ELSEVIER

Interacting with Computers 11 (1999) 237–254

Interacting with Computers

Commentary

Wanted: psychologically relevant, device- and event-independent work analysis techniques

Kim J. Vicente*

Cognitive Engineering Laboratory, Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, Canada

Received December 1997; received in revised form July 1998; accepted August 1998

Abstract

This article offers a commentary on Richardson, Ormerod, and Shepherd (in press) while building on the previous discussion in this journal of the relative merits of task analysis and systems analysis in human–computer interface design [1,2,7]. The SGT scheme described by Richardson et al. represents a valuable contribution to the work analyst's toolkit. However, it is limited in the extent to which it can identify the information requirements associated with unanticipated events. The abstraction hierarchy [23] is an event-independent work domain analysis technique that can be used to overcome this limitation while still satisfying the criteria of device-independence and psychological relevance. Future research should integrate the complementary advantages of SGT and the abstraction hierarchy into a single, unified framework for work analysis.

Keywords: Task analysis; Systems analysis; Human–computer interaction; Cognitive engineering; Process control

1. Glossary

This nomenclature is a subset of the one created by Vicente (in press) to contribute some conceptual clarity and discipline to the work analysis literature (cf. [6]). All terms that are defined in the Glossary have been capitalized.

- *Action* — the goal-directed behavior of an Actor. The object of actions can be another Actor or the Work Domain itself.
- *Actor* — a Worker or Automation.
- *Affordance* - a goal-relevant description of the world (e.g. Work Domain) that describes an opportunity for Action defined with respect to the capabilities of a particular Actor [11].

* *E-mail address:* benfica@mie.utoronto.ca (K.J. Vicente) URL: www.mie.utoronto.ca/labs/cel/

0953-5438/99/$ - see front matter
PII: S0953-5438(98)00067-8

- *Automation* — a mechanical, electrical, or computerized Actor. Automation acts on other Automation or on the Work Domain.
- *Behavior* — the dynamic state of a System, whether it be the System being acted on (Work Domain) or the System doing the acting (Worker or Automation). Compare with Action.
- *Closed System* — a System that does not interact with its environment (i.e. a System without Disturbances). Compare with Open System.
- *Coherence-Driven Work Domain* — a Work Domain that does not impose dynamic, environmental constraints on the goal-directed Behavior of Actors. Compare with Correspondence-Driven Work Domain.
- *Controls* — the part of an Interface that is used by Workers to act on Automation or on the Work Domain.
- *Correspondence-Driven Work Domain* — a Work Domain that imposes dynamic, environmental constraints on the goal-directed Behavior of Actors. Compare with Coherence-Driven Work Domain.
- *Device* — the computer-based information system available to Workers, including both the Interface and Automation.
- *Device-Dependent* — an adjective describing Work Analysis techniques whose results are contingent upon a particular Device, or class of Devices.
- *Device-Independent* — an adjective describing Work Analysis techniques whose results are not contingent upon a particular Device, or class of Devices.
- *Displays* — the part of an Interface that is used by Workers to obtain information about a Sociotechnical System.
- *Disturbances* — factors that have not been anticipated by analysts but that can affect the state of a System.
- *Event* — a happening or occurrence.
- *Event-Dependent* — an adjective describing Work Analysis techniques that are contingent upon a finite class of initiating Events.
- *Event-Independent* — an adjective describing Work Analysis techniques that are not contingent upon a finite class of initiating Events.
- *Function* — a goal-relevant structural property of a Work Domain. An Affordance that is relevant to the Purposes for which the Work Domain was designed. Affordances that are relevant for other Purposes are not considered Functions.
- *Goals* — the states to be achieved, or maintained, by an Actor. Note that Goals are attributes of Actors, not Work Domains, and that they are dynamic (unlike the Purposes for which a Work Domain is built, which are relatively permanent). Compare with Purposes.
- *Interface* — the computer-based means by which Workers obtain information about, and control the state of, a Sociotechnical System. Composed of Displays and Controls.
- *Open System* — a System that interacts with its environment (i.e. a System with Disturbances). Compare with Closed System.
- *Purposes* — the overarching intentions that a Work Domain was designed to achieve. Note that Purposes are properties of Work Domains, not Actors, and that they are relatively permanent (unlike the Goals of Actors, which change over time). Compare with Goal.

- *Sociotechnical System* — a System composed of technical, psychological, and social elements.
- *System* — a set of inter-related elements that share a common purpose. The boundary defining the System is specified by the Analyst, and is not inherent in the System. Several equally meaningful boundaries can be drawn, although some may be more useful than others for certain purposes. The System can be an Actor (i.e. Worker or Automation), the thing being acted on (i.e. a Work Domain), or both (i.e. a Sociotechnical System).
- *Task* — a sequence of Actions that can or should be performed to achieve a Goal.
- *Task Analysis* — a form of Work Analysis that specifies what Goals should be achieved, or also how they should be achieved. Compare with Work Domain Analysis.
- *Work Analysis* — any technique for analyzing human work. Exemplars include Task Analysis and Work Domain Analysis.
- *Work Domain* — the System being controlled, independent of any Worker, Automation, Event, Task, or Device.
- *Work Domain Analysis* — a form of Work Analysis that identifies the functional structure of the Work Domain. Compare with Task Analysis.
- *Workers* — the people who participate in and act in a Sociotechnical System. Synonymous with operators and users.

2. Introduction

This article contributes to the thoughtful and constructive discussion that has been taking place in the pages of this journal regarding the value of task analysis and systems analysis techniques in the design of interactive computer systems [1,2,7,27]. This topic is of critical importance to the field of human–computer interaction (HCI). Previous experience has shown that the value of any particular design is typically constrained by the thoughtfulness and thoroughness of the analysis that preceded it. If interfaces are to support users in their work, then the design of those interfaces must be based on an intimate understanding of that work context. This understanding can be systematically obtained using some form of *work analysis*, a term that will be used in this article to refer to any technique, whether it be task- or system-based, that can be used to analyze human work contexts (see the Glossary for a definition of this and other terms used in this article).

