



Using intentional models for the interface design of multi-level systems

ALAN W. COLMAN AND YING K. LEUNG

*School of Information Technology, Swinburne University of Technology,
PO Box 218, Hawthorn, Victoria 3122, Australia.*

E-mail: {alan.colman@detir.qld.gov.au, yleung@swin.edu.au}

(Received 27 May 1999 and accepted in revised form 17 January 2000)

In this paper, it is argued that the design of computer interfaces for complex, multi-layered systems needs to take into account the differing intentional models that are held by different types of users of such systems, and that there is a strong correlation between the job roles of individuals and the level of abstraction of the mental models held by such users. An approach to the analysis and design of complex multi-layered systems based on the analysis of job roles to elicit such models is suggested and linked with other techniques of task analysis and object-oriented analysis and design. The methodology is illustrated with the interface analysis for an automatic environmental chemical analyser.

© 2000 Academic Press

KEYWORDS: intentional models; user interface design; multi-layered systems; object-oriented methodology; human-computer interaction.

1. Introduction

Many computer programs and all computer systems are layered to provide different sets of functionality to different types of users. For example, operating systems are separated from application programs; compilers and interpreters make possible high-level languages that allow users to write programs in human-friendly sentences rather than in assembly or machine code; standards, such as the ANSI/SPARC model for database management systems, define different levels of functionality for different types of users and so on. Layered systems are ubiquitous in computing. One aspect of good interface design is that the information presented by the interface is compatible with the user's ability to process that information. If design methodology is to foster this compatibility, it needs to take into account how users represent knowledge. In a system where different users interact with different layers[†] of the system, the methodology needs to cater for the various representations of the different users.

Our focus in this paper is on representations of problem domain *knowledge*, both within the users' heads and within the "system images[‡]" presented by the computer. The

[†]The term "multi-layer" or "multi-faceted" has been taken to indicate multiple sets of functionality at varying levels of abstraction, not necessarily a strict hierarchy of layers providing services to each other as, for example, in the ISO Reference Model for Open Systems Interconnection.

[‡]Norman's (1983) term meaning the physical images presented by the system based on the designer's conceptual model.

aim is to provide guidelines for the design of multi-layered interfaces. These guidelines should help designers better match the system images the interface presents with the mental models held by different types of users, without fragmenting the consistency of representation that allows the different layers to be integrated into a coherent model by the user.

This paper will first define complex and multi-level systems and provide an explanation of the relationship between intentional models and structural and functional models. It then describes how layers and intention in the interface can be represented and presents techniques where intentional models can be used for system design. An approach for the analysis and design of complex multi-layered systems based on the analysis of job roles is suggested, based on the elicitation of such models and linked with other techniques such as task analysis and object-oriented analysis and design. The methodology is then illustrated with the interface analysis for an automatic environmental chemical analyser.

2. Complex and multi-level systems

Although multi-layered systems are ubiquitous in computing, not all layered systems present a readily accessible user interface to all levels. For example, the lower levels of an operating system may not be directly accessed at all by users. In this paper, we are concerned with multi-layered systems that present user interfaces for a number of their layers. The interfaces for multi-level systems may involve entirely separate interfaces for each of the layers (e.g. a hierarchical database data definition language is distinct from the database data manipulation language), or may combine the different levels of functionality into a single interface (e.g. a spreadsheet allows the user to define the structure of the data and input the data from within the one interface).

The systems we are concerned with here are complex in the sense that the problem domain is structurally complex and multi-faceted, and the user must maintain a mental model of the problem domain in order to interact with it. Each layer within the problem domain represents different types of relations, often at a different level of abstraction. For example, the Open Systems Interconnection (OSI) model for data communications is complex in the sense that layers represent different levels of abstraction of the communication process. A network management program may attempt to represent a number of these levels of abstraction. A financial spreadsheet, on the other hand, is not complex in this sense as the relationships are all of the same type at the same level of abstraction.

Layering of functions and knowledge also occurs in work roles, in particular in those roles in which users are interacting with different levels of a complex system. It can therefore be argued that the stratification of knowledge into different levels of abstraction is a determinant in the stratification of both job roles and complex computer systems. By understanding this stratification, designers of multi-level systems can create a better match between the system images and the mental models the users construct of the system.

3. Mental models

In recent years the study of "mental models" has been prominent, particularly in HCI, because there is evidence that users of systems and devices, to some extent, represent and

reason about these systems using simplified mental simulations of the real-world system (e.g. Johnson-Laird, 1983; Gentner & Stevens, 1983). This paper will examine mental models of a problem domain from, as Rasmussen (1990) terms it, a “cognitive engineering” perspective, which is concerned with the operational aspects of mental models in work situations.

Various authors have differentiated types of mental models. While there is variation in the terminology, two types of mental model are often distinguished in the literature: structural models and functional models. Structural models are associated with an “expert” understanding of a system — an understanding of how the system works, with “some plausible cascade of causal associations connecting the input and the output” (Carroll & Olson, 1988, p. 51). Functional models, on the other hand, describe not just the behaviour or functionality of a system, but also the interactive behaviour of human and machine.

If the aim of system design is to provide consistency between the user’s mental model and that presented by the system, we need to consider what sort of model or system image to present to the user to enable them to construct their own *appropriate* mental model. The difficulties with the two categories of models described above can be summarized as follows.

- Purely functional models have no structure but are merely sets of task-action mappings. It is precisely this structure that makes mental models useful to the user, in assisting the user to predict the behaviour of the system in non-routine situations (Halasz & Moran, 1983).
- Structural models do not provide an adequate basis for the construction of system images. Users find it difficult to construct such models (see Preece, 1994, pp. 135–136). Users, given a structural model, find it difficult to derive the function of the system. Even experts tend to rely on heuristics rather than structural models (Taylor, 1994).
- The level of abstraction in complex structural models is arbitrary. The reductionist explanation that relies on mechanistic causality is not how people tend to explain the behaviour of systems.
- Functional and structural models do not describe the purpose of the system from the user’s perspective—the ‘why’ or intentionality of the system.

