In: Proceedings of Qualitative Reasoning (QR2019)

Model-based Decision Support Systems - Conceptualization and General Architecture

Peter Struss
Department of Computer Science, Technical University of Munich, Germany
struss@in.tum.de

Abstract

The paper presents an attempt to conceptualize decision support and various generic subtasks and to develop a general architecture of intelligent decision support systems. We decompose the task of decision support into subtasks whose input, output, and function are characterized. This is based on a small number of elementary concepts: such as "situation", "goal" and "action", and "process". Based on the resulting generic architecture for intelligent decision support systems, we show how it can be realized using a model-based approach, more specifically, process oriented modeling and model composition.

1 Introduction

A survey of definitions or characterizations of decision support systems (DSS) and of proposals for general architectures or (sub-) functions of DSS delivers rather disappointing results. Many of the offered definitions boil down to "A DSS is a computer system (or set of tools) that supports making decisions", which turns any data base, MATLAB, excel, even Google into a DSS. When architectures are proposed, they are often presented as a huge set of tools and computational steps embedded in a confusing web of interconnections, often heavily emphasizing data-driven techniques. The components are mainly characterized as various alternative or complementary techniques, rather than by the function they implement.

A systematic analysis and a conceptualization of DSS seem to be missing. This is not an academic question aiming at delivering a set of definitions. It is a practical necessity, if we are interested in the systematic re-use and integration of different tools and methods. And it is a prerequisite for establishing requirements on DSS and its components, especially when we want to build "intelligent DSS".

The conceptualization and architecture proposed in the following does not claim to offer a general account for all existing DSS. Its objective is to provide the foundation for the design and comparison of knowledge-based DSS and the identification of subtasks and modules and their principled interfaces. This leads to a second disclaimer: although we are fully aware that observations, knowledge, and inferences can be subject to a significant degree of uncertainty, we do not explicitly represent this at this stage of the formalization. The consideration behind this is that uncertainty (e.g. in terms of probabilities) can be **added** to the presented concepts, potentially "softening" results and introducing ambiguity and alternatives.

We start with the general concepts and tasks of decision making. Then we introduce the concepts and the algorithms underlying a model-based realization of a DSS und finally a number of challenges and open questions.

2 Conceptualization of a DSS

2.1 Decision Making

A decision has to be taken only if there is a **choice**. It means picking one from a set of alternative options. These options are different ways to **act**. If the possible ways to act are complex (rather than only a single action), we call them a plan, which is a set of actions, possibly with a particular order or temporal extent. Hence, a decision has a set of plan options as an input and results in choosing one of them. The necessity to act and the criteria for the selection depend on two crucial concepts: the **goals** that should be achieved (or maintained) and the current (or an assumed) **situation**. Hence, the basic concepts are related as depicted in a class diagram of the Unified Modeling Language (UML) of Figure 1.

Hence, the activity of "selecting the best plan for achieving or pursuing goals in a given situation" characterizes decision making in the narrow sense only. However, a prerequisite for this step is that

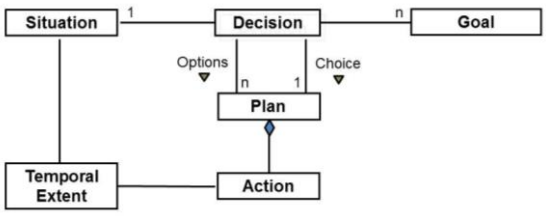

Figure 1 Decision and related concepts

- an understanding of the situation and
- a set of possibly effective plans

have been generated beforehand. These are the main subtasks that require knowledge, domain expertise, and reasoning.