3. Conceptual groundwork

3.1. What is ‘‘The System’’?

Before reviewing the points that have already been made by participants in this ongoing discussion, it is important to lay down some conceptual foundations. The term system has appeared frequently in the aforementioned articles, but perhaps

surprisingly, it has not been explicitly defined by all of the authors. This is an important oversight because there are several candidate objects that can be viewed as ''the system'', including:

1. the combined worker–computer dyad;
2. the sociotechnical system comprising the worker–computer–work domain triad.

The lack of definition of ''system'' presents a problem because different theoretical and empirical claims make sense, depending upon the definition one is implicitly adopting. As we will see later, if one considers the worker–computer dyad to be the system, one may reach very different conclusions than someone who considers the worker–computer–work domain triad to be the system. At the very least, the object of analysis will change depending on the definition one adopts for the system, and as a result, the very concept of systems analysis will differ accordingly.

Although it is difficult to be sure, it seems that the various participants in this discussion have implicitly adopted different conceptions of the term ''system''. [1,2] seems to be using the term to refer to the worker–computer dyad, as evidenced by the following quotations:

- ''HCI focuses on the interaction between humans and computers'' ([2], p. 248)
- ''The object of the [data] model is the human–computer system'' ([2], p. 248).

Benyon then goes on to describe the ''human side'' and the ''computer side'' of this system. This dyadic view seems to be shared by Diaper and Addison [7], who refer to apparently analogous task-oriented and computer-oriented representations (p. 128). Richardson et al. [27], on the other hand, seem to be using the term system to refer to the worker–computer–work domain triad, with the work domain being a process plant in their case. At the risk of stating the obvious, note that the process plant is neither a worker nor a computer; it is a physical plant that has an inertia all its own based on the dynamic laws that govern its behaviour. Thus, Richardson et al. seem to have adopted a different definition of the term system than did Benyon and Diaper and Addison.

One would think that when these authors refer to systems analysis techniques, the ''system'' they refer to would correspond to that identified in these definitions. However, this does not seem to be the case. Instead, it seems that from the human–computer dyad perspective, systems analysis refers to an analysis of the data flow in the computer, whereas from the human–computer–work domain triad perspective, it refers to an analysis of the work domain. In both cases, systems analysis does not focus on workers or their tasks. But because a different perspective is being adopted in the two cases, the object of systems analysis differs accordingly.

Thus, we are faced with two difficulties. First, the term ''system'' is being used in two senses. One refers to the collective entities that are involved in HCI, and either consists of two objects (worker and computer) or three (worker, computer, and work domain). The other usage of ''system'' refers to the object of a systems analysis, and corresponds either to data flow in the computer or to the nature of the work domain outside of the computer. This problem could have been avoided by using the more differentiated terms ''data flow

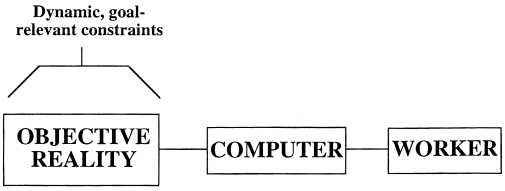

Fig. 1. Correspondence-driven work domains (adapted from [32]). See text for description.

analysis'' and ''work domain analysis'' instead of the vaguer term systems analysis. However, since the discussion so far has used the latter term, I will continue with that usage later when reviewing the authors' claims.

The second difficulty is that the object of systems analysis seems to differ for the various authors. Of course, there is nothing inherently wrong with this situation. It is well known that the choice of how to define system boundaries is an arbitrary one (e.g. [16]). There is no one correct definition. Rather, different definitions are useful for different purposes. Nevertheless, for the purposes of this discussion, the apparently differing definitions of the term systems analysis may get in the way of meaningful debate. If authors are indeed using this term in different ways, then some of their disagreements may simply be caused by the fact that the authors are talking about different entities. These differences and disagreements may arise because different authors are primarily concerned with different types of HCI problems.

3.2. Correspondence- vs. coherence-driven work domains

A very simplistic way to categorize HCI problems is by distinguishing between correspondence- and coherence-driven work domains [32]. These terms are borrowed from the philosophy literature (e.g. [19]) but nevertheless capture a distinction that has some practical utility. As shown in Fig. 1, *correspondence-driven* work domains have an external reality outside of the worker–computer dyad that imposes dynamic, environmental constraints on the goal-directed behavior of workers. For example, engineers working on a collaborative design project must take into account the intentions and actions of other engineers, otherwise their efforts will not be coordinated and the project goals will be compromised. Similarly, airline pilots must take into account the positions of other aircraft and the terrain if they are to achieve their goals. Also, the decisions made by a financial analyst must take into account the actual behavior of the stock market, otherwise the analyst's goals cannot possibly be consistently achieved. These are all examples of correspondence-driven domains because they have a physical or social reality that must be respected by any goal-directed actor working in the respective application domain, human or otherwise.

