Problem Solving in Control of Discrete-Event Systems

LENKO GRIGOROV lenko.grigorov@banica.org
School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada
KAREN RUDIE karen.rudie@queensu.ca
Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario K7L 3N6, Canada

Abstract— **Human performance in problem solving in the context of supervisory control of discrete-event systems is discussed. Software packages for such problems usually offer only rudimentary support in the form of visual data entry and algorithmic computations. Other activities such as system modeling and verification of the solution have to be done without computer support. This work is a first step in understanding how humans approach discrete-event control problems. An observational study is described, where subjects are video-taped solving supervisory control problems. One video recording is used to construct a taxonomy of the task and then the different activities of the subject are encoded accordingly. The data are analysed to discover patterns of human activity. These results can be used to guide the design of software for computer-aided problem solving in this context.**

Keywords: Discrete-Event Systems, Supervisory Control, Human Problem Solving

I. INTRODUCTION

The field of research investigating the control of discrete-event systems (DESs) was established with the seminal work of Ramadge and Wonham [1]. One of the main strengths of their theory of Supervisory Control of DESs is that there exists a polynomial-time algorithm which, given a set of hard constraints, can compute automatically the most permissive system supervisor. The first implementation of this algorithm appeared in the software package TCT [2].

The user interface of this software package has been critiqued unfavorably [3], however, even the latest versions offer no significant improvements. More recently, newer software packages such as [4], [5], [6] have improved the experience of the user by taking advantage of the graphical abilities of the interface of the desktop computer. The visibility of DES models is improved by displaying them as Finite-State Machines (FSMs). As well, the user can interact with these representations directly via the mouse cursor. For example, in IDES, a "pen-and-paper" user interface paradigm is used. The user can create consistent structures of both states and transitions by clicking inside the model view without having to switch between different "modes of operation".

Despite all advances in user-friendliness, all reviewed software for supervisory control of DESs suffers from two main disadvantages:

1) The representation of very complex DES models is not helpful for the user. Some DES models may have a state space of a size greater than 10^6. The visual FSM representation of such models is meaningless to the user; as is a simple list of states and transitions. Humans have a limit on the amount of information they can attend to at any moment in time [7]. It is practically impossible to discover significant relationships between such enormous number of entities if each one has to be examined separately.
2) The main functionality around which the software is designed is the performance of the raw computations of DES algorithms. Additional support is provided only for the input of DES models. However, there are other activities that take place in DES problem solving. These include the following:
   - creating a formal model of a real system
   - the verification of the result of computations
   - the implementation of the result of computations

   Existing software does not offer support in these areas.

Researchers in Human-Computer Interaction (HCI) point out [8], [9] that our understanding of the human factors in computer use and in information visualization is very rudimentary. There are no central theories which reliably and robustly describe human thinking and performance when dealing with a complex and interactive device such as a computer. Thus, a simple solution to the problems mentioned earlier is not possible.

The main goal of the research, a part of which this study describes, is to create an approximate model of the cognitive processes involved in DES problem solving. The collected data, due to their inherent limitations, do not allow for a reliable statistical analysis. However, they may indicate trends which can be useful in the design of a new user interface for DES software. While standard evaluative methods for software interfaces, such as user testing and interviews, may reveal problems with a specific design choice, they are not able to uncover the reasons that underlie the success or failure of a design. This is why we believe that understanding the cognitive process of problem solving is an important first step to the design of effective interfaces. Such understanding would allow the software to use predictions from the cognitive model online—to make the work of DES researchers more seamless and offer enhanced support throughout the process of problem solving.

No literature discusses formally the topic of DES problem solving as exhibited by human subjects. However, some authors make anecdotal observations. In [3], the authors discuss the usability issues of the TCT package as it pertains to the activities carried out in DES problem solving. The two main problems discovered are that certain commonly used algorithms are not available and that the interface of the program is not graphical. The authors recommend that graphical images be used to represent states and transitions which could, in turn, be labeled using alphanumerical symbols. In our opinion, the authors' claims that this would make a DES system model truly understandable is not well substantiated, especially when one considers the limitations of human perception and the difficulties of designing good visualizations of huge amounts of data [9] (real-world models may have thousands of states). In [10], Leduc describes his experience in implementing the control solution for a complex DES system of 10^{16} states. His recommendations are that large components be broken down into small interacting modules (and he considers a module with three hundred states as too big). The modules should be classified as "fundamental" and "interaction". This simplifies the design process since the relations between modules becomes more apparent. Unfortunately, this work focuses mostly on the technical aspects of the implementation of control and the discussion of the designer's experiences is very brief. In [11], the author points out that the modeling of the system, specification and supervisor is a more arbitrary process where each affects the other's design. This is especially pertinent to situations where the system is not yet built, or where the system is implemented using programmable circuitry. In the latter case, there is great freedom in designing the "unrestricted" behavior of the system, since a large part of the behavior is actually generated by the control circuitry.