2.2 Example from Water Treatment

In the following we will use a simplification of a real problem encountered in a previous project on drinking water treatment in the city of Porto Alegre, Brazil ([Roque *et al*., 2003]) as an illustrative example.
In this scenario, customer complaints about an unpleasant **metallic taste of the drinking water** were the starting point. In order to determine appropriate countermeasures, experts from the municipal department of water and sewage in Porto Alegre, DMAE, analyzed the situation and finally identified the reasons behind the observation:

- The cause for the taste was an unusually **high concentration of iron** in the drinking water, which was already present in the treatment plant.
- This had been **pumped in** from a particular reservoir, Lomba do Sabão, more specifically from its upper water layer, the epilimnion.
- The dissolved iron in this layer did **ascend** from the lower water layer, the hypolimnion.
- The increased iron concentration, in the hypolimnion was caused by a higher rate of **re-dissolving of solid iron**, which was contained in the sediment.
- This increased re-dissolving was due to an unexpected **change in the pH** towards acidic conditions of the hypolimnion as a result of a phase of algal bloom that had occurred recently.

Based on this analysis, reducing the iron concentration in the plant by creating an oxidation process through the introduction of oxidation agents such as chlorine or ozone was the appropriate intervention, since neither removal of the solid iron from the sediment nor the modification of the pH in the reservoir was a feasible solution.

2.4 High-level Decomposition of a DSS

A DSS should support or automate the selection of a plan for achieving certain goals based on a set of observations. Doing this in a way that deserves the attribute "intelligent" certainly implies determining whether or not and to what extent carrying out a plan will effectively transform the given situation to one that satisfies the goals. This, in turn, requires the step to interpret the observations in order to derive a more complete picture of the current situation, which includes, in particular, a representation of the present causal interactions in the system, such as the causal story behind the metallic taste of the drinking water in the above example. We call this subtask, which certainly requires deep domain knowledge, **situation assessment**. Another subtask, which is necessary, unless the different options to choose from are given *a priori*, is the generation of plans that promise to achieve the goals. The inputs to this **therapy proposal** step are the situation, the goals, and the potential actions that can be taken. Together with the **decision making** step, i.e. the choice of one plan from the options, we obtain the high-level decomposition of the DSS shown in Figure 2.

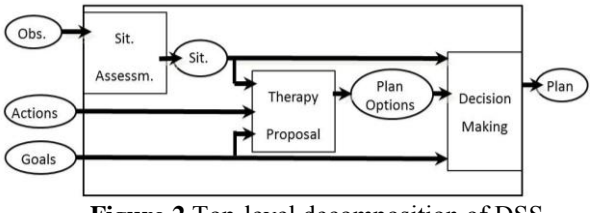

Figure 2 Top-level decomposition of DSS

2.3 Therapy Proposal

The initial step in therapy proposal (see Figure 3) is to evaluate the situation with respect to the given goals, i.e. determining which goals are violated and, perhaps, in which way (e.g. a particular threshold exceeded or undershot, relevant objects missing, or unwanted ones being present). This **situation evaluation** may not only be necessary for the current situation, but also for future situations with or without applying actions. This requires a **prediction** module, which produces the description of such forecast situations, e.g. through simulation. The discrepancies between situations and goals are an input to **goal generation**. This is needed, although we already have a set of explicit goals, because, usually, we need to determine some intermediate goals that guide the search for a plan. If the iron concentration in drinking water is above the threshold, even after a remedial action, the respective goal will be violated for some time, while a feasible intermediate goal will be reducing the iron concentration. Together with the situation description and the available actions, they are an input to the **plan generation** module.

2.5 Decision Making

The last step leading to a decision is **plan evaluation**, by accumulating **situation evaluation**, i.e. checking whether and to what extent they establish situations that fulfill the (intermediate or general) goals. Again, this analysis may not only be carried out for a single situation, but for a sequence of situations created by executing a plan with several steps or situations in the evolution of the system after the intervention has been finished.

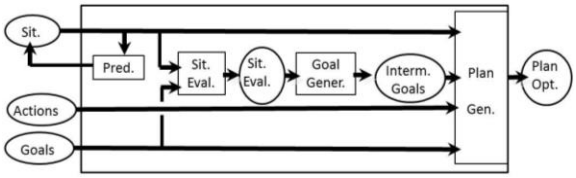

Figure 3 Decomposition of therapy proposal

Therefore, prediction may be involved in this subtask, too. This is, usually, a knowledge-intensive task, since determining the impact of actions on the system and its evolution requires an understanding of the driving forces and interactions in the system. In contrast, the step of picking one plan from a set of suitable ones based on the plan evaluation is a more or less formal task based on the plan evaluation, which is guided by weights and priorities of goals and cost of plan execution, factors that are external to the domain model.