As shown in Fig. 2 coherence-driven work domains do not have any dynamic, environmental constraints that must be respected. A simple example is a person working with a word processor. Such work domains do not have a dynamic physical or social reality, outside of the worker and the computer, that must be taken into account by workers

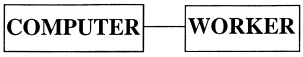

Fig. 2. Coherence-driven work domains (adapted from [32]). See text for description.

in order for goals to be achieved. In these cases, the focus of attention is solely on the worker–computer dyad without reference to an external environment.[1]

This distinction corresponds to the difference between working through computers and working with computers, respectively.[2] More importantly, the distinction can be used to understand the apparent differential use of the term ''system'' by the aforementioned authors. More specifically, Benyon [1,2] and Diaper and Addison [7] appear to consider the worker–computer dyad as the system, probably because they are primarily concerned with HCI problems in the realm of coherence-driven work domains. Thus, their focus seems to be on the interaction between humans and computers. In contrast, Richardson et al. [27] appear to consider the worker–computer–work domain triad as the system, probably because they are primarily concerned with HCI problems in the realm of correspondence-driven work domains. Like Richardson et al., my research is primarily in the area of process control (e.g. [33,36]). Thus, the problems with which I am most concerned fall squarely in the realm of correspondence-driven domains. Accordingly, my focus is on the interaction between humans, computers, and the work context. As a result, the arguments presented later are based on the frame of reference illustrated in Fig. 1, and may not be as useful or valid for other types of HCI problems.

In the remainder of this article, I consider the characteristics that work analysis techniques should have if they are to meet the challenges imposed by correspondence-driven work domains. As we will see, the fact that correspondence-driven work domains are open systems that are subject to unanticipated disturbances has important implications for work analysis techniques. These implications have not been considered in the ongoing discussion centred around task and system analysis methods.

4. A tale of three criteria

At least three criteria can be used to compare various forms of work analysis techniques, including both task- and system-based approaches.

1. *Device-independence* — To what extent is the analysis technique independent of the

[1] A case could be made that coherence-driven work domains do not really exist because such systems (e.g., word processing) are always embedded in a larger system that is correspondence-driven (e.g., engineering design). In this view, the distinction between coherence–driven and correspondence–driven work domains is an artifact of where an analyst decides to draw a boundary around the work domain. I am sympathetic to this view, but a proper defense of this claim is outside the scope of this article.

[2] It could be argued that the distinction between ''working through computers'' and ''working with computers'' is spurious because the latter is just a restricted view of the former. This may very well be true, but it does not change the fact that some researchers and designers focus primarily on people's interaction with computer systems (e.g. see any ACM SIGCHI Proceedings), whereas others focus primarily on people's interaction with the work context beyond the computer systems (e.g., Rasmussen et al., 1994; Vicente, in press).

characteristics of the existing device that is used to perform the work, and the current work practices induced by that device? Device-independent work analysis techniques may be preferable because they allow designers to divorce themselves from existing design sub-optimalities, thereby opening up new possibilities for device functionality.[3]

2. *Psychological relevance* — To what extent does the analysis technique result in a representation or model that is psychologically plausible? Psychologically relevant work analysis techniques are preferable because they produce representations that naturally support users' thought processes.
3. *Event-independence* — To what extent does the analysis technique help to identify information requirements that are independent of a particular class of pre-enumerated events, tasks, situations, or contexts? Event-independent work analysis techniques are useful because they provide users with support for a broader range of events, including those that are unfamiliar to workers and that have not been anticipated by designers ([37]).

These criteria can be used to summarize some of the major points that have already been made in this ongoing discussion. In addition, the criterion of event-independence opens up a new point that has not been made before.

4.1. Device-independence

The main point in Benyon's [1] article, which started off this discussion, was that task analysis techniques are device-dependent. As a result, they inherit the deficiencies exhibited by the interfaces that are currently being used by workers to do the job. As an alternative, Benyon advocated the use of systems analysis techniques (e.g. data flow diagrams), which he claims have the benefit of being device-independent. As a result, such techniques can lead to fundamental changes in device functionality, rather than just incremental changes to current practice.

It is not completely clear whether Benyon [1] originally intended his critique to apply to all task analysis techniques or just to some. On p. 102, he claims that systems analysis techniques are more effective than ''any'' task analysis techniques. On p. 103, he states that the deficiencies he will be discussing are inherent in ''many'' task analysis techniques. On p. 113, he states that it is ''impossible'' for task analysis techniques to be device-independent. Thus, there are two versions of Benyon's critique, a strong version applying to all task analysis techniques and a weaker version applying only to many. Fig. 3(a) provides a schematic representation of the strong form of the argument.

In my view, there can be little argument that at least some task analysis techniques suffer from the problems identified by Benyon. The dependence between current practice and existing technology has been acknowledged by other HCI researchers, perhaps most elegantly in Carroll et al. [5] task–artifact cycle. Thus, the weaker version of Benyon's critique seems to be relatively uncontroversial. The point of debate seems to centered on

[3] There are many situations where interface designers must work within the context of previous designs, and so opportunities for designing new functionality may be limited. In these cases, device-independence may not be desirable, but the resulting design may not be as good as it could be if it is inheriting deficiencies or limitations of the previous interface.

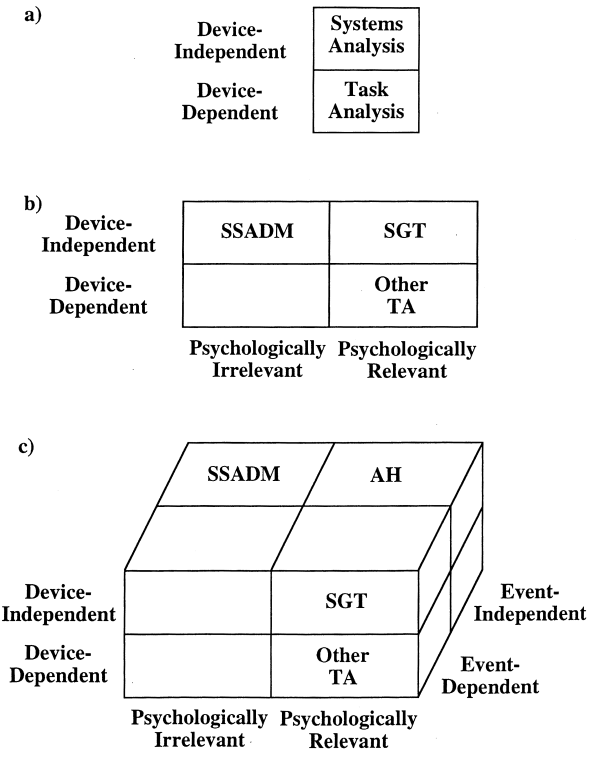

Fig. 3. (a) A graphical representation of Benyon's [1]main point, based on the criterion of device-independence. (b) A graphical representation of Richardson et al's [27] main point, based on the additional criterion of psychological relevance. c) A graphical representation of the main point of this commentary, based on the additional criterion of event-independence. (SGT: Sub-Goal Template; TA: Task Analysis; AH: Abstraction Hierarchy.)

the strong version of the argument — do all task analysis techniques suffer from device-dependence?