An example of how problem solving in humans can be examined and how the collected information can be used in software design can be seen in the work of Rogers *et al.* [12], [13], [14]. The authors conducted an observational study of how radiologists, both novices and experts, use X-ray images to make diagnoses. The relevant activities and the types of information accessed were recorded and analysed to determine common patterns [12], [13]. Then, the results were used in the creation of a model of the cognitive activity of radiologists in this task [15]. A software application was developed to aid in the process of diagnosis [14]. The organization of the information presented to the users was tailored to match the the mental model. As well, specific operations were available to manipulate the data so that a successful diagnosis becomes more likely.

An approach similar to that of Rogers *et al.* was undertaken in our work to investigate the cognitive processes involved in problem solving of control of DESs and to identify what data are accessed and how they are accessed. We describe the methods used in the investigation. Then, the preliminary results are presented, followed by a discussion of the findings. At the end, we make some conclusions and discuss the future direction of the research.

II. Methods

The observation of human cognitive activity is not possible directly. Early on in cognitive research, researchers used the method of *introspection* where subjects would report as objectively as possible their own experiences [16]. However, this method proved to be scientifically unsatisfactory due to the highly inconsistent data that were collected. In [17], Ericsson and Simon describe a new method for the collection of data, called *protocol analysis*, which has since become a *de facto* standard when human cognitive activity is examined. The method relies on asking the subjects to "think aloud" while performing the given task. The verbalization is recorded and later analysed. Unlike the introspection reports, the verbal information is unstructured and is believed to interfere much less with the performance of the task.

A. Observational study

In order to investigate the cognitive activity involved in solving DES problems, we recruited five subjects of both genders, all of whom were graduate students with some knowledge of DES control. Two subjects had experience only through a course on DES supervisory control theory and the others had done research in the area. The subjects were presented with DES problems and asked to "think aloud" while solving them. A video and audio recording of their performance was made. Furthermore, any additional artifacts produced, such as written records or computer files, were retained.

Each subject performed under two conditions (the order of the conditions was randomly assigned). In one condition, the subjects had to solve a simple problem which had been extensively discussed during the DES course each subject had taken. It involves a factory setup of machines, buffers and a testing unit. In the other condition, the subjects had to solve a problem which was formulated to be as close as possible to the factory problem, however, in a hospital setting. The entities in this problem consist both of equipment and of people (doctor, nurse, etc.) Both problems were cast in the original paradigm of supervisory control, i.e., without extensions such as partial observability, [18], etc. The subjects were free to solve them using modular control, [19], however, this was not a requirement. In order to reduce the influence of the solving of one problem on the solving of the other, subjects performed under both conditions with a minimum of one week's time in between.

The tools that were available to all subjects were pen and paper and a computer running a version of the IDES software package. The software allows for the graphical modeling of FSMs and for the performance of the algorithms needed to compute a supervisor for a system. The subjects had up to one hour to solve the problem under each condition and they were free to switch between the pen and paper and the software at any point and as many times as they wished.

B. Methods of analysis

The analysis of the data collected proceeded in separate stages according to the recommendations described in [13]:

1) Use prior experience to define a taxonomy of DES problem solving;
2) Refine the taxonomy from the observed data;
3) Encode the observations as per the proposed taxonomy;
4) Analyse the encoded form to find patterns and the interleaving of cognitive activities.

1) Preliminary taxonomy: The initial definition of the taxonomy resulted in the following outline:

1) Understand the problem. This includes activities such as reading the description of the problem, deciphering any diagrams, creating one's own explanatory diagrams, etc.
2) Define the problem formally. This may be done in parallel with understanding the problem. It includes activities such as determining what the system is and what the control specification is, describing the dynamics of the system and the desired behavior, deciding on the controllability of events etc. The description of the dynamics involves low-level activities such as drawing states and transitions, denoting states as initial and/or marked and others. After the formal definition is ready, the subject may "cross a line" and start referring only to the formal model for all subsequent tasks.
3) Mathematical computations. This includes activities such as inputting the formal model into software (which may be done while defining the problem formally), composing DES modules and running the supervisor generation algorithm.
4) Verify output. This stage may be omitted by the subject if they decide to trust the output from the mathematical computations (which is likely when the output is huge). Otherwise, the verification may proceed by using the verbal or the formal definitions. Manual verification includes activities such as glancing to discover irregularities in the result, performing the computations by hand and cross-referencing with the output, counting states/transitions, tracing of strings, etc. Automated verification may involve algorithms such as checking language containment (or the containment of arbitrary strings) or checking controllability. If the output is considered unsatisfactory, the subject may return to any of the previous stages, depending on the specific problem discovered.

