Organisation of design activities: Opportunistic, with hierarchical episodes

Willemien Visser

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

ORGANISATION OF DESIGN ACTIVITIES:

OPPORTUNISTIC, WITH HIERARCHICAL EPISODES

Willemien Visser - INRIA

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

ORGANISATION OF DESIGN ACTIVITIES:

OPPORTUNISTIC, WITH HIERARCHICAL EPISODES

First version: June 1993

Revised version: May 1994

Abstract. The organisation of actual design activities, even by experts involved in routine tasks, is not appropriately characterised by the retrieval of pre-existing plans, but is opportunistic (possibly with hierarchical episodes at a local level, but not globally hierarchical). Actually executed design actions depend, at each moment t, on the evaluation of actions proposed at t-1. These proposals can be made by pre-established plans, but also by other action-proposal knowledge structures. This position is supported by results from diverse empirical design studies. A major reason why design activities are organised opportunistically is that, even if designers possess plans which they may retrieve and use, the designers very often deviate from these plans so that their activity satisfy action-management constraints, of which the most important is "cognitive economy". Two types of variables underlying this opportunism are discussed: "situational" and "processing".

If design is opportunistically organised, a support system which imposes a hierarchically structured design process will probably handicap designers. Suggestions for systems offering "real" support are formulated.

Keywords. Organisation, Planning, Design activity, Opportunistic organisation, Control, Cognitive cost, Expertise, Routine task, Nonroutine task.

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

1. INTRODUCTION

This paper defends the claim that, even for experts involved in routine design, retrieval of pre-existing plans does not characterise the organisation of their actual activity appropriately. The arguments that will be used are supported by results from empirical studies which show that even if, when confronted with routine design tasks, expert designers may use pre-existing plans they possess, their activities will be organised opportunistically, because they often will prefer to proceed with actions other than those proposed by these plans. At the cognitive level, to which this paper will limit itself, this choice is guided by action-management considerations, the most important being cognitive economy.

This introduction will first present the distinction between "planning" and "organisation" and our hypothesis concerning the relationship between planning, the routine character of a task and a designer's expertise. Next, a second subsection presents and critiques planning models, focusing on the schema-based programming approach. In section 2, diverse empirical studies on design in different domains are presented, nearly each of them showing one or more factors contributing to the opportunistic character of the organisation of the activity. Section 3 presents "situational" and "processing" factors underlying the organisation of design activities and proposes elements for a model of design which can account for opportunism. Section 4 first presents a summary of the results on the relationship between the routine character of a task and the organisation of the activity. It then closes with a discussion of some consequences of our conclusions for design support tools.

1.1 Plan and organisation - Routine tasks and expertise

Several distinctions and definitions will be important in this text. They are presented in Table 1.

Hoc (1988b) considers a plan to be "a schematic and/or multilevel representation of an object (procedure or state), elaborated and/or used to guide activity". The schematic and anticipatory characteristics of a plan may guide subjects

in at least two ways: a procedural* plan by providing them with the structure of the activity, a declarative* plan by showing them the final or intermediary states of the activity.

Planning can be considered from several non-exclusive viewpoints: as a problem-solving activity with its own components (in the case of plan construction*), or as a component of another task, such as design. In this text, planning is examined as a component activity of design, alongside other components, such as the construction of

Plan	The mental representation(s) constructed and/or retrieved by designers in order to organise, i.e. anticipate and guide, their design activity.
Organisation	The structure of the actual design activity, such as it may be observed by an external observer.
Declarative plan	By showing the structure which the final or intermediary states of the activity must have, this type of plan allows the anticipation of states this activity must attain or go through.
Procedural plan	This type of plan guides the activity by representing both the co-ordination between the actions to be realised and elements of the control structure.
Plan retrieval	Planning by retrieval of particular specific pre-existing plans[1] or instantiation of plan schemas, both memory representations which result from plan constructions in previous activities.
Plan construction	Planning by construction of new plans, using pre-existing plans and other components.
Problem	(see text)
Routine - Nonroutine	(see text)
Expert - Novice	(see text)

Table 1. Main distinctions and definitions used in the text

problem representations and the proper problem-solving activities of solution development and solution evaluation (see Visser, 1991). Design may itself be a component of a more global problem-solving task; e.g., in

* Terms followed by this sign appear in Table 1.

1 "Plan" and "subplan" are relative notions (like "problem" and "subproblem"). When there is no risk of confusion, the term "plan" (or "problem") may refer to subplans (or subproblems), as well.

programming, design is supposed to come before coding (but see the software-design studies presented below in Section 2.2, in particular Visser, 1987).

The central distinction in the present text, between plan* and organisation*, refers to the difference between (the structure of) designers' mental representations and the structure of their actual activity.

Routine and nonroutine design - Expert and novice designers. A distinction between "routine" (or "familiar") and "nonroutine" (or "creative", "insightful" or "innovative") is commonly used in design studies in A.I. (e.g., see Brown and Chandrasekaran, 1989; Navinchandra, 1991). In cognitive psychology, the distinction is rather found in global analyses of problem solving (not especially in design studies). In this text, the distinction made by Mayer (1989) is adopted: "routine problems are familiar problems that, although not eliciting an automatic memorized answer, can be solved by applying a well-known procedure. Although the problem solver does not immediately know the answer to a routine problem, [they know] how to arrive at an answer. For example, the problem 888 x 888 is a routine problem for most adults. In contrast, nonroutine problems are unfamiliar problems for which the problem solver does not have a well-known solution procedure and must generate a novel procedure." (p. 40)

Note that the "routine" or "nonroutine" character of a design task depends on the designers' knowledge with respect to the "problem" they are confronted with, and on other characteristics of the problem situation. A "nonroutine" task for one designer may constitute a "routine" task for a colleague. This second designer may retrieve a pre-existing procedure leading to the answer or may not even need to proceed with "problem solving" for the execution of the task (when "simple" retrieval of the required answer is sufficient).

An analogous remark holds for the degree to which a designer may be considered "expert" or "novice": "expertise" depends on the situation one is confronted with. Nevertheless, the characterisation of a designer as an "expert" or a "novice" is often used more globally: "expertise" is generally expertise in a domain, and not expertise in one particular problem or even type of problem. Designers with an extensive experience in tasks in their domain are generally considered "experts" (in the domain), whereas their colleagues who -still- have little experience in these tasks are -still- considered to be "novices" (in the domain).

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

The dimension "routine-nonroutine (tasks)" can subsume the dimension "expertise", but in the presentation of the studies reviewed in this text, the values in the two dimensions will be used according to the terminology and analysis adopted by the authors of the studies analysed. An authors' decision on this point is not going to be questioned, because the tasks and subjects are rarely described in sufficient detail for such questioning. Expressions such as "(moderately) difficult", "novel" and "complex" will be considered to be synonyms of "(rather) nonroutine"; "familiar" as a synonym of "routine".

Planning, the routine character of a task and a designer's expertise. At first sight, the following "naïve" hypothesis could be thought to express the relationship between the two forms of planning and routine vs. nonroutine design:

- retrieval of pre-existing plans is sufficient for planning routine design tasks

whereas

- plan construction characterises nonroutine rather than routine design.

In terms of levels of "expertise", the corresponding hypothesis would be that:

- novices in a domain possess few pre-existing plan structures,
  so their planning tends to be constructive

whereas

- experts in a domain possess many pre-existing plans,
  so they tend to proceed by plan retrieval.

The data reviewed below impose, however, a revision of these hypotheses, leading to our stand:

even for experts involved in routine design, retrieval of pre-existing plans does not characterise the organisation of their activity appropriately.

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

1.2 Plan-based approaches to cognitive activities

This subsection presents briefly the plan-based model of programming, an activity in which design plays an important role. This presentation will be followed by a number of critiques of the model. With respect to more general planning models, only a brief mention will be made of some recent critical reactions to these models.

General planning models. Classic approaches to planning often suppose hierarchical plan structures (e.g., in psychology, Miller, Galanter and Pribram, 1960; in A.I., Sacerdoti, 1974). Presentations and discussions of these models can be found in many places. Allen, Hendler and Tate (1990) present the historically important papers on planning and action in the domain of A.I. (see also Chapman, 1987). Hoc (1988b) provides a cognitive-psychology approach, mainly based on results from studies in programming and classic problem-solving tasks. From a cognitive-ergonomics viewpoint, Bisseret (1987/1990) presents a critical presentation of hierarchical vs. opportunistic planning, defending his ideas by references to empirical research into design.

The "situated action" model by Suchman (1987/1990) formulates one of the main critiques of the concept "plan" (implying "pre-existing -")(see also the research by Chapman, 1987, on "improvisation" in action). Suchman claims that the main guidance of purposeful action -i.e. of most action- does not come from plans: "actions are primarily situated, and … situated actions are essentially ad hoc" (p. ix), i.e. "taken in the context of particular, concrete circumstances" (p. viii). "Plans are best viewed as a weak resource for what is primarily ad hoc activity." (p. ix) "A basic research goal for studies of situated action … is to explicate the relationships between the structures of action and the resources and constraints afforded by physical and social circumstances." (p. 179)

The schema-based programming model. The notion of "plan" is the central concept of one of the main theoretical approaches to programming taken in psychology, the schema-based, or plan-based, approach (see Détienne, 1990b). In this context, the notion "programming plan" is generally used as a synonym of "programming schema", but Ormerod's (1990) discussion of the difference between the two notions argues for preferring the term of "schema" in this context (see also the distinction made by Rist, 1991).

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

"Programming plans" are supposed to be the representational structures expressing the knowledge possessed by programmers. Such as these plans have been identified, this knowledge is rather "static": programming plans tend to express the representation of programs rather than that of the (programming) activity (see Davies, 1989, p. 488). This focus on static aspects may be due to the experimental tasks traditionally used to study and identify programming knowledge. Indeed, even if it may be claimed that "the same kind of knowledge is used by the processes of program composition and program comprehension" (Détienne, 1990b, pp. 205-206), research on "programming plans" has been mainly conducted in tasks focusing on comprehension -which is quite different from production (see Bisseret, 1987/1990, p. 214).

The critiques of the schema-based approach to programming emphasise mainly the important role of strategic aspects, which are neglected by this approach (see the studies referred to in Gilmore, 1990; see also Davies, 1989). Gilmore (1990) intends the "strategic approach … as a complementary, rather than alternative, explanation of expertise" (p. 224). The plan-based theory falls short if (or when) it considers acquisition of expertise as exclusively, or even mainly, the acquisition of knowledge (implying "declarative", i.e. not covering the mastery and use of procedures and strategies).

2. PLANNING AND ORGANISATION: EMPIRICAL STUDIES

An examination of the history of research on the organisation of activity in design tasks shows, overall, four stages: a first period in which the terms of "hierarchy" or "opportunism" are not relevant, as they had not yet been used: the period "before the conflict"; a second period in which several studies concluded that design was organised in a hierarchical way; a third period when various authors showed the design activities they examined to be opportunistically organised; a fourth period -in which we still are- in which several researchers have begun to qualify the "conflict" between "hierarchy" and "opportunism" (see Visser, 1992b).