So far, we characterized the inputs and outputs of the DSS and its various modules by informal concepts. To lay the foundations for a model-based solution to DSS, we continue by providing more precise, but also specialized definitions of these concepts more precisely.

3 Process-model-based DSS - Concepts

The main concepts we used in the description of the interfaces and their intuitive meanings were:

- Observation: "What is the case" (information)
- Situation: "What is going on" (applied knowledge)
- Goal: "What should be the case" (objectives)
- Action: "What can be done" (potential)
- Plan: "What should be done" (intention)

The above concepts are all assertions about the world under different modalities, which means we need to fix a basic ontology for statements about the considered domain, a class of real or potential systems. We decide to categorize such statements as propositions about

- **Structure**: in terms of a set of existing objects and object relations between them
- **Behavior**: by constraining quantities or properties of the objects to a particular range.

Figure 4 introduces the unifying concept of an **assertion** in a UML static structure diagram, depicting that (non-)existence is assigned to structural elements, which are objects and object relations. Objects have quantities, and assertions can associate a certain range to them.

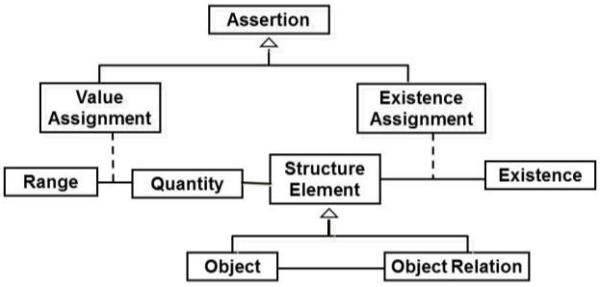

Figure 4 Assertions w.r.t. quantity and object (relation)

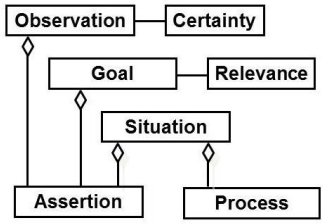

Figure 5 Observation, Goal, and Situation

While observations and goals are just collections of assertions, a situation or, more precisely, a representation of a situation is meant to capture not only a completed description of the observed system, but also reflect our understanding of the underlying causal interdependencies. Hence, we represent it as a set of assertions and a collection of processes (Figure 5).

Processes (Figure 6) represent the mechanisms that are the driving forces of the system's dynamics (or, rather, our model thereof). Processes are characterized by a collection of **preconditions**, which are assertions, and **effects**, which are assertions or influences, i.e. their impact on the system dynamics, i.e. their contribution to changes of variables.

4 Process-model-based DSS - Algorithm

The architecture and the interface and functional definitions of its modules are still very general. This is deliberate, because our goal is providing a generic way of structuring DSS that allows us to combine different techniques and models that implement different modules or sub-modules.

In this section, we describe our model-based solution to implementing an intelligent DSS along the lines presented above. It will turn out to provide a very coherent way of implementing different modules of the architecture. Its core is the concept of a process, which was introduced in section 3 and which is embedded in an inference scheme that opens ways, in particular, to implement (semi-)automatic situation assessment and therapy proposal. This implementation is based on a rigorous specification of the involved subtasks (see [Heller and Struss, 2002]), which is grounded in a logical reconstruction of Qualitative Process theory [Forbus, 1984] and exploits theories and techniques from model-based problem solving ([Struss, 2008]).

