself, and thus also having a control step, this form of recursion may lead to a long sequence control of control of control of control etc. This sequence should end somewhere, in order to keep the control process finite. This ending is obtained in a natural way. Control of control and control of control of control etc may become difficult. Since the control process has the same structure as the basic level, its steps may also be done according to the lower box of Figure 3.

When control becomes more difficult, such as in control of control, it becomes more likely that a person cannot deal with it any more being (exposed to mental overload) and then it is likely that (s)he falls back on either a template-solution or trial and error. This ends the recursion and thus keeps the sequence of control of control of control etc finite. In fact, the author believes that in many cases the first level of control is already a template. This is in agreement with the finding that persons usually

[5] It takes some 20 minutes, before a person is able to work on a task with full focus (ref).

become expert in a single subject area/task area or just a few of these. Each subject/task area (or problem type, or decision type) has a specific kind of control.

4.11 Action within steps

We discuss what happens within the steps, as far as did not already do this.

Distinction by kinds of knowledge

As seen above, the actions within steps (except the execution step) can be distinguished by the knowledge involved: see the lower box of Figure 1. The control process chooses among these, and determines the order of execution. Recursion differs from iteration among steps. In iteration, each part of the same process is given attention for a number of times. In recursion, a new process is invoked. For example: when executing the conceived task 'drilling a hole', and need to determine what to do precisely, then you have to do recursion (since it is a new task). When re-planning whether to drill or do something else, then it is called iteration (since it is concerned with the original task). In conceiving tasks: conceiving a small part is done by recursion after decomposition.

The combination of knowledge and reasoning on the one hand and trial and error on the other hand requires decomposition and composition – subtasks for which the same sources of knowledge hold. For instance when you try to develop an improved knife, you may use an old grip design and an existing type of connection between the grip and blade, and design a new blade. The latter is done in a new process, i.e. by means of recursion.

Decomposition/composition

Decomposition is very common in daily life, including problem solving, decision making and scenarios engineering. The reason for decomposition is that many tasks are easier to do in parts, i.e., decomposed. After each part is finished, they are combined into a whole. In other words: composition. The separate parts usually have interrelations that should be taken in consideration. It is beneficial to choose the decomposition in such a manner that no or only few such interrelations exist, since then the individual parts can be dealt with without much overhead for integrating them.

4.12 Conscious and unconscious thinking in DMP

In Section 2.1, we already introduced the distinction between conscious and unconscious thinking, and the combination thereof proposed by Dijksterhuis. We adopt his approach, by including the mixture of conscious and unconscious in each step. We recap the approach: (1) Collect relevant information; (2) Do something else (but, aware of the later steps, so that the unconscious brains can start working); (3) Do that task unconsciously; (4) Check the result consciously for obvious errors and mistakes. We add a step: (5) If the conscious approach and the unconscious approach are in conflict, then try to find the cause. If this fails, wait. If there's no time to wait: follow what you like most – unconscious or conscious result. If you have to account for your decisions, such as in a legal judgment, you should follow the conscious approach; however, you may still find guidance by means of intuition.

There is a risk in consciously checking an unconsciously obtained result: the persons who checks may be biased towards the outcome, or opposed to the outcome. This may then lead to the inclusion of less relevant reasons, and the omission or relevant reasons. Of course, care should be taken to avoid this, by carefully checking all the reasoning steps.

Since it is difficult to estimate the importance (weight) of each goal, the result of the unconscious approach may be used to reproduce weights for the conscious approach – by reasoning backwards from the result to the goals and weights. For the goals, a rough scale suffices, since persons are unable to make precise estimations. For instance, use the groups important-average-not important, or: must-may.

5 Discussion, conclusions, and future research

The results are discussed briefly, in Section 5.1, then we draw conclusions, in Section 5.2, and finally we make suggestions for future research, in Section 5.3.

5.1 Discussion

On the basis of the preceding, it seems that our expectation of Section 1, that it should be possible to formulate a domain-independent, top-level procedure, has turned out right. The procedure is non-trivial, in the sense that it does indeed show the way.

Differences to other methods

All known approaches fit in our procedure, in the sense that their steps can be mapped onto DMP steps. However, DMP is more comprehensive regarding the steps (i.e., the domain knowledge excluded), in the sense of providing wider and deeper process steps. In width, all decision activities are included, including control of these activities (as an activity itself). In depth, the content of all activities is detailed down to either the level of basic mental tasks, such as copying an old solution, observing, and guessing (trial and error), or to the level of recursion to a new decision-making task.