Already in 1980, Green, in a paper on "Planning a program", concludes a discussion of structured programming methods by advancing the idea that "good programmers …. leap intuitively ahead, from stepping stone to stepping

stone, following a vision of the final program; and then they solidify, check, and construct a proper path. That proper path from first move to last move, in the correct order, is the program, their equivalent of the formal proof." (p. 306) Green notes that the author who introduced the concept of "stepwise refinement", Wirth, is himself "quite explicit; having described his stepwise refinement, [Wirth] says 'I should like to stress that we should not be led to infer that the actual program development proceeds in such a well organised, straightforward, top-down manner. Later refinement steps may show that earlier ones are inappropriate and must be reconsidered.'"(Green, 1980, p. 306)

Until the "Second Workshop on Empirical Studies of Programmers" in 1987 (Olson, Sheppard and Soloway, 1987), no empirical studies, except the famous one by Hayes-Roth and Hayes-Roth (1979), however, had shown -and analysed- the opportunistic character of the organisation of design activity. Most empirical studies -especially in the domain of software design- had been conducted in order to examine how (rather than "if" and, only if yes, "how") designers organise their activity hierarchically. Authors, in their presentation of results, insist on the hierarchical aspects of design, generally just noticing one or more "exceptions", presented as "details" with respect to the general hierarchical organisation. This is not surprising if, as stated by Jeffries, Turner, Polson and Atwood (1981) -the most notable example of this approach- the starting point of an empirical study on software engineering are the existing prescriptive software-engineering methods.

At the 1987 ESP Workshop, two papers were presented which questioned the traditional software-design models (such as those defended by Jeffries et al., 1981, and Adelson and Soloway, 1988). These papers were the Guindon, Krasner and Curtis (1987) and Visser (1987) studies, presented below, both making proposals for other models: Guindon, Krasner and Curtis in terms of "serendipitous" design, Visser in terms of "opportunism".

Position defended. With respect to planning, the following position will be defended in this paper -for design activities, but other types of activity could be concerned as well. Actions taken in a design activity cannot be predicted -i.e. completely- as stemming from pre-existing plans; in no way are they systematic. Often, in addition to the design actions which the plan proposes for execution, designers perceive "opportunities" -other possibly interesting design actions- mostly due to processing of data which they have at the time: specifically, the state of their design in progress, their representation of this design and their knowledge, and information at their disposal,

information they receive, and information they construct. The choice of the one design action to be executed, then, depends on criteria for action selection, the most important being cognitive economy (see below §3.3) (for more details, see Visser, 1990).

Studies reviewed: diverse types of design tasks. In order to defend this position, results from empirical studies conducted on design will be analysed. These studies were carried out on various types of design tasks (reference will also be made to another type of ill-defined problem solving). This paper is, indeed, not restricted to one particular type of design. Most empirical design studies have been conducted on software design (and it is in this domain that the "hierarchical-design" position has been introduced and most strongly defended, but among the studies conducted in other domains, also, some have concluded that design is hierarchically organised, whereas others show its opportunistic organisation.

Ten studies will be analysed. Half of them have been conducted on software design and half on other design activities; half of them concluding that, globally, design is hierarchically organised and half that the organisation of design is opportunistic. Together with five other studies, here only alluded to, they make up the studies presented in Visser (1992b); Table 2 gives their main characteristics.

reference	$ or £	expertise relative to the problem situation	type of design	results relative to hierarchy-opportunism
Adelson and Soloway (1988)	$	experts - a "novel" task "challenging to them"	software design: design of an electronic-mail system	systematic top-down breadth-first design without exceptions
Adelson and Soloway (1985)	$	experts - "mixed-familiarity" problems (see text)	software design: design of an electronic-mail system & design of an interrupt handler	top-down breadth-first design, but with deviations when problem domain is familiar
Jeffries, Turner, Polson and Atwood (1981)	$	experts - rather easy problems	software design: design of a page-keyed indexer	top-down breadth-first design, but with exceptions
Byrne (1977)	$	experts - "a familiar and practiced task"	meal planning	top-down breadth-first design, but with exceptions ("goal reordering")

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

Davies (1991)	$	experts - rather simple problems	software design: simple programming problems	broadly top-down design with opportunistic local episodes
Hayes-Roth and Hayes-Roth (1979)	$	experts - familiar task	errand planning	incremental, opportunistic planning
Guindon, Krasner and Curtis (1987)	$	experts - "novel" problem	software design: lift-control problem	"serendipitous" design
Ullman, Dietterich and Staufer (1988)	$	experts - rather routine problems	mechanical design problems	opportunistic, "locally controlled" design
Visser (1990)	$	expert - "mixed" routine problem	mechanical design: functional specifications	opportunistically organised design
Visser (1987)	$	expert - "mixed" routine problem	software design: automatic machine-tool control	opportunistically organised design
Bisseret, Figeac-Létang and Falzon (1988)	$£	experts - "a rather difficult problem"	traffic signal setting	opportunistic design
Kant (1985)	$£	"moderately or fairly sophisticated in algorithm design" (no experts) - complex problems	software design: computational geometry algorithm design	opportunistic design
Malhotra, Thomas, Carroll and Miller (1980)	$£	college students (no experts) - moderately complex problems	restaurant design	"subjects claimed to have designed top-down" (-> "self-reported strategy variables": no correlation with final designs)
Rist (1990)	$£	experts - rather simple problems	software design: simple programming problems	designers' activities may be organised hierarchically if corresponding schema possessed; if not, opportunistically organised activity

Voss, Greene, Post and Penner (1983)	$£	experts - moderately complex problems	political science problem solving	"no evidence of a well-developed solution plan" "subproblems encountered" when "exploring the implications of a proposed solution"

Table 2. Studies reviewed here (= $) and studies reviewed in Visser (1992b) (= £)

With respect to possibly cognitively relevant differences between the design activities in different domains, few ideas have been formulated. Most authors work in one particular domain and do not establish comparisons with other domains. An exception are Ullman, Dietterich and Staufer (1988) who contrast domains where form and function can be aligned (a functional decomposition corresponds directly to a form decomposition; e.g. software and circuit design) with domains where individual forms are engineered to do many functions simultaneously (engineering, e.g. mechanical design). According to the authors, an important consequence of this difference is that in design of the second type, each decision can potentially affect every subsequent decision, because a goal may be achieved by modifying a previously specified form, rather than by introducing a new form. Design activities in these domains, thus, would still be more exposed to an opportunistic organisation. Opportunism has, indeed, been "discovered" in a task other than software design, i.e. in errand planning (Hayes-Roth and Hayes-Roth, 1979). Afterwards it has however also been identified in software -and other- design activities.

Presentation of the reviewed studies. The presentation of these studies will be as follows:

• we will start with the software-design studies pleading most strongly for a hierarchically organised design organisation (Adelson and Soloway, 1985, 1988; Jeffries, Turner, Polson and Atwood, 1981),

• present succinctly one study in another design domain (meal planning), which does not focus on the hierarchy-opportunism question, but which tends to show a hierarchically organised activity (with some exceptions) (Byrne, 1977),

• say some words about an "in-between" software-design study (Davies, 1991),

• then present the study on errand planning in which Hayes-Roth and Hayes-Roth (1979) proposed to model planning as an incremental, opportunistic process,

• followed by two software-design (Guindon, Krasner and Curtis, 1987; Visser, 1987) and two "other-design" studies (Ullman, Dietterich and Staufer, 1988; Visser, 1990), all four concluding that design is opportunistically organised.

2.1 Empirical studies: design is hierarchically organised, but ...

The first three papers presented below are often considered to be "the" empirical studies showing the hierarchically structured character of design.

Adelson and Soloway, in their famous 1988 paper, present the "balanced development" model of design. They present the only software-design study, as far as we know, without "exceptions" to the hierarchical model. Three expert software designers receive a "novel" task which is "challenging to them" (design of an electronic-mail system): nevertheless, all three systematically implement a top-down breadth-first strategy.

Handling a problem at one level, a designer may think of related elements at another level. Whereas for other designers, in other studies, this often constitutes a deviation-triggering situation, Adelson and Soloway's experts stick to their plan. When they are capable to, they maintain these "related elements" in memory and retrieve them at the appropriate moment. Frequently they make "notes to themselves" about things to remember later in the design process (constraints, partial solutions, or potential inconsistencies). Adelson, Littman, Ehrlich, Black and Soloway (1985) posit the existence of "demons", "active information gatherers" which would remind the designer to incorporate such information into the design once the appropriate level of analysis has been reached.

This study, thus, seems to have shown that "even" in a nonroutine software design task, experts may organise their activity in a completely hierarchical way. Let us confront these results with those from other research conducted by the authors (Adelson and Soloway, 1985) and reaching the following conclusion: only in rather familiar domains is deviation from "systematic expansion" possible (i.e. without a considerable risk of problems) - in nonroutine tasks, experts (need to) organise their activity in a hierarchical way, because they cannot "allow" themselves deviations!

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

In their 1985 study, Adelson and Soloway examine several design problems, varying in their degree of "familiarity to the observed designers". An interesting observation in the context of the present paper has been made on an expert who solved two problems. The first was the mail-system problem, which constituted for this designer a "mixed" problem with respect to familiarity in that it concerned an "unfamiliar object" in a "familiar domain". The second problem, the design of an interrupt handler, was globally "familiar" to the designer: it concerned a "familiar object" in a "familiar domain", but it included an "unfamiliar" subproblem: the designer did not know the particular chip used as the interrupting device. Once the designer had developed an abstract solution to the global problem, he turned his attention to the functionality of the unfamiliar subproblem and explored it in detail. This differed from the way he handled the mail-system problem of "mixed" difficulty, where "exploration was cut off sooner and postponed via the making of notes". The authors explain this difference as the designer "allowing" himself to deviate from "systematic expansion" when he is familiar with a problem domain. "If the mental model is lost from working memory it can easily be reconstructed if it has been constructed frequently in the past. As a result, details can be explored … and then the mental model can be reconstructed when the designer is ready to continue systematic expansion." (p. 1358)

Jeffries, Turner, Polson and Atwood (1981) study four experts and five novices. The problem (design of a page-keyed indexer) is of "moderate difficulty" for the novices ("upper-division undergraduate" level), thus it may be expected that it is a rather easy problem for the experts. They start their presentation of results asserting that "almost all subjects approached the problem with the same global control strategy", problem decomposition, which is done, in general, by a progressive top-down expansion (or "successive refinement") of the design, expanding it in a breadth-first manner at each successive level. A close reading of the paper shows, however, that this "systematic" top-down breadth-first strategy has numerous "exceptions". One expert out of four, indeed, seems to implement it systematically, but the other experts deviate more or less from this strategy. The authors present some examples of situations in which these "exceptions" may occur: a designer may choose to deviate from the "advocated order" when he realises that a component has a known solution, is critical for success, or presents special difficulties. Some examples of behaviours shown by the three experts who deviated from the "optimal" strategy, both from its top-down and from its breadth-first component, are the following:

• deviation from a top-down expansion: descending in the decomposition tree, but coming back up afterwards (e.g., to introduce a whole new solution decomposition level); starting the decomposition in the middle of the tree; making interruptions for digressions at a level other than the current one (e.g., to handle other subproblems, or to define primitive operations, i.e. elements at the lowest level);
• deviation from a breadth-first expansion: starting the decomposition by processing only some branches of the tree; working simultaneously on two distinct branches.