4. Layers and stages in human–computer interaction

Many authors have distinguished different layers of interaction between humans and computers. One way to differentiate layers is to distinguish between the tool domain and the problem domain of a system (Colman & Leung, 1995).§ The user model of a computer control or design system can be described on a number of layers on a problem domain — tool continuum including: problem domain → representation of problem domain → symbolic tools or affordances (Gibson, 1966) of symbolic objects specific to problem domain → generic computer software interface (menus, icons, etc.) → hardware tool interface (mouse, keyboard etc). When a user executes a task to achieve a goal

§A number of other authors have made similar distinctions, many based on the linguistic distinction between syntax and semantics—Carroll and Olson (1988 p. 46) differentiate between “task-knowing” and “system interface-knowing”. Van der Veer, Wijk and Felt (1990, p. 135) differentiate task, semantic, syntax and keystroke levels. Taylor (1998, pp. 212–215) summarizes other layered approaches.

the user interaction with, or modelling of, these levels can be viewed as stages task execution–evaluation process (Norman, 1984; 1988).

The tool levels of the interface include the physical interface and the interactive elements of the system (menu, commands). Much of the cognitive science based work in HCI has been a study of interaction at the tool level — the study of *task execution* (e.g. based on the analogy of the human mind with an information processor; Card, Moran & Newell, 1983).

To highlight the distinction between tool and problem domains, an analogy with an architect's task is apposite. If we want to describe the knowledge required to design a house, we need to be concerned with knowledge of design principles and building regulations. The architect needs to maintain a and mental model of the proposed building (problem domain knowledge) and understand the current state and meaning of the design (representation of problem domain knowledge), as well as knowledge of drafting skills (tool knowledge). The knowledge of the tools is secondary (knowledge of how to draft a house may change depending on whether you are using a pencil or computer) to the knowledge of the problem domain (how to design a house and what is the design of the house at the current stage of design).

In certain types of computer system such as computer-aided design or control systems, the problem domain representation is critical to the effective use of the system. At the problem domain level *task acquisition* is particularly important. Task acquisition requires the user to have an understanding of the current state of development of the particular problem domain. The system interface needs to provide feedback to the user on this current state.

Work by Taylor and others (Taylor, 1988; Taylor, Farrell, Hollands & Gamble, 1998) on Layer Protocol Theory (LPT) sees the problem domain and tools used to manipulate the domain as layers on a protocol stack that represent different levels of the task acquisition/execution process. Different levels of the abstraction in the problem domain, for example, the reference to a place on a map by name or by coordinates, are regarded as *protocol nodes* that receive an intention from a higher layer. This intention (primal message) defines a reference state (model) in the node. The sender node uses the services of lower layers to achieve the desired outcome in the corresponding receiving node. In LPT, the problem domain and the tool domain are different layers on the stack used at different stages of task execution.

In this paper, we are concerned with the user's modelling of the problem domain, rather than the process or stages of interaction with the tool (the dialog between the human and the computer). We are interested in different levels of abstraction in the problem domain. From this perspective the tool is transparent to the task. It is the mental models of the problem domain, rather than just the tool interface, that we need assist the user to develop if we are to provide the user with an adequate basis for task acquisition in complex systems. In the LPT schema, we are concerned with how the reference state of a problem domain is acquired and what process can be used to define the boundaries of layers in the protocol stack.

In this sense, layers of abstraction in the problem domain can be seen as independent from, and orthogonal to, layers (stages) of abstraction in the manipulation of the tool. The model of the problem domain in the architect's head is the same whether she is using a pencil or a computer to design a house. However, while the problem domain is

independent of the tool, the user still needs to be able to manipulate the problem domain (in a control system) or problem domain representation (in a design system). For task execution, there still needs to be a link between the problem domain level and the tool levels in the protocol stack. In later sections, we will discuss how this link can be provided by the affordance of the objects in the interface or the provision of symbolic tools compatible with the user mental model of the problem domain.

5. Intentional explanations and models

Searle (1984, p. 74–77) argues that to explain the structure of human action, we have to resort to the *intention* of the phenomena — “the feature by which our mental states are directed at, or about, or refer to, or are of objects and states of the world other than themselves” (p. 16). Searle maintains that the explanation of human behaviour in terms of the intention of the behaviour is both the most effective and common form of reasoning about human behaviour. Searle notes that there are two main components to intentional states. The “content” is objects, actors and actions in the mental representation. The “psychological mode” or “type” relates to whether the action is desired, feared, intended or believed by the person holding the representation. For example, two people may share structural or functional explanations of a system, but the intentional context in which they are put may differ.¶ A physicist with no understanding of the rules of billiards and an expert billiards player may have the same structural explanation of the mechanics of billiard ball behaviour, but have very different understandings of what is occurring on the table during a game. Note that for a successful player, some understanding of the “content” — of the mechanics of billiard ball behaviour — is necessary. Such mechanistic explanation (a structural model), however, is within a broader context of intentional explanation based on the player’s understanding of the participants’ likely intentions as structured by the rules of the game, and as mediated non-mental capacities of the players (their level of motor skills, eye–hand coordination, etc.).

Intentional explanations provide a higher-level framework for lower levels of explanation by providing a context for those lower-level explanations. Users’ structural explanations of systems at a mechanistic level tend to be poorly formed and disjointed. Intentional models can be viewed as a “glue” that hold these disjointed, lower-level explanations together, and to check whether these lower-level explanations make sense in terms of the goals of the system. To extend Searle’s concept of an intentional state — just as an intentional state of an individual functions against a background of human capacities (abilities, skills, habits, etc.) that are not themselves mental states (Searle, p. 68) — an *intentional model* can be defined as a network of intentional explanations and states that consist, in part, of non-intentional structural and functional explanations and fragmentary models. An intentional model contextualizes the user’s understanding of the ‘content’ of a system by referring to the goals of the system.

Figure 1 illustrates the relationship between an intentional model, and the models that make up the “content”—the structural and functional models. These intentional explanations constitute the individual’s perception of the purpose of the system that is

¶In this sense, many “scientific” explanations have an intentional element. For example, evolution can be viewed as survival of the fittest animal or we can view animals as mere by-products of competing genes. Both explanations are based on the same observed facts — what differs is what or who is considered the intentional agent.