We also found that the case studies fit, in hindsight. The cases provided details not found in the known approaches, and they showed that approaches found in literature were realistic though sometimes very coarse. Waterfall models showed unrealistic; however, this was no news. We also found that iteration plays a role between all steps.

As usual in research, much is already known, and only a small part is new. DMP also has many elements from existing approaches. Domain independent procedures exist already. (e.g., in AI and in SE.). DMP differs from the existing procedures in that its steps are different and provide a better description of the actions observed in the cases. In particular:

- The steps differ from those in the existing procedures, e.g. by distinction of functional-physical scenarios and original-derived scenarios, and *perceived* present situation, *predicted* given scenario, and a separate control step.

- The content of some of the individual steps includes recursion to DMP.
- The control process is of the same type as the basic process steps. The control process makes iteration possible over all steps.
- Intuition is included, next to explicit reasoning.
- The switching of attention among different steps, as described in Chapter 2 as typical behaviour of humans having various causes, is reflected by the control step that performs iteration over the basic steps, and the fact that 'control' includes explicit reasoning with a goal as well as trial and error (possibly due to being tired).

These differences are partly the result of using realistic insights in thinking. These differences also will make DMP more effective than existing methods. The procedure was created as a *description*. Now that it has been formulated explicitly, it may be used as a *prescription.* In other words, it is a guideline that may be used for improved decision making, both by human beings and by computers.

In addition to being more effective in general, we mention that DMP seems to solve the problem posed by Hofstadter, of getting stuck, as mentioned in the introduction. DMP may jump out of such a situation due to its using not only knowledge and reasoning, including experience, but also trial and error within the steps, in combination with a control step that checks progress.

Differences to SE

In addition to the differences mentioned above, in SE, a risk analysis is included as a separate task in the control step. In DMP the risk assessment is part of the conception of scenarios, i.e. integrated the design task, considering allowed risk as a requirement. This allows for a description of risk analysis as a subtask, thus recursively as a decision-making task itself – with the same structure as a decision-making process, thus the same schema may be reused.

Also, in DMP the separate control step is further defined: as having the same structure as a decision-making process, including recursion. We understand that this difference in approach of dealing with risk is somewhat academic; in practice, systems engineers often do treat risk along with the requirements.

Both the iterative loops in SE re-occur in DMP as 'conceiving of scenarios'. DMP supports the first loop of SE by the two steps of setting requirements and conceiving of scenarios, under the control of an explicit control process that allows for iteration. DMP supports the second loop by applying recursion, i.e., invoking a new problem-solving procedure for finding a physical scenario (or design) to a given functional scenario. In this manner, DMP provides an answer to *how* a physical design may be created effectively. Another advantage of DMP over the process of SE is that DMP also details the control process and provides the same extensive set of techniques for both iterative loops. Apart from these differences, DMP has many elements of SE.

Finally, SE provides other useful details for the steps, such as how to formulate requirements properly. Therefore, it is suggested to consult SE in addition to DMP.

5.2 Conclusion

On the basis of the above, we draw the following tentative conclusion: *DMP, as defined above, is an overall, subject-independent and task-independent procedure that is more effective for making decisions by a single person than other known procedures.*

We expect that the *efficiency* of DMP will depend on details that still need to be chosen and are domain or task-dependent; in particular, the control step needs such further details. The choice and content of the steps in DMP cause the increased effectiveness of DMP, as indicated above, in the discussion.

5.3 Future research

First of all, (more) experimental evidence is needed. Now DMP is a theory, although based on extensive practical experience. In addition, we mention two other interesting directions for future work: (1) a formalisation of the procedure. A formalisation, in terms of computer-implementable terms, will demand a more *precise* and thus better formulation of the procedure. (2) A further refinement of the procedure. In particular, details pertaining to *how* each step is performed are needed since we gave an elementary description only.

References

A. Aamodt and E. Plaza, Case-based Reasoning: Foundational Issues, Methodological Variations and System Approaches, *AI Communications* 7 (1) (1994) 33-59

K. Beck, *Extreme Programming explained; embrace change*. (2nd edition) (Addison-Wesley Professional, Reading, MA 2006)

P. Benner and C. Tanner, Clinical Judgment: How Expert Nurses Use Intuition, *The American Journal of Nursing*, (1987) 87:1 (23-31)

I. Bratko, *Prolog programming for Artificial Intelligence* (2nd edition) (Addison-Wesley, Reading, MA, 1990)

D.M. Breuker, Personal communication (2006)

B.G. Buchanan and E.H. Shortliffe, *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project* (Addison-Wesley, Reading, MA, 1984)

W.J. Clancey, Heuristic Classification. *Artificial Intelligence* 25 (3) (1990) pp. 289-350

J.C. Cool, F.J. Schijff and T.J. Viersma, *Regeltechniek* [*Control Engineering*] (Elsevier, Amsterdam/Brussel, 1979).