Byrne (1977), in one of the early empirical studies on design, examines a domain other than software: meal planning. The author considers it to be "a familiar and practiced task" in which every subject can be considered to be an expert. As in the study by Hayes-Roth and Hayes-Roth (1979) (on errand planning), the "planning" studied by Byrne covers both the global meal-design activity and its planning component. Byrne did not focus in particular on the "hierarchical" - "opportunism" dimension, but his study provides data relevant to this question. The observed activity mainly seems to be organised hierarchically, and most decisions with respect to the choice of a course (a meal component) are made in a top-down fashion. However, some deviations of the *a priori* plan structures are observed. In spite of a pre-existing "standard" plan for processing the subgoals of meal planning, Byrne observes "goal reordering", generally leading to an order which allows goal satisfaction without any risk of backtracking: the subjects start by the goal which constrains (most?) the other(s). Byrne considers this to be the "easiest" order, i.e. the order which "economizes on effort".

Davies (1991) studies simple programming problems. We called this study an "in-between" study. Davies, indeed, judges that "the clear dichotomy between top-down and opportunistic approaches that is implied in previous work may be unfounded" (p. 186). He proposes a "model of program design … that tries to integrate existing views by characterizing program design tasks as broadly top-down with opportunistic local episodes" (p. 176) -exactly the opposite conclusion of ours! He focuses on the design strategies adopted by experts solving rather simple programming problems (the most difficult one does not take more than 78 minutes to be solved by an experienced programmer/designer). Davies concludes that "while opportunistic episodes may occur at any point in the evolution of a program" -in his study they appear rather as the task progresses than at early design stages- "the programming activity itself is hierarchically structured and proceeds in a largely top-down fashion" (Davies, 1991, p. 173). Our

major objection to this conclusion is that the top-down model may be considered as a special case of the opportunistic model (but not vice versa), a point raised originally by Hayes-Roth and Hayes-Roth (1979) that will be discussed below.

2.2 Empirical studies: design is opportunistically organised

This section presents five studies concluding that design activities are organised opportunistically.

Hayes-Roth and Hayes-Roth's (1979) famous study on errand planning is often used as "the" reference for the opportunistic nature of "planning". This "planning", as examined by the authors, was both a component activity of the global activity and the main activity itself (as in Byrne, 1977). Therefore the "errand planning" the authors examined may be considered as the activity of "designing a plan for the errands", where the resulting "plan" is a route connecting all errands. We will refer to the main activity studied by the authors (errand planning) as "design" (of plans), reserving "planning" for the component activity. This allows us to distinguish the main activity and the planning component, and to compare the activity studied by the authors with the other design activities presented in this paper.

In the model elaborated by the authors, plan design is conceived as an incremental, opportunistic process (see also Hayes-Roth, Hayes-Roth, Rosenschein and Cammarata, 1979). A tentative solution, i.e. a "plan", is elaborated by several co-operating cognitive "specialists" (knowledge sources) making decisions concerning which they can interact and communicate via the "blackboard". This common data structure is partitioned into several planes containing conceptually different categories of decisions: one is the "plan", the others are the "executive", the "meta-plan", the "plan abstraction" and the "knowledge base" planes. The assumptions of the model, which generalises the theoretical architecture of the Hearsay-II system, are illustrated with a subject's "thinking aloud" protocol, for which the authors show how the model can produce it. In Hayes-Roth, Hayes-Roth, Rosenschein and Cammarata (1979), the authors assert that the main purpose of their simulation is to test the sufficiency of the model as a psychological theory. Until its use to model planning -a generation problem- Hearsay-II had been applied exclusively to

interpretation problems. The success in modelling planning is, thus, considered to attest to the utility of the Hearsay-II framework as a general model of cognition.

The subject's processing was, indeed, multi-directional: the sequences of his basic actions, i.e. his decisions, included both top-down and bottom-up instances; in addition to being formulated at abstract, high levels, plans were also formed at low levels in the absence of corresponding higher-level plans. The authors note that, compared with a fixed, one-directional planning approach, "the bottom-up component in multi-directional processing provides a potentially important source of innovation in planning" (Hayes-Roth and Hayes-Roth, 1979, p. 306). They refer to similar remarks made by Feitelson and Stefik (1977) concerning the planning process of an expert geneticist who is interpreted to proceed in an event-driven way with the aim of "fishing for interesting possibilities" (ibid.).

The plan elaboration observed by Hayes-Roth and Hayes-Roth was neither systematic with respect to the breadth-first - depth-first dimension: the subject proceeded neither strictly in one, nor in the other way. Combined with the observations presented above about the absence of systematicity on the top-down - bottom-up dimension, this leads to the conclusion that the subject's actions did not fit a simple hierarchical structure. His design grows by "incremental accretion": "each new decision [is related by the planner] to some subset of his previous decisions. … The developing plan need not grow as a coherent integrated plan. Alternative subplans can develop independently either within or between levels of abstraction. The planner can incorporate these subplans into the final plan as she or he wishes." (p. 304)

Jeffries et al. (1981) criticise the conclusion drawn by Hayes-Roth and Hayes-Roth that design is opportunistically organised; they assert that the result may be due to the task and the subjects' level of expertise: none of the subjects would have had an extensive experience with errand-planning tasks. We believe, on the contrary, that every subject is more or less an expert in this task, given its "everyday" character (see Byrne's remark about the "familiar and practiced" character of the meals-planning task)!

Guindon, Krasner and Curtis (1987) studied eight professional programmers working on a realistic problem which took them some two hours to solve. This lift-control problem presented "novelty because none of [the] designers

had designed an identical system in the past, although they had worked on related real-time, concurrent, or embedded systems" (Guindon, 1990, p. 314).

The protocols of two designers were analysed in detail (see also Guindon, 1990). The observed design activities were qualified as "serendipitous": even if the observed design behaviour is interspersed with top-down decomposition episodes, control of problem solving proceeded by recognition of partial solutions, at different levels of abstraction or detail, without the designers having previously decomposed the problem into subproblems. This recognition of solutions to subproblems, often at a level of detail or abstraction other than that of the currently handled problem, may be triggered by familiar aspects of the problem environment that happened to be focused on. Understanding and elaborating the requirements through mental simulations often led to the sudden discovery of new -added or inferred- requirements. Such a discovery contributes to the "serendipitous" character of design when the corresponding solutions are developed immediately, rather than being delayed with a mental or written note. Examination of external solution representations also led to recognition of solutions to subproblems before "their turn". The designers expanded their solutions by rapidly shifting between different levels of abstraction and different parts of the solution decomposition and by developing low-level partial solutions prior to a high-level decomposition. With regard to the two designers under study in the analyses, all the "unbalanced" activities taken together made up 52% of the total of their design activities.

Ullman, Dietterich and Staufer (1988) studied how two mechanical design problems, each of a different type, were solved by experienced mechanical designers who could be expected to have a high degree of expertise for the selected problems. The authors conclude that their design process is "controlled locally, at the episode and task level. The designer does not formulate and then execute a global design plan execution. Instead, a central concept is developed and gradually extended to accomplish the design goals" (Ullman, Dietterich and Staufer, 1988, p. 36). Kant (1985) makes a similar remark, observing the importance of what she calls a "kernel idea": design generally starts by "selecting and sticking" with an idea which is "quickly selected from those known to the designer … [who] lays out the basic steps of the chosen idea and follows through with it unless the approach proves completely unfeasible (Kant, 1985, p. 1362) (see also the "primary generator" proposed by Darke, 1979/1984).

Ullman, Dietterich and Staufer (1988) notice that, firstly, designers progress from systematic to opportunistic behaviour as the design evolves (as noticed also by Davies, 1991) and, secondly, they do not always keep their designs "balanced". "Initially, …. subjects … established an initial agenda for solving the problem. This was followed by a systematic period of conceptual design. The subjects usually followed an organised plan of attack at this early stage. As the design progresses, subjects became more opportunistic." (Ullman, Staufer and Dietterich, 1987, p. 64) Attention may switch from the current problem to other (usually adjacent) problems which are suddenly "noticed": "ideas and problems are triggered by noticing patterns or configurations in the sketches. Sometimes, the focus of attention is immediately shifted to one of these, perhaps because the designer doesn't want to lose the 'good idea' by forgetting it." Another hypothesis formulated by the authors is that, "as the design progresses, the complexity increases to the point where the designer can only keep a small portion of the whole design in short term memory or in front of him or her in the form of a sketch. Consequently, an agenda is formed by what he or she can remember and see." (Ullman, Staufer and Dietterich, 1987, p. 65)

Little planning was observed: in only 13% of the design "episodes". The observed plans were usually rather "short-range, near-term plans. Their main purpose seems to be to evaluate whether a proposed task is worth performing at the current time. … Plans, once formulated, were not followed very exactly." (Ullman, Dietterich and Staufer, 1988, p. 43) Another hypothesis put forward by the authors is that plans are formed "prior to tasks in which many distractions are possible" because they may provide "a kind of defense against the possible distracting effects" of information that will be encountered during the coming episodes.

Concerning "balanced development", the authors "hesitate to draw any conclusions from [the] observations because of the ambiguity of the term…. We define it as the effort to keep all elements of a design at the same level of abstraction while moving the design toward completion. [However] we do not yet have an objective definition of levels of abstraction" (Ullman, Staufer and Dietterich, 1987, p. 65).

Visser (1990) studied the definition of the functional specifications for the control part of a "programmable controller" (a computer specialised in the control of industrial processes, here a machining process) which was to control an automatic machine tool. The conditions of this study -and the next one (Visser, 1987)- will be presented

with somewhat more detail than the previous ones, because -as already noticed above- we think that it is due to our studies having been conducted on real, professional design tasks that we were able to identify, not the opportunistic organisation of design -which had been shown already in other studies- but the main criteria and several new types of "opportunities" which are underlying this opportunism.

The study was carried out on a professional, industrial designer involved in a real design project, globally a "routine" task, but with subtasks having "nonroutine" characteristics. The designer, an experienced mechanical engineer, was observed during a period of three weeks full time, throughout his current task(s). His main task during this observational period was functional specification.

The study has focused on the differences between the designer's mental representation of the organisation of his activity (his "plan" of his activity), and the actual organisation of this activity (the "organisation" of his activity). In order to obtain data on his plan, we asked the designer to describe his activity, explaining -and insisting on the fact- that we were interested in the actual organisation, not in a rationalisation of it, such as a linearisation or any other type of structuring. He presented us with a description most appropriately represented by a "hierarchically structured" plan[2] (represented in Visser, 1990). In addition to another argument presented below, this description by the designer of a plan as representing -according to him- his actual activity, made us conclude that the designer, indeed, possessed a complete solution schema for the specification problem: what we examined afterwards, through our observation, was <u>if</u> -and if yes, <u>how</u>- he used this plan.

