BEHAVIOUR & INFORMATION TECHNOLOGY, 1996, VOL. 15, NO. 4, 213–225

Planning and knowledge about strategies: their relationship to work characteristics in software design

SABINE SONNENTAG

University of Amsterdam, The Netherlands

Abstract. This paper describes an empirical study of software design processes in which both cognitive (i.e. planning the work process, knowledge about strategies) and organizational (i.e. work characteristics) factors were examined. Thirty-five software designers with an average professional experience of 6.6 years worked on a software design task in a laboratory setting. Thinking-aloud protocols were analysed, and additional interview and questionnaire data were gathered. It was found that software designers do very little explicit planning but have a broad knowledge of useful strategies. Results of regression analyses indicated that the amount of explicit planning and knowledge of strategies is predicted by the amount of design work to accomplish, communication and cooperation requirements, and control at work.

1. Introduction

1.1. *Research on planning within information system design*

Within the cognitive-oriented research on information systems design, planning plays an important role. Planning can refer to various targets. First, planning can refer to a future object or situation. Within this conceptualization, planning is equated with designing (Black 1990, Strohschneider and von der Weth 1993). The whole process of developing ideas and concepts about how a system should work and look like is termed 'planning the system'. Here, all what happens within the software design process is planning.

Second, planning can refer to the layout of central features of the system (Brooks 1977, Pennington and Grabowski 1990). Within this view, a program or a design is regarded to be composed of a set of plans (Rist 1991). It is assumed that design tasks can be performed skilfully by relying on programming plans or other schema-like structures (Soloway and Ehrlich 1984, Rist 1989). Also the recent discussion on stepwise refinement versus opportunistic design (Guindon 1990, Visser 1990) shares—at least implicitly—this concept of planning.

Third, planning can refer to the action process of a person or a group of persons. According to Miller *et al.* (1960:16), planning controls 'the order in which a sequence of operations is to be performed', i.e. planning determines a person's course of actions. In contrast to the other conceptualizations, here the primary focus of planning is on what a person will *do* and not on how the *product* of the person's action will look like. This process-planning can be seen as a kind of meta-activity that guides the product-planning process.

There is a large body of studies on cognitive processes following the first and second conceptualization of planning (e.g. Brooks 1977, Davies 1990, Visser 1990). Only a few studies with student subjects were performed referring to the third meaning of planning (e.g. Sutcliffe and Maiden 1992). Thus, little is known about how professional software designers plan their own work process. Planning the work process is necessary for efficient cooperation and project management (Pietras and Coury 1994). Additionally, research showed that high performance persons do more planning in the third sense than do others (Dörner and Schölkopf 1991, Hacker 1986). Therefore, one purpose of this paper is to examine how professional software designers plan their work, i.e. how they determine the course of their actions.

Besides research on cognitive processes within systems design, there is some research on broader issues concerning work situations and organizational factors within software design (Brodbeck and Frese 1994, Curtis *et al.* 1988, Hornby and Clegg 1992, Kraut and Streeter 1995). Up to this point there is little cross fertilization between these organizational and cognitive lines of research (Clegg 1994,

cf. Waterson and Clegg 1994, for an exception). Reasons are: first, as Curtis (1986) pointed out, little is known whether and how cognitive processes identified in experimental studies can be generalized to large-scale design processes within organizational settings. Second, cognitive processes within design processes have been studied in a mainly descriptive way (Goel and Pirolli 1992, Guindon 1990) or in relationship to design methods and experience (e.g. Adelson *et al.* 1985, Chatel and Détienne 1994, Lee and Pennington 1994). The question remains still unanswered whether or not a software designer's cognitive processes are also influenced by the person's experience within his or her everyday work situation. Therefore, the second purpose of this paper is to examine the relationship between software designers' work situations and their cognitive, mainly planning processes in design.

1.2. *Planning as determining the course of action*

Planning—in the third sense—one's course of action is an important step within the action process. An action process consists of the development of goals, orientation and integration of information including prognosis of future events, generation of plans, decision among competing plans, execution of the task including monitoring, and processing of feedback (Dörner 1989, Frese and Zapf 1994, Hacker 1986). It is important to note that these steps do not necessarily follow this order (Hacker 1986). Within the planning phase it is decided what should be done in order to achieve a goal and when to take these actions.

Within the design of information systems, planning can occur both at the project and at the individual level. At the project level, planning aims at scheduling time, allocating tasks, and organizing cooperation. At the individual level, planning means that a person decides how to organize his or her own work process. For example, the person decides with which part of the task to start or what to do when necessary information is missing.

One could argue that the availability of software design methods makes individual planning unnecessary. However, there are at least three reasons why some sort of planning should also occur when software design methods are used. First, at a very global level, the person must decide how to apply the procedure proposed by the method to the task in question. Second, design methods do not describe every single step that has to be performed. Thus, additional planning is needed. Third, as was shown by the work of Guindon (1990) and Visser (1990), software designers do not follow necessarily the course of actions prescribed by design methods. This means that software designers find other ways of organizing their personal work process than that suggested by design methods. One possibility of organizing this work process is planning.

There are great variations among planning processes and plans generated in these processes concerning depth of detail, inclusion of back-up plans, *a priori* hierarchization, and time-frame (Frese and Zapf 1994). Another important distinction has to be made between explicit and implicit planning. Explicit planning occurs when a person consciously plans an action or a sequence of actions. Being confronted with a design task never performed before and reflecting on and deciding what to do first and what to postpone is an example of this type of planning. Also thinking of *pros* and *cons* of specific procedures that could be applied is explicit planning. Implicit planning takes place when a person determines the course of his or her actions by relying on a sort of routinized procedure, i.e. without explicitly reflecting on what to do. This is the case when the person has already existing strategies and heuristics that are applied to the problem in question. For example, a person who regularly starts working on a task by writing down all that he or she already knows about the problem does not have to plan this step explicitly. Nevertheless, the person's work process is guided by an implicit plan. Although there might be no obvious planning, following an implicit plan differs substantially from non-planning where no decision on future actions is made. When planning implicitly, heuristics and meta-cognitive knowledge are applied (cf. Brown 1988, Gleitman 1985, Semmer and Frese 1985).