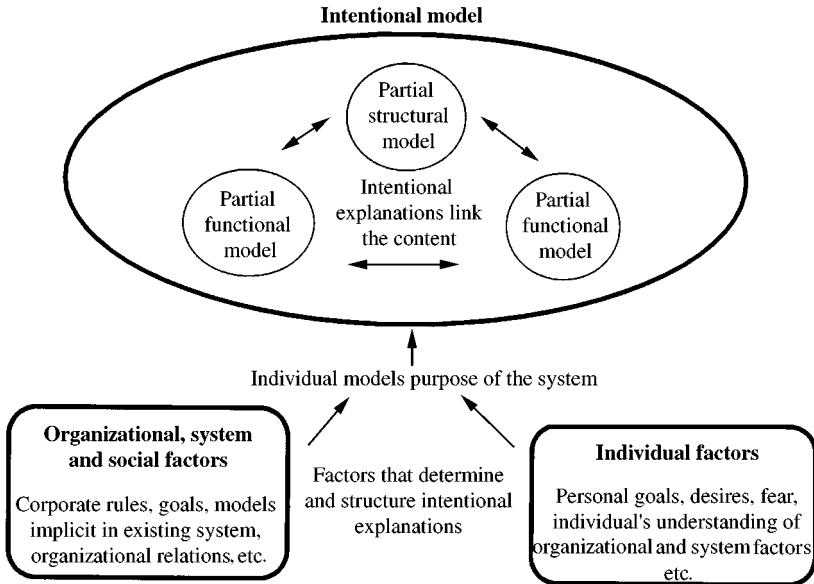


FIGURE 1. Relationship between intentional models and structural/functional models.

informed by the user's goals or intentions in that work situation. A user's construction of a purpose for a system, is itself likely to be an intentional model of the system at a higher-level abstraction.

It should be noted that intentional models are not well defined, may well be inconsistent and cannot be used as a basis for machine computation. They are not 'engineering' models. Rather they accord with the observation that mental models are "incomplete, unstable, easily confusable, based on presumption" (Norman, 1983). Carroll, Mack and Kellogg (1988) have noted in relation to metaphors (a particular type of mental model where the user uses pre-existing knowledge of an analogous system) that composites of mismatched and incomplete metaphors are often employed by a user to explain aspects of a domain. To interact with a system a person may hold a number of, possibly inconsistent, mental models. Yet it is by means of their intentional models that people actually reason about real-world systems.

6. Intentional models and multi-level systems

How are intentional models relevant to interaction with complex (computer) systems in the workplace? Rasmussen (1990), in his study of the cognitive control of actions in the workplace, recognizes the importance of different notions of causality. Mental models are "representations of the relation structure of the causal environment and work content. Many different kinds of relationships are put to work during reasoning and inference at this level ..." (Rasmussen, 1990, p. 58). Rasmussen argues that complex systems are often more effectively regarded as "intentional systems controlled by motives or intentions together with the constraints of the environment" (p. 62) Rasmussen argues

that, for the individual operator, the shifting up and down the abstraction hierarchy is important to understand the system for problem solving (p. 60).

Each operator's construction of the intention of a system is unique to that individual. Each operator brings a pre-established complex of attitudes, intentions and knowledge to a job — a complex that effects construction of intentional models. Notwithstanding these individual variances, there are organizational factors that allow us to make useful generalizations about the construction of intentional models in the work situation. There can be many factors which determine the division of job roles within an organization (see Rasmussen, 1994, pp. 98–99). Here we are concerned with two factors that reinforce the correlation between work roles and intentional models likely to be held by an individual performing a role.

- Work roles have purposes or goals defined within the context of the organization's system.
- Work roles have prerequisite knowledge in particular the *type* of knowledge required often varies between roles.

The stratification of knowledge into different levels of abstraction is a determinant in the stratification of both job roles and complex systems. By understanding this stratification, designers of multi-level systems can create a better match between the system images and the mental models the users construct of the system.

To discuss the types of knowledge associated with these strata, we need an epistemology that will allow us to generalize different levels of abstraction. Rasmussen (1986, Chapter 4) identifies a number of these levels of abstraction.

- Functional meaning-purpose—value structures, intentions.
- Abstract function—information processing level, semantic content of the physical signals, organizing principles of organizations.
- Generalized function-functional properties represented more generally without reference to physical process or equipment.
- Physical function—physical processes or functions.
- Physical form—physical objects and location.

The higher the level of abstraction the more the system will be defined in terms of its organizational context and the more likely explanations of behaviour of the system will be couched in terms of intentionality of the system in its organizational context. In complex systems where many individuals perform different functions according to their work role, the designer will have to present corresponding system images at the appropriate level of abstraction. If we are to construct software systems whose system images are compatible with the users of the system at different levels, we need to understand the nature of the intentional models likely to be constructed by operators in various roles. Furthermore, we need to establish what factors are likely to affect the creation of intentional models by individuals performing various job roles.

In summary, by identifying the goals associated with particular job roles, and by examining the factors in the workplace that lead to the stratification of knowledge in job roles, it should be possible to develop conceptual models and system images at appropriate levels of abstractions for those work roles. Because this stratification is, in part, based on the intentional “means-end dimension”, the resultant system images should enable the

operator to reason more effectively in a top-down manner about the likely behaviour or 'intention' of the system.

7. Representing layers and intention in the interface

The task of the designer, as Rasmussen (1994, p. 134) expresses it, is to develop an interface "that (1) will lead users to adopt effective mental models, but that (2) do not conflict with and established stereotypes belonging to the given professions or company". Designing a system with multiple layers that correspond to the intentional models associated with different work roles, does not necessarily involve designing a discrete interface for each job role. Rather, it involves determining what layers are appropriate, and representing intention in the interface.

