Solving-Oriented and Domain-Oriented Knowledge Structures: Their Application to Debugging Problem Solving Activity

Slim Masmoudi
University of Tunis 1
Faculty of Human and Social sciences of Tunis
Department of Psychology
slim.masmoudi@fisht.rnu.tn

Abstract. This paper describes and analyzes two different sub-systems of knowledge structures - namely "domain-oriented" and "solving-oriented" knowledge structures - in the results of a case study based on interviews and experimental sessions. This case study consisted of the analysis of the activities of an expert programmer in his debugging of a complex computer system. The paper describes the approach adopted to study the conceptual domain-oriented) structures in relation to the cognitive (solving-oriented) patterns in the context of problem solving activities. The results appear to be consistent with the role of some solving-oriented cognitive patterns in debugging activity and the contextual effects of the problem on the conceptual structures.

1. Introduction

Since the GPS (General Problem Solver) theory developed by [30], several studies have tried to elucidate problem solving strategies and the different stages and operations developed by the human or artificial solver, from an initial state to a final one. These studies, and according to the information processing point of view, have contributed to the better view we now have of the different types of problems and of the role of a set of components, namely problem-space, problem representation, dual search, hypothesis formulation, planning, problem states, and goal-driven solving (for a review, see [18], [39]). Nevertheless, very few studies have addressed the relationships between problem-solving, on the one hand, and knowledge structures, and in particular conceptual nets, on the other (see [25]). This may be due to the methodological and empirical difficulties relative to the study of conceptual structures in problem solving contexts, particularly in complex contexts such as debugging. This may also be due to theoretical foundations which do not take into account relationships between data and concepts, and the role of activation processes during problem solving activity.

The present work is based on the assumption that there are two sub-systems of knowledge structures relevant to problem solving: Solving-oriented knowledge and Domain-oriented knowledge structures. Solving-oriented knowledge structures are cognitive patterns of closely related features of a sub-problem space or a sub-problem.
state. Domain-oriented knowledge structures are conceptual nets which represent domain and problem related knowledge. In addition, two activation processes are considered: data-driven and conceptual-driven activation. In the first, concepts are activated by data selected from stimuli and through the instantiation of human cognitive patterns (e.g. the wrong value of the window ordinate on the screen activates some concepts related to the window in question). In the second, some concepts of the conceptual nets are activated by other concepts and thus become available for the cognitive patterns (e.g. a concept related to the window in question activates an other concept related to the general function of the display, which provides information about one or several attributes).

2. Problem Solving and Knowledge Structures

Problem solving activities are often studied in relation to the strategies and heuristics developed by the subject to produce a solution. This point of view, however, does not take into account the role of knowledge structures, their organization and their activation. Neither does it satisfactorily elucidate the dynamic processes between stimuli and knowledge structures on the one hand, and between knowledge structures themselves during problem solving, on the other.

As an alternative, a different problem solving activity model is proposed here. This model consists of three modules: stimulus structures, knowledge structures and dynamics of resolution (Fig. 1).

![Figure 1: Model of the problem solving activity. Stimulus structures are the mental representation of the stimulus found in the solving context and essentially in the problem space. The dynamics of resolution is mainly the reflection of the interaction between stimulus structures and knowledge structures.](image)

The key question is to elucidate how one organizer and uses what s/he knows about the problem, and how s/he organizes and uses what s/he knows about his/her domain of expertise during problem solving. In this respect, we start from the assumption that there are essentially two knowledge structure sub-systems which contribute to problem solving. The first, solving-oriented, sub-system is made up of cognitive patterns that represent, manage and process the problem data (data-driven information), activate concepts and aim to solve sub-tasks of the problem. The second, domain-oriented, sub-system is composed of conceptual nets that represent and manage the concepts of the domain (conceptual-driven information); these concepts are directly activated by the cognitive patterns (Fig. 2).
These two sub-systems are different in the sense that each includes different functional and structural relations. This distinction is based on the nature of information to be processed (either data or concepts), and its relevance to problem-solving. Hence, it is believed to be more appropriate than the one based on the content of information (declarative or procedural).