It can be assumed that within professional design of information systems both explicit and implicit planning occurs. Because of the short innovation cycles within software design, many software designers are often confronted with problems, methods, and tools they never dealt with before. Here, explicit planning is necessary. However, although problems, methods, and tools differ, there are some general principles how to approach a task. Some of these principles are both proposed by design methods and acquired through practice. Therefore, it is assumed that implicit planning takes place as well.

My empirical study aimed at describing individual planning processes within software designers. It was assumed that both explicit and implicit planning occurs. Because knowledge of useful strategies and heuristics is needed in order to do successful implicit planning software designers' knowledge of such strategies and their application in design work was also examined.

1.3. *Influence of work situations on planning and knowledge about strategies*

Research has shown that the way people work on design

tasks is related to the length of their professional experience and the design methods used. For example, Batra and Davis (1992) reported that experienced subjects working on a database design task first concentrated on a holistic understanding of the problem and then developed the conceptual model (cf. also Davies, 1990; 1991; Jeffries, Turner, Polson, & Atwood, 1981; Rist, 1991).

In contrast to this large body of research concentrating on years of experience and design methods as predictors of design behavior little is known about the effect of people's work situation on the way they accomplish a design task. If it was found in empirical research that characteristics of a person's work situation have an influence on the use of effective strategies, implications for work redesign would arise.

Within the broader area of industrial and organizational psychology studies showed that a person's work situation is indeed related to the way this person performs a task. For example, von Papstein and Frese (1988) found that expertise acquired in a training course was only transferred to the actual work situation if job decision latitude was high. Earley *et al.* (1990) reported a positive relationship between task component complexity and quality of task strategy. Another study revealed that complexity of work and control at work were related to personal initiative at work (Frese *et al.* 1996). Persons with high complexity and high control showed higher initiative, i.e. they made more suggestions on how to deal with difficult situations and approached the situations more actively.

It is assumed that this relationship between a person's work situation and the way this person accomplishes his or her tasks holds also for software designers. There are at least two explanations for such a relationship. First, a work situation can enable or impede work behaviours. For example, if control at work is low, there is little opportunity for planning one's own work process. Second, a work situation can have a socialization effect on work behaviours and activity (Frese 1982). Task requirements and job characteristics require specific work behaviour (e.g. quick reactions, long-term planning, or cooperation) in order to accomplish the task. By carrying out the task frequently, people develop relevant skills, experience their approach to be successful, and automatize it with practice (Frese and Zapf 1994, Hacker 1986). Skills and behaviour developed in work situations generalize to a certain degree and are transferred to other work situations as well as to other contexts including leisure activities (Kohn and Schooler 1983, Meissner 1971). For example, the frequent experience of high control at work provides the opportunity to explore various work procedures and therefore results in the availability of a wide range of strategies and heuristics that can be applied in other situations as well. Frese *et al.* (1996) found empirical evidence for such a socialization effect.

In the study described below three central aspects of software designers' work situations were examined: amount of design work accomplished in the everyday work situation; communication and cooperation requirements; control at work. The extent to which a designer is confronted with design tasks in the work situation is assumed to be important for the way this person approaches a design task. People who do a great amount of design work have acquired experience of how to cope with such tasks. They have already ready-made strategies and heuristics on how to approach design tasks and how to come to solutions. Thus, it is assumed that compared to designers who seldom perform design tasks, software designers who perform a high amount of design work do little explicit planning. Additionally, due to their intensive experience with design tasks, they should have more knowledge of useful strategies.

In general, work in information systems design is characterized by high communication and cooperation requirements (Brodbeck *et al.* 1993, Kraut and Streeter 1990, Olson *et al.* 1992). However, there is also a certain variability in the degree to which individuals are engaged in such cooperative activities (Curtis *et al.* 1988, Sonnentag 1995). When cooperative requirements are high, there is a great need for planning. This concerns tasks to be accomplished cooperatively but also holds for tasks that are performed individually, because information, knowledge, and work products provided by others have to be incorporated in their own design process at the right time. In order to make the cooperation efficient explicit planning is necessary. Strategies and heuristics that are only planned implicitly without being communicated to others are not sufficient for effective task performance.

Therefore, it is assumed that software designers who experience high communication and cooperation requirements within their work situations show more explicit planning than do designers who are not confronted with such requirements. However, when a person plans explicitly that he or she also has knowledge of useful strategies and procedures that could be used implicitly can not be excluded. Accordingly, it is hypothesized that there is no relationship between communication and cooperation requirements and knowledge about useful strategies.

Control at work is a crucial variable describing the work situation of individuals and groups showing effects both on well-being and performance (Frese 1989, Karasek and Theorell 1990, Wall *et al.* 1992). A positive relationship between control at work and planning within the design process can be assumed. This relationship is due to two processes. First, it is necessary to have a certain degree of control at work in order to plan, i.e. in order to be able to determine one's course of action. Otherwise planning is not possible. Second, high control, i.e. a lot of decision

possibilities requires planning because one *must* decide how to proceed. It is assumed that this positive relationship holds for both explicit planning and knowledge about useful strategies.

The hypotheses concerning the relationship between software designers' work situations, planning processes, and knowledge about strategies can be summarized as follows:

(1) The amount of design work to be accomplished is negatively related to explicit planning and positively related to the knowledge of useful strategies.
(2) Communication and cooperation requirements are positively related to explicit planning but show no relationship with knowledge about useful strategies.
(3) Control at work is positively related to explicit planning and knowledge about useful strategies.

2. Methods

2.1. *Sample*

Thirty-five persons participated in the study. All subjects were professional software designers working in software development projects. The average professional experience within software development was 6.6 years (SD = 2.6). The mean age was 33 years (SD = 4.7). Thirty-one per cent of the participants were females.