7.1. DETERMINING THE LAYERS

In deciding what layers to present to the users a couple of issues need to be addressed, other than determining the appropriate level of abstraction. Firstly, the system designer needs to decide to what extent models are elicited from existing users, or to what extent a single 'correct' model is imposed. Rasmussen, Pejtersen and Goodstein (1994, pp. 50–53) categorize work domains on a continuum. At one end of the continuum, loosely coupled domains are structured by the actor's intentions, work rules and practices. At the other end of the continuum are highly structured and tightly coupled domains such as automated process plants. In the latter type of tightly coupled work domain, it is important that the model presented by the interface of the system is close to the structural model of the system so that users can develop the 'correct' model to enable them to respond appropriately to non-routine circumstances. In more loosely coupled systems where users have more autonomy as actors, it may be appropriate to have the interface reflect pre-existing models or metaphors held by typical users.

The second design consideration in determining the layers to be presented in the interface is to what extent there is integration of system images of the various levels. Systems in which there is a distinct separation of job roles into independent functions may not need to have an integrated interface. On the other hand, systems in which there is a progression through levels of the system in the process of learning to use the system, or where a user has to use multiple levels of abstraction to perform his/her role will benefit from an integration of the system images at various levels. In this case, we need to look for techniques that will help us find commonalities between the various mental models of users in their different work roles, so that we can integrate the system images presented at different levels of the interface. The aim would be to develop an interface based on a 'deep' conceptual model of a problem domain, from which various consistent interface 'views' can be elaborated. We will discuss techniques for doing this in the next section.

From our discussion of intentional models in previous sections, the importance of maintaining context across different layers of the system will be apparent. An intentional explanation references higher levels of abstraction to explain why the system behaves the way it does. Without the contextual link that integrates the different levels of abstraction, an intentional explanation is not possible.

7.2. REPRESENTING INTENTION

Representing intention in a system interface can be attempted through firstly the development of virtual actors who embody aspects of the system, and secondly through the construction of an appropriate (dramatic) space where the elements suggest (“afford” in Gibson’s 1966 terminology) the likely course of action. Laurel in her book *Computers as Theatre* (1993) proposes intentional explanations of action as a basis of the design of computer interfaces. Laurel advocates the use of dramatic techniques as the basis for software design — as a means of engaging the user. In a drama, as the plot progresses, the possible outcomes are constrained through the characters and action (Laurel, p. 105). The interaction of the characters provide the audience (users) with a means of anticipating the action through their understanding of the characters’ intentional state. While the attribution of anthropomorphic features to a virtual actor may be of use in software such as computer games where there is a dramatic context, it is of little use in complex systems. Such attribution of human characteristics to objects in a work environment does little to suggest the likely behaviour of those objects as anyone who has been annoyed by Microsoft’s Office Assistant can attest.

Although the use of characterization may be of limited benefit in an interface, an understanding of the intentional model likely to be held by users performing a job role — an understanding of how users perceive the goals of the system, including their own roles — will help us anticipate how users reason top-down about the system. This will allow us to develop system images that present compatible “explanations” of the behaviour of the system. Design based on intentional models is more about designing spaces with appropriate objects (a stage design and props in dramatic terms) in which autonomous actors can perform their work roles, and ensuring consistency in the definition of objects across levels of abstraction — than prescribing a set of tasks to be undertaken.

8. Techniques for system design using intentional models

A process to assist the development of system images that are compatible with the intentional models constructed by users in different job roles is proposed here. These techniques are an adjunct to object-oriented analysis. A number of major tasks are identified. The process through these tasks, in practice, is likely to be iterative rather than a strict sequence. The tasks are the following

1. Identify major job roles and the goals associated with each role.
2. Describe taxonomy and intentional model for each role.
3. Develop a conceptual design model based on integrating the taxonomies across levels, and any preferred model.
4. Map integrated model to an object model.

8.1. IDENTIFYING JOB ROLES AND GOALS

Much work has been done on the analysis of work tasks in human resource management (HRM), human factors research and HCI. In HCI this work is part of, what is broadly termed, task analysis. Because of the mental nature of many computer tasks, cognitive

task analysis has been a particularly important strand of this research. Task analysis in HCI has generally concentrated on an individual's tasks as they relate to a system, from a high level of abstraction such as the analysis of goals and tasks such as in hierarchical tasks analysis (Annett & Duncan 1967; Stammers & Gray, 1971), though to finer levels of granularity in the analysis methods and procedures (e.g. Card *et al.*, 1983. GOMS methodology). The study of the performance of tasks has often taken a reductionist approach based on analogies between cognitive processes and information processing.

While much of HCI research has tended to study the individual's execution of a discrete task on a computer, other fields such as Requirements Analysis have focused on the organizational context in which the tasks take place, including the sharing of tasks between groups of people. Jirotko and Goguen (1994) provide an overview of attempts to look at systems from a social as well as a technical aspect. Such approaches have attempted to meld user needs, organizational structure and job design.

Of particular relevance is the ORDIT methodology (Dobson, Blyth, Chudge & Strens 1994), which describes an organization as a set of related work roles. The ORDIT methodology defines two types of role—a *functional* role which is the relationship between an agent and an activity, and a *structural* role which is the relationship between agents and corresponds to the social aspect of the role (p. 99). Both type of roles need to be described to describe an intentional model of a system. A user's understanding of their functional role will be based on the 'content' of their model of the system—the problem domain knowledge required to perform their work role. This problem domain knowledge will be modelled at a level or levels of abstraction appropriate to the performance of their role. The structural role (as distinct from a structural model) defines the intentionality of the system—the user's relationship with the rest of the organization and the purpose for their role.

To be effective as an input to system design, the description of functional work roles needs to be set in the context of a top-down organizational perspective. This is because intentional models are always defined in terms of higher goals and purposes. These higher-level goals are not always explicitly defined in an organization, but are implicit in the organization structures. Rasmussen *et al.* (1994, p. 29), in their study of modern work organization, point out that job roles are often better defined by their goals and constraints, rather than set procedures. "A typical task sequence often does not exist in a modern advanced work setting. As a result, generalizations cannot be made in terms of work procedures revealed by means of a classical task analysis but instead should be made in terms of the objectives, functions, and resources active in prototypical situations, and the related information requirements".