Dept. of Defense, Systems Management College, *Systems Engineering Fundamentals* (Prepared by the Defense Acquisition University Press, Fort Belvoir, Virginia, 2001)

A.D. de Groot, *Thought and Choice in Chess* (Mouton, The Hague, 1965)

A. Dijksterhuis, M.W. Bos, L.F. Nordgren and R.B. Van Baaren, On making the right choice: the deliberation-without-attention effect, *Science* (February 17th 2006) 311, no. 5763 (1005-1007)

G.A. Dumont and M. Huzmezan, Concepts, methods and techniques in adaptive control.. In: *Proceedings of the American Control Conference* (2002) 2 (1137- 1150) Dept. of Electr. & Comput. Eng., British Columbia Univ., Vancouver, BC

C.E. Garcia D.M. Prett and M. Morari, Model predictive control: theory and practice, *Automatica* (1989) 25 (335-348)

M.S. Gazzaniga, R.B. Ivry and G.R. Mangun, *Cognitive Neuroscience: the biology of the mind* (W.W. Norton & Company. New York/London, 1998)

D.R. Hofstadter, *Gödel, Escher, Bach: an eternal golden braid* (Vintage Books Edition, New York, 1980)

G.A. Miller, The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. (*The Psychological Review*, 1956) 63 (81-97)

A. Newell, *Unified Theories of Cognition* (First Harvard University Press paperback edition, third printing, 1994)

A. Newell and H.A. Simon, *Human Problem Solving* (Prentice-Hall, Inc., Englewood Cliffs, N.J., 1972)

D.W. Patterson, *Introduction to Artificial Intelligence and Expert Systems* (1990)

S.L. Pfleeger, *Software Engineering; theory and practice* (Prentice-Hall, Upper Saddle River, NJ, Third edition, 2006)

F.K. Reilly, *Investment Analysis and Portfolio Management* (The Dryden Press, 3rd International Edition, 1989)

N.C. Rowe, *Artificial Intelligence through Prolog* (Prentice-Hall, London, UK, 1988)

N.F.M. Roozenburg and J. Eekels, *Product Design, Fundamentals and Methods* (1995) Wiley, Chichester, UK.

D.E. Rumelhart, J.L. McClelland, and the PDP Research Group (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations* (MIT Press, Cambridge, MA, 1986)

H.G. Schmidt, P.P. Hobus, V.L. Patel and H.P.A. Boshuizen, Contextual factors in the activation of first hypotheses: Expert-Novices differences (*Annual Meeting of the American Educational Research Association*, Washington, 1987)

G. Schreiber, B. Wielinga and J. Breuker, *KADS: A principled approach to Knowledge-based system development* (Academic Press Inc, San Diego, CA 92101, 1993)

C.E. Shannon. Programming a computer or playing chess. *Philosophical Magazine (41) (1950)* (256-275)

H.J. van den Herik, Computerschaak, *Schaakwereld en Kunstmatige Intelligentie*, PhD Thesis (Delft University of technology, Delft, 1983)

R.W. van der Pol, *Knowledge-based Query Formulation in Information retrieval*, PhD Thesis (Phidippides, Cadier en Keer, The Netherlands, 2000) SIKS Dissertation Series no. 2000-5.

H. Visser, Principles and procedures of productive problem solving. In: *Il Laboratorio di Scienze Cognitive – Attività ottobre 1989 – novembre 1999. V. Braitenberg and Ch. G. Orsini (eds.). Vol. 1. Saggi e lavori*. (Trento: Università degli Studi di Trento, 2000) (161-193)

J. Zabczyk. *Mathematical Control Theory: An Introduction*. Boston, MA: Birkhäuser, 1993.

BIOGRAPHY

Ruud W. van der Pol is a mechanical engineer and has a Phd from Maastricht University, in Information Retrieval. He is self-employed at Thoughtwell, as a consultant in systems engineering and patents.

Floris J. Wiesman is a computer scientist and has a PhD from Maastricht University, in Information Retrieval. He works as an assistent professor in Amsterdam Medical Center, Dept. Medical Informatics. His interests include information retrieval, electronic patient files and the semantic web.