2.2. *Procedure*

The study was composed of three parts. First, participants had to work on a software design task, the Lift Control Problem (cf. Guindon 1990). The goal of this task is to design a software system that controls the movement of N lifts between M floors by taking into account various realistic constraints. Participants were given two hours to work on this task. They were not restricted to any specific design method, programming language, or notation and were free to write down everything they wanted, including sketches, notes, and questions to themselves. Products produced by the designers ranged from rough verbal specifications to detailed pseudo-code notations. While working on the task participants were asked to think aloud, i.e. to verbalize all their thoughts. Verbalizations were tape-recorded, later verbally transcribed, segmented, and categorized. On average, a thinking-aloud protocol consisted of 493 segments (SD = 188). After designers had finished working on the Lift Control task they were

Table 1. Category system for analysing thinking-aloud protocols and results from protocol analysis.

Category	Description	Example	r[a]	M[b]	SD[b]
Planning ahead	Reflecting on and making decisions about the course of action	This is not yet clear to me. Therefore, I note this as an open question. If there is some time left later, I'll do something with it.	0.69	1.7	1.4
Local planning	Thinking about the next step	Now I have to read it again in order to understand it	0.82	4.7	2.0
Problem comprehension: requirements	Reading and reflecting on requirements	What does this mean: 'all floors should be given equal priority'?	0.87	15.6	9.2
Problem comprehension: scenarios	Reflecting on typical problems and scenarios within the lift domain	Yes, I expect this when I am in a lift: that I go in one direction and that I do not go up and down while being in the lift.	0.96	5.5	4.7
Solution development	Designing the outline of the software system	I just think of a rule for checking the status. I write: if the first state of n is smaller than the second state...	0.86	52.0	10.5
Solution evaluation	Evaluating the designed solution	Let's go through it again. We have a button for every lift...	0.68	10.6	7.8
Comments	Commenting on the own work process	That is the drawing I did in the very beginning.	0.72	6.2	4.3
Task-irrelevant statements	Statements do have nothing to do with the task requirements	This is embarrassing, really embarrassing. Do others behave like I do? Do others succeed within two hours...?	0.81	2.9	4.2
Others	E.g. clarifying the experimental setting		0.86	0.8	0.6

Note:
[a] Agreement among two raters at the protocol level (n = 20 protocols). All correlations are significant at the 0.001 level.
[b] M and SD referring to percentage of segments in thinking-aloud protocols.

asked to rate its complexity and difficulty. Forty-three per cent of the participants reported that the Lift Control task was at least as complex as a task that they had to accomplish in their everyday work situation. With respect to difficulty 66 per cent of the designers rated the Lift Control task at least as difficult as a task in the everyday work situation.

As a second part of the study software designers participated in an interview that was performed after they had finished working on the design task. The interview included questions about the task, implicitly used strategies, and knowledge about useful strategies. The third part of the study was a questionnaire including questions about the designers' every day work situation and professional experience. The software designers were paid for participating in the study.

2.3. *Measures*

2.3.1. *Explicit planning*: The amount of explicit planning was assessed by the analysis of the thinking-aloud protocols. For performing this analysis a category system was developed comprising nine categories. This category system is shown in table 1. It differentiated between two types of explicit planning: planning ahead and local planning. Planning ahead means that a person reflects on how he or she wants to proceed, for example, what to do first and what to postpone. Local planning is characterized by a person's mere statement what he or she intends to do next without extensively reflecting on it. Often, local planning sounds like a short introductory statement into the following solution or evaluation statement. It is important to note that planning ahead is not restricted to early phases of task completion but can also occur during later phases. For example, one can plan ahead after having finished the first parts of the task by reflecting about what steps to perform next.

Twenty out of the 35 thinking-aloud protocols were categorized by two raters. For computing the reliability at the protocol level, the percentage for every category within the protocol rated by the first rater was correlated with the corresponding percentage within the protocol rated by the second rater. Correlations between the two raters are given in the fourth column of table 1. The correlations of $r = 0.69$ for planning ahead and $r = 0.82$ for local planning are indicators for a good reliability at the protocol level. In the case of disagreement, the ratings of the more experienced rater were used for further analysis. This rater also categorized the remaining 15 protocols.

2.3.2. *Implicitly used strategies*: Because implicit strategies are not necessarily verbalized while thinking aloud (cf. Ericsson and Simon 1993), an interview method was used in order to assess such meta-cognitive strategies. After participants had completed the design task they were asked whether they had used 'higher-order principles' while working on the task. An example was given: 'A higher-order principle could be to start with the most simple part of the task—or to start with the most difficult part'. Software designers reported the principles they had applied. These reports were verbally recorded and later categorized by using a fine-grained category system comprising of 22 categories. However, some of these categories remained empty. In order to compute the reliability of the categorization, strategies reported by 20 designers were categorized by two raters. An agreement of 77.3% was obtained (Cohen's Kappa = 0.75) indicating good reliability (Landis and Koch 1977).

2.3.3. *Knowledge about useful strategies*: Knowledge about useful strategies was also assessed during the interview by adapting a procedure suggested by Wolff (1989; cf. Hacker 1992). Software designers were told to imagine an inexperienced colleague having to work on that design task. Then designers were asked to say what recommendations they would make to this colleague. The recommendations were verbally recorded and later categorized by using the category system described in the preceding paragraph. Again, data provided by 20 designers were categorized by two raters. Here, an agreement of 85.3% resulted (Cohen's Kappa = 0.84) that can be regarded as a very good reliability.

2.3.4. *Work situation*: The software designers' work situation was assessed by a questionnaire. The amount of design work and communication and cooperation requirements were ascertained by single questionnaire items. Designers were asked to indicate the percentage of working time they spent on a design task and the percentage of working time they spent on communication, cooperation, or coordination. Control at work was assessed by a 6-item questionnaire scale developed by Semmer (1984) and adapted to clerical work by Zapf (1991). The scale included items such as 'Can you decide *how* you do your work?'. Cronbach's alpha was 0.81.

3. Results and discussion

3.1. *Analysis of thinking-aloud protocols: explicit planning*

The two right hand columns of table 1 show the results of

Table 2. Percentage of software designers using strategies implicitly and recommending strategies to an inexperienced colleague.