It may not be possible or desirable to strictly define the job roles that interact with a system. Many application programs are written for an industry rather than for a single organization. In such cases, typical job roles can still often be defined. These typical job roles are determined by a number of factors: the different types of knowledge required for different tasks common to these industries; common organizational factors including the different information needs of various level of management; job classification standards; and social factors such as the educational system's structuring of knowledge, professional organizations creating standards of knowledge and barriers to entry to various job roles; and so on. The description of a work role of a user should provide the goals, responsibilities, tasks, knowledge and skills associated with that role. In other words, the description

should provide the elements of an intentional model—the meaning (goal), context (organizational, social, personal) and content (knowledge, skills) of the role.

8.2. DESCRIBING INTENTIONAL MODELS FOR DIFFERENT ROLES AT APPROPRIATE LEVELS(S) OF ABSTRACTION

Once a description of the job role in terms of goals, tasks, knowledge and skills required has been developed, we need to describe an intentional model whose knowledge content is at an appropriate level of abstraction. The content of intentional models, as defined above, is made up of partial functional and structural models. We cannot necessarily associate one role with one level of abstraction because the partial models that make up the content of the intentional models may be at different levels of abstraction. The levels of abstraction can range from high level to low level as defined by Rasmussen above. For the top three functional layers the partial model will be a “black box”. The user is aware of the black box’s purpose, inputs and outputs. Lower levels of abstraction will involve some structural understanding that enables the user to infer behaviour of the partial model.

Eliciting the mental models associated with various roles can be derived from primary or secondary sources. Primary sources involve the users, who interact with the system in various roles, articulating their understanding of the system. Requirements analysis provides a number of techniques for elicitation. Yet in order to structure this understanding in terms of an intentional model we need to ensure the model is described from the perspective of the user and at the level of abstraction at which the user is thinking. Approaches^{||} that attempt to elicit the participants’ own account of the system are more appropriate to describing intentional models than traditional analysis documentation techniques (system charts, DFDs) that describe the system at the designer’s level of abstraction.

Difficulties also exist in the use of verbal descriptions, verbal protocol analysis, and so on, to elicit mental models at the appropriate level(s) of abstraction (Norman, 1983, p. 11). The problem with language in use in the workplace is that lower levels of abstraction are often embedded in the language, even though the user may not be using that lower level of abstraction as a structural model. For example, a PC user may use a term like “reboot the computer” to mean turning the computer off and then on again, without any understanding of the notion of a bootstrap program loading the operating system. The proliferation of jargon in the computer industry makes it very easy for the designer to presume the existence of understanding at lower level of abstraction than the user actually holds. Elicitation techniques such as Soft Systems Methodology’s rich pictures (Checkland & Scholes, 1990) probably form a more reliable guide to the mental models a user holds because they are jargon free, and can be more expressive of the relationships between the elements of the model.

Depending on what level of the system we are describing, secondary sources for eliciting intentional models include: existing system documentation, role descriptions, work instructions, quality assurance procedures, etc. Again, care needs to be taken because the level of abstraction of the secondary source may not match that of the user.

^{||}See Jirotko and Grougen (1994, p. 5) for an overview of such approaches.

System documentation is written by designers/developers who will tend to have a low-level structural understanding of the system. Conversely, organization/management documentation is usually written by those who have a good understanding of the purposes and structure of the organizational context of the system. A person performing a role at a lower level of the organizational hierarchy may not share this understanding of the organization's intentional structure. In this latter case, however, the goal hierarchy within an organization usually ensures consistency of the specific goal with the broader organizational goal.

Dobson *et al.* (1994) point out that organizational requirements — the social, or in our terms, the intentional, context — often need to be 'debated' rather than 'captured or elicited'. These requirements, such as "power structures, obligations and responsibilities, control and autonomy, value and ethics", are often embedded in organizational structure and policy (p. 88). The different perspectives on the function and purpose of a system held by different work roles will result in disparate intentional models. We discuss the handling of disparate models below.

The result of this phase of analysis of mental models should be a set of intentional, functional and structural descriptions of the system. A why, what and how of the system — one for each work role. These descriptions consist of one or more (partial) mental models described in terms of actors and objects at various levels of abstraction. The aim of these descriptions is not to provide a coherent structural model, but to describe how a typical user performing a role understands the system and how the user understands his/her interaction with the system. For the purposes of analysis, an intentional explanation is just as valid as a structural explanation.

The elicitation of an intentional model is not necessarily a detailed description of everything a particular user understands about a system. Rather, it is a prototypical user's 'world view' of the purpose of major objects in the system and how they interact. It is the development of an "object world" through a "naming discourse" that names part of the design and their interactive behaviour (Bucciarelli, 1988; cited in Rasmussen *et al.*, 1994, pp. 165–166). This discourse is an iterative process undertaken with group individuals performing the same role.

Once intentional models have been elicited for each role, the designer needs to develop a conceptual model of the system that takes account of both the existing models held by users, and any preferred or 'correct' models of the system that the designer wants to implement.

8.3. ANALYSIS AND DESIGN DOCUMENTATION TECHNIQUES FOR INTENTIONAL MODELS

The system descriptions obtained for each role need to be documented in a way that reflects the goals, objects, relationships and dynamics of the intentional model associated with the role. These techniques can be used for the analysis of existing models and documentation of the system design based on the designer's conceptual model that in turn becomes the basis of a set of system images. The description of the system is not just an abstraction of the 'objects' in the system, but the behaviour of the objects and the way we interact with those objects. Such abstractions are what Johnson (1989) refers to as the taxonomic substructure in his categorization of knowledge structures.

For Johnson, the taxonomic substructure is a collection of object-action pairs. Colman and Leung (1995) have suggested an extension to KAT using object modelling documentation techniques, in particular, the Object Modelling Technique's object model (OMT—Rumbaugh *et al.*, 1991). Using object models as a documentation tool has two advantages. Firstly, they allow us to express the richness of the taxonomic substructure, in particular, the attributes and methods/actions associated with each object and the relationship between objects. Secondly, the object model notation allows easier translation from the domain of task analysis, where we are modelling the user's knowledge, to software design where we are constructing a system representation. Consequently, there is no sharp distinction between the analysis and design phases of the system.