3. Problem Solving and Knowledge Structures in a Debugging Context

Many attempts have been made to model the debugging activity without considering at the same time the two knowledge structure sub-systems. Instead, scholars have focused solely on either of the two aspects ([22], [32], [33], [40], [41]). In addition, there has been no obvious distinction between the different knowledge structures that can play a crucial role in complex problem solving activities such as debugging. [40], for instance, like [22], proposed a model based on a concatenation of operations through a set of stages, while [32] introduced the role of the abstraction level in program comprehension.

4. Knowledge Structures

This section consists in the presentation of the basic notions which will be made use of in the subsequent discussion.
4.1 Knowledge Schemas

The notion of schemas alone, is not sufficient to describe and explain the cognitive processes underlying knowledge structures. This makes it necessary to consider the different activation and operating processes of the schema, such as the meaning construction processes studied by Kintsch [22]; see also [29], and the context dependency aspects of concepts reviewed by [2]; see also [3]). Thus, the study of these processes in the context of a problem solving activity becomes important. As knowledge schemas, the expert programmer's cognitive patterns elucidated in the debugging activity are contextualized because they process data stemming from the bugs stimuli. They, consequently, activate some concepts from the highly contextualized conceptual structures.

4.2 Concepts and Semantic Networks

Basically, a concept is the cognitive meaning of a term and the smallest unit of thought processor. Concepts are used to recognize an object as an instance of the concept, to produce or to understand sentences in which the concept is expressed and to develop constructs or cognitive systems using the concept in question.

First put forward by [6] and reviewed by [5], this notion has since been acquiring more and more importance in cognitive science as an expert knowledge representation technique. In a semantic/conceptual network, and according to the Spreading Activation Theory (SAT), concepts are represented by nodes, which are interconnected by links or labeled relations. Various types of relationships can define links between concepts. The most common representation of conceptual networks, as shown in Fig. 3 (form A), is unihierarchal and can be qualified as static. The alternative suggested by the present work (Fig. 3, form B) represents the knowledge core as a spiral cloud found a central concept, the concept which has the greatest member of direct and indirect links (i.e. embeddedness). This representation, inspired from [15] and [38], is dynamic because the central concept and the general structure of the network can evolve according to the context.

5. Conceptual Structures and Expertise

5.1 Conceptual Chunking and Expertise

Chalking is a mechanism which puts together representation units (such as concepts, schemas, scripts). These units resist from the integration and the embeddedness of smaller units that lose their autonomy, thus creating new storage automates. Such

---

1 See [11], [13], [20], [31], [34], [35], [36]; [37]; Concerning program comprehension, see [10], [11], [12], [32], [33]; Concerning planning processes, see [21].

2 proposed by [5].
representation units emphasize the sophisticated and refined hierarchy in the conceptual organization. Concepts and chunks are used unconsciously by experts, through activation processes, to direct and to interpret perceptual groupings and configurations (see [7], [19]). Conceptual chunking plays a crucial role in the expert problem solving activities [18], [28]). The following sections will show how concept chunks can be activated to direct the programmer’s cognitive patterns, which, in turn, select data from stimuli structures.

Fig. 3. Semantic network representations: comparison between two forms.

5.2 Conceptual Structures and Contextual Clues

Knowledge nets represent retrieval structures through activation processes that depend both on direct stimulus information and on contextual information. Contextual is relevant not only to encoding processes, but also to activation processes. [14] presented a review and experimental data showing the importance of contextual information particularly for the expert (for an extended review, see also [2]). It was argued in [26], [27] that, for an expert programmer, contextual features are more important than direct stimulus information. This is even more the case in problem solving activities such as debugging, where contextual clues help the expert have a dynamic mental representation of the bug, reinforce his hypothesis and activate the relevant concepts.