Strategy	Strategy implicitly used	Strategy recommended
Simple parts first	40.0	14.3
Divide and conquer	40.0	28.6
Intensive problem comprehension	28.6	37.1
Top down	20.0	22.9
Evaluating the solution	20.0	20.0
Visualization	20.0	17.1
Exploration	17.1	8.6
Systematic search for information	14.3	8.6
Important parts first	11.4	8.6
Applying software design method	8.6	8.6
Explicit planning	5.7	5.7
Bottom up	5.7	2.9
Difficult parts first	5.7	0.0
Completeness and disciplinarity as goal	5.7	5.7
Modularity as goal	2.9	0.0
Documentation	2.9	2.9
Cooperation with colleagues	NA	31.4
User involvement	NA	2.9
Taking enough time	0.0	8.6
Using already existing solutions	0.0	5.7
Clarifying organizational questions	0.0	2.9
Others	8.6	20.0

Note: NA = not applicable.

protocol analysis. Subjects spent most of the time on developing the solution (52.0%). For problem analysis, i.e. considering requirements and building scenarios, 21.1% of the time was used. Evaluation of the solution took 10.6% of the time. In general, there was a rather great variability within each category. For example, time spent on solution development ranged between 29.7 and 74.8% of the working time. Concerning explicit planning, analysis of thinking aloud protocols showed that software designers spent 6.4% of their working time for explicit planning. Most of this time was used for local planning (4.7% of total working time), while only 1.7% of total working time was spent for planning ahead. This indicates that especially planning ahead very seldom occurred in the work process, even less often than task-irrelevant statements (2.9%).

3.2. *Interview data: implicitly used strategies*

For every category it was computed how many software designers reported having used such a strategy. Results are shown in the left hand column of table 2. The most often reported strategies were 'simple parts first' (40.0%), 'divide and conquer' (40.0%), 'intensive problem comprehension' (28.6%), 'top down' (20.0%), 'evaluating the solution' (20.0%), and 'visualization' (20.0%). Only 5.7% of the subjects reported having used explicit planning as a strategy.

The high percentage for the category 'simple parts first' might be an overestimation because this category was given as a sample item in the instruction. However, 'difficult parts first' that was the opposite sample item was mentioned by only 5.7% of the participants. This suggests that the strategy of starting with the simple parts first was indeed given priority over the opposite strategy of concentrating on difficult parts first. The low percentage with which explicit planning was reported confirms the findings of protocol analysis showing little explicit planning. Therefore, these results suggest that software designers have strategies other than explicit planning that guide their work, especially starting with simple parts, dividing the problem into manageable subproblems, and spending a lot of time on problem comprehension. Explicit planning plays only a minor role in determining the work process.

3.3. *Interview data: knowledge about useful strategies*

Software designers' knowledge about useful strategies—measured as recommendations for an inexperienced colleague—is shown in the right hand column of table 2. The most often recommended strategies were 'intensive problem comprehension' (37.1%), 'cooperation with colleagues' (31.4%), 'divide and conquer' (28.6%), 'top down' (22.9%), 'evaluating the solution' (20.0%), and visualization (17.1%). Again, explicit planning was recommended

Table 3. Means, standard deviations, and intercorrelations for variables used in hierarchical and logistic regression analyses.

		M	*SD*	1	2	3	4	5	6	7	8	9	10	11
1	Years of professional experience	6.6	2.6											
2	Amount of design work[a]	18.5	11.7	0.14										
3	Communication and cooperation requirements[a]	39.3	23.6	0.02	0.30									
4	Control at work[b]	3.9	0.6	– 0.20	– 0.07	0.36								
5	Planning ahead[c]	1.7	1.4	– 0.17	0.02	0.25	– 0.27							
6	Local planning[c]	4.7	2.0	0.04	0.01	– 0.08	– 0.30	0.36						
7	Simple parts first (implicitly used)	0.4	0.5	– 0.07	– 0.14	– 0.62	– 0.34	– 0.09	0.08					
8	Divide and conquer (implicitly used)	0.4	0.5	– 0.22	– 0.10	0.10	0.06	0.11	0.40	0.05[d]				
9	Intensive problem comprehension (implicitly used)	0.3	0.5	– 0.09	– 0.28	0.13	– 0.05	0.19	0.07	0.25[d]	0.00[d]			
10	Intensive problem comprehension (recommended)	0.4	0.5	– 0.03	0.10	0.11	0.08	– 0.18	0.22	0.14[d]	0.10[d]	0.04[d]		
11	Cooperation with colleagues (recommended)	0.3	0.5	0.13	0.06	0.13	0.43	0.00	– 0.00	0.29[d]	0.05[d]	0.02[d]	0.24[d]	
12	Divide and conquer (recommended)	0.3	0.5	– 0.09	0.42	0.14	0.07	0.02	– 0.09	0.00[d]	0.25[d]	0.25[d]	0.17[d]	0.02[d]

Note:
$N = 35$.
[a]Per cent of working time.
[b]Range: 1–5.
[c]Per cent of segments in thinking-aloud protocols.
[d]Contingency coefficient.

Table 4. Results from hierarchical regression analyses predicting planning ahead and local planning.

Type of explicit planning and predictors	*B*	*SE B*	β	R^2	R^2
Planning ahead					
Step 1					
Years of profession experience	– 0.0008	0.0010	– 0.15	0.02	0.02
Step 2					
Years of professional experience	– 0.0013	0.0009	– 0.24		
Amount of design work	– 0.0002	0.0002	– 0.12		
Communication and cooperation requirements	0.0003	0.0001	0.48*		
Control at work	– 0.0117	0.0043	– 0.48*	0.28*	0.26*
Local planning					
Step 1					
Years of professional experience	0.0000	0.0013	0.00	0.00	0.00
Step 2					
Years of professional experience	– 0.0006	0.0013	– 0.08		
Amount of design work	– 0.0000	0.0003	– 0.01		
Communication and cooperation requirements	0.0000	0.0001	0.02		
Control at work	– 0.0120	0.0064	– 0.36§	0.12	0.12

Note:
*$p < 0.05$.
§$p = 0.0704$.

only by 5.7% of the subjects. This indicates that software designers have a rather broad knowledge of useful strategies including knowledge about the positive effects of cooperation. Explicit planning is not regarded as a recommendable strategy.