2) Refined taxonomy: When the actual video recordings were considered, it became apparent that the taxonomy based on previous experience is not suitable for the encoding of the data. The taxonomy is interpretive in nature: low-level events have to be assigned to classes even though the mapping is not unambiguous. Furthermore, many low-level events are not defined very precisely and some activities are even omitted. Thus, we decided to create a separate task taxonomy where only low-level events are defined. The goal was to minimize the degree of interpretation necessary to apply the taxonomy by referring, when possible, only to the mechanical properties of events.

The main distinction of events is along four axes:

- The subject verbalizes an idea;
- The subject examines the problem definition;
- The subject models using pen and paper and
- The subject models using software.

Additional events such as looking at the computer screen, interrupting the problem-solving process (e.g., to drink water), switching between modeling with pen and paper and with the software, etc. were also included but not assigned to a specific axis. Besides the main four axes, events were further refined as to whether they pertain to DES modules (separate parts of the system and control specifications), events, states, transitions, or DES algorithms. Certain stages such as model inspection or model verification may also be identified if possible. At last, each event may have a parameter to specify the context: for example, the parameter of a "module" event will be a specific module (such as "buffer 1"). A more detailed description of the current taxonomy, as derived from the observation of one video session, is shown in Table I. We expect that as more video sessions are examined, it will be expanded. For example, the subject in this video session did not draw any additional explanatory diagrams, while other subjects were observed to do so. Thus, events for diagram drawing will have to be added.

Some types of events	
Axis	Description
A	Announce idea verbally
C	Perform using the provided software
M	Perform using pen and paper
R	Read written material

Some types of entities referred to	
Ref	Description
C	Computation/algorithm
E	Event
M	Module
S	State
T	Transition

Some types of stages	
Stage	Description
I	Inspection
V	Verification

Some types of actions	
Act	Description
B	Modify appearance
C	Count
I	Make initial
R	Remove

Some combinations	
Code	Description
AS	Verbalize thought of a particular state
ATV	Verbalize thought of verifying a transition
CM	Create a module using the software
CTR	Remove a transition using the software
CC	Invoke a DES algorithm using the software
CB	Change the appearance of the model in the software
MSI	Make a state initial in the pen-and-paper model
ME	Write down a new event on the paper
MTIC	Inspect the pen-and-paper model by counting the transitions

TABLE I

PARTIAL LIST OF THE TAXONOMY USED IN MARKING UP EVENTS IN THE VIDEO RECORDINGS.

3) Data encoding: After the current taxonomy of events was prepared, we proceeded with the markup of the video session which was used to prepare the taxonomy. In order to set the timestamps of events as they occur in the video sequence, we used software designed for the creation of subtitles. Each event was marked up as a subtitle. The output of the software, a file with subtitles, was further processed by a custom-made application which converts it into a form suitable for the following analysis. The markup was done by a single person.

4) Analysis algorithms: The sequence of events in the problem solving, as obtained from the markup of the video session was analysed using n-grams and clustering. N-gram analysis [20] is the determination of the frequency of occurrence of a specific sub-sequence of n items in a larger sequence. For example, in the sequence "abcdbbc", the 2-gram (or *bigram*) 'bc' occurs two times, while the 3-gram 'bcd' occurs only once. In our study we computed both absolute and relative ratios of n-grams. An absolute ratio is the ratio of the number of occurrences of a given n-gram to the total number of n-grams in the sequence. We use the term "relative ratio" to refers to the ratio of the number of occurrences of an n-gram to the number of occurrences of all n-grams which start with the same $n-1$ items. For example, in the sequence "abcdbbc", the relative ratio of the bigram 'bc' is $2/3$ since there are two occurrences of 'bc', one occurrence of 'bb' and no other bigrams starting with 'b'. In comparison, the absolute ratio of 'bc' is $2/6$ since 'bc' occurs twice and in the sequence there are six bigrams in total.

The markup of the video sequence results in a sequence of low-level events. This is convenient for the analysis of how certain actions interleave, however, it is difficult to use the data to answer questions at the meta level, e.g., what type of activity does the subject engage in at a given moment. Thus, we decided to use unsupervised clustering of the data to obtain aggregates.