In their commentary on Benyon [1], Diaper and Addison [7] make a convincing case against the strong critique of task analysis. In my view, their primary point is that ''some aspects of sequence and thus task flow are properties of the work and quite independent of any particular device'' (p. 133). If so, then the door is opened up for more sophisticated forms of task analysis that capture only the constraints that are inherent in the task, independent of any particular device. In terms of Fig. 3(a), it seems to me that Diaper and Addison's point is that not all task analysis techniques belong in the lower cell, as Benyon claimed. At least some techniques belong in the upper cell in Fig. 3(a).

In his reply, Benyon [2] seems to back off of the strong version of his argument, stating: ''There are doubtless situations in which task analysis will be effective, but there are many situations in which it would appear to produce a limiting and overly device-dependent design'' (p. 258). However, the question of device-(in)dependence is not the only relevant criterion in comparing the utility of task- and systems-based approaches to work analysis.

4.2. Psychological relevance

In their recent article, Richardson et al. [27] make two points in response to Benyon's [1] original claims. First, they reinforce Diaper and Addison's [7] position by arguing that not all task analysis techniques suffer from device-dependence. In particular, they point out that the sub goal template (SGT) technique provides a way of conducting a device-independent task analysis. According to Richardson et al. [27] SGT achieves this objective by analyzing ''user/operator goals rather than an existing task implementation'' (p. 2). Thus, by identifying what needs to be done rather than how it is to be done with a particular device, device-independence may be achieved. Second, Richardson et al. [27] make a stronger point, namely that task analysis techniques can actually be superior to the systems analysis techniques advocated by Benyon. This advantage is based on the criterion of psychological relevance. Richardson et al. [27] support their claim by comparing an exemplar of task analysis (SGT) with an exemplar of systems analysis (SSADM) in the context of a detailed case study. This instructive example shows that basing a work analysis on worker goals is more psychologically relevant because it identifies information requirements that are not uncovered by a goal-neutral (i.e. psychologically less relevant) systems analysis.

These two points are summarized graphically in Fig. 3(b), albeit in an exaggerated bipolar form. Not only are some task analysis techniques, specifically SGT, device-independent, but they are also more psychologically relevant than systems analysis techniques such as SSADM. Thus, Richardson et al. [27] turn around Benyon's [1] claims: some task analysis techniques are actually *superior* to systems analysis techniques.

In my view, the additional claim about psychological relevance is well taken. Any representation of work must be psychologically plausible, otherwise it will lead to interfaces that are difficult for workers to comprehend, and therefore, use. If there is one thing that we know about human behaviour, it is that people are highly goal-directed. The goal-relevant description provided by SGT seems to be more psychologically relevant than the goal-neutral, logical descriptions produced by SSADM.

Benyon [1,2] does not seem to appreciate this point, as evidenced by his data-centered view:

- ''Data provides a common currency between human and machine processing'' ([1], p. 119).
- ''Data is the most important and probably the only thing which humans have in common with computers'' ([2], p. 256).

There is no inherent meaning in data, in the absence of some goal or purpose. Moreover, there is no reason why workers and computers cannot share goals and purposes. In fact, one could argue, as do Richardson et al. [27], that unless goals and purposes are shared, interaction will not be as fluent as it could otherwise be.

Benyon [2] counters that this task-based approach is problematic because ''a complex relationship exists between data, users and tasks'' (p. 253). What he does not point out, however, is that this complex relationship is inescapable. If the analyst and designer do not confront it and provide support for it in the interface, then workers will be left to deal with

this complex relationship unaided. In short, goal-sensitivity is a fact of life that can be ignored but not escaped, and those who most ignore, least escape.

4.3. Event-independence

Although the SGT approach advocated by Richardson et al. [27] has a great deal of merit and can probably be productively applied in many contexts, there are some important situations for which it is not very well suited. These limitations are centred around the criterion of event-independence, a factor that had not received any attention in this ongoing debate. This problem was not identified by Richardson et al. [27] nor by Shepherd [30] in an earlier discussion of SGT, so I will devote the remainder of this section to explaining it.

4.3.1. The event-dependence of SGT

All task analysis techniques, including SGT, are inherently event-dependent (i.e. they only apply to the pre-enumerated class of events anticipated by analysts). To see why this is the case, we can use Umbers and Reiersen's [31] application of task analysis to the design of a safety system in the nuclear industry, the prototypical correspondence-driven work domain. The starting point for any task analysis is the (implicit or explicit) specification of a class of initiating events. This step is unavoidable for the simple reason that if we do not know what the initiating event is, we cannot determine what the task should be. After all, different initiating events can impose very different demands on workers ([31], p. 445). The typical response to this problem is to make a set of assumptions about the likely state of the work domain for particular classes of events ([31], p. 448). But of course, the validity and utility of the task analysis is only as good as the validity of those assumptions: ''unrealistic or inaccurate assumptions can invalidate the results'' ([31], p. 448).