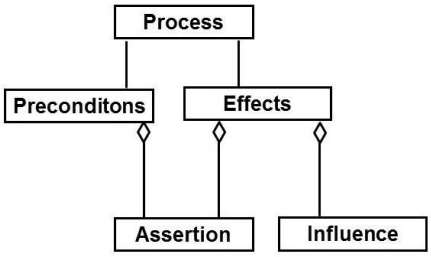

Figure 6 Process

In the following subsections, we summarize the representation of elementary processes, which are collected in a domain library, and then discuss the task of composing them to form a model of a complex system. Then we outline a solution to situation assessment, which is seen as the task of model composition. On this basis, situation evaluation and goal generation can be implemented. Therapy proposal turns out to be an instance of model composition, too.

4.1 Process-oriented Modeling

In section 3, a process was represented as a pair of conditions and effects, which both contained assertions, while the effects captured impacts of the process informally stated as **influences**. Turning the informal semantics of a process, namely that the effects will be established whenever the preconditions are satisfied, into logic, a process becomes an **implication**, i.e if the conditions are satisfied, the effects are also true:

$$\text{StructuralConditions} \wedge \text{QuantityConditions} \Rightarrow \text{StructuralEffects} \wedge \text{QuantityEffects},$$

where StructuralConditions and StructuralEffects are ExistenceAssignments, and QuantityConditions and QuantityEffects contain ValueAssignments as described in section 3. In addition, QuantityEffects contain influences. Influences capture the impact of a process on the dynamics of the systems, i.e. how quantities change, but, nevertheless, are beyond the expressiveness of differential equations. For instance, the result of the process of iron re-dissolving cannot simply be stated as a positive derivative of the iron concentration in the hypolimnion. This is because, in the composed system model, there may be other, counteracting processes, which may compensate or over override the impact of re-dissolving and lead to a negative derivative of the iron concentration. All the local model of re-dissolving may state is a positive **contribution**, a positive influence. In an approximate way, it specifies a partial derivative of the iron concentration: the iron concentration in the water can be a function of several variables, and the partial derivative w.r.t. the iron concentration in the sediment is positive. Whether it really increases or not, depends on other influencing variables, as well.

In the step of model composition, the various influences on each variable will be combined, e.g. in a linear combination, like the different inflows and outflows to a container are added. An essential aspect of this step is that **all** relevant influences and, hence, processes are assumed to be known, which may have to be retracted in subsequent reasoning steps in order to find a plausible explanation of the observations. This step of influence resolution implies that, **if the model does not contain any influence on a variable, the variable does not change**. Similarly, objects and relations added to the model need to be introduced as part of the StructuralEffects of some process. As a consequence, a model that contains a change in a variable or an object or object relation that are not caused by (i.e. occurring in the effects of) at least one process is inconsistent.

4.2 Situation Assessment

Situation assessment was introduced as the task of construction a causal account for what has been observed. It is performed by constructing situations, i.e. sets of assertions and processes that are consistent with the observations based on a domain library of processes.

In a process-oriented framework, situation assessment **is** model composition. The resulting model needs to be complete w.r.t. the effects of the included processes (and the processes triggered by these effects etc.). But the more important aspect in situation assessment is its completeness regarding causation – everything that has been observed or hypothesized requires a process in the situation causing it (with an important exception, as discussed below).

Candidates for such explanatory situations are constructed in a cycle of completing them in a (causally) forward, deductive direction (i.e. adding effects and their consequences) and in a backward, abductive direction (by hypothesizing preconditions and their potential causation). In contrast to the former step, the latter is usually not unambiguous (there may be different causal accounts for the observations). Hence, situation assessment will deliver alternative results, and the user may have to pick the most plausible one.

How can the extension of model in this causally upstream direction be enforced? The key to this lies in the fact stated above that a situation that is lacking a causal account for a change or a StructureElement is not simply incomplete, but inconsistent. For determining possible "repairs" of inconstant intermediate situation hypotheses, consistency-based diagnosis ([de Kleer-Williams, 1984], ([Struss, 2008]) is applied.