This mental representation of his activity, i.e. a hierarchical action structure accompanied by a control procedure covering it, reflects a procedural* plan. The designer may have thought that he actually followed this plan, but our observations showed that he did not systematically: he deviated from his plan whenever other possible actions or local plans that he perceived, i.e. "opportunities", were judged more "interesting" from a cognitive-economy viewpoint, i.e. were cognitively more economical. The plan described by the designer was certainly a mental representational structure -even the main one- which guided his activity, because he always came back to it after

[2] The plan is said to be "hierarchically structured" and not “hierarchical", to avoid confusion with “hierarchical planning” as described by Sacerdoti (1974).

more or less deviation actions (this constitutes the other argument for the assertion that the designer possessed and used this representation). It was followed, however, only as long as the designer did not perceive any other "cognitive-cost opportunity", because, as soon as he did, he deviated from his plan.

We identified both the main factors leading to alternative-to-the-plan design-action proposals -leading to an opportunistically organised activity if selected to be executed- and the control criteria used in order to select, from these proposals, the action to be executed. They will be presented in the next section.

The difference between a designer's mental representation of the organisation of his or her activity, and the actual organisation of this activity may be related to Suchman's (1987/1990) remark about the "reconstruction" of plans when presented in retrospect. According to Suchman, "it is only when we are pressed to account for the rationality of our actions … that we invoke the guidance of a plan. Stated in advance, plans are necessarily vague, insofar as they must accommodate the unforeseeable contingencies of particular situations. Reconstructed in retrospect, plans systematically filter out precisely the particularity of detail that characterises situated actions, in favor of those aspects of the actions that can be seen to accord with the plan." (Suchman, 1987/1990, p. ix) Contrary to Suchman, we believe that these plans described by the designers are, indeed, used to guide their activity, but that they are not the only guiding structures.

Visser (1987), in a companion study to Visser (1990), observed a rather experienced software engineer, for four weeks full time, throughout his design of the software for controlling an automatic machine tool, using the specifications defined by the mechanical engineer. Like the functional-specifications, the software design task was globally a "routine" task with subtasks having "nonroutine" characteristics[3].

[3] The software designer was of course supposed to deliver a "correct" program. There was, however, an entire, independent "testing and debugging" stage following his intervention on the machine-tool project. This third "stage" was executed by another software engineer, and took place in the workshop, in parallel with the mechanical testing of the machine tool (Visser, 1988a). Visser (1988b) gives a global presentation of the longitudinal three-stage study; Visser (1992a) presents -through examples from the functional-specifications design, the software design, and a third of her empirical design studies (on aerospace-artefact design)- an analysis of design at three levels: the global organisation, the strategies, and the problem-solving actions level.

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

With respect to the organisation of the activity, we tried to break down the software designer's activities into consecutive "stages". The criterion adopted to distinguish two stages was the following: a sequence of actions was considered to constitute one "stage" when the actions were homogeneous with respect to their function (e.g., planning, problem understanding, solution development, solution evaluation) (see also Pennington and Grabowski, 1990). Using this criterion, a first stage and a second stage could be neatly identified:

• Problem representation. The first day, the software designer "studied" and "analysed" the specifications he had received. Actually, he skimmed through the some 50 cm of A4 documents corresponding to these specifications.
• "Programming planning". During one hour, the designer plans his programming activity. This leads to a "programming plan". He does this planning along two lines:
- Breakdown of his task into program parts, according to the relative urgency with which various colleagues (especially in the workshop) need the different parts. This leads the designer to distinguish in the program three parts, which he plans to handle consecutively.
- Breakdown of the program into modules, according to the different machine-tool functions. For this decomposition, the designer re-used the order of modules on the listing of an "example" program that he had previously written for a similar machine tool.

After these first two stages (Problem representation and Programming planning), occupying all together one day, the designer directly started to code -and continued for over four weeks. This coding, however, was, of course, interrupted frequently for other activities, in particular because the design of the software to be coded was not at all finished. However, these interruptions were not systematic. In addition to coding, the designer carried out one or the other of the different types of activities which may be considered as components of a design activity, in particular planning, problem understanding, solution development, solution evaluation (where the "problem" and the "solution" are design problems and solutions). These different types of activities did not necessarily occur iteratively in a fixed order. Thus, there were no separate "stages" in the designer's activity during the rest of the four weeks, neither a "design" nor a "coding" stage. The designer's activities during this period will be simply qualified as "programming".

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

Therefore, except during the first short "programming planning" stage, the designer's "planning" was "local" and "punctual". During the entire period in which he was "programming", his activities took place at varying problem-solving levels and concerned larger or smaller entities (at the design level, e.g., a function or a machining operation; at the coding level, a module or an instruction). The designer had a "programming plan" inspired by the order of the modules in the "example" program, that he re-used heavily, but he deviated from this plan if another order was judged more economical from the point of view of cognitive cost. Local deviation actions, or alternative local plans leading to more global deviations, were triggered by various causes which might be analysed in the same terms as those proposed for modelling the deviations observed on the mechanical designer involved in his specification task.

3. OPPORTUNISTIC ORGANISATION OF DESIGN: WHAT, WHY AND HOW?

Underlying the claim defended in this paper is the idea that, in real design tasks, designers may often find it "profitable" to deviate from their plan. Indeed,

- even if designers **possess a solution plan** for their problem,
- and if they **can** and, in fact, **do retrieve this plan** to solve their problem (which is often possible for experts confronted with routine design),
- yet if **other possibilities for action** ("opportunities") are also perceived (which is often the case in real design)
- and if the designers **evaluate the cost of all possible actions** ("cognitive" and other costs), **as they will do in real design**,
- the **action selected** for execution will **often** be an action **other than the one proposed by the plan** (it will be a selected opportunity).

This line of reasoning requires that one takes into consideration a factor which has not been considered in most studies, i.e. the evaluation of possible design actions with respect to action-selection criteria. Another condition has to be fulfilled for our approach to be relevant: regularly, during a designer's activity, there must be opportunities

leading to proposals of "other" actions, in addition to those proposed by the plan. These conditions will generally not be satisfied in experimental studies in psychological laboratories.

The results presented above, in §§2.1-2.2, have led us to conclude that expert design activities, even in routine tasks, are opportunistically organised, but this does not exclude that, sometimes -in particular, under artificially restricted conditions- a solution can be developed completely according to a pre-existing plan (e.g., see Adelson and Soloway, 1988). In this section the main factors of opportunism will be described and elements will be presented for a model of design which is able to take into account its opportunistic organisation.

A characterisation of "opportunism" will be followed by a discussion of two types of factors which contribute to the way in which a person solves a problem, thus contributing to the organisation of an activity: "situational" factors, characterising "problem situations", and "processing" factors, qualifying a subject's knowledge sources and their use in problem solving. "Situational" factors are considered to result from a combination of task factors and subject factors, and therefore to depend upon the subject's knowledge and processing of the task. That is why the "processing" factors are considered to be the most relevant ones in a model of design activities. The two types of factors remain, of course, to be articulated -something which is only alluded to in this paper.

3.1 Opportunism

The authors who introduced the concept of "opportunistic planning" in psychology, Hayes-Roth and Hayes-Roth (1979), consider the activity observed in their study, i.e. sequences of errand decisions, to be "opportunistic" because "at each point in the process, the planner's current decisions and observations suggest various opportunities for plan development. The planner's subsequent decisions follow up on selected opportunities" (p. 276). "Each decision is motivated by one or two immediately preceding decisions, rather than by some high-level executive program" (Hayes-Roth, Hayes-Roth, Rosenschein and Cammarata, 1979, p. 381).

Our approach to "opportunism" is inspired by Hayes-Roth and colleagues. In their model, planning is under the control of the "executive" which "selects one of the invoked specialists to execute its action" (p. 291). In the same

way, we distinguish between a level of action "proposal and execution" (by knowledge sources or "specialists") and another, control level where actions are "selected" for execution. The two levels may be articulated according to the following iterative sequence (see Visser, 1990):

(i) proposal of one or more actions

(ii) evaluation of the proposed action(s), leading to selection of one action

(iii) execution of the selected action

(iv) back to (i)

In an "opportunistically" organised activity, the actions proposed at a moment t depend on the data which designers have at that moment: these are mainly the designers' representation of the state of their design in progress (thus, the result of the preceding design actions), their knowledge (domain knowledge, design knowledge, and more general problem-solving knowledge), the information at their disposal and received from other sources (technical documents; the problem specifications; the requirements provided by the client -and modifications upon these requirements; remarks and suggestions made by colleagues or other information provided by them) and the information constructed by the designer (examples will be presented below). In Hayes-Roth and Hayes-Roth's experimental setting, the problem specifications which were given to the subjects did not receive any later addition or modification -as they do in real-life tasks. It might be due to this setting that the authors did not identify the important role of information other than that stemming from a designer's previous actions and knowledge.

The actions to be executed depend, at each moment t, not on a pre-established plan followed without any possibly previous comparison with other action proposals, but on the selection of one action, from all those proposed at t-1, based on action-management evaluation criteria (mainly "cognitive economy"). Thus, ultimately, it is this evaluation of the proposed design actions which causes a designer's activity to become opportunistically organised. "Ultimately", because if all action proposals are made by a pre-existing hierarchically structured plan, the activity will, completely or mainly, be organised in a hierarchical manner.

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

N.B. Sometimes not "completely", but "mainly", because, even if they use only one or more pre-existing plans, designers can decide to not follow them completely. They can skip actions proposed by the plan they are following. This may occur when the planned action P is judged as costing too much, not compared to a currently proposed "alternative" action -no such actions are proposed- but to P itself if executed later. Action P may cost too much when the procedure for obtaining the information required by the planned action is judged as being too expensive, e.g. because the information is not directly available (values are to be calculated, or documents are to be examined), is very expensive to obtain now (the colleague who possesses the information is absent) or is impossible to obtain now (the client has not yet made a decision required for the planned action). Another factor may be what several authors call the "difficulty" of the planned design action (see below). In all these cases, the planned action P is postponed and a local plan is generally formed for re-proposing P, either as soon as the conditions leading to its postponement no longer prevail, or at a particular moment later in the design process.

In conclusion: we join Hayes-Roth and Hayes-Roth (1979, p. 307) who propose, in order to solve "the apparent conflict" between the top-down, successive refinement model and the opportunistic model, to consider the refinement model as a special case of the opportunistic model. Thus, an opportunistically organised activity may have hierarchical episodes, i.e. at a local level, but its global organisation is not hierarchical (the opposite position is defended by Davies, 1991).

3.2 Problem situations: which characteristics can make a designer's activity opportunistically organised?

This subsection will set out how characteristics of "problem situations" may contribute to the approach which solvers take to a problem, and, consequently, to the way in which they organise their activity.

"Situational" characteristics. Rather than characterising a particular task as a "problem", independently of a subject (designer) or other cognitive system confronted with this task, we consider that a "problem" only exists relative to a subject (see Hoc, 1988b; Mayer, 1989). If one defines a "situation" as "a functional system constituted by a task and a subject" (Leplat and Hoc, 1983, p. 49), a situation constitutes a "problem" or not, for a particular subject,

depending on the representation this subject constructs of her or his task. We follow Mayer (1989) (see his definitions presented above in § 1.1), in considering a situation to constitute a "problem" for a subject if the representation of the situation constructed by the subject does not elicit a memorized answer, but if the subject needs either to apply a well-known procedure ("routine problems"), or to generate a novel procedure ("nonroutine problems") to solve the problem.