As pointed out earlier, correspondence-driven work domains are complex, open systems which means that they are subject to disturbances that will not be anticipated by designers. As a result, it is impossible to analyze all of the initiating events because they are essentially infinite in number. No matter how many events analysts postulate, there will always be other unanticipated events that can occur. Further, it is precisely these unanticipated events that pose the greatest threat to system safety [36]. The history of industrial accidents is filled with such examples [15,22,25]. Large-scale disasters occur in situations that are unfamiliar to workers and that have not been anticipated by system designers. Because task analyses are based on a set of initiating events, by definition they are unable to cope with unanticipated events. For example, one investigation from the nuclear industry [14] found that 100% of the abnormal events surveyed that were not handled successfully by workers consisted of unanticipated events for which there was no procedure or for which the existing procedure had to be modified. One might think that completeness could be achieved by prescribing a default set of actions that operators should take in any unanticipated situation. However, the variety exhibited by unanticipated events strongly suggests that this approach has its limits. Each unanticipated event has a unique set of instigating conditions, and thus demands a different set of compensatory actions.

It is clear from the description provided by Richardson et al. [27] that SGT suffers from these limitations. For example, they state that in order to perform such an analysis ''the

exact nature of operators' tasks must be known'' (p. 12), and that ''all that is required is to know what the operators' tasks are and the order in which these are carried out'' (p. 15). The evidence cited above shows that this endeavour is bound to be incomplete in correspondence-driven work domains. In short, because it is a form of task analysis, SGT cannot — by definition — deliver a work domain analysis because of its event-dependence (see Glossary). In fact, the limitations of SGT with respect to unanticipated events were recognized by Shepherd [30]: ''in cases where independent decision making from operators is sought, it is difficult to anticipate the information that would suffice'' (p. 1433).

The limited robustness of SGT to disturbances such as variability in initial conditions may not be limited to potentially catastrophic events. For instance, at one point Richardson et al. [27] stated that SGT focuses ''on the goals that the operator must achieve rather than the actions used to achieve them'' (p. 16). At the same time, however, they state that ''Specifying an idealised procedure for each of the operator's tasks using the SGT scheme becomes part of the design process'' (pp. 16–17). There seems to be a contradiction here. Given that goals are hierarchically decomposed into increasingly detailed layers of subgoals using SGT and that plans put those subgoals into some temporal order, it is not clear how such a task representation is any different from a procedure consisting of a sequence of actions. At what point is a subgoal different from an action? Moreover, it is also not clear that the plans in SGT accommodate the variability that is typically encountered in correspondence-driven work domains. For example, Norros [20] provides several examples showing that workers must frequently achieve the same goals in different ways (i.e. using different plans) as a function of the current context. Since there can be a great deal of variability in initial conditions, workers must exhibit an equal amount of adaptive variability to achieve a particular goal [34]. It is not clear whether the ''a priori logical analysis of the order in which subgoals must be achieved in order to satisfy the overall goal'' ([27], p. 14) can accommodate this requisite variability.

A good example of this point is provided by Shepherd [29] in an application of task analysis to maintenance training for mechanical fitters in a chemical company: ''A major problem that had to be dealt with at the outset was that fitters, like most craftsmen, are supposed to do anything they are called upon to do: there is apparently no one task to analyze'' (p. 328). One way to deal with this dilemma is to specify the task goal at a high (i.e. vague) level. For example, Shepherd [29] identified one task goal as requiring workers to ''Note any problems'' (p. 335). This tactic tries to overcome ignorance about the precise goal to be pursued by specifying a very broad goal that subsumes all of the more specific goals that might be associated with unanticipated events. The problem with this approach is that it merely provides a place holder for what workers are supposed to do. It does little to identify the particular information or knowledge that workers require to cope with the novelty imposed by unanticipated events.

The conclusion of this critique is not that techniques such as SGT are not useful, but rather that other work analysis techniques are required to make up for their limitations. But is there an alternative? Some researchers believe that there is not. For example, Shepherd [30] stated: ''By definition, if a problem prompting a decision-making task is genuinely novel, there can be no confident provision of the information necessary to solve it'' (p. 1433; see also [18]). This conclusion is absolutely correct, if we limit ourselves to

task-based approaches. However, it overlooks the fact that there are relationships and information that are potentially relevant independent of any particular event. These invariants can be found by analyzing, not operator tasks, but the functional structure of the work domain itself. In other words, rather than model what workers should do, we can instead model the action possibilities associated with the work domain.

4.3.2. Analytical redundancy as a path to event-independence

The logic behind this model-based approach to dealing with unanticipated events is very similar to analytical redundancy techniques developed in control theory [10]. These techniques derive their power from the fact that work domain constraints play a crucial role in disturbance management. For instance, when a process plant is functioning as intended, the variables describing the state of the plant are inter-related by a number of constraints (see the example presented later). These constraints result from the specific structure and purpose of the plant, as well as the physical laws that govern it. When a disturbance occurs, one or more of these constraints are violated. That is, the process constraints provide a referent for defining expected plant behavior, and deviations from that referent can signify a disturbance. Thus, the residual from the analytical redundancy calculations provides an indication of whether the process may be in a normal or faulty state.

Note that this approach is event-independent. Regardless of the initiating event, the alternatives that are available for action are limited by the functional structure of the work domain. Workers can only use components that are available, and components can only be used for functions that they are capable of satisfying. By identifying these functional constraints on worker action, we can determine what information workers may need to deal with unanticipated situations where they have to adapt and improvise a solution. Note that the fundamental question is one of control requirements — what information is needed, given the demands of the problem? Questions about usability, while certainly very important, are subservient because effective control cannot be achieved with a usable interface that does not have the information that is required to do the job [34]. Note also that, in this view, the goal of interface design is to reveal how the work domain actually functions. If the resulting design is not consistent with the mental model of the operator, then it is the operator's mental model that needs to be changed, not the interface, for the work domain is indifferent to what the operator believes. The laws of physics are unforgiving.