Unsupervised clustering [21] is a method to assign data items to separate classes without having some prior idea of how many of these classes there should be and what the criterion of distinction is. The clustering algorithms utilize a measure of the distance between individual items as the base for generating classes and assigning items to them. A number of algorithms are available, however, none of the algorithms seemed suitable for the task of clustering our data. Thus, we implemented a custom clustering algorithm (Fig. 1). Each data item is assigned a type depending on a chosen combination of its properties. For example, items in our study may be assigned type depending on what the low-level events refer to (i.e., modules, events, states, transitions or algorithms). Clusters also have types and the type of a cluster is intended to reflect the type of the majority of its items. Let $type$ stand for the type of item or cluster and $time$ stand for the time of an item or the mean time of all items in a cluster. Given two clusters, A and B, let C denote the union of all clusters between them, inclusive; $C = \bigcup_{time(D)\in[time(A),time(B)]} D$. In other words, C represents the cluster which will result in the merging of A and B. The distance, d, between A and B is computed as follows.

```
cluster($list$) {
/* returns $\Lambda$, a set of clusters */
$\Lambda := \emptyset$
for($a \in list$)
    $A := \{a\}$
    type($A$):=type($a$)
    $\Lambda := \Lambda \cup \{A\}$
if($|\Lambda| <= 1$)
    return $\Lambda$
do
    $T := \{(I,J)|I,J \in \Lambda, time(I) < time(J)\}$
    $M := \{(A,B)|d(A,B) = \min_{(I,J)\in T} d(I,J)\}$
    $(C_1,C_2)$ :=random($(A,B) \in M$)
    if($d(C_1,C_2) > 0$)
        break
    for($C \in \{D|$time$(D) \in [$time$(C_1)$,time$(C_2)]\}$)
        $\Lambda := \Lambda \setminus \{C\}$
        /* type($C_1$) remains unchanged */
        $C_1 := C_1 \cup C$
    $\Lambda := \Lambda \cup C_1$
while($|\Lambda| > 1$)
return $\Lambda$
```

Fig. 1. Custom clustering algorithm

If $type(A) = type(B)$, then

$$\begin{aligned} d(A,B) \quad = \quad & \sum_{c\in C} abs(time(C) - time(c)) \\ & +N \times |\{c \in C|type(c) \neq type(A)\}| \\ & -P \times |\{c \in C|type(c) = type(A)\}|, \end{aligned}$$

otherwise $d(A,B) = \infty$. Here P and N are coefficients which can be used to parametrize the algorithm. The algorithm starts off with one cluster per item; and the type of each cluster equals the type of the item contained. Then, the algorithm enters a loop where, at each iteration, the most beneficial merging of clusters is performed (the two clusters with least distance). Notice that only the merging of clusters of the same type is considered (since the distance between clusters of different types is ∞). Furthermore, all clusters on the time line between A and B will also be merged into the new cluster. The loop terminates when there is only one cluster left or when the distance becomes positive.

III. Preliminary results

The preliminary results of the analysis of one video session are presented next. The subject was given the task of finding a supervisory control solution for a DES consisting of a factory setup. Two machines and a test unit are interconnected via buffers into a processing line. The control specification is to prevent underflow or overflow of the buffers. A more detailed description of this problem can be found in [22]. The subject proceeded by modeling the system and specification first on paper. In the subsequent discussion, we will call this period of the problem-solving "paper period". After this, the subject used the IDES software to input the model, perform algorithmic computations, verify the solution and correct the model. Subsequently, we will refer to this period as "computer period".

A. Duration

One comparison of interest is how long different parts of the problem solving lasted. If we take into account all interruptions, such as necessary software setup procedures, the paper period lasted 12:06 min, while the computer period lasted 34:22 min. This makes almost a 1/3 ratio of how the solving activity was distributed in time. It has to be considered, however, that the paper period included only the reading and understanding of the problem and the initial formal modeling. The computer period included the data entry, the performance of the computations and the verifications of the produced solution. The latter resulted in the subject discovering an error in their formal model, so the model had to be fixed and the result re-verified. Given these activities, it is not surprising that the subject spent 7:17 min examining the problem definition during the paper period. They spent only 3:47 min in the computer period examining the problem definition and the models they had produced.

During the encoding of the video session it became apparent that the subject engages very frequently in modifications of the visual appearance of the data. This included most frequently repositioning of states and transition labels. A further look into this revealed that the subject modified the visual appearance on 243 occasions during the computer period. This consists of approximately 33% of all encoded events for the period. If one considers that each modification lasts at least 2 sec, this means that the subject spent at least 8 min, or 23% of the computer period, changing the visual appearance of the models.

B. Reference of entities

In order to get a better understanding of how a task is performed, it is important to look at what tools are used and in what sequence. From the full transcript of the video session, we extracted only the data which reflect this aspect of problem solving. For each encoded item, we observed whether the subject refered to one of the following entities: modules (M), events (E), states (S), transitions (T) or computations (C). By "reference" we mean the subject mentions such an entity or directly manipulates it (such as the drawing of a state on the sheet of paper). Then, we looked at the sequence at the low level and, after clustering the data, also at the high level.