3.2.1 Problem characteristics

Four "problem characteristics" are distinguished in the literature on design and other ill-defined problems. Given our approach according to which a situation constituting a "problem" for one designer does not necessarily constitute a "problem" for a colleague, one or more of these characteristics might be better considered "situational" characteristics.

Problem structure or structuredness. Hayes-Roth and Hayes-Roth (1979) assert that "planners might usefully apply a top-down approach to planning whenever the problem at hand exhibited an inherent hierarchical structure" (p. 306). As an example of a such a problem, the authors present the menu-planning problem used by Byrne (1977), whereas their own errand-planning problem "did not exhibit any obvious hierarchical structure" (p. 307). The reasoning followed by Hayes-Roth and Hayes-Roth seems to presuppose that the "structure" of a problem is the structure of its final solution.

Guindon (1990) claims that, with respect to the approach taken by a subject who has to solve a problem, the "structuredness" of this problem is an important variable. She identifies three factors as contributing to a problem's "structuredness": the degree of "novelty" of the problem, the degree of completeness of its specifications and the number of knowledge domains to be used in solving it.

The structure of a problem may be presented more or less clearly. Carroll, Thomas, Miller and Friedman (1980), in a study of the design of a library system, examine the influence of the structure of the explicit problem presentation on four dependent variables characterising the solution or the solution process. They show that problem

presentations which are explicitly more hierarchically structured result in more stable design processes and more clustered solutions.

Time allotted for the task. Hayes-Roth (1979, quoted in Hayes-Roth and Hayes-Roth, 1979) has studied a variable which can be considered to be an "objective" "problem characteristic": the "amount of time available for plan execution". She "successfully induced alternative planning approaches by manipulating [this] amount of time.… For problems that imposed severe time constraints, most subjects adopted a top-down approach. For problems that imposed minimal time constraints, most subjects adopted a bottom-up approach." (Hayes-Roth and Hayes-Roth, 1979, p. 307)

Delay of implementation. Voss, Greene, Post and Penner (1983) consider that one of the two factors making social-science problem solving quite difficult is the "delay from the time a solution is proposed and accepted to when it is fully implemented". This delay affects in particular solution evaluation. "Naturally, a good solution anticipates changes in conditions, but anticipation can be quite difficult." (ibid.) This argument of course also holds for design problems.

Inter-solvers agreement. The other factor taken by Voss et al. (1983) to make particularly difficult the problem solving in social science (and which may also apply to design problems), is "the lack of agreed-upon solutions". Comparing the social-science problems used in their study with those "used thus far in problem-solving research", the authors argue that the specificity of problems in the domain of the social sciences is their "lack of agreed-upon solutions": "social science problems seldom have solutions about which experts are in complete agreement" (p. 169). Voss and Post (1988) relate this point to Reitman's (1965) analysis of the role of agreement in constraint satisfaction among a community of solvers. In Reitman's terms, there is disagreement in the domain of social sciences regarding how to fill open constraints. That is why, according to Voss and Post, "it is possible … to have ill-structured physics problems.… for example, in new areas of research" (p. 263). The authors refer to Tweney (1981) for evidence which suggests that, in these conditions, problem solving in physics highly resembles political-science problem solving.

Analysing the composition of a fugue, Reitman was, indeed, one of the first authors to conduct an empirical study on design problems, and to characterise these ill-defined problems (see also Reitman, 1964). He considers that the continuum ranging from well-defined formal problems to such ill-defined empirical problems as composing a fugue is related to the idea of "ambiguity": for ill-defined problems, there is little or no agreement, over a specified community of problem-solvers, regarding the referents of the problem attributes, permissible operations, and their consequences. That is why ill-defined problems do not have "correct" (or "incorrect"), but more or less "accepted" solutions (Reitman, 1964).

3.2.2 Subject characteristics

Two "subject characteristics" are described below. In addition to the "classic" subjects variable "expertise", another, rather vague factor will be presented: "individual differences".

Expertise. In the problem-solving literature, differences in expertise between subjects have generally -even if implicitly- been considered as differences in level of expertise (expert vs. novice); differences in type of expertise have rarely been studied (but see Visser and Falzon, 1992, who have compared the way in which two expert designers in the same domain, but with different design experiences, differ with respect to various aspects of their knowledge organisation).

In the problem-solving literature, this expertise differentiating experts and novices has been analysed as depending on various factors, both static and dynamic (see Glaser, 1986; Glaser and Chi, 1988; Holyoak, 1991). In studies on design, however, most authors have focused on designers' static knowledge structures. In the studies on software design these structures have mainly been formalised in terms of "plans" and/or "schemas". Studies focusing on dynamic and strategic variables have shown that these differences in knowledge structures led novice and expert designers confronted with a same design task to adopt different strategies: experts tend to adopt a "depth-first", whereas novices a "breadth-first" strategy (see § 1.2, where the main critiques of the plan-based approach were presented, according to which this approach neglected the importance of strategic differences between experts and novices).

Individual differences. This label is often used in clinical studies to cover one or more generally not specified "personality" variables. Hayes-Roth and Hayes-Roth (1979) use the term when they present the results of the Hayes-Roth (1979) experiment, in which "many of [the] subjects exhibited a strong proclivity to adopt a bottom-up approach regardless of problem characteristics. Even with explicit instruction, some subjects persisted in using the bottom-up approach." (Hayes-Roth and Hayes-Roth, 1979, p. 307)

3.3 Problem processing: why and how does a designer's activity become opportunistically organised?

We suggested above that, in order for an activity to be organised opportunistically, there must be, in addition to the "planned" actions, "opportunities" which are also suggesting actions. These suggestions stem from the opportunities being taken into account and processed by knowledge sources other than the plan. These knowledge sources and the action-selection criteria will now be presented.

3.3.1 Cognitive control criteria for action selection

This text focuses only on cognitive variables, but, of course, other criteria may also play a role in action selection: preferences for solutions or procedures due to personal variables or the "culture" of the enterprise: e.g., certain solutions are considered to be more "beautiful" or "elegant", or certain procedures are considered to be "better" in a particular enterprise.

In Hayes-Roth and Hayes-Roth's (1979) model, control of the planning process is done by an "executive", which has three levels: "priorities", "focus" and "schedule". Schedule decisions resolve conflicts between competing specialists "on the basis of relative efficiency, reliability, etc." (p. 290). Unfortunately, the paper provides no examples of these criteria in the analysed subject protocol.

Cognitive economy: the main selection criterion. We suppose that, from an information-processing viewpoint, the main goal underlying the designers' choice from several possible actions for the organisation of their activity is

supposed to be "cognitive economy" (Visser, 1990). At each step in the design process, control selects one action from those proposed. If an alternative-to-the-plan action is more profitable than the planned action from a cognitive-economy viewpoint, designers adopt this proposal: they, then, deviate from their plan.

The cognitive cost of an action depends on the number of information units to be processed and the nature of this processing (both the cost of accessing the required information, and that of using it). On the one hand, an action may be interesting from the viewpoint of cognitive cost provided that the information required for executing it is available, i.e. accessible without processing a large number of information units. This question applies to actions proposed in a concept-driven way, i.e., in the present context, actions proposed by the designer's pre-established plan(s). On the other hand, an action may become interesting because information available allows for its execution. Thus, e.g., if a solution procedure just developed by the designer to solve a subproblem can be applied on other subproblems, an important part of the information required for solving these subproblems is accessible at a particularly low cost.
N.B. The fact that a procedure formally "can be applied" is of course not relevant if the designer does not "perceive" this possibility. It is the perception of possibilities for actions "in" data "coming by" that plays a very important role in "opportunities" arising. The importance of such data-driven aspects in the organisation of design activities has also been underlined by other authors: Kant (1985) uses the terms of "discoveries" which are made and "good ideas" that the designer gets from the data.

Davies (1991) considers that "opportunistic episodes arise largely as a consequence of simple cognitive failures …. related to working memory capacity limitations" (p. 185). In our model, the limited capacity of working memory is only one of the various factors contributing to a designer choosing one action rather than another: it does not favour an opportunistic or a top-down approach. On the one hand, plans which -if followed- would lead to a top-down approach may be interesting from a cognitive-economy viewpoint because they organise into a relatively small number of chunks (better manageable in terms of working memory capacity limitations) the information-processing elements necessary to execute the design actions. On the other hand, alternative action proposals -leading, if the action is selected, to an opportunistically organised activity- may also be interesting from a cognitive-economy viewpoint. Because alternative actions lead to plan deviation, the "gain" of plan deviation must, of course, be greater

than its "cost", i.e. the cost of resuming the plan. Plan resumption requires plan retrieval from memory, which generally may be relatively inexpensive for an expert, especially in routine tasks. Thus, the gain of plan deviation may be relatively high when an "alternative" action costs less, if executed now, than if executed later when it would be "its turn" according to the plan (several examples are presented below; others may be found in Visser, 1990).

"Difficulty": another term for "cognitive cost"? Several authors suppose that the "difficulty" of a problem domain, a design project, a design action, etc. may determine whether designers follow their pre-established plan, form a local plan, or deviate from their plan.

Both Ullman, Staufer and Dietterich (1987) and Davies (1991) observe that as design evolves, designers progress from systematic to opportunistic behaviour. Ullman, Staufer and Dietterich (1987) notice that at early stages, expert subjects usually follow an organised plan of attack, but that, as the design project progresses, "the complexity increases to the point where the designer can only keep a small portion of the whole design in short term memory or in front of him or her in the form of a sketch" (Ullman, Staufer and Dietterich, 1987, p. 65) Adelson and Soloway (1985) put forward that designers may "allow" themselves to deviate from "systematic expansion". Jeffries et al. (1981) observe that the difficulty of an action proposed by the plan followed by a designer may lead the designer to skip this action.

If the "difficulty" of a problem plays a role in the approach taken to the problem, one may ask which problem-solution aspects are processed first: the difficult ones or the easy ones? Kant (1985) observes that the most difficult, most problematic problem-solution aspects are handled first (an aspect of the algorithm which is a potential problem "is more likely to be expanded to ensure that the algorithm as a whole is feasible", p. 1366). Also, according to Guindon, Curtis and Krasner (1987), two general heuristics used by designers are to select the most difficult subproblem first, and to postpone the solution of a subproblem if it is known to be trivial. In a study by Malhotra, Thomas, Carroll and Miller (1980) on novice designers, a majority of the subjects claim to have tackled the more difficult problems first. However, neither this, nor other "selfreported strategy variables" were correlated with the final designs. Spohrer (quoted in Rist, 1990) suggests that either "simplest-first" or "hardest-first" orders are possible, but Rist (1990) himself observes (or only supposes?) that these "more rational methods" are reserved to

experts and that, before acquiring this expertise, novices tend to proceed by random selection. In conclusion: even if, put together, the elements proposed by these various authors do not constitute a completely non-ambiguous answer, most authors seem to think that designers start by the most difficult aspects.