6. Method

Two different techniques were applied to study both solving-oriented and domain-oriented structures and the relations between them. These are verbal protocol and

3 Activation is unconscious and does not need conscious and logical reasoning.
techniques based on semantic networks. These techniques were used in an extended case study of the activities of an expert programmer in his debugging of complex computer system, through genuine (not artificially created) bugs (see [28]). In addition, a questionnaire was prepared and distributed to 60 expert programmers, in order to verify our assumptions about the knowledge structures identified in the case study.

6.1 Verbal Protocol Tracing and Analysis

6.1.1 Tracing Technique
During the debugging, use has been made of active tracing. This corresponds to the real time activity during which the expert programmer describes what he does, both by giving verbal descriptions of his thoughts during the solving process, and by inserting in this description all elements that belong to the solving context, and more particularly, elements that are relevant to his activity. The programmer's activity has also been recorded using a built-in program included in the system to debug.

6.1.2 Types of Corpses to Analyze
From the raw verbal protocols gathered, and according to the segmenting technique, two types of RTD (Real Time Debugging) corpses were produced. The "RTD I" corpses are the result of a "macro-structural" segmenting which is based on type I textual units. These units correspond to phrases connected around a verbal core with all necessary ingredients to make sense (e.g., subject, object). These corpses allow the study of the global solving actions, the different cognitive sub-activities implicit in the solving process, and the solving-oriented knowledge structures. The "RTD II" corpses are based on a "micro-structural" segmenting. A type II textual unit corresponds to a fragment of a proposition that either announces something important in the activity or introduces an important concept. The "RTD II" corpses allow the study of the global solving actions through the fact that the programmer used, as well as the conceptual structures, through what has been called concept-words in the present research (e.g., window, list, defin, grep).

6.3 Corpus Indexing and Analysis
A general model of debugging activity indexing (Table 1) was used to index the "RTD I" corpses, in order to identify cognitive patterns and cognitive sub-activities. This model contains 12 global units and a total of 30 analysis units. The "RTD I" corpses are indexed by three indexing and analysis units which are "action", "global action" and "concept-word". The analysis of the indexed corpses is based on statistical computations.
6.2 Semantic Network Technique

6.2.1 Experimental Procedure to Study Conceptual Structures

Each concept-word identified in the corpus is used as an experimental stimulus manipulated in an experiment. The stimuli are successively presented to the expert programmer who is asked to react by providing as quickly as possible the word which each stimulus triggers in his mind. The result of the experiment is a set of concept/association pairs (Fig. 4).

Table 1. General model of debugging activity indexing.

<table>
<thead>
<tr>
<th>1. Bug representation</th>
<th>8. Cost management</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Information search</td>
<td>8.1 management of the cognitive cost</td>
</tr>
<tr>
<td>2.1 dynamic search</td>
<td>8.2 management of the material cost</td>
</tr>
<tr>
<td>2.2 static search</td>
<td>9. BAC (Bug Appearing Conditions)</td>
</tr>
<tr>
<td>2.3 search about state</td>
<td>9.1 BAC generalization</td>
</tr>
<tr>
<td>2.4 search about process</td>
<td>9.2 BAC fine-tuning</td>
</tr>
<tr>
<td>3. Information search goals</td>
<td>10. Hypothesis formulation</td>
</tr>
<tr>
<td>3.1 well-working/badly-working comparison</td>
<td>10.1 problem reformulating</td>
</tr>
<tr>
<td>3.2 determining when</td>
<td>10.2 new hypothesis</td>
</tr>
<tr>
<td>3.3 determining how</td>
<td>10.3 hypothesis generalization</td>
</tr>
<tr>
<td>3.4 determining why</td>
<td>10.4 hypothesis fine-tuning</td>
</tr>
<tr>
<td>3.5 determining what</td>
<td>11. Hypothesis testing</td>
</tr>
<tr>
<td>4. Interpretation</td>
<td>11.1 positive test strategy</td>
</tr>
<tr>
<td>5. Stimulus</td>
<td>11.1.2 positive target test</td>
</tr>
<tr>
<td>5.1 textual stimulus</td>
<td>11.1.1 positive hypothesis test</td>
</tr>
<tr>
<td>5.2 experimental stimulus</td>
<td>11.2 negative test strategy</td>
</tr>
<tr>
<td>6. Observed phenomena</td>
<td>11.2.1 negative hypothesis test</td>
</tr>
<tr>
<td>6.1 provoked stimulus</td>
<td>11.2.2 negative target test</td>
</tr>
<tr>
<td>6.2 spontaneous stimulus</td>
<td>12. Specific domain knowledge evocation</td>
</tr>
<tr>
<td>7. Nature of activity cost</td>
<td></td>
</tr>
<tr>
<td>7.1 cognitive cost</td>
<td></td>
</tr>
<tr>
<td>7.2 material cost</td>
<td></td>
</tr>
</tbody>
</table>