The suggestion that work domain analysis might be a useful means of dealing with unanticipated events may be suspect, given Richardson et al. [27] critique of systems analysis techniques based on the criterion of psychological relevance [see Fig. 3(b)]. However, as Benyon [1] pointed out, there are many different examples of systems analysis techniques. While techniques such as SSADM may have limited psychological relevance, it is possible that there are other systems analysis techniques that are not subject to such limitations. This point has not been made explicit in this ongoing discussion. For example, Richardson et al. [27] state that systems analysis techniques focus on ''specifying the logical structure of the system to be built'' (pp. 6–7). But as it turns out, there is at least one systems analysis technique — or better, work domain analysis technique — that focuses on identifying the functional structure of the work domain, rather than its logical

Table 1
Examples of interfaces that were designed according to a systems analysis conducted with the abstraction hierarchy

Reference	Application domain
Burns and Vicente [4]	Conventional power plant simulation
Dinadis and Vicente [9]	Power plant feedwater subsystem
Dinadis and Vicente [8]	Aircraft engines and fuel systems
Itoh et al. [12]	Full-scope nuclear power plant simulator
Jamieson [13]	Petrochemical process unit
Pejtersen [21]	Fiction library information retrieval
Reising and Sanderson [26]	Pasteurizing plant simulation
Sharp [28]	Neonatal intensive care
Vicente and Rasmussen [35]	Thermal-hydraulic process control microworld
Xu [38]	Hypertext information retrieval

structure. That technique is Rasmussen's [23] abstraction hierarchy (AH), an approach to work domain analysis that was developed outside of the software engineering community.

4.3.3. The abstraction hierarchy

The basic insight behind the AH is very similar to that which led Gibson [11] to formulate his concept of *affordances*, namely that it is possible to describe the world (e.g. the natural environment, a process plant) in a way that makes reference to action capabilities. For example, rather than describing a process plant in context-free engineering terms, it is possible to describe it from a functional perspective based on the purposes for which the plant was designed. The advantage of such a functional description is that it supports goal-directed problem solving. There is a great deal of empirical evidence to indicate that people reason effectively and naturally using an AH representation in domains that require practical problem solving (see [23,33,36] for reviews). Thus, like SGT but unlike some other systems analysis techniques, the AH framework satisfies the criterion of psychological relevance.

However, unlike SGT, the AH also satisfies the all important criterion of event-independence. This point can be appreciated by noting that an analysis of the functionality of the work domain is *not* equivalent to an analysis of its failed functionality. Faults and abnormalities cannot be found in an AH representation because no events of any kind are represented (see the example below). Because it is based on the logic of analytical redundancy described above, the AH is event-independent. Unlike SGT and other task analysis techniques, the object of representation is the work domain being controlled by workers, not the worker's goals. In short, the AH is a psychologically relevant work domain analysis technique that is device-independent because it makes no reference to an existing interface [1] and that is also event-independent because it is not explicitly tied to a particular class of events. As such, it provides a representation of the work domain that workers can use to reason during unanticipated situations. These insights are represented graphically in Fig. 3(c).

4.3.4. Design implications

To some, it is not clear how the AH delivers a set of information requirements that designers can use in a tangible way. In part, this reaction is due to the fact that the rationale behind the AH has yet to be comprehensively presented in a pedagogical form.[4] In part, the reaction is also due to the fact that examples showing the practical value of the AH have only been developed relatively recently. Table 1 lists a number of examples of interfaces that were designed based on the requirements identified by an AH systems analysis. Note that the list includes application domains that are technical (e.g. process control) as well as humanistic (e.g. fiction retrieval), continuous (e.g. nuclear power) as well as discrete (e.g. hypertext), and primarily steady state (e.g. power plant) as well as primarily dynamic (e.g. neonatal intensive care). These examples show that the AH can indeed provide designers with practical insights that can be turned into concrete design products for a wide variety of application domains.

Unfortunately, a detailed example showing how the AH can be used to identify information requirements is beyond the scope of this commentary. The interested reader can consult the references in Table 1, the tutorial provided by Bisantz and Vicente [3], and detailed theoretical descriptions can be found in several other sources ([23,24,36]). However, we can provide an example to illustrate the key property of the AH, namely its event-independence.

Consider a very simple process control system consisting of one reservoir holding a volume of water (Vol), an input valve (V_{IN}) that regulates the flow into the reservoir, and an output valve (V_{OUT}) that regulates the flow out of the reservoir (see [34]). One of the constraints that governs such a process is the law of conservation of mass. If the process is operating normally, we would thereby expect that the rate of change of volume in the reservoir is determined by the difference between the flowrate of mass going through V_{IN} and that going through V_{OUT}. This constraint can be stated more formally as follows:

$$\frac{dVol(t)}{dt} = \frac{MI(t) - MO(t)}{\rho} \tag{1}$$

where Vol(t) = reservoir volume or level; MI(t) = mass input flowrate; MO(t) = mass output flowrate; and ρ = density.

Now consider what kinds of disturbances could affect this very simple work domain. There are literally an infinite number. There could be a leak in the reservoir. If the building housing the process develops a hole right above the reservoir, then water can leak into the reservoir on a rainy day. If there is a fire in the building housing the process, then someone may decide to bail water out of the reservoir to put out the fire. A disgruntled employee may decide to pour quantities of some other liquid into the reservoir on a regular basis. We could go on and on with this list, adding more and more implausible events. If we begin to consider multiple faults occurring simultaneously, then the possibilities increase exponentially. The main point to take away is that it is not possible to enumerate a priori all of the things that can go wrong. There are always events that can, and do, happen that designers did not anticipate. What can we do to help workers deal with those situations?