This "difficulty", which seems to play a role in the approach taken to a problem, is it only another term for "cognitive cost", is it a "type" of "cognitive cost", is it a situational characteristic, a factor of "cognitive cost", …? (see Byrne, 1977, for whom the "easiest" component-processing order is the one which "economizes on effort"). The question -and the fact that the answer is not obvious- may be taken to show that these notions are not (not yet?) sufficiently specified.

"Cognitive economy" in A.I.. Lenat, Hayes-Roth and Klahr (1979) use the term "cognitive economy" to describe the productivity of systems, heightened without sacrificing expressibility. "*Cognitive economy is the degree to which a program is adapted to its environment*, the extent to which its internal capabilities (structures and processes) accurately and efficiently reflect its environmental niche." (p. 3) The authors propose a certain number of techniques enabling programs "to (semi-)automatically improve themselves and thus increase their productivity". One family of techniques, "caching", is contrasted with psychological ideas of economy and the authors present psychological evidence in order to support their idea.

"Caching" is "storing the results of frequently-requested searches, so [that] they needn't be repeated over and over again; i.e., *intelligent redundancy*" (p. 3). The authors refer to psychological experiments by Rips, Shoben and Smith (1973) and by Hayes-Roth and Hayes-Roth (1975, 1977). These authors, indeed, presented counterexamples to the hierarchical retrieval schemes proposed by earlier cognitive psychological research on semantic networks (particularly, Collins and Quillian, 1969, 1972). Hayes-Roth and Hayes-Roth (1975) proposed "an adaptive model of memory, in which all learned relations are represented directly, with strength proportional to experience. There is also good evidence for redundancy in the network, with multiple routes connecting nodes representing particular pairs of concepts." (Hayes-Roth and Hayes-Roth, 1975, p. 519, quoted in Lenat, Hayes-Roth and Klahr, 1979, p. 28) (see also the concept of "knowledge compilation" introduced by Anderson, 1986, as "the general learning mechanism")

Only "caching" techniques are discussed by the authors in reference to psychological data. The other techniques and characteristics proposed by Lenat, Hayes-Roth and Klahr (1979) as being useful for intelligent systems "having more potentially interesting things to do than they have resources to pursue" (p. 41) could, however, also be psychologically "plausible". These are techniques for "dynamic self-monitoring and self-modification" (i.e. learning), "expectation-filtering", and the use of "multiple levels of abstraction". All these are characteristics which the authors suppose to be needed by "A.I. systems … addressing the same question facing intelligent humans: 'What would I most like to accomplish next, and how can I do that economically?'" (p. 42).

A second criterion: "Importance" of the design actions proposed. This second action-selection criterion only plays a role when the designer has to choose from several actions at equal cost. Examples of two factors contributing to the importance of an action are the importance of the type of the action and the importance of the type of object concerned by the action (for more details, see Visser, 1990). E.g., fixing the omission of an operation which has been forgotten seems to be an important action, independently of the type of operation: as soon as the designer discovers that he has forgotten an operation, he fixes it. Verification is an important action if it concerns operation "durations", but not if it concerns operation "identifiers": the engineer frequently deviates from his plan in order to verify the duration of an operation, but never to verify its identifier.

Other criteria? In addition to the heuristics for ordering problems dependent on their difficulty, Guindon, Curtis and Krasner (1987) propose other problem-selection heuristics:

• if the solution of a subproblem affects, or is necessary for, the solution to another subproblem, attempt to solve the former subproblem first (see Byrne, 1979's "goal reordering" depending on the constraints between goals);

• if a subproblem is at a too low level of detail, postpone trying to solve it;

• if, after a certain amount of effort, no satisfactory solution has been reached to a subproblem, shift the focus on the subproblem, by temporarily abandoning it.

Do these heuristics translate what we call action-selection criteria? The first one concerns a dependence between problems: according to Byrne, this variable contributes to the cognitive cost of the corresponding action(s).

"Cognitive cost" seems the variable underlying the last strategy, expressed as it is in terms of "effort". In order to relate the second heuristic to the action-selection criteria, an analysis of its motivations is needed: why should one postpone solving a low-level problem? What happens -to other problems and to the conditions of them being solved- if one solves a "too" low-level problem?

3.3.2 "Opportunities" leading to the proposal of cognitively interesting actions

According to our approach to "opportunism" presented above, the main factors leading to alternative-to-the-plan design actions stem from "taking advantage" of "opportunities", i.e. from data-driven processing of information, (permanent) knowledge and (temporary) design representations. Taking advantage of these "opportunities" rather than following a pre-established plan will, indeed, lead to an opportunistically organised activity.

Reformulating to some extent the six processes presented in Visser (1990, where more details and examples can be found), we identified six categories of data candidates for being "taken advantage of". In these categories, two distinctions are used: (i) between different types of data: information from external sources, knowledge (rather "permanent" mental structures) and mental design representations (rather "temporary" mental structures); and (ii) between different ways in which designers may encounter these data: the designers may receive the data from an external source, they may construct them themselves (e.g., by analysis, exploration, or taking a different viewpoint on them), and they may "come across" them when "drifting". The first distinction, between different types of data, is unambiguous, but the second one is subject to discussion. Some possible other distinctions will be discussed below. E.g., construction of information could have been a category embracing all the others: nonetheless, separate categories have been created, instead, in order to emphasise particularly interesting observations.

The examples presented below come from the observations made on the designer of the functional specifications (Visser, 1990). Examples could also have been taken from the software-design study (Visser, 1987), but we have chosen to refer to only one study in order to avoid the risk of confusion between similar but different design components used in the two studies.

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

1. Information provided to the designer by an external information source (in particular, the client or a colleague).The client may communicate modified requirements to the designer (see examples in Morais, 1987). Colleagues may present designers with new, relevant information, because they are specialists in other domains, and / or have another viewpoint on the current design state. In both cases, designers could receive this information from an external source and decide to put it aside until later, when it would be "its turn" to be processed in the context of a planned action. In general, such information is, however, taken into account immediately. This may be because it applies to aspects of the design already handled: therefore it involves additions or modifications to be made in the design as it exists already -and not in parts of the design still to be handled. However, the reason may also be that, once the information has been processed to decide if and when the designer will need it, the main part of the processing required for the corresponding specification action has been done. The main part of the action's cognitive cost has, thus, been "paid": if the action were postponed, this "expense" would be "for nothing".

This existence of external information sources providing data at unforeseeable moments, and concerning unforeseen design aspects, introduces an important difference between professional design and design such as it may be observed in controlled experiments. It provides one of the sources of opportunism in "real" design.

2. Information the designer "comes across" when "drifting". In Visser (1990), we defined "drifting" as involuntary attention switching to a design action other than the current one, noting that rather than calling it "involuntary", one might consider it as "not yet explained". It may lead to phenomena verbalised as "I am thinking of X", or "I might rather do Y now" because "I am afraid to forget it otherwise".

We suppose that drifting occurs in particular when it is difficult to attain the current goal. In these conditions, designers often look, at various places, for possibly useful information. During these explorations, they can come across data whose use is "obvious", but for a design action other than the current one. Therefore, drifting is likely to occur during information search that is not guided by precise hypotheses concerning the location of the information or its retrieval procedure.

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

Example. The designer, involved in the specification of an operation Ox, finds its ending conditions hard to define; he thinks and hesitates: "What shall I put there? ... I don't know". He does not seem to know which information is required to define these ending conditions and which information source(s) could be consulted. He, then, interrupts his current definition (operation Ox) to define the ending conditions for the following operation, Oy, remarking: "There, I know what to put".

On the one hand, drifting may occur at different problem-solving stages: it may occur when designers are looking for information required for the current design action, thus before they have identified this information; or once they have got the information, but before they have used it for the current design action. On the other hand, designers may drift from information provided by different sources, external information sources or their own knowledge (an internal information source).

Example. The designer is specifying an operation Ox using a "tool plate", i.e. a document providing information on operation durations. During this consultation, he is "struck" by a modification of other operation durations made by colleagues[4]. He interrupts the specification of Ox in order to take into account this modification in the specifications of these other operations.

In every case of drifting, the designer comes to focus on information other than that required for the current design action, except when one considers opportunities of the following category also as "drifting".

3. Information constructed by considering data from another viewpoint. Information used -or looked up, but not yet used- for the current design action may also -or rather- be used for another action because it is also -or rather- considered from a viewpoint other than the one taken on the information as it is used for the current action.

Example. Several times during his functional-specifications task, the designer happens to consider information under use also from a mechanical viewpoint, leading him to switch to his mechanical-specifications task, which, in principle, was finished before the functional specifications.

---

[4] Different designers collaborate on the design project, using and modifying the same documents.

We noticed above that this category could have been put together with the previous one, i.e. "drifting". It could also be considered as a subcategory of the following one, construction of information.

4. Information constructed as a by-product of problem-solving actions, such as analysing the problem specifications, developing and evaluating solutions, exploring the implications and consequences of a proposed solution (this category has also been identified by other authors; see Guindon, 1990; Kant, 1985; Voss et al., 1983).

Example. Frequently the specification of an operation Ox leads the designer to verify a "corresponding" operation Ox'. Ox' may be the "same" operation taking place on another workstation (such as the Fast Advance Movements of the Second Phase Workstation and the First Phase Workstation), or the "same" operation at another speed (such as the Fast and Slow Advance Movements). Such deviations may occur because processing an aspect of Ox makes the designer think that he has forgotten, incompletely specified or ill specified, this aspect of Ox'.

5. Mental representations of design objects activated by the representations used for the current design action, through activation-guiding relationships existing between the two sets of representations. Four of these relationships leading to switches between design-object representations being activated were identified: analogy, prerequisites, opposites and interaction.

The "goal reordering" observed by Byrne (1977) may be related to this process. This reordering of the processing of design components was made in order to avoid the need for backtracking. In order to achieve such an "optimal" order of goals, the design-component representations must be related in memory in terms of their mutual constraints, or in terms allowing these constraints to be calculated.

Example. The designer considers "first-phase tooling" operations (operations made in order to shape the rods) as analogous to their corresponding "second-phase tooling" operations (finishing the rods). He continually switches between the specifications of such analogous tooling operations: often he takes advantage of the specification of a first-phase tooling operation O1 to specify, by adaptation of this O1 specification, its corresponding second-phase

tooling operation O2. Frequently, an O1 specification "makes him think" of an omission or error made on the corresponding O2 operation.

6. Mental representations of design procedures available because they have just been developed for the current design action. According to his plan, the designer constantly switches from one type of design-component aspect to be specified to another type, and, thus, from one type of design procedure to develop and / or apply to another. Sometimes, however, he decides to specify the same aspects of a series of design components "in a row" (see Détienne's, 1991, "reuse in a row").

Example. Having specified a particular aspect of a design component Cx leads the designer to also specify this aspect of one or several other components Cy - Cz. E.g., the designer defines the starting conditions of five consecutive operations ("consecutive" according to the plan). It also occurs that a series of aspects is specified of a series of design components: the starting and ending conditions of a number of consecutive operations are defined "in a row".