For each debugging session, the list of concept/association pairs resulting from the experiment represents the first material of analysis. The produced semantic network is called “Out-of-Context Semantic Network” (OCSN), since the programmer is not in a problem-solving situation (debugging).

The second material of analysis corresponds to other pair lists. The first element of each pair represents a concept-word as stated in the verbal protocol. Each concept-word is associated with the one that follows it. The succession of concept-words as stated in the verbal protocol describes the contextual dynamic links which relate each concept-word to the one that follows it. The resulting semantic network is called “Contextualized Global Semantic Network” (CGSN), since it corresponds to the problem-solving situation (debugging activity).
6.2.2 Network Analysis Technique

Use was made of a computer program called "SensNet"([16], [17]) adapted for the purpose of the present research. This program produces a visual representation of the programmer's knowledge core. For each concept, it calculates two values: $n_i$ that corresponds to the number of concepts to which it is directly related, and $n_{i+}$ (embededness) which is the sum of $n_i$ and the number of concepts to which those of the first level are related. The advantages of this analysis are the following:

- it provides the possibility to represent the expert's knowledge as a dynamic semantic network,
- it makes it possible to identify differences between two situations (contextualized situation and experimental uncontextualized situation) according to the structure produced, to the network complexity and to the central concept,
- it makes it possible to identify the central concept, to visualize the chunk of a concept and to compare different networks between them.

To sum up, this analysis shows the relationships between the cognitive (solving-oriented) patterns and the conceptual (domain-oriented) structures. Thus, it shows how these conceptual structures evolve through the problem solving activity.

7. Results and Discussion

Domain-oriented and solving-oriented knowledge structures were firstly studied through the "General model of indexing debugging activity". The results of the
implementation of this model (Table 1) highlight the importance of essentially three
cognitive patterns (Table 2): Bug Cognitive Pattern (BCP), Bug Appearing
Conditions Cognitive Pattern (BACC) and Hypothesis Pattern (HP).

Table 2. This table summarizes the occurred of textual units indexing the different patterns.
Each cognitive pattern has an occurrence rate significantly bigger than the average calculated
over the total of textual units (A = 920 textual units / 12 indexing units = 76.7; P = 8.3%). The
importance of these patterns and their attributes has been confirmed by the questionnaire (Table 3).

<table>
<thead>
<tr>
<th>Cognitive Pattern</th>
<th>N</th>
<th>%</th>
<th>t</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCP</td>
<td>187</td>
<td>20.3</td>
<td>6.8</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>BACC</td>
<td>191</td>
<td>20.6</td>
<td>6.4</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>HP</td>
<td>129</td>
<td>14</td>
<td>3.7</td>
<td>&lt; 0.01</td>
</tr>
</tbody>
</table>

7.1 Solving-Oriented Knowledge Structures

7.1.1 Bug Cognitive Pattern (BCP)

This solving-oriented knowledge structure provides the necessary and sufficient attributes which allow the programmer bug detection and the mental representation triggering a hypothesis formulation process. This pattern is instantiated and updated along the debugging process, through the visualization of different stimuli and the experimental manipulations of the programmer.

Fig. 5. Schematic representation of the “Bug Cognitive Pattern” and its illustration with the “move-link” bug. The BCP is supposed to have a decoding capacity of all information related to the bug.