1) Low-level analysis: We consider the analysis of the original data low-level since the data reflect low-level activities—such as the actual placement of a transition with a mouse click or the verbal reference to a DES module. Such data may reveal some regularities in how a person would perform the task of finding a DES control solution. We used bigram analysis to obtain the absolute and relative ratios (see Section II-B.4) of all bigrams which appeared in the process of problem solving, both in the paper period and in the computer period. A chart of the data is shown in Figs. 2 and 3.

In Fig. 2 the bigrams are sorted according to their frequency during the paper period. It can be seen that the most common bigrams are the ones that refer to the same entities, i.e., "S,S", "T,T", and "E,E". This means that the subject was likely to continue working with the same entities without interruption. This is further supported by the high relative ratios of these bigrams; the relative ratios serve as a predictor of how likely is the occurrence of a given bigram if the subject refers to the first entitiy in the bigram. For example, in the paper period, if the subject referred to a transition, it is about 65% likely that the next entity they will refer to will be a transition—since the relative ratio of the bigram "T,T" is 65%. It can also be seen that the subject quite frequently alternated between working with states and with transitions (bigrams "S,T" and "T,S"). This is probably due to the way the subject drew their DES models.

In Fig. 3 the bigrams are sorted according to their frequency during the computer period. When comparing the data with the ones from the paper period, it can be seen that the histogram of the ratios is similar: most common are the bigrams with entities of the same type and the bigrams of alternation between states and transitions. These bigrams, as in the paper period, have high relative ratios. It is interesting to notice, however, the differences between the two periods. It appears that only in the computer period did the subject have a longer stretch of working with modules; the "M,M" bigram has high absolute and relative ratios only in that period, while, during the paper period, dealing with modules usually occurred only alongside other activities. Another interesting observation is that working with events occurred in a very different sequence during the two periods. While in the paper period, the bigrams "E,M" and "M,S" had high absolute and relative ratios, in the computer period, the bigrams "M,E" and "E,S" dominated both ratios. Observations of the video session confirm that during the modeling on paper the subject preferred to list the DES events, then proceed with the modeling of a module. During the computer modeling, the interface of the software required that a module be created first, before entering any events.

2) High-level analysis: While the low-level analysis reveals how problem solving is executed, it is not suitable for the investigation of the significant steps which the subject takes. Thus, the clustering algorithm described in Section II-B.4 was used on the low-level data to obtain groups which signify related activities. An example of the result of the clustering algorithm for the subject's reference to DES modules is shown in Fig. 4. Each cluster is tagged with the entity to which the subject consistently refers during the given set of activities. Thus, clusters can be viewed as aggregations of the low-level data and can be analysed in the same way.

The n-gram analysis of the clustered data from the paper period did not reveal any consistent patterns, except a general trend towards modeling by using entities in the sequence "M,S,T". The most significant part of the results for the 4-gram analysis of the computer period are shown in Fig. 5. It can be seen that the most frequent sequence of entities referred to during the use of the software is "M,E,S,T" (or a rotation thereof). In other words, the subject dealt with a module, then the events, then the states and then the

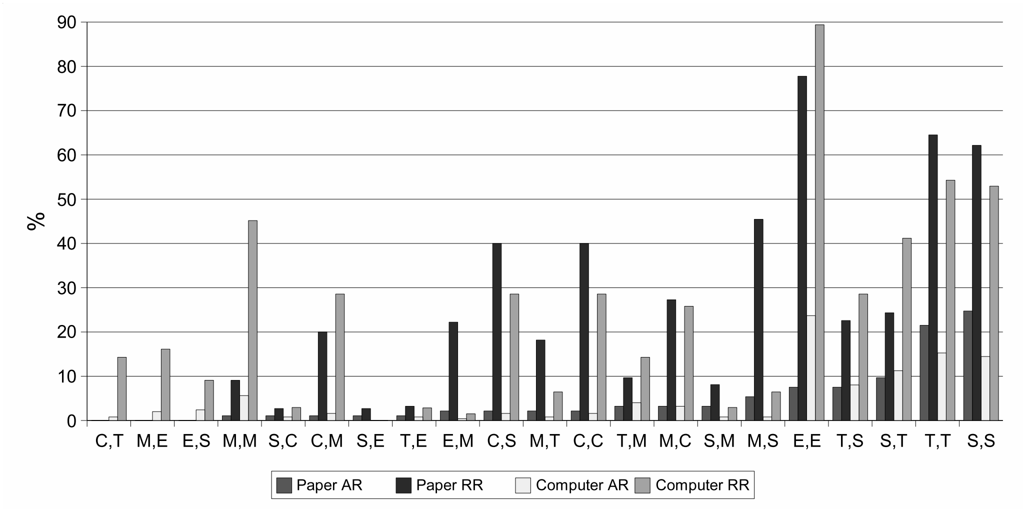

Fig. 2. Chart of the ratio of occurrences of bigrams. The data are sorted according to the absolute ratio of occurrence during the pen-and-paper modeling. 'Paper' refers to ratios during pen-and-paper modeling. 'Computer' refers to ratios during use of software. 'AR' stands for absolute ratio; 'RR' stands for relative ratio.