3.4. *Relationship between characteristics of the work situation and explicit planning*

Table 3 shows the means, standard deviations, and intercorrelations for all variables used in hierarchical multiple and logistic regression analyses.

Results from hierarchical multiple regression analysis predicting explicit planning in the thinking-aloud protocols from the designer's work situation are displayed in table 4. Years of professional experience were entered in the first step as a control variable. In the second step, the amount of design work, communication requirements, and control at work were included as measures of the software designers' work situation. Separate analyses were performed for planning ahead and local planning.

In the first equations years of professional experience was not a significant predictor of planning ahead and local planning. Work situation variables entered in the second step accounted for 28% of the variance of planning ahead ($p < 0.05$) with communication and cooperation requirements and control at work showing significant beta weights. Software designers with high communication requirements in their work situation and low perceived control showed a higher amount of planning ahead than did designers with low requirements and high control at work. This means that with respect to planning ahead, hypothesis 2 was supported by the data, while hypothesis 1 was not supported and hypothesis 3 was contradicted by the data. This finding must be seen in the light that, within the experimental situation, all participants experienced the same, relatively high control: no restrictions concerning design methods or notations were imposed on the designers. This relatively high control within the experimental situation is also reflected by the fact the 65.7% of the participants asked about the required detail of the design and if specific steps were expected to be performed. All designers were free to choose the procedure and sequential order of steps they wanted. This means that all had the possibility to plan. Therefore, those who were not familiar with situations in which no prescriptions concerning the work process is made had to plan and structure their work in order to cope with this high control situation. In contrast, designers

Table 5. Results from logistic regression analyses predicting knowledge of useful strategies.

Strategy mentioned and predictors	*B*	*SE B*	Wald	df	Chi^2
Intensive problem comprehension					
Step 1					
Years of professional experience	– 0.0447	0.1373	0.11	1	0.11
Step 2					
Years of professional experience	– 0.0475	0.1434	0.12		
Amount of design work	0.0166	0.0332	0.25		
Communication and cooperation requirements	0.0049	0.0173	0.08		
Control at work	0.1203	0.6884	0.03	3	0.57
Cooperation with colleagues					
Step 1					
Years of professional experience	0.0947	0.1393	0.46	1	0.46
Step 2					
Years of professional experience	0.2845	0.1835	2.40		
Amount of design work	0.0376	0.0394	0.91		
Communication and cooperation requirements	– 0.0168	0.0218	0.59		
Control at work	2.9422	1.2102	5.91*	3	9.72*
Divide and conquer					
Step 1					
Years of professional experience	– 0.0980	0.1531	0.41	1	0.43
Step 2					
Years of professional experience	– 0.1935	0.1875	1.07		
Amount of design work	0.1171	0.0555	4.45*		
Communication and cooperation requirements	– 0.0051	0.0227	0.05		
Control at work	0.3779	0.8985	0.17	3	7.20§

Note:
*$p < 0.05$.
§$p = 0.0659$.

experiencing control every day had no need to plan because they knew how to deal with a high control situation.

Local planning was not predicted by the software designers' work situations. Thus, the respective hypotheses were not supported. This indicates that in contrast to planning ahead, local planning does not help in dealing with high control.

3.5. *Relationship between characteristics of the work situation and knowledge about useful strategies*

For strategies that were recommended by at least 25% of the subjects (i.e. 'cooperation with colleagues', 'intensive problem comprehension', and 'divide and conquer') it was analysed whether software designers' work situations were related to the knowledge about these strategies. Logistic regression analyses were performed again following a hierarchical procedure: years of professional experience were included in the first step, and the amount of design work, communication and cooperation requirements, and control at work were entered in the second step. Results are shown in table 5.

None of the three recommended strategies was predicted by years of professional experience entered first into the equations. Recommending 'cooperation with colleagues' was predicted by work situation variables entered in the second step ($Chi^2 = 9.72$; df $= 3$; $p < 0.05$) with control at work being a significant predictor (Wald statistic $= 5.91$; $p < 0.05$; partial correlation $= 0.30$). Communication and cooperation requirements were no significant predictor (Wald statistic $= 0.59$; n.s.; partial correlation $= 0.00$). Thus, analysis revealed that with respect to 'cooperation with colleagues', Hypotheses 2 and 3 were supported by the data. This indicates that perceived control at work was related to knowledge about this strategy while the amount of communication and cooperation requirements was not.

Mentioning 'divide and conquer' as a useful strategy was predicted nearly significantly by variables of the work situation ($Chi^2 = 7.20$; df $= 3$; $p = 0.0659$). Software designers performing a high amount of design work more often recommended the 'divide and conquer' strategy than did designers with a low amount of design work to accomplish (Wald statistic $= 4.45$; $p < 0.05$; partial correlation $= 0.25$). Communication and cooperation requirements did not contribute significantly to the prediction of knowledge about 'divide and conquer' (Wald statistic $= 0.05$; n.s.; partial correlation $= 0.00$). This means that with respect to the strategy 'divide and conquer', hypotheses 1 and 2 were supported. The amount of design work to accomplish this was positively related to knowledge of this strategy while communication and cooperation requirements were not.

'Intensive problem comprehension' was not predicted by the software designers' work situation. Thus, with respect to this strategy, there was no empirical support for hypotheses 1 and 3.

3.6. *Relationship between characteristics of the work situation and implicitly used strategies*

No hypotheses had been formulated with respect to the relationship between work situation and implicitly used strategies. Nevertheless, for exploratory reasons, logistic regression analyses were performed for those implicitly used strategies that were mentioned by at least 25% of the subjects (i.e. 'simple parts first', 'divide and conquer', and 'intensive problem comprehension'). The same hierarchical procedure as described above was followed. No significant effects of professional experience and work situation on the reporting of 'divide and conquer' ($Chi^2 = 0.84$; df $= 3$; n.s.) and 'intensive problem comprehension' ($Chi^2 = 5.64$; df $= 3$; n.s.) were found. However, the implicitly used strategy 'simple parts first' was significantly predicted by work situation variables ($Chi^2 = 17.561$; df $= 3$; $p < 0.01$) with communication and cooperation requirements being a significant predictor yielding a negative sign (Wald statistic $= 5.31$; $p < 0.05$; partial correlation $= -0.27$). Thus, software designers experiencing low communication and cooperation requirements reported more often having begun with simple parts.