Results emerging from the questionnaire show the importance of the instantiation of the BCP attributes. Table 3 summarizes the possible instantiation percentages of the expert programmers who refer to an attribute.

For further analysis of the BACC, see [27] Because of space constraints, this point cannot be investigated here.
Results show the significant importance of the different BCP attributes identified according to verbal protocols, and their different instantiations.

<table>
<thead>
<tr>
<th>Pattern attribute</th>
<th>expected value</th>
<th>Attribute instantiation / %</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Defining the program’s abnormal behavior</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Defining the initial state</td>
<td>data input</td>
<td>initialization variables</td>
<td>before execution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Defining the user action</td>
<td>expected cause</td>
<td>leading to the last expected result</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Defining the final state</td>
<td>first computing step</td>
<td>program’s end</td>
<td>executed code</td>
</tr>
<tr>
<td></td>
<td>incorrect with expected result</td>
<td>after the bug appearing</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Defining the expected value</td>
<td>predicted value</td>
<td>result provided from other computing mean</td>
<td>specified result</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Defining the deviation</td>
<td>quantitative/qualitative deviation</td>
<td></td>
<td>gradual deviation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Defining the values domain of the expected result</td>
<td>interval</td>
<td>type of result</td>
<td>valid/invalid</td>
</tr>
</tbody>
</table>

### 7.1.2 Hypothesis Pattern (HP)

The hypothesis is defined, in the present work, as a solving-oriented knowledge structure. The expert programmer is supposed to instantiate, throughout the debugging session, the attributes of this structure in interaction with the bug mental representation (BCP).
7.2 Domain-Oriented Knowledge Structures

Conceptual domain-oriented structures are activated by the contextual clues (data) through the solving-oriented structures. Four aspects were adopted for the study of the contextual effect: semantic network complexity, general structure, central concept, and central micro-network.

CGSN (contextualized) is more complex than OCSN (uncontextualized) since there are more links between its elements. The debugging context has an effect on the programmer's knowledge core of activated concepts, as shown by the greater complexity of the network. Instead, debugging context activates more concepts and provides temporary relations between them. According to the contextual clues, the central concept and the micro-network differ in CGSN and OCSN. The central concept in CGSN, which is the most embedded one, has a strong relationship with the bug solution (the error consists, in this example, in a missing attribute - an unpredicted case - of the "declignot" function).
Thanks to its multiple activations throughout the problem-solving activity, and its high degree of interconnectedness, a concept becomes central and strongly related to the bug location. The activation processes described above depend highly on the different cognitive pattern instantiations (RCP, BACCP, and EFP).

With the semantic network technique, the conceptual chunk of a selected concept can be visualized. For each selected concept, this chunk displays the other related concepts, both in COSN and OCSN. Significant differences were found between the embeddedness of the related concepts of a chunk from COSN to OCSN. Results showed also strong relationships between the cognitive patterns and the conceptual structures (essentially the different chunks) throughout the debugging activity process.

8. Conclusion

This paper is based on the assumption that a distinction has to be made between solving-oriented and domain-oriented knowledge structures. It is supposed that the latter correspond to conceptual structures which are activated in interaction with the cognitive patterns of some aspects of the problem space. These patterns play a crucial role in decoding stimulus and contextual information, and in domain-oriented concept activation. The discussion of a part of the presented results emphasizes this distinction, which needs to be further explained and emphasized using other experimental techniques, over other problem-solving activities. This distinction does not contradict strong relationships between the two knowledge structure sub-classes. It may be a theoretical foundation of future problem-solving simulations.

The semantic network representation, that was implemented through the "SemNet" technique, seems to be a powerful means to study conceptual structures and contextual effects. Indeed, it makes it possible to distinguish the stable knowledge...
cues and the contextual-based one. Nevertheless, it would be interesting to investigate, in future studies, the activation processes and the interactions between the cognitive patterns and the conceptual structures, in other complex problem solving activities.

References


5 Conceptual domains, conceptual progress during a problem solving activity and solving-oriented knowledge structures are further investigated in [28].