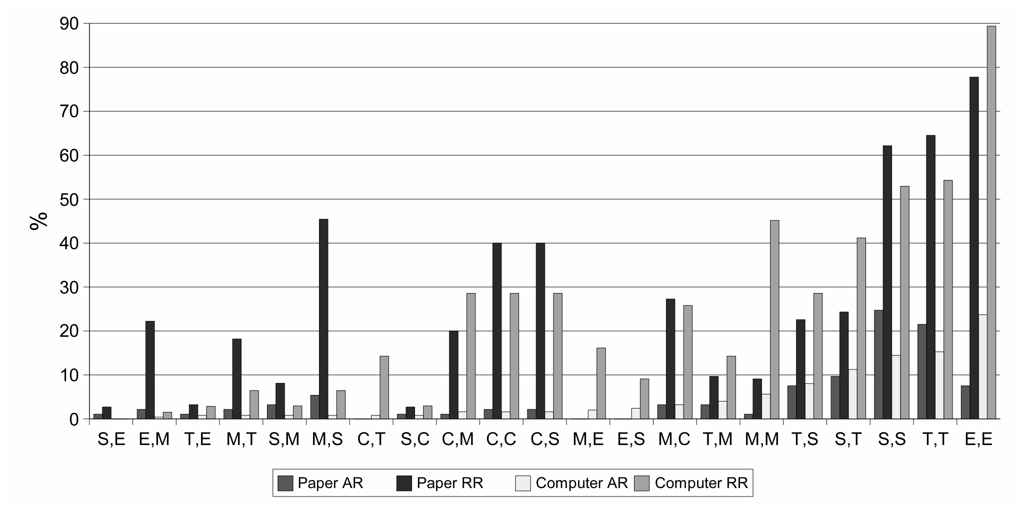

Fig. 3. Chart of the ratio of occurrences of bigrams. The data are sorted according to the absolute ratio of occurrence during the modeling with software. 'Paper' refers to ratios during pen-and-paper modeling. 'Computer' refers to ratios during use of software. 'AR' stands for absolute ratio; 'RR' stands for relative ratio.

transitions. This result was supported further by the bigram and 3-gram analysis of the same data. Visual inspection of the clustering and the video recording reveals that this is, indeed, the sequence which the subject preferred to follow. However, it has to be noted that this sequence is prevalent only in the first part of the computer period, i.e., when the subject inputs the models into the software. It is possible that the exact form of the sequence is influenced by the software which imposes constraints on the order in which data can be input (e.g., it is not possible to input events unless a module is specified).

Another result which can be noticed from the data for the computer period is that computations (C) are most referred to after references to modules. The only bigrams which contain a 'C' in the second position are "M,C" and "S,C" and the ratio of occurrence of "M,C" is seven times greater than the ratio of "S,C". This indicates that the subject usually had to "back-up" from considerations of particularities (such as states and transitions) and think at the module level in order to consider the application of algorithms.

The study of the distribution of references over time revealed that both during the paper period and the computer period, the subject refers to events (E) only during the first stages of modeling. The set of events is relatively stable once built. Another thing noticeable through visual inspection is that, in the computer period, when the subject is verifying their solution, in the beginning they focus primarily on states and then primarily on transitions. This is seen on both occasions of verification.

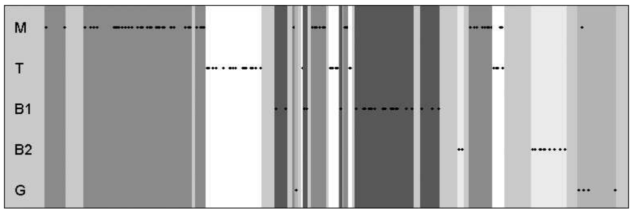

Fig. 4. Clustering of the beginning of the transcript for the period when the subject used software for problem solving. Data points are arranged on the Y axis according to the DES module they refer to. The X axis represent the flow of time. Uniformly colored intervals of points show clusters.

C. Reference of modules

After looking at how tools (modeling entities) are used in the process of problem solving, it is important to see what model data are accessed in what way. For this purpose, we extracted from each encoded item in the video session what DES module it referred to. The modules considered were machine 1 or 2 (M), test unit (TU), buffer 1 (B1), buffer 2 (B2), complete system composed of the machines and the test unit (G), complete specification composed of the two buffers (E) and supervisor (U).