This result indicates that designers who do not have to meet cooperation requirements in their everyday work situation tend to start with the most convenient parts without paying attention to other issues. However, one should be cautious with further interpretations because 'simple parts first' was the sample item and only 14.3% of the subjects regarded 'simple parts first' as a recommendable strategy. In general, analyses revealed stronger relationships between work situation and knowledge about strategies than between work situation and implicitly used strategies. This suggests that the experience of specific work situations is predominantly associated with a person's *knowledge* about useful strategies but not with (the report of) their implicit application.

4. Overall discussion

The study showed that professional software designers do very little explicit planning when working on a design task. Instead, they have a broad knowledge of other useful

strategies that can guide the work process. Additionally, it was found that the amount of planning ahead and knowledge about useful strategies are predicted by characteristics of the work situation.

The amount of explicit planning is not only small in absolute figures but also small compared to the findings reported by Sutcliffe and Maiden (1992). That study applied a similar conceptualization of planning than was used in the present study. Participants in the study by Sutcliffe and Maiden (1992) spent substantially more time in planning. The differing findings of this and the present study may be due to the different samples studied.The sample studied by Sutcliffe and Maiden (1992) consisted of students with only a few months of experience in systems analysis while professional software designers with many years of professional experience participated in the present study. Various theoretical approaches agree that cognitive processes are automated through practice (e.g. Semmer and Frese 1985, Shiffrin and Dumais 1981). Thus, the professionals' longer experience compared to that of the students can account for the differences in explicit planning. It can be concluded that designers in the present study used routinized and already existing strategies. This is confirmed by the designers' self-reported 'high-order principles'.

However, there are some alternative explanations for the low amount of explicit planning in software designers' work process. One might argue that designers did plan explicitly but did not verbalize it. However, two reasons speak against this assumption. First, if the assumption was true, then in retrospective reports participants probably would have said that they planned. But only 5.7% of the participants mentioned explicit planning. Thus, it is not very plausible that the designers planned to a great degree but did not verbalize it. A second reason refers to individual differences in difficulties with verbalizations. There was of course a certain variability in the participants' ability to verbalize. This variability is reflected in the length of thinking-aloud protocols. If it was true that designers planned without verbalizing it, one would expect that those with poor verbalizations, i.e. short protocols, verbalized relatively less planning because they planned silently. Additionally, one would assume that those with very fluent verbalizations, i.e. long protocols, verbalized relatively more planning. However, there were no substantial correlations between length of thinking-aloud protocol and explicit planning ($r = 0.04$ for planning ahead and $r = 0.04$ for local planning). Thus, although the argument that designers explicitly planned without verbalizing can not be ruled out completely, these data speak against this assumption.

Another potential explanation for the low amount of explicit planning is that the task did not require planning. In general, planning is not necessary, if the task is perceived to be very easy. However, participants' assessment of complexity and difficulty shows that perceived difficulty and complexity were not low. This indicates that planning was not completely obsolete. There is another argument why planning would have been helpful or even necessary. Many participants did not manage to finish their work within two hours. Therefore, some sort of time planning would have been necessary. Taking these arguments together it can be concluded that—although the task did not force planning—task characteristics did not make explicit planning unnecessary. However, as was found by retrospective interviews, software designers relied more on ready-made strategies.

The study demonstrated that the experiences people have within the organizational context are related to design behaviour and knowledge while length of professional experience is not. This questions the notion that within professional systems development length of experience is a substantial factor for predicting work behaviour and performance (cf. Sonnentag (1995) for a related argument). One can assume that professional experience can account for performance differences between students and professionals, but after a certain degree of professional experience has been acquired other aspects become more important. As the present study shows characteristics of the everyday work situation might be such variables.

The use of questionnaire measures for assessing work situations has been often criticized (e.g. Kasl 1986). One major critique in using such measures is that these measures share common method variance (Campbell and Fiske 1959) with dependent variables leading to an overestimation of relationships. However, this critique does not hold for the present study, since only work situation variables were assessed by a questionnaire; other variables were ascertained by analysis of thinking-aloud protocols and open-ended interview questions.

Although not tested causally, results of the present study are in accordance with those of other studies (e.g. Frese *et al.* 1996) reporting socialization effects of working conditions on individual behaviour. With respect to theory these findings have implications for explaining work behaviour and performance. Approaches that only consider individual variables such as length of experience fall short. Present and previous conditions at the work place with their learning and socialization potential have to be taken into account as well.

One might argue that the implications of the study are limited due to the Lift Control task studied. Of course, within any one study not all aspects of professional software design can be taken into account. For example, for developing a solution for the Lift Control problem neither a long term focus nor communication or cooperation were required. However, the task incorporated aspects of typical

software design tasks such as a certain degree of complexity and poor structure (cf. Guindon 1990). Additionally, this procedure of taking the same experimental task for every participant has one important advantage compared to the study of tasks in real work environments. With the procedure chosen in the present study, one can be sure that individual differences in the design process can not be due to differences in the task setting in which the task has to be accomplished. These differences in the individual work process must be due to other variables such as the work situation outside the setting of the study or professional experience.

'Divide and conquer' is one important top down strategy within software design (e.g. Budgen 1994). Working cooperatively is an important factor for success in software development projects (Brodbeck 1993, Curtis *et al.* 1988, Walz *et al.* 1993). However, knowledge about these strategies is not shared by every software professional. Only designers who do a high amount of design work and who experience high control at work regard these strategies as important recommendations for an inexperienced colleague. Therefore, with respect to this study's practical implications, it can be concluded that high control at work should be provided. In order to become familiar with the 'divide and conquer' strategy, software designers should accomplish a high amount of design work.