A disadvantage of the OMT is that the object model is a *static* model. Although it expresses the potential behaviours of individual objects, it does not express either the functional or dynamic aspects of the system. For this OMT relies on a separate functional model (similar to a data flow diagram) and a dynamic model (state transition diagram). These models are "orthogonal" (Rumbaugh *et al.*, 1991, p. 6) to the object model and each other. This can result in difficulties in integrating them in the final design.

An alternative approach, and one that accords better with the notion of an intentional system, is that of a responsibility-driven approach to object-oriented analysis (Wirfs-Brock, Wilkerson & Wiener, 1990). "Responsibilities are meant to convey a sense of purpose of an object and its place in the system. The responsibilities of an object are all the services it provides ..." (p. 62). Such responsibilities are fulfilled in *collaboration* with other objects in the system (p. 90). Such an approach is precisely an intentional analysis of the system in that it is analysis in terms of the purpose and context of the system. We could describe it as a job description for the objects (virtual and human) in the system. The anthropomorphic view of a system as a group of "cooperating and collaborating agents" aids conceptualization in the software design process (p. 7).

Jacobson's (1992) 'use-case' approach is similar to Wirfs-Brock in viewing objects as agents, and also provides an effective framework for describing the dynamics of a system. A 'use-case' is a collection of possible sequences of interactions between the system and its "external actors", *related to a particular goal* (Cockburn, 1996). The major difference between a use-case and a scenario is that a use-case documents the variations (problem space) in sequences of actions. Each sequence is tracked until its goal is achieved, or the sequence is abandoned. In Searle's terms, the intentional states own "conditions of satisfaction" are achieved or otherwise (Searle, 1984, p. 60). The actors may be people or computer objects.

A useful technique for documenting the object model to be 'run' through use-cases is CRC (Class-Responsibility-Collaborators) diagrams or cards (Cunningham, 1994). The CRC technique explicitly spells out the responsibilities and collaborators of each class (actor/agent). In terms of the ORDIT methodology (Dobson *et al.*, 1994), the responsibility of an actor can be viewed as the functional role and the relationships with the collaborators is the structural role. The CRC approach allows the system designer (and the user assisting the designer) to remain focused on the system as an intentional system when running the use-case scenarios. This process parallels the ways user 'run' their mental models of the system. The focus on responsibility models the system as an intentional system.

The taxonomies elaborated for each role should consist of a description of the system from the point of view of each actor, and show the relationship between actors. The elaboration of the taxonomies could be done each as a separate process, or combined with the process of integrating them as described in the next section.

8.4. DEVELOPING CONCEPTUAL AND OBJECT MODELS

Once a model has been elicited for each role, the designer needs to establish the following.

- To what extent a preferred model needs to be implemented.
- To what extent these models will be integrated.

Where models need to be integrated to overcome differences between different levels of abstraction or user roles, a conceptual model based on the 'invariant structure' of elicited or preferred intentional models needs to be developed. This invariant structure is important for two reasons.

- Stable objects and their behaviour allow the implementation of a direct manipulation style of interface. Rasmussen (1994, p. 114) argues that the sources of regularity near the surface of the interface allows the user to predict the likely behaviour of a system using direct perception of the affordances of the object or structure (Gibson, 1979; Norman, 1988) "An artifact that is well-designed should, through appropriate use of invariant features make obvious what it is for and how it should be used" (Rasmussen, 1994, p. 116).
- We need to establish the commonality between the objects in each of the levels of abstraction in order to facilitate learning and movement between these levels. The consistent representation of certain objects across system images provides an intentional context for those images and the mental models underlying them, allowing the user to employ intentional explanations by referring to higher levels of abstraction.

Figure 2 illustrates the relationships between different types of explanation, levels of abstraction, and the importance of consistent representation of objects across levels of abstraction.

There are two approaches to the elaboration of an integrated, invariant conceptual model. A formal approach involves a process of 'generification' (Johnson, 1989, p. 172) to establish the 'centrality and 'representativeness' of the objects. The 'stability' (Colman & Leung, 1995, p. 8) of objects across models (system images) then needs to be analysed. Colman and Leung (1995) elaborate a method for developing an integrated interface from different system images in a developmental system (a system whose system images change over time). This involves analysing the "visibility" of system objects across various system images and using these "stable" objects as the basis for the interface design. A similar approach could be taken to the analysis of named objects across different levels of abstraction of the system. An alternative to the formal analysis of objects across levels, is to have a team of representatives of various job roles negotiate the names and definition of objects within the system—a "naming discourse" (Bucciarelli, 1988). This is the approach taken in the example analysis described in the next section and proves an effective way of establishing a common nomenclature. It may well be

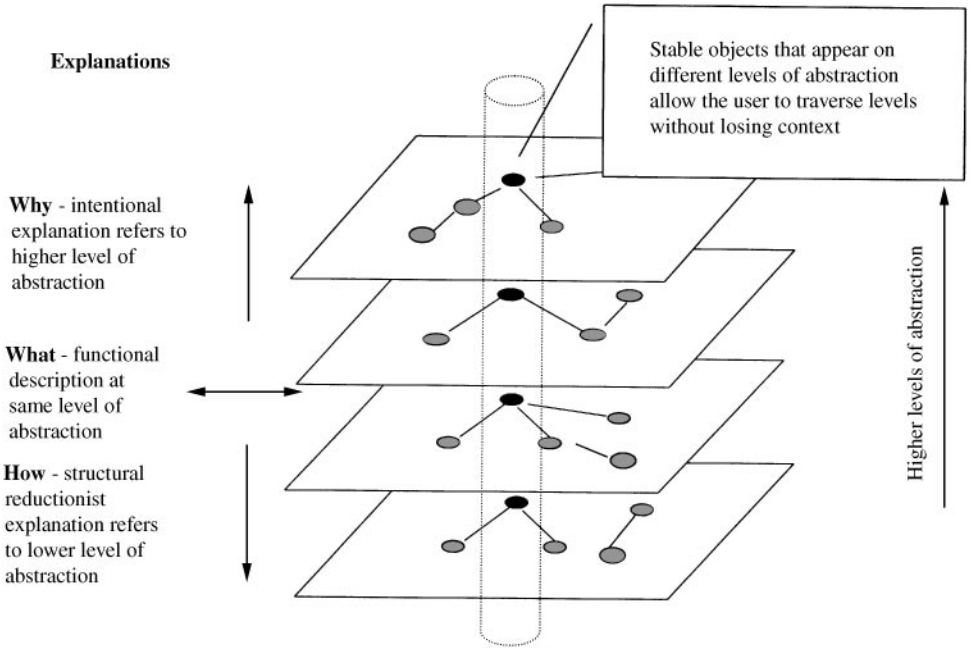


FIGURE 2. Moving between levels of abstraction with different types of explanation: (●) object, (—) relationship.

worth undertaking the negotiation prior to detailed elaboration of individual taxonomies. This shared object model can be used to run the use-cases as described above.