The logic of analytical redundancy, on which the AH is based, provides a viable answer

[4] This deficiency will be remedied in a forthcoming book (Vicente, in press).

Table 2
Relative advantages and disadvantages of task analysis and work domain analysis [34]

	Task	Work domain
Mental economy	Efficient	Effortful
Ability to adapt to unforeseen contingencies	Brittle	Flexible
Scope of applicability	Narrow	Broad
Ability to recover from errors	Limited	Great

to this question. The solution is to turn the problem on its head. Instead of trying to figure out all of the things that can go wrong — a hopeless task — we can instead define how the work domain should be functioning. That is precisely what work domain constraints do — they provide an operational definition of what it means for the work domain to be operating normally. Then, when constraints are violated, there is an indication that something is wrong. The technique is equivalent to definition by exclusion — no matter how bizarre the fault, it should affect one or more of the constraints governing the work domain.

For example, each of the faults described above will violate the mass balance constraint in Eq. (1). Therefore, if an interface presents information about each of the variables that enter into the constraint [i.e. $dVol(t)/dt$, MI(t), MO(t)] and the way in which those variables are normally inter-related, then workers should be able to detect the disturbance by noticing the violated constraint. Moreover, they can use information about exactly how the constraint is violated to diagnose the nature of the disturbance in sufficient detail to take appropriate compensatory actions. For instance, in the case of a leaky roof, workers could detect that the rate of change of volume is greater than it should be, given the measured inflow and outflow. This information alone is not sufficient to identify the leaky roof. However, it is enough to determine that too much water is entering the reservoir, and thus, that it may be appropriate to decrease the input valve setting or increase the output valve setting. This example shows how the AH can be used to provide support for coping with unanticipated events without identifying any tasks, events, or faults.

In practice, the situation is much more complicated, of course. We never know beforehand which constraints will be violated by a particular fault. That is why it is important to identify all of the goal-relevant constraints. The AH provides a structured and systematic framework for conducting this analysis.

4.3.5. Comparative analysis

To be fair, it is important to point out that there is a cost to adopting the AH as a work analysis technique. The added generality caused by event-independence results in an added burden in effort. This trade-off can be made clear with an analogy to spatial navigation [34]. Work domain analysis representations (such as the AH) are like maps, whereas task analysis representations (such as SGT) are like directions.

Table 2 shows that the comparative advantages and disadvantages of directions and maps are analogous to those between task and work domain analyses. Task analyses are efficient because they identify what needs to be done, and perhaps even how it should be done. But, as a result of this specificity, task analyses are brittle because they do not provide the support required to adapt to unanticipated events (see earlier). Task analyses

are also narrow in their generality because they are only applicable to the tasks that have been identified up front, or even more narrowly, to the ways of doing the task that have been identified up front. Finally, and as a result, task analyses are also limited in their ability to support recovery from errors.

As shown in Table 2, work domain analyses have a complementary set of strengths and weaknesses. Their primary disadvantage is that they do not tell workers what to do. They merely describe the capabilities of the work domain on which workers will be acting. As a result, work domain analyses put greater demands on workers. The good news is that work domain analyses are flexible because they provide workers with the information they need to generate an appropriate response, on-line in real-time, to events that have not been anticipated by designers. Moreover, work domain analyses also have a broader scope of applicability. Because they merely show what the work domain is capable of doing — independent of any particular event — they provide workers with the discretion to meet the demands of the job in a variety of ways that suit their preferences or the particular needs of the moment. Finally, work domain analyses also provide workers with the flexible support they need to recover from errors.

5. Conclusions

People interact with, or through, computers in a very large number of application domains that present a diverse set of design challenges. Given this breadth, there is no reason to believe that ''one size fits all'' when it comes to work analysis techniques. For the particular case of correspondence-driven work domains, it may be important for work analysis techniques to be device-independent so that future designs do not inherit the weaknesses of previous generations. It is also important that such techniques be psychologically relevant so that workers can comprehend and effectively utilize the resulting design. Richardson et al. (in press) have made a strong case that the particular form of task analysis known as SGT satisfies both of these criteria and thus represents a valuable contribution to the analyst's tool kit. However, correspondence-driven work domains also require event-independent work analysis techniques to identify the information requirements associated with unanticipated events. To ignore this requirement is to ignore the most significant threat to the safety of complex systems, as analyses of industrial accidents have shown. The particular form of work domain analysis known as the AH can satisfy this criterion, thereby complementing SGT. Future research should investigate the relationship between these techniques in further detail. Ideally, system and task models should be integrated [17], along with strategy models, human operator models, and social-organizational models, into a single, unified approach to work analysis for correspondence-driven work domains [24,34].

Acknowledgements

The writing of this article was sponsored by grants from the Natural Sciences and Engineering Research Council of Canada. I would like to thank Chris Miller of Honeywell

Technology Center, Tom Ormerod, and an anonymous reviewer for their constructive comments.