The n-gram analysis of the low-level data for both periods reveals that the subject, once working on a specific module, is likely to continue working on it (i.e., sequential activities are not likely to refer to different modules). This is evident even in the 4-gram analysis (where "M,M,M,M" and "B1,B1,B1,B1" are the 4-grams with highest absolute and relative ratios). This result, however, is fully expected since a person is highly unlikely to be working concurrently on two modules and, for example, model transitions by making one in each and alternating between the two modules.

Again, in order to look at how the subject refers to modules at the high level, we used the clustering algorithm to aggregate the data. A part of the resulting clusters is shown in Fig. 4. The n-gram analysis of the clusters did not reveal any notable results with one exception: in the second part of the computer period the subject dealt most frequently with

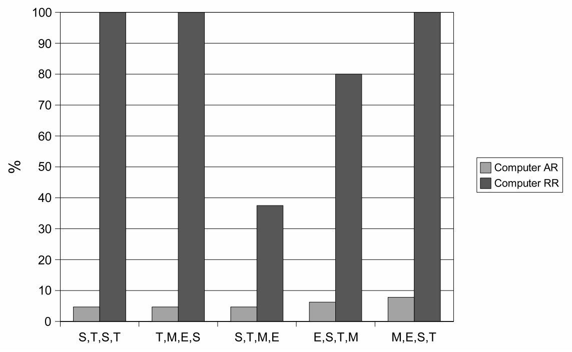

Fig. 5. Chart of the ratio of occurrences of 4-grams with references to modules, events, states, transitions, and computations. The data are sorted according to the absolute ratio of occurrence during the modeling with software. 'AR' stands for absolute ratio; 'RR' stands for relative ratio.

the supervisor computed for the control solution (U). One third of all clusters referred to the supervisor. This is not unexpected, since the subject dedicated a large portion of the computer period to the verification of the computed solution.

The visual inspection of the clustered data reveals a few interesting observations. In the paper period, the modeling proceeded by referring roughly to the sequence "M,T,G,U,B1,B2,E", while in the computer period, the sequence was roughly "M,T,B1,B2,G,E,U". The problem description refers to the modules in the order "M,T,B1,B2"; most likely this played a role in the order in which the DES modules were modeled. The significant difference between the two periods is that during the pen-and-paper modeling, the subject chose to investigate the composition of the complete system (G) before modeling the buffers, while during the modeling with software, this was delayed until all individual modules had been modeled. The appearance of a reference to the supervisor (U) during the paper period does not imply that a supervisor was built (indeed, it is impossible to do it without first modeling the specifications). The reference was only brief and verbal—indicating that the thought of the supervisor module had crossed the subject's mind. Another thing to note is that in the computer period, during the modeling of both buffer 1 and buffer 2, the subject interrupted the process to look first at the machine modules and then at the test unit module (see Fig. 4). The reason, as indicated by the subject, was that they wanted to check again the exact properties of the events in these modules since they needed to enter the same events when modeling the buffers.

IV. Discussion and conclusions

The results presented in this paper are only preliminary in the sense that only one video session of the observational study has been analysed. The process of defining the task taxonomy, marking up of the video record, and analysing the data was rather exploratory since no previous work exists on the problem solving of DES supervisory control. At this point it is too early to draw any significant conclusions, however, as described next, some useful information can already be recognized from the analysed data. The main contributions of this work are the following:

- The proposition of a formal approach to the investigation of human performance in DES problem solving, using information about the fundamental cognitive processes;
- The creation of an initial taxonomy of the problem solving activity from the human perspective; and
- The collection of performance data from an observational study and the development of a method for analysis.

Our findings show that, indeed, there are some discrepancies in how humans proceed in solving a DES control problem using pen and paper versus software. During the pen-and-paper modeling, the subject spent very little time actually creating new information (the models) in comparison to how much time they spent examining existing information (reading the problem statement). During the use of the computer, they spent the majority of the time creating new information rather than examining existing information; the only exception was during the verification of the solution. One surprising result is that the visual appearance of the models was of such great importance to the subject—as they spent about a quarter of the time adjusting the appearance of the models. This indicates that one of the areas where DES software designers should focus on is the advancement of the visual representations.

The discrepancies between the workflow during pen-and-paper modeling and modeling with software are, in our opinion, most probably due to two factors:

1) the different stages of problem solving: when pen and paper was used, the subject was translating verbal specifications into a formal model, while when the software was used, the subject was translating a written formal model into a digital model; and
2) the interface of the software imposed constraints on the modeling which are not present when pen and paper are used.

If software is to offer support in the process of initial modeling (not only in the input of existing models), it is advisable that as many constraints as possible be removed. The user should not be forced to follow a specific sequence of steps only because this is what makes most sense in the underlying implementation of the software. Examining what information is accessed and how during pen-and-paper problem solving may give important feedback to software designers.