In a layered system, it can be difficult to identify objects that suit all levels of abstraction. A useful metaphor at a high level of abstraction may be based on objects that do not exist at a physical level (Garland, 1993). In the automatic laboratory example we examine below, there is no physical object called an “analyser”, yet this abstraction proves important for users at higher levels of abstraction who are familiar with traditional laboratory processes. There may be no single perfect integrated model that can match the disparate objects at different objects at different levels of abstraction. A balance may need to be found between the benefits of using an integrated model, and the cognitive load for users to whom the representation may appear unnatural or artificial.

9. The Aqualab—an example application

The Aqualab, manufactured by Greenspan Technology Pty Ltd, is an automatic wet-laboratory system that analyses water quality *in situ* (rivers, lakes, etc.). It is a multi-layered system in the sense that it has different types of users needing to interact with different sets of functionality in the system. The Aqualab is a complex and adaptable system that can be viewed from a number of different perspectives: environmental data logger, chemical laboratory, process control system, physical machine, etc. The Aqualab comes in a number of configurations. Aqualabs are custom-built to suit individual

customers' requirements. Customers are able to choose configurations that will measure up to approximately 20 physical and chemical parameters.

At the time of analysis, the Aqualab had been in prototype and early release form for a period of approximately 12 months. The original interface to Aqualab had been adapted from existing data acquisition software, and was an engineer's view of the system. The design process has been an attempt to redefine the architecture of the Aqualab and make the interface more compatible with the different types of user who interact with it.

9.1. THE WORK ENVIRONMENT AND JOB ROLES

As the product is being sold to many different work environments and industries, the definition of job roles has to be based on a generic functional analysis of the type of people likely to interact with the system, rather than an analysis of position descriptions within a particular company or industry.

The Aqualab is a moderately coupled system in that the system controls a physical process system. These physical processes in turn impose constraints in how the system can be manipulated by the user. On the other hand, the system is a stand-alone monitoring system rather than a system coupled to a broader industrial control system. This gives users autonomy in how they set up the system, subject to the level of control to which they have access. The user of the Aqualab can be best described in Rasmussen's terms as a "constrained autonomous user" (Rasmussen, 1994, p. 177). This implies that the model projected by the interface does not have to accurately map to the structural model. The conflict between pre-existing models that typical users hold of the system, and how the system physical structure is discussed below.

The work role definitions were arrived at after discussions with marketing and technical people at the manufacturer, and discussions with various industry representatives. In the workplace, individuals may combine one or more of these work roles, although it is unlikely an individual would be doing a fractional part of one of these work roles. The generic work roles identified were Environmental Manager, System Manager, Service Technician, Production Engineer and Production Technician.

9.2. THE INTENTIONAL MODELS

An initial attempt was made to eliciting various employees' concepts of the Aqualab through the drawing of rich pictures (Checkland, 1981). The elicitation of mental models at this stage was not to establish a definitive structural model of how the system worked, but to develop an overview of the taxonomic substructure of the knowledge associated with each role. Participants were encouraged *not* to try to reduce their explanation of the system to a lower level of abstraction, but to describe the system at the level with which they interacted with it.

In Section 5, we proposed that both the goals and knowledge held by a person performing a job role would influence their construction of a model of the system. The rich pictures drawn by participants confirmed this proposition, with the resulting objects and relationships reflecting the participant's work role and background knowledge. Marketing people and managers drew high-level views of the system with the Aqualab as

TABLE 1. *Mappings between work role and model*

Role	Goal	Rasmussen's level of Abstraction	Model/view of Aqualab
Environmental manager (client)	Provision of accurate environmental information	Functional meaning-purpose Abstract function	Information only—minimal view of Aqualab abstract function only
System manager (client)	Provision of accurate data from Aqualab to host computer	Abstract function Generalised function	Information system—Aqualab as black box
Production engineer (manufacturer)	Design of logical system that will perform valid chemical tests	Generalised function Physical process Physical form	Aqualab as chemical control system
Production technician (manufacturer)	Implementation of system design	Physical process Physical form	Aqualab as control system
Service technician (client)	Maintenance of physical system	Physical form	Aqualab as conglomeration of physical components and fluids.

a black box and an emphasis on the people in the system. Personnel with a background in chemistry, and a responsibility for ensuring the validity of the chemical tests, viewed the system as an automated chemical laboratory. Electrical engineers with a background in data acquisition, on the other hand, tended to conceive of the system as a data logger with some control functions used to control pumps and valves.

When the participants' views were categorized in terms of their corresponding work roles, a mapping was created between the work role and the model of the system (Table 1). The work roles interacted with the system at different levels of abstraction and each level categorized according to Rasmussen's schema.

The categorization of functional roles according to Rasmussen's levels of abstraction is a useful process, because the different levels represent the boundaries between different types of knowledge required to be able to perform the job. As can be seen in the case of the Aqualab, there is a reasonably close correspondence between job role and level of abstraction. Some job roles, such as the *service technician*, require knowledge of the system at only one level of abstraction and will not, therefore, have to continually cross the boundaries between levels of abstraction. Individuals performing other roles, such as the *production engineer*, have to understand the system on a number of levels of abstraction, and will have to shift between these levels to perform their work role. It is important that the system design facilitate the easy transition between these levels. The

next section examines the process used to maximize consistency between levels of abstraction.