The algorithm (depicted in Fig. 7) starts with the set of initial observations, i.e. Assertions, which may be treated as facts or assumptions, and enters the cycle

- **Forward completion**: applying processes as implications
- **Consistency check**: if the situation is consistent, it is a candidate for a proper situation assessment
- **Diagnosis**: consistency-based diagnosis identifies assumptions that contribute to the inconsistency and, hence, candidates for revisions of the model that may (ultimately) lead to a consistent situation. Such assumptions can be initial observations or the closed-world assumption involved in influence resolution.
- **Candidate selection**: usually, there will several possible revisions. Either they are pursued in parallel, or a heuristic or a user selects one option. If a closed-world assumption is involved, it is related to a variable or StructureElement.

Figure 7 The cycle of forward and backward completion

- **Backward completion**: searches the library for processes that affect the variable or StructureElement associated with the retracted closed-world assumption. Again, there is usually a choice to be made. The result is a modified situation that is entered to forward completion.

Note that backward completion is inherently infinite; te demanding for causal explanations would not terminate (unless there exists a process without preconditions or a cyclic causation, which would indicate a modeling error), and no consistent situation could be generated.

Asking for causes of the preconditions has to be limited: while existence of iron at high concentrations in the tank and the water layers demands for an explanation, the presence of iron in the sediment does not have to be justified by another process, because this would be beyond the scope of the model (and of the interest). We call these preconditions that cut off the search for explanations **introducibles**. They characterize the boundaries of the model and the desired explanations.

What is treated as introducible may heavily depend on the specific task. If the unexpected pH is taken for granted, the search may stop here, because it is possible to determine an appropriate intervention. In a different setting, one may be interested in tracing it back to algae bloom and its creation by an abundance of nutrients caused by agricultural runoff, in order to take preventive actions regarding this root cause. The search is confined by the scope of the domain library that is assumed to contain all the processes known to be relevant to the class of problems to be solved. Each structural element and influence on a variable that does not appear as an effect of any process in the library has to be treated as an introducible. Do we have to consider this as a weakness of process-model-based situation assessment? No, because this is not different from the performance of a human expert who cannot and will not generate explanations beyond the causal accounts (s)he is a aware of.

The result of this solution to situation assessment is a set of candidate situations, each of the being

- A **minimal** situation
- containing **all facts**
- otherwise only **introducibles** and
- **effects** of occurring processes
- **closed** w.r.t. **effects**,
- with a **maximal** set of **assumptions** holding.

This means, while the factual observations are preserved, some assumptions may be dropped (e.g. the assumption that the pH in the reservoir was neutral). In addition, there are introducibles and their consequences.

4.3 Situation Evaluation and Goal Generation

If situation assessment, perhaps after filtering by the user, has committed to a particular situation (or a disjunction of situations), situation assessment has to check for violation of goals. This is a straightforward consistency check of the assertions established by the situation and the goals. An actual value (range) of a variable may be disjoint w.r.t. the goal range, more specifically, it may be above or below this range ("iron concentration above the threshold"). Or the (non-)existence of a structural object required by a goal may contradict the situation. If the existence of an object is required by a goal, but not result from a process contained in the situation, this creates an inconsistency, as in situation assessment.

From the detected goal violations, intermediate goals can be generated in a mechanical way: the creation of a non-existent object remains a goal. If a quantity is higher (lower) than the range specified by the goal, then a negative (positive) derivative becomes an intermediate goal.

4.4 Therapy Proposal

Therapy proposal operates similar to situation assessment, but aiming at consistency with the (intermediate) goals (rather than with observations). The causally upstream completion of the model is bounded by **action triggers** as the introducibles (i.e. every change applied to the situation is ultimately caused actions), and the downstream completion helps to check whether the actions may violate some of the (intermediate) goals. Therapy proposal creates

- A **minimal** structure
- containing all **facts**
- otherwise only **action triggers** and
- **effects** of occurring processes
- **closed** w.r.t. **effects**
- with all **fixed goals** and a **maximal** set of **relaxable goals** holds.

As suggested by this characterization, situation assessment and therapy proposal can be performed by the same algo-

rithm, by interchanging observations and goals, and natural introducibles and action triggers (see [Heller, 2001]).