The last two categories of opportunities stem from taking advantage of design procedures being available "at low cognitive cost", because having just been developed and, thus, -that is our hypothesis- still available as one single chunk in (working) memory. Indeed, retrieving from memory a procedure which constitutes one single chunk is less expensive than having to develop this procedure from pieces (later on, when it will no longer be available as one chunk in memory).

3.4 Problem situations and problem processing: concluding remarks

This section may be concluded by two remarks: the first concerns the nature of the "problem characteristics", the second the relation between the situational and the processing characteristics.

As noticed, so-called "problem" characteristics may be related, not to the "problem" task alone, but to the situation, i.e. the combination of task and subject. In particular, this is the case of "problem structure": it constitutes a

dimension in which the value depends on the subject confronted with the problem situation (see also the remark by Ullman, Staufer and Dietterich, 1987, concerning the difficulty to give an "objective" definition of the abstraction levels of a problem).

Some elements for the necessary articulation between situational and processing characteristics have been presented along this section. We noticed how the "goal reordering" observed by Byrne (1977) might be due to the activation of representations of design objects (here meal courses corresponding to the goals) by the representations used for the current design action (design of the current meal course).

The relations between problem-situational variables and the processing of the problem may be mediated by action-selection criteria: the structure of a problem situation may be supposed to contribute to the "cognitive cost" of its processing. Byrne considers the "easiest" component-processing order -result of the "goal reordering- to be the one which "economizes on effort".

These relations and dependencies remain to be examined further. As noticed, a necessary step in this investigation is the specification of the different notions which have been used: in particular "cognitive cost", "effort" and "difficulty".

4. DISCUSSION

Before closing with a discussion of some of the consequences of our conclusions for design support tools, this section will sum up the main points of this paper.

Summary of the results. The results discussed in Section 2 were considered as supporting our stand that, even for experts involved in routine design, retrieval of a pre-existing plan does not characterise the organisation of their actual activity appropriately. Such a plan which -if it is followed- may lead to systematically organised activities, is supposed to be only one of various action-proposing knowledge structures. Often, these knowledge sources make

designers perceive "opportunities", i.e. other possibly interesting design actions. The choice of the one design action to be executed then depends on action-selection criteria, the most important being cognitive economy. Pre-existing plans may be interesting from a cognitive-economy viewpoint because executing an action for which such a schematic memory representation is available may cost relatively little if all schema variables relevant for execution have constant or default values. But if other knowledge structures propose relatively more economical actions, designers may deviate from their plan. In particular this is true for experts, who may be supposed to possess -or else to be able to construct without great difficulty- a representation of their activity which allows them to resume their plan later on, when it once again becomes profitable to do so from the viewpoint of cognitive economy. Having to compare several action proposals and taking into account the cognitive cost of an action are two characteristics which probably only appear in "real" design. This may explain why in laboratory experiments, one mainly observes systematically organised design activities.

Section 3 presented two types of factors contributing to the possibly opportunistic organisation of an activity: "situational" factors, characterising "problem situations" (i.e. task-subject couples), and "processing" factors, qualifying a designer's knowledge sources and their use in design problem solving. Because "situational" factors depend upon a designer's knowledge and processing of the design task, "processing" factors are considered to be the most relevant ones for a model of design activities, but the two types of factors remain to be articulated.

Design support tools: assisting opportunistically organised activities? The conclusions of this paper may inspire, at least, two approaches to design assistance: given the opportunistic organisation of the activity of designers working "in total freedom", one may either try to use tools to prevent this type of proceeding, or offer designers such tools as would assist them in their "natural" way of proceeding.

Therefore, examples of questions which may be asked are the following. Should a designer be allowed to have an opportunistically organised activity? If so, should the designer be assisted in this activity -or should assistance be restricted to systematic aspects of the activity, at local levels or even only at the global level? If one wishes to assist the designer (also) in the opportunistic aspects of the activity, how might this be done?

This text is a post-print of Visser, W. (1994). Organisation of design activities: Opportunistic, with hierarchical episodes. *Interacting with Computers, 6*(3), 239-274 (Executive summary: 235-238).

In an evaluation study of a programming environment which supports a particular top-down approach (a retrospective strategy), but in which plan revision is made very tedious, Hoc (1988a) shows that professional programmers working with this system may generate non-optimal solutions when, in order to repair planning errors, they awkwardly modify modules at a very low level.

Lebahar (1989) observes that the use of CAD in architectural design modifies the actual activity by inducing a hierarchical organisation in the design solutions according to the "producible" objects, i.e. depending on the possibilities CAD makes available to its users.

Green, Gilmore, Blumenthal, Davies and Winder (1992) note that "a classic example of the problems of [object-oriented programming systems] during the design process is caused by the imposition of top-down design, such as the requirement to state the class hierarchy before the methods can be defined" (p. 18). The Smalltalk-80 environment is presented to illustrate these problems. Détienne (1990a) observes similar problems in her evaluation of another object-oriented environment, O2.

We judge that assistance tools should be compatible with the actual activity. If design is opportunistically organised, a support system which supposes -and therefore imposes- a hierarchically structured design process will at the very least constrain designers and will probably even handicap them (see Visser and Hoc, 1990).

Most existing tools are, however, backed up by prescriptive or analytical, task-experience based models of design, i.e. not by data on the actual activity to be supported, and they are limited to certain tasks during the "detailed design" stage, e.g., to the "editing" (coding) task in text or program production, or to "drawing" in many engineering tasks; they do not support "conceptual" design.

In a paper focusing on the planning component in the production of text (not "editing" but "writing"), Bisseret (1987/1990) distinguishes between the plan of the product (the final text) which is hierarchical, and the plan of the activity (in our paper referred to as its "organisation") which is opportunistic. He notices that most planning tools are plan editors which are based on a strictly hierarchical top-down model, and which, consequently, do not really

support the planning process. He presents some research on tools which would not impose such a hierarchical planning structure.

Hoc (1988b) makes similar remarks with respect to assistance tools for software engineering, which are mostly program editors. The problem, however, is not only one of developing an appropriate tool for supporting the use of a given language; it is also -or rather- a problem of the programming language as an appropriate tool. Green (1990) notices that "unfortunately for [designers], not all devices make it possible [to use some version of opportunistic planning]. Some programming languages, for instance, pretty well forbid it. So - what do people do? Do they build their programs top-down, as they are instructed? No. They abandon the programming language instead. They use informal languages which allow them to explore ideas. They develop their thoughts opportunistically, and then they rationalise and rebuild those thoughts in a coherent, well-structured way." (p. 2)

The empirical studies presented in this paper -and others which may be found in the literature- might lead to several suggestions for systems which offer "real" support. Such systems should

- assist organisational activities:
  - not impose a hierarchical planning structure, but allow designers, if they are following a plan, to deviate from, and even abandon, this plan;
  - support plan resumption, but not presuppose, and still less impose, it;
- assist representational activities:
  - allow designers to work at different levels of representation (different levels of detail, of abstraction, of design decomposition);
  - make available multiple visual pictorial and non-pictorial, more symbolic or verbal representations (see Guindon, 1992), adapted to the designers according to their level of expertise, and adapted to the different viewpoints they may take on the design problem/solution state;
  - allow easy changes of representation mode (of problems and solutions), and support switching between representation levels and viewpoints;
- assist the management of memory limitations, offering displays and facilities for:
  - presentation of intra- or inter-level information in parallel;

- presentation of constraints on the solution order;
- maintaining a trace of abandoned, interrupted or postponed tasks which may require backtracking;

• assist the re-use of designs:

- support analogical reasoning, in particular access to analogs and other candidates for re-use (see also Falzon and Visser, 1989; Visser and Trousse, 1993).

Related, more detailed and slightly different requirements for design assistance tools may be found in other papers (see Ullman, Staufer and Dietterich, 1987; Whitefield, 1986). The recent research on "design rationale" may also provide elements, even if this approach is not in particular focusing on organisational aspects of the design activity. Design rationale "articulates and represents the reasons and the reasoning process behind the design and specification of artifacts" (Carroll and Moran, 1991), thus it may supply data required, e.g. to assist representational activities, or even to support plan resumption.

Prototypes and experimental systems implementing some of the requirements presented are, indeed, under development (see Green et al., 1992; Guindon, 1992). However, no commercial tools integrating these elements are available, and such tools are the ones which are used in industry, i.e. in the "real" design which has been central in the present paper.

REFERENCES

Adelson, B., Littman, D., Ehrlich, K., Black, J., and Soloway, E.; 1985; Novice-expert differences in software design; Human-computer interaction - INTERACT '84; B. Shackel (Ed.); Amsterdam: North-Holland.

Adelson, B. & Soloway, E.; 1985; The role of domain experience in software design; IEEE Transactions on Software Engineering; (SE-11); 1351-1360.

Adelson, B., & Soloway, E.; 1988; A model of software design; The nature of expertise; M. T. H. Chi, R. Glaser & M. J. Farr (Eds.); Hillsdale, N.J.: Erlbaum.

Allen, J., Hendler, J., & Tate, A.; 1990; Readings in planning; San Mateo, CA: Morgan Kaufmann.

Anderson, J. R.; 1986; Knowledge compilation: the general learning mechanism; Machine learning. An artificial intelligence approach (Vol. II); R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.); Los Altos, Calif.: Morgan Kaufmann Publishers.

Bisseret, A.; 1987/1990; Towards computer-aided text production; Cognitive ergonomics. Understanding, learning and designing human-computer interaction; P. Falzon (Ed.); London: Academic Press; (Reprinted from Research report N° 665; Rocquencourt: INRIA)

Bisseret, A., Figeac-Létang, C., & Falzon, P.; 1988; Modeling opportunistic reasonings: the cognitive activity of traffic signal setting technicians; (Research Report N° 898); Rocquencourt: INRIA.

Brown, D. C., & Chandrasekaran, B.; 1989; Design problem solving. Knowledge structures and control strategies; London: Pitman.

Byrne, R.; 1977; Planning meals: Problem-solving on a real data-base; Cognition; (5); 287-332.

Carroll, J. M., Thomas, J. C., Miller, L. A. & Friedman, H. P.; 1980; Aspects of structure in design problem solving; American Journal of Psychology; (93); 269-284.

Carroll, J. M., & Moran, T. P.; 1991; Introduction to this special issue on design rationale; Human-Computer Interaction; (6); 197-200.

Chapman, D.; Planning for conjunctive goals; Artificial Intelligence; 1987; (32); 333-377.

Darke, J.; 1979/1984; The primary generator and the design process; Developments in design methodology; N. Cross (Ed.); Chichester: Wiley; (Originally published in Design Studies; (1); 36-44)

Davies, S. P.; 1989; Skill levels and strategic differences in plan comprehension and implementation in programming; People and computers V; A. Sutcliffe & L. Macaulay, L. (Eds.); Cambridge: Cambridge University Press.

Davies, S. P.; 1991; Characterizing the program design activity: neither strictly top-down nor globally opportunistic; Behaviour & Information Technology; (10); 173-190.

Détienne, F.; 1990a; Un exemple d'évaluation ergonomique d'un système de programmation orienté-objet, le système O2; Actes d'ERGO-IA '90, Biarritz, 19-21 septembre 1990.