The rest of the information obtained through analysis is mostly confirmatory in nature. As expected, problem solving proceeds in "chunks" of related activities, e.g., relating to the same module. Furthermore, it can be seen that thinking about DES computations requires considerations at the level of modules, not states and transitions—and this is not a surprise.

A. Future work

The research presented in this paper is only the first step of a much larger investigation. All ten video sessions have to be encoded and then analysed. The taxonomy of DES problem solving has to be revised to include activities which other subjects engage in but the current subject does not. Additional methods for pattern discovery should also be employed to supplement n-gram analysis and clustering.

The main goal of this study is to create an approximate model of the cognitive processes involved in DES problem solving. If this succeeds, we will use the information to design a new user interface for DES software and implement a back-end which will use the cognitive model to enhance the software support offered to DES researchers. In addition, the new interface will have to be tested to check what impact the introduced changes have on the performance of users.

As a separate venue of investigation, an online statistical analysis of the low-level actions in the software may allow us to distinguish and create "use profiles"—to adapt better to the requirements of users.

V. Acknowledgments

We would like to thank Brian Butler from the Department of Psychology, Queen's University and Dorothea Blostein from the School of Computing, Queen's University. Their useful advice helped in setting up the observational study and analysing the data.

This work was supported, in part, by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.
[2] "CTCT software." Department of Electrical and Computer Engineering, University of Toronto, Canada. Available at http://www.control.toronto.edu/DES/.
[3] C. M. Enright and M. Barbeau, "An evaluation of the TCT tool for the synthesis of controllers of discrete event systems," in *Canadian Conference on Electrical and Computer Engineering*, vol. 1, Vancouver, BC, Canada, September 1993, pp. 241–244.
[4] "DESUMA software." Department of Electrical Engineering and Computer Science, University of Michigan, USA. Available at http://www.eecs.umich.edu/umdes/toolboxes.html.
[5] "IDES software." Department of Electrical and Computer Engineering, Queen's University, Canada. Available at http://www.ece.queensu.ca/directory/faculty/Rudie.html.
[6] "Supremica." Department of Signals and Systems, Chalmers University of Technology, Sweden. Available at http://www.supremica.org/.
[7] N. Cowan, "The magical number 4 in short-term memory: A reconsideration of mental storage capacity," *Behavioral and Brain Sciences*, vol. 24, no. 1, pp. 87–114, 2001.
[8] A. J. Dix, J. E. Finlay, G. D. Abowd, and R. Beale, *Human-Computer Interaction*, 2nd ed. Prentice Hall Europe, 1998.
[9] R. Spence, *Information Visualization*. Addison-Wesley, 2000.
[10] R. J. Leduc, "PLC implementation of a DES supervisor for a manufacturing testbed: An implementation perspective," Master's thesis, Department of Computer and Electrical Engineering, University of Toronto, 1996.
[11] M. M. Wood, "Application, implementation and integration of discrete-event systems control theory," Master's thesis, Department of Electrical and Computer Engineering, Queen's University, 2005.
[12] E. Rogers, R. C. Arkin, M. Baron, N. Ezquerra, and E. Garcia, "Visual protocol collection for the enhancement of the radiological diagnostic process," in *Proceedings of the First Conference on Visualization in Biomedical Computing*, Atlanta, Georgia, USA, May 1990, pp. 208–215.
[13] E. Rogers, R. C. Arkin, and M. Baron, "Visual interaction in diagnostic radiology," in *Proceedings of the Fourth Annual IEEE Symposium on Computer-Based Medical Systems*, Baltimore, Maryland, USA, May 1991, pp. 170–177.
[14] E. Rogers, "VIA-RAD: a blackboard-based system for diagnostic radiology," *Artificial Intelligence in Medicine*, vol. 7, pp. 343–360, 1995.
[15] ——, "A cognitive theory of visual interaction," in *Diagrammatic Reasoning*. The AAAI Press and MIT Press, 1995, pp. 481–500.
[16] J. R. Anderson, *Cognitive Psychology and Its Implications*, 6th ed. New York, New York, USA: Worth Publishers, 2005.
[17] K. A. Ericsson and H. A. Simon, *Protocol Analysis*, revised ed. Cambridge, Massachusetts, USA: The MIT Press, 1993.
[18] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information Sciences*, vol. 44, no. 2, pp. 173–198, 1988.
[19] W. M. Wonham and P. J. Ramadge, "Modular supervisory control of discrete-event systems," *Mathematics of Control, Signals, and Systems*, vol. 1, pp. 13–30, 1988.
[20] C. Y. Suen, "N-gram statistics for natural language understanding and text processing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 164–172, 1979.
[21] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 2nd ed. San Diego, CA, USA: Academic Press, 2003.
[22] W. M. Wonham, "Notes on supervisory control of discrete-event systems," Available at http://www.control.toronto.edu/DES/, July 2004.