While the basic outline of these inferences suggests a straightforward solution, we are indeed facing a fundamental problem: which elements of the situation actually persist while the actions considered by therapy proposal are applied? After all, the actions aim at modifying the situation. The answer to this question is based on the consideration that absolute values change continuously (e.g. the iron concentration remains above the threshold), while the derivatives may be discontinuously influenced by the action (i.e. the derivative of iron concentration flipping its sign immediately). As a consequence, we cannot simply preserve derivatives in the situation in which we apply the actions. However, we cannot drop them altogether, because only those affected by the actions and the processes triggered by them may change, while the others will persist. In consequence, we turn all derivatives into assumptions with the effect that they will be dropped if enforced by the therapy proposal and otherwise persist.

5 Discussion

Although we had a theoretical basis and an implementation of a similar process-oriented diagnosis engine, G+DE, some 20 years ago [Heller and Struss, 2002], the re-specification for DSS and the analysis of envisioned applications reveal the impact of both already known inherent restrictions and practical problems to be solved. We briefly discuss some of them, which will determine future work.

Snapshot-based analysis: right now, the analysis assumes that a situation, i.e. a causal explanation can be constructed within a (qualitative) temporal snapshot. More technically, the analysis does not go backward beyond integration steps. If they are included, there could be concurrent changes in the system, and different orders of their temporal occurrences would have to be considered, which complicates the analysis.

Intervention plans: Similarly, the generated interventions are simply collections of actions, considered to be executed in parallel, rather than in a particular order or a certain point in time.

Observations: Uncertainty of observations is an issue in practical settings. The current solution treats it only in a black-grey-white fashion by allowing observations to be assumptions or defaults, that may be retracted. What we deliberately ignore in the general concepts, is the step of obtaining what we call observations from raw data that are available. We simply assume there is a way to gain information about the situation at the level that is given by the domain library. We do not attempt to address this in a general way, because we believe that the required preprocessing of the available data is usually very application-specific.

Controlling the search: in a rich domain, there will be many branches to pursue during the search for consistent situations. Lacking general heuristics for determining promising foci, we envision an interactive solution, where the user selects the branches to be pursued. However, this creates the necessity of providing him with a clear and informative picture of the state of the search, which is non-trivial

Determining introducibles: defining the boundary of the search in terms of specifying introducibles is a necessity, but far from being straightforward. It will usually be impossible to expect a user to define introducibles in a comprehensive way beforehand, because it would require anticipating the potential causal explanations generated by the DSS. Again, the only feasible solution appears to be an interactive one, where the user decides on the fly, whether or not something needs further causal analysis.

Acknowledgements

Many thanks to Ulrich Heller for contributing to the foundations of this work and to Hosain Ibna Bashar, Iliya Valchev, and Xin Ruan for discussing and implementing the ideas.

References

[de Kleer-Williams, 1987] J.D. de Kleer and B.C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 31(1), 1987.

[Forbus, 1984] K. Forbus. Qualitative process theory. *Artificial Intelligence*, 24, 1984.

[Heller, 2001] U. Heller: Process-oriented Consistency-based Diagnosis-Theory, Implementation and Applications. Dissertation, Fakultaet fuer Informatik, Technische Universitaet Muenchen, 2001.

[Heller and Struss, 2002] U.Heller, P.Struss: Consistency-Based Problem Solving for Environmental Decision Support. In: *Computer-Aided Civil and Infrastructure Engineering* 17 (2002) 79-92. ISSN 1093-9687

[Roque *et al*., 2003] Waldir Roque, Peter Struss, Paulo Salles, Ulrich Heller: Design de um sistema de suporte à decisão baseado em modelos para o tratamento de ÁGUA. In: *I Workshop de tecnologia da informação apliacada ao meio ambiente - cbcomp2003*, 2003, Itajaí, SC , Anais do III Congresso Brasileiro de Computação, 2003, v. 01, n. , p. 1894-1906, ISSN 1677-2822.

[Struss, 2008] P. Struss: Model-based Problem Solving In: van Harmelen, F., Lifschitz, V., and Porter, B. (eds.). *Handbook of Knowledge Representation*, Elsevier, 2008, ISBN-13: 978-0-444-52211-5, pp. 395-465