References

[1] D. Benyon, The role of task analysis in systems design, Interacting with Computers 4 (1992) 102–123.
[2] D. Benyon, Task analysis and system design: The discipline of data. Interacting with Computers 4 (1992) 246–259.
[3] A.M. Bisantz, K.J. Vicente, Making the abstraction hierarchy concrete, International Journal of Human–Computer Studies 40 (1994) 83–117.
[4] C.M. Burns, K.J. Vicente, A comprehensive experimental evaluation of functional displays in process supervision and control. (ABB Final Report, CEL 97-05). Toronto: University of Toronto, Cognitive Engineering Laboratory, 1997.
[5] J.M. Carroll, W.A. Kellogg, M.B. Rosson, The task–artifact cycle, in: J.M. Carroll (Ed.), Designing interaction: Psychology at the human–computer interface. Cambridge University Press, 1991, pp. 74–102.
[6] J. Dewey, A.F. Bentley, Knowing and the Known. Beacon, Boston, MA, 1949.
[7] D. Diaper, M. Addison, Task analysis and systems analysis for software development, Interacting with Computers 4 (1992) 124–139.
[8] N. Dinadis, K.J. Vicente, Designing functional visualizations for aircraft system status displays. International Journal of Aviation Psychology, in press.
[9] N. Dinadis, K.J. Vicente, Ecological interface design for a power plant feedwater subsystem, IEEE Transactions on Nuclear Science 43 (1996) 266–277.
[10] P.M. Frank, Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy — A survey and some new results, Automatica 26 (1990) 459–474.
[11] J.J. Gibson, The ecological approach to visual perception. Houghton-Mifflin, Boston, MA, 1979.
[12] J. Itoh, A. Sakuma, K. Monta, An ecological interface for supervisory control of BWR nuclear power plants, Control Engineering Practice 3 (1995) 231–239.
[13] G.A. Jamieson, Ecological interface design for petrochemical processing applications, (Unpublished masters thesis). Toronto: University of Toronto, Department of Mechanical and Industrial Engineering, 1998.
[14] J.V. Kauffman, G.F. Lanik, R.A. Spence, E.A. Trager, Operating experience feedback report — Human performance in operating events (NUREG-1275, Vol. 8). US Nuclear Regulatory Commission, Washington, DC, 1992.
[15] N.G. Leveson, Safeware: System Safety and Computers. Addison-Wesley, Reading, MA, 1995.
[16] M.D. Mesarovic, D. Macko, Y. Takahara, Theory of Hierarchical, Multilevel, Systems. Academic Press, New York, 1970.
[17] C.M. Miller, K.J. Vicente, Toward an integration of task and work domain analysis techniques for human–computer interface design, in: Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting, 1998.
[18] C.M. Mitchell, Models for the design of human interaction with complex dynamic systems, in: Proceedings of CSEPC 96: Cognitive Systems Engineering in Process Control. Kyoto University, Kyoto, Japan, 1996, pp. 230–237.
[19] W.H. Newton-Smith, The Rationality of Science, Routledge and Kegan Paul, Boston, MA, 1981.
[20] L. Norros, System disturbances as springboard for development of operators' expertise, in: Y. Engeström, D. Middleton (Eds.), Cognition and Communication at Work. Cambridge University Press, 1996, pp. 159–176.
[21] A.M. Pejtersen, The Book House: An icon based database system for fiction retrieval in public libraries, in: B. Cronin (Ed.), The Marketing of Library and Information Services 2. ASLIB, London, 1992, pp. 572–591.
[22] J. Rasmussen, Man–machine communication in the light of accident records (S-1-69). Danish Atomic Energy Commission, Research Establishment Risø, Roskilde, Denmark, 1969.
[23] J. Rasmussen, The role of hierarchical knowledge representation in decisionmaking and system management, IEEE Transactions on Systems, Man and Cybernetics SMC-15 (1985) 234–243.

[24] J. Rasmussen, A.M. Pejtersen, L.P. Goodstein, Cognitive systems engineering. Wiley, New York, 1994.
[25] J. Reason, Human Error. Cambridge University Press, 1990.
[26] D.V.C. Reising, P.M. Sanderson, Designing displays under ecological interface design: Towards operationalizing semantic mapping, in: Proceedings of the 42nd Annual Meeting of the Human Factors and Ergonomics Society. HFES, Santa Monica, CA, 1998.
[27] J. Richardson, T.C. Ormerod, A. Shepherd, The role of task analysis in capturing requirements for interface design, Interacting with Computers 9 (4) (1998) 367–384.
[28] T.D. Sharp, Progress towards a development methodology for decision support system for use in time-critical, highly uncertain and complex environments (Unpublished doctoral dissertation). University of Cincinnati, Department of Electrical and Computer Engineering, Cincinatti, OH, 1996.
[29] A. Shepherd, Maintenance training, in: B. Kirwan, L.K. Ainsworth (Eds.), A guide to task analysis. Taylor and Francis, London, 1992, pp. 327–339.
[30] A. Shepherd, An approach to information requirements specification for process control tasks, Ergonomics 36 (1993) 1425–1437.
[31] I.G. Umbers, C.S. Reiersen, Task analysis in support of the design and development of a nuclear power plant safety system, Ergonomics 38 (1995) 443–454.
[32] K.J. Vicente, Coherence- and correspondence-driven work domains: Implications for systems design, Behaviour and Information Technology 9 (1990) 493–502.
[33] K.J. Vicente, Improving dynamic decision making in complex systems through ecological interface design: A research overview, System Dynamics Review 12 (1996) 251–279.
[34] K.J. Vicente, Cognitive work analysis: Towards safe, productive, and healthy computer-based work. Erlbaum, Mahwah, NJ, in press.
[35] K.J. Vicente, J. Rasmussen, The ecology of human–machine systems II: Mediating ‘‘direct perception’’ in complex work domains, Ecological Psychology 2 (1990) 207–250.
[36] K.J. Vicente, J. Rasmussen, Ecological interface design: Theoretical foundations, IEEE Transactions on Systems, Man and Cybernetics SMC-22 (1992) 589–606.
[37] K.J. Vicente, F. Tanabe, Event-independent assessment of operator information requirements: Providing support for unanticipated events, in: Proceedings of the American Nuclear Society Topical Meeting on Nuclear Plant Instrumentation, Control and Man–Machine Interface Technologies. ANS, LaGrange Park, IL, 1993, pp. 389–393.
[38] W. Xu, Externalizing a work domain structure on a hypertext interface using an abstraction hierarchy: Supporting complex search tasks and problem solving activities (Unpublished doctoral dissertation). Miami University, Department of Psychology, Oxford, OH, 1996.