The study revealed that aspects of the work situation are related to cognitive processes within design. Thus, further research is needed that takes into account organizational and cognitive issues in *one* study. With respect to cognitive issues such research can show the relative importance of organizational factors compared to purely cognitive variables in predicting behaviour and performance in cognitive tasks. With respect to organizational issues it can be expected that such research will not only show which organizational factors are related to performance but will also identify the mediating cognitive processes.

Acknowledgements

This research was supported by a grant from the German Research Community (DFG; So 295/1-1, 1-2) that is gratefully acknowledged. Special thanks are due to Christa Speier, Michael Frese, Chris Clegg, and two anonymous reviewers for their helpful comments on earlier versions of this paper. Correspondence concerning this article should be addressed to Sabine Sonnentag, who is now at the Department of Psychology, University of Amsterdam, Roetersstraat 15, NL-1018 WB Amsterdam, The Netherlands. Electronic mail may be sent to: ao_ sonnentag@macmail.psy.uva.n1.

References

Adelson, B., Littman, D., Ehrlich, K., Black, J., and Soloway, E. 1985, Novice-expert differences in software design, in B. Shackel (ed.), *Human–Computer Interaction – INTERACT '84* (Elsevier, Amsterdam), 473–478.

Batra, D. and Davis, J. G. 1992, Conceptual data modelling in database design: similarities and differences between expert and novice designers, *International Journal of Man-Machine Studies*, **37**, 83–101.

Black, A. 1990, Visible planning on paper and on screen: the impact of working medium on decision-making by novice graphic designers, *Behaviour and Information Technology*, **9**, 283–296.

Brodbeck, F. C. 1993, *Kommunikation und Leistung in Projektarbeitsgruppen. Eine empirische Untersuchung an Software-Entwicklungsprojekten*, unpublished Dissertation, University of Giessen.

Brodbeck, F. C. and Frese, M. 1994, *Produktivität und Qualität in Software-Projekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung* (Oldenbourg, Munich).

Brodbeck, F. C., Sonnentag, S., Heinbokel, T., Stolte, W., and Frese, M. 1993, Tätigkeitsschwerpunkte und Qualifikationsanforderungen in der Software-Entwicklung. Eine empirische Untersuchung, *Softwaretechnik-Trends*, **13**, 31–40.

Brooks, R. 1977, Toward a theory of cognitive processes in computer programming, *International Journal of Man-Machine Studies*, **9**, 737–751.

Brown, A. L. 1988, Metacognition, executive control, self-regulation, and other, even more mysterious mechanisms, in F. E. Winert and R. H. Kluwe (eds), *Metacognition, motivation, and understanding* (Erlbaum, Hillsdale, NJ).

Budgen, D. 1994, *Software Design* (Addison-Wesley, Wokingham).

Campbell, D. T. and Fiske, D. W. 1959, Convergent and discriminant validation by the multitrait-multimethod matrix, *Psychological Bulletin*, **56**, 81–105.

Chatel, S. and Détienne, F. 1994, Expertise in object-oriented programming, in R. Oppermann, S. Bagnara and D. Benyon (eds), *ECCE7, Seventh European conference on cognitive ergonomics. Human-computer interaction: From individuals to groups in work, leisure, and everyday life* (Gesellschaft für Mathematik und Datenverarbeitung, Bonn), 143–159.

Clegg, C. 1994, Psychology and information technology: The study of cognition in organizations, *British Journal of Psychology*, **85**, 449–477.

Curtis, B. 1986, By the way, did anyone study any real programmers?, in E. Soloway and S. Iyengar (eds), *Empirical Studies of Programmers* (Ablex, Norwood), 256–262.

Curtis, B., Krasner, H. and Iscoe, N. 1988, A field study of the software design process for large systems, *Communications of the ACM*, **31**, 1268–1287.

Davies, S. P. 1990, Plans, goals and selection rules in the comprehension of computer programs, *Behaviour and Information Technology*, **9**, 201–214.

Davies, S. P. 1991, Characterizing the program design activity: neither strictly top-down nor globally opportunistic, *Behaviour and Information Technology*, **10**, 173–190.

Dörner, D. 1989, *Die Logik des Mißlingens* (Rowohlt, Hamburg).

Dörner, D. and Schölkopf, J. 1991, Controlling complex systems; or, expertise as ''grandmother's know-how'', in K. A. Ericsson and J. Smith (eds), *Toward a General Theory*

of Expertise: Prospects and Limits (Cambridge University Press, Cambridge), 218–239.

Earley, P. C., Lee, C. and Hanson, L. A. 1990, Joint moderating effects of job experience and task component complexity: Relations among goal setting, task strategies, and performance, *Journal of Organizational Behavior*, **11**, 3–15.

Ericsson, K. A. and Simon, H. A. 1993, *Protocol Analysis. Verbal Reports as Data*, Revised edition (MIT Press, Cambridge).

Frese, M. 1982, Occupational socialization and psychological development: an underemphasized research perspective in industrial psychology, *Journal of Occupational Psychology*, **55**, 209–224.

Frese, M. 1989, Theoretical models of control and health, in S. L. Sauter, J. J. Hurrell Jr, and C. L. Cooper (eds), *Job Control and Worker Health* (Wiley, Chichester), 107–127.

Frese, M., Kring, W., Soose, A. and Zempel, J. 1996, Personal initiative at work: differences between East and West Germany, *Academy of Management Journal*, **39**, 37–63.

Frese, M. and Zapf, D. 1994, Action as the core of work psychology: a German approach, in H. C. Triandis, M. D. Dunnette and J. M. Hough (eds), *Handbook of Industrial and Organizational Psychology*, Vol. 4 (Consulting Psychologists Press, Palo Alto, CA), 271–340.

Gleitman, H. 1985, Some trends in the study of cognition, in S. Koch and D. E. Leary (eds), *A Century of Psychology as a Science: Retrospections and Assessments* (McGraw-Hill, New York), 420–436.

Goel, V. and Pirolli, P. 1992, The structure of design problem spaces, *Cognitive Science*, **16**, 395–429.