Détienne, F.; 1990b; Expert programming knowledge: a schema-based approach; Psychology of programming; J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.); London: Academic Press.

Détienne, F.; 1991; Reasoning from a schema and from an analog in software code reuse; Fourth Workshop on Empirical studies of programmers, New Brunswick, N.J., December 6-8, 1991.

Falzon, P., & Visser, W.; 1989; Variations in expertise: implications for the design of assistance systems; Designing and using human-computer interfaces and knowledge based systems; G. Salvendy & M. Smith (Eds.); Amsterdam: Elsevier. Also available at http://hal.inria.fr/hal-00643770/fr/

Gilmore, D. J.; 1990; Expert programming knowledge: a strategic approach; Psychology of programming; J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.); London: Academic Press.

Glaser, R.; 1986; On the nature of expertise; Human memory and cognitive performances; F. Klix & H. Hagendorf (Eds.); Amsterdam: North-Holland.

Glaser, R., & Chi, M. T. H.; 1988; Overview; The nature of expertise; M. T. H. Chi, R. Glaser & M. J. Farr (Eds.); Hillsdale, N.J.: Laurence Erlbaum Associates.

Green, T.; 1980; Programming as a cognitive activity; Human interaction with computers; H. T. Smith & T. R. G. Green (Eds.); London: Academic Press.

Green, T.; 1990; Models of structure and models of perceiving structure (draft version); Proceedings of ECCE-5, Fifth European Conference of Cognitive ergonomics, Urbino (Italy), September 3-6, 1990.

Green, T. R. G., Gilmore, D. J., Blumenthal, B. B., Davies, S., & Winder, R.; 1992; Towards a cognitive browser for OOPS; International Journal of Human-Computer Interaction; (4); 1-34.

Guindon, R.; 1990; Designing the design process: exploiting opportunistic thoughts; Human-Computer Interaction; (5); 305-344.

Guindon, R.; 1992; Requirements and design of DesignVision, an object-oriented graphical interface to an intelligent software design assistant; CHI'92; 499-506.

Guindon, R., Curtis, B., & Krasner, H.; 1987; A model of cognitive processes in software design: An analysis of breakdowns in early design activities by individuals (MCC Technical Report N° STP-283-87); Austin, TX: MCC.

Guindon, R., Krasner, H., & Curtis, B.; 1987; Breakdowns and processes during the early activities of software design by professionals; Empirical Studies of Programmers: Second Workshop; G. Olson, S. Sheppard & E. Soloway (Eds.); Norwood, N.J.: Ablex.

Hayes-Roth, B., & Hayes-Roth, F.; 1979; A cognitive model of planning; Cognitive Science; (3); 275-310.

Hayes-Roth, B., & Hayes-Roth, F., Rosenschein, S., & Cammarata, S.; 1979; Modeling planning as an incremental, opportunistic process; Proceedings of the 6th International Joint Conference on Artificial Intelligence, Tokyo, 20-08-1979.

Hoc, J. M.; 1988a; Towards effective computer aids to planning in computer programming. Theoretical concern and empirical evidence drawn from assessment of a prototype; Working with computers: theory versus outcomes; G. C. van der Veer, T. R. G. Green, J. M. Hoc, and D. Murray (Eds.); London: Academic Press.

Hoc, J.M.; 1988b; Cognitive psychology of planning; London: Academic Press.

Holyoak, K. J.; 1991; Symbolic connectionnism: Toward third-generation theories of expertise; Toward a general theory of expertise: Prospects and limits; K. A. Ericsson & J. Smith (Eds.); Cambridge, MA.: Cambridge University Press.

Jeffries, R., Turner, A. A., Polson, P. G., & Atwood, M. E.; 1981; The processes involved in designing software; Cognitive skills and their acquisition; J.R. Anderson (Ed.); Hillsdale, N.J.: Erlbaum.

Kant, E.; 1985; Understanding and automating algorithm design; IEEE Transactions on Software Engineering; (SE-11); 1361-1374.

Lebahar, J.C.; 1989; La dialectique "compétence humaine/compétence artificielle" dans les processus de conception aidée par ordinateur; Contributions écrites - Premières journées de Psychologie du travail, Ergonomie et Psychopathologie du travail. PIRTTEM - CNRS, Paris, 13 et 14 juin 1989.

Lenat, D. B., Hayes-Roth, F., & Klahr, P.; 1979; Cognitive economy (Working paper HPP-79-15); Stanford: Stanford University, Computer Science Department, Heuristic Programming Project.

Leplat, J., & Hoc, J.M.; 1983; Tâche et activité dans l'analyse psychologique des situations; Cahiers de Psychologie Cognitive; (3); 49-63.

Malhotra, A., Thomas, J. C., Carroll, J. M. & Miller, L. A.; 1980; Cognitive processes in design; International Journal of Man-Machine Studies; (12); 119-140.

Mayer, R. E.; 1989; Human nonadversary problem solving; Human and machine problem solving; K. J. Gilhooly (Ed.); New York: Plenum.

Miller, G. A., Galanter, E., & Pribram, K. H.; 1960; Plans and the structure of behavior; London: Holt, Rinehart & Winston.

Morais, A.; 1987; Acquisition d'un outil de spécification (GRAFCET) pour décrire un procédé automatisé (Rapport de Recherche INRIA n°631); Rocquencourt: INRIA.

Navinchandra, D.; 1991; Exploration and innovation in design. Towards a computational model; New York: Springer.

Olson, G., Sheppard, S., and Soloway, E. (Eds.); 1987; Empirical Studies of Programmers: Second Workshop; Norwood, N.J.: Ablex.

Ormerod, T.; 1990; Human cognition and programming; Psychology of programming; J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.); London: Academic Press.

Pennington, N. & Grabowski, B.; 1990; The tasks of programming; Psychology of programming; J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.); London: Academic Press.

Reitman, W.; 1964; Heuristic decision procedures, open constraints, and the structure of ill-defined problems; Human judgments and optimality; M. W. Shelley & G. L. Bryan (Eds.); New York: Wiley.

Rist, R. S.; 1990; Variability in program design: the interaction of process with knowledge; International Journal of Man-Machine Studies; (33); 305-322.

Rist, R.; 1991; Models of routine and non-routine design in the domain of programming; Artificial Intelligence in Design (Preprints of the Workshop of the Twelfth International Joint Conference on Artificial Intelligence, Sydney, Australia, 25 August 1991); J. S. Gero & F. Sudweeks (Eds.); Sydney: University of Sydney.

Sacerdoti, E.; 1974; Planning in a hierarchy of abstraction spaces; Artificial Intelligence; (5); 115-135.

Suchman, L. A.; 1987/1990; Plans and situated actions; Cambridge: Cambridge University Press; (Originally published)

Ullman, D., Dietterich, T. G., & Staufer, L. A.; 1988; A model of the mechanical design process based on empirical data; AI EDAM; (2); 33-52.

Ullman, D., Staufer, L. A., & Dietterich, T. G.; 1987; Toward expert CAD; Computers in Mechanical Engineering; (6, 3); 56-70.

Visser, W.; 1987; Strategies in programming programmable controllers: a field study on a professional programmer; Empirical Studies of Programmers: Second Workshop; G. Olson, S. Sheppard & E. Soloway (Eds.); Norwood, N.J.: Ablex. Also available at http://hal.inria.fr/hal-00641376/fr/

Visser, W.; 1988a; L'activité de comparaison de représentations dans la mise au point de programmes; Le Travail Humain, Numéro Spécial "Psychologie ergonomique de la programmation informatique"; (51); 351-362. Also accessible at http://hal.inria.fr/hal-00642569/fr/

Visser, W.; 1988b; Giving up a hierarchical plan in a design activity (Research Report N° 814); Rocquencourt: INRIA. Also accessible at http://hal.inria.fr/inria-00075737/fr/

Visser, W.; 1990; More or less following a plan during design: opportunistic deviations in specification; International Journal of Man-Machine Studies; (33); 247-278. Also accessible at http://hal.inria.fr/inria-00633544/fr/

Visser, W.; 1990; Evocation and elaboration of solutions: Different types of problem-solving actions. An empirical study on the design of an aerospace artifact; COGNITIVA 90. At the crossroads of Artificial Intelligence, Cognitive science, and Neuroscience. Proceedings of the third COGNITIVA symposium; T. Kohonen & F. Fogelman-Soulié (Eds.); Amsterdam: Elsevier. Also accessible at http://hal.inria.fr/inria-00000165/fr/

Visser, W.; 1992a; Designers' activities examined at three levels: organization, strategies & problem-solving; Knowledge-Based Systems; (5, 1); 92-104. Also accessible at http://docs.google.com/fileview?id=0B7OqbBIc3_7bY2Y2ZmRlZDktMDFiZS00YjNlLWI2YWMtZTI3Zjc5ZGY5Mzlm&hl=en

Visser, W.; 1992b; Design organization: There is more to expert knowledge than is dreamed of in the planner's philosophy (Rapport de Recherche N° 1765); Rocquencourt: INRIA. Also accessible at http://hal.inria.fr/inria-00077005/fr/

Visser, W., & Trousse, B.; 1993; Reuse of designs: desperately seeking an interdisciplinary approach; Introduction; Proceedings of the Workshop of the Thirteenth International Joint Conference on Artificial Intelligence "Reuse of designs: an interdisciplinary cognitive approach", Chambéry (France), August 29, 1993; W. Visser (Ed.); Rocquencourt: INRIA. Also accessible at http://hal.inria.fr/inria-00117275/fr/

Visser, W., & Falzon, P.; 1992; Catégorisation et types d'expertise. Une étude empirique dans le domaine de la conception industrielle; INTELLECTICA; (n° 15); 27-53. Also accessible at www.intellectica.org/archives/n15/15_04_Visser-Falzon.pdf

Visser, W., & Hoc, J.M.; 1990; Expert software design strategies; Psychology of programming; J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.); London: Academic Press. This chapter is also accessible in Dr Alan

Blackwell's Course material 2010–11 for "Usability of Programming Languages" http://www.cl.cam.ac.uk/teaching/1011/R201/ with the following "Note on copyright: This book is currently not available in print. A selection of chapters has been provided, with the permission of the editors, for non-commercial educational and research use only. Members of the Psychology of Programming Interest Group (PPIG) continue to conduct research as outlined here, and it is intended that specific chapters will be replaced with updated versions when individual authors are able to provide them. Volunteer assistance with OCR of these chapters, to assist those authors who no longer have their original manuscripts, would be appreciated."

Voss, J. F., Greene, T. R., Post, T. A., & Penner, B. C.; 1983; Problem-solving skill in the social sciences; The psychology of learning and motivation (Vol. 17); G. Bower (Ed.); New York: Academic Press.

Voss, J. F., & Post, T. A.; 1988; On the solving of ill-structured problems; The nature of expertise; M. T. H. Chi, R. Glaser & M. J. Farr (Eds.); Hillsdale, N.J.: Laurence Erlbaum Associates.

Whitefield, A.; 1986; An analysis and comparison of knowledge use in designing with and without CAD; Knowledge engineering and computer modelling in CAD. Proceedings of CAD86. Seventh International Conference on the Computer as a Design Tool; A. Smith (Ed.); London: Butterworths.