Guindon, R. 1990, Designing the design process: exploiting opportunistic thoughts, *Human–Computer Interaction*, **5**, 305–344.

Hacker, W. 1986, *Arbeitspsychologie* Huber, Bern.

Hacker, W. 1992, *Expertenkönnen. Erkennen und Vermitteln* (Verlag für Angewandte Psychologie, Göttingen).

Hornby, P. and Clegg, C. 1992, User participation in context: a case study in a UK bank, *Behaviour and Information Technology*, **11**, 293–307.

Jeffries, R., Turner, A. A., Polson, P. G. and Atwood, M. E. 1981, The processes involved in designing software, in J. R. Anderson (ed.), *Cognitive Skills and Their Acquisition* (Erlbaum, Hillsdale, NJ), 255–283.

Karasek, R. and Theorell, T. 1990, *Healthy Work, Stress, Productivity, and the Reconstruction of Working Life* (Basic, New York).

Kasl, S. V. 1986, Stress and disease in the workplace: A methodological commentary on the accumulated evidence, in M. F. Cataldo and T. J. Coates (eds), *Health and Industry. A Behavioral Medicine Perspective* (Wiley, New York), 52–85.

Kohn, M. L. and Schooler, C. 1983, *Work and Personality. An Inquiry into the Impact of Social Stratification* (Ablex, Norwood).

Kraut, R. E. and Streeter, L. A. 1990, Satisfying the need to know: interpersonal information access, in D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds), *Human–Computer Interaction INTERACT '90* (Elsevier, Amsterdam), 909–915.

Kraut, R. E. and Streeter, L. A. 1995, Coordination in software development, *Communications of the ACM*, **38**, 69–81.

Landis, J. R. and Koch, G. G. 1977, The measurement of observer agreement for catagorial data, *Biometrics*, **33**, 159–174.

Lee, A. and Pennington, N. 1994, The effects of paradigm on cognitive activities in design, *International Journal of Human–Computer Studies*, **40**, 577–601.

Meissner, M. 1971, The long arm of the job: a study of work and leisure, *Industrial Relation*, **10**, 239–260.

Miller, G. A., Galanter, E. and Pribram, K. 1960, *Plans and the Structure of Behavior* (Holt, New York).

Olson, G. M., Olson, J. S., Carter, M. R. and Storrosten, M. 1992, Small group design meetings: an analysis of collaboration, *Human–Computer Interaction*, **7**, 347–374.

Pennington, N. and Grabowski, B. 1990, The tasks of programming, in J. M. Hoc, T. R. G. Green, R. Samurcay and D. J. Gilmore (eds), *Psychology of Programming* (Academic Press), London.

Pietras, C. M. and Coury, B. G. 1994, The development of cognitive models of planning for use in the design of project management systems, *International Journal of Human–Computer Studies*, **40**, 5–30.

Rist, R. S. 1989, Schema creation in programming, *Cognitive Science*, **13**, 389–414.

Rist, R. S. 1991, Knowledge creation and retrieval in program design: A comparison of novice and intermediate student programmers, *Human–Computer Interaction*, **6**, 1–46.

Semmer, N. 1984, *Stressbezogene Tätigkeitsanalyse: Psychologische Untersuchungen zur Analyse von Stress Am Arbeitsplatz* (Beltz, Weinheim).

Semmer, N. and Frese, M. 1985, Action theory in clinical psychology, in M. Frese and H. Sabini (eds), *Goal Directed Behavior: The Concept of Action in Psychology* (Erlbaum, Hillsdale, NJ), 296–310.

Shiffrin, R. M. and Dumais, S. T. 1981, The development of automatism, in J. R. Anderson (ed.), *Cognitive Skills and their Acquisition* (Erlbaum, Hillsdale, NJ).

Soloway, E. and Ehrlich, K. 1984, Empirical studies of programming knowledge, *IEEE Transactions on Software Engineering*, **10**, 595–609.

Sonnentag, S. 1995, Excellent software professionals: experience, work activities, and perceptions by peers, *Behaviour and Information Technology*, **14**, 289–299.

Strohschneider, S. and von der Weth, R. (eds) 1993, *Ja, mach nur einen Plan. Pannen und Fehlschläge–Ursachen, Beispiele, Lösungen* (Huber, Bern).

Sutcliffe, A. G. and Maiden, N. A. M. 1992, Analysing the novice analyst: cognitive models in software engineering, *International Journal of Man–Machine Studies*, **36**, 719–740.

Visser, W. 1990, More or less following a plan during design: opportunistic deviations in specification, *International Journal of Man–Machine Studies*, **33**, 247–278.

von Papstein, P. and Frese, M. 1988, Transferring skills from training to the actual work situation: the role of task application knowledge, action styles, and job decision latitude, in E. Soloway, D. Frye and S. B. Sheppard (eds), *Human Factors in Computing Systems, ACM SIGCHI Proceedings, CHI'88* (Addison Wesley, Washington).

Wall, T. D., Jackson, P. R. and Davids, K. 1992, Operator work design and robotics system performance: a serendipitious field study, *Journal of Applied Psychology*, **77**, 353–362.

Walz, D. B., Elam, J. J. and Curtis, B. 1993, Inside a software design team: knowledge acquisition, sharing, and integration, *Communications of the ACM*, **36**, 63–77.

Waterson, P. E. and Clegg, C. W. 1994, Cognitive and organizational issues in programming in the large: Preliminary findings from a case study, *Proceedings of Sixth Annual Workshop of the Psychology of Programming Interest Group (PPIG-6)*. Open University, Milton Keynes, 6–8 January 1994.

Wolff, S. 1989, Knowledge Acquisition and possibilities for

eliciting expert knowledge, in F. Klix, N. A. Streitz, Y. Waern and H. Wandke (eds), *Man–Computer Interaction Research. MACINTER II* (Elsevier, Amsterdam), 413–421.

Zapf, D. 1991, Stressbezogene Arbeitsanalyse bei der Arbeit mit unterschiedlichen Bürosoftwaresystemen, *Zeitschrift für Arbeits- und Organisationspsychologie*, **35**, 2–14.