9.3. COMBINING DISPARATE TAXONOMIC SUBSTRUCTURES AND THE DEVELOPMENT OF SYSTEM IMAGES

These prototypical models have different taxonomies, with a number of inconsistent definition of 'objects' in the system. For example, should a sensor be defined as just the detector that measures the result of the chemical test, or should it represent the whole sub-system that performs the test? Difficulties were also encountered in defining the relationships between objects as a result of conflicting mental models of how the systems should be viewed. For example, those people familiar with data acquisition of physical parameters thought the best generalized functional level abstraction of the Aqualab was as a data logger with a number of physical sensors attached. The difficulty with this model is that, although it may simplify the view of the Aqualab for someone at the information-system level, it conflicts with the actual construction of the Aqualab that uses a system of pumps and valves to combine samples and reagents to undertake chemical analysis. Such disjunction between models leads to difficulty in moving between levels of abstraction.

In the case of the Aqualab design process, detailed taxonomic substructures were not developed for each level or role. Rather a somewhat protracted series of meetings—a "naming discourse" (Bucciarelli, 1988)—was conducted with a number of participants with different work roles, in an attempt to develop a generic taxonomy at the abstract-functional and generalized-functional levels of abstraction. In this process, the nomenclature of objects tended to create more conflict than the function of the proposed objects or the relationship between objects. Agreement on nomenclature was often arrived at by the introduction a new term not associated with any of the pre-existing models—e.g. "analyser" and "scheduler".

The advantage of defining new terms is that there are no preconceived mental models associated with the new term. This allows the discussion to be maintained at the appropriate level of abstraction. When the object model is elaborated each object can have a different representation at each level of abstraction—that is the object would be polymorphic. For example, the System Manager would view an 'analyser' as something that is added to a 'schedule' and measures a parameter. On the other hand, a Production Engineer could view an 'analyser' as a series of 'test steps' that define states of a 'plumbing process'.

Once a generic taxonomy was agreed, a class-responsibility-collaborators analysis was undertaken. Use-case scenarios were used to further elaborate the object (class) and the relationships between them. Each of the work roles was represented by a human-actor and included as an object in the analysis. An overview of the conceptual/object model developed by this process is illustrated in Figure 3.

The diagram is divided horizontally between five different levels of abstraction, and divided vertically between human actors and system objects (classes). The arrows between object/actors indicate responsibility/collaboration relationships, with the direction of the arrow indicating how to read the relationship descriptor—e.g. "Scheduler *triggers* Analyser". The detailed CRC description for each class has not been reproduced here.

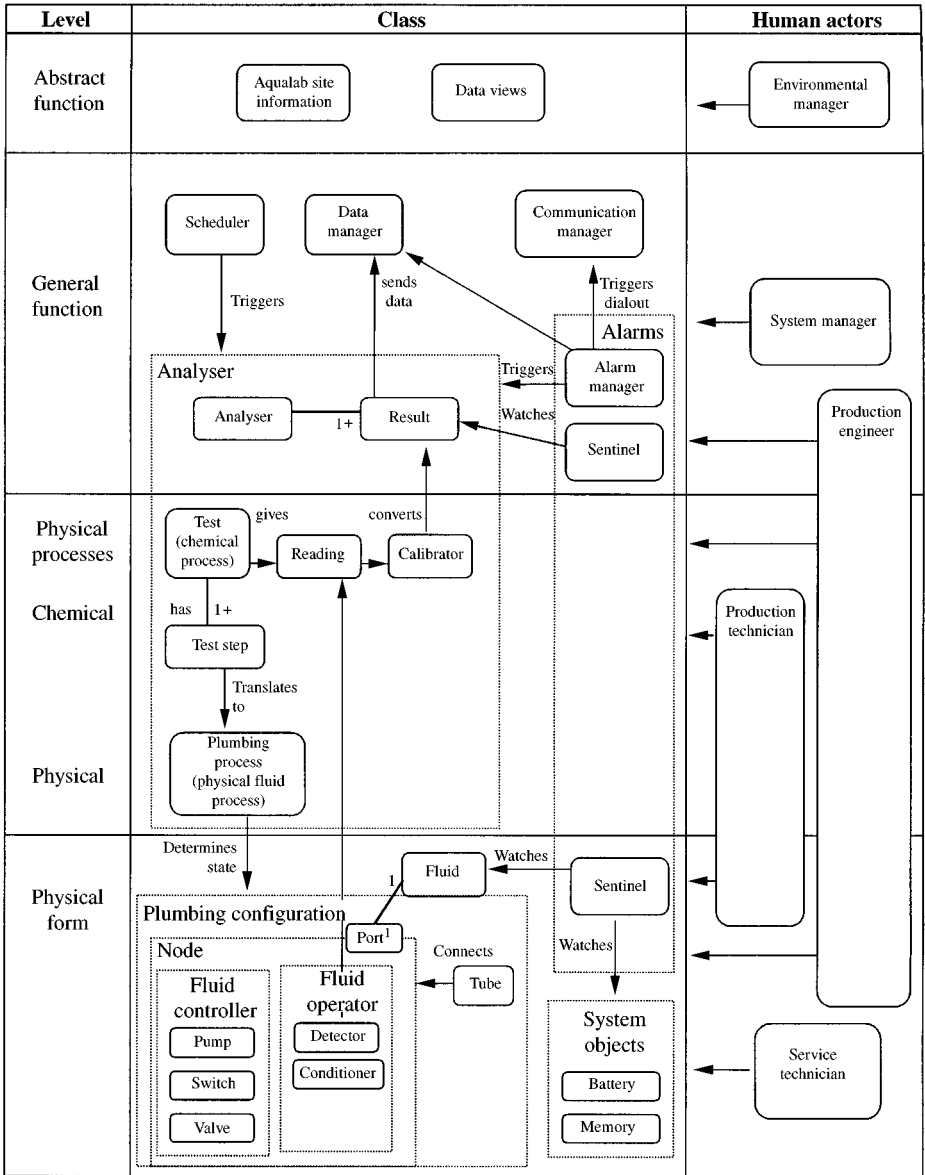


FIGURE 3. Conceptual/Object model from CRC analysis.

Note that some of the human actor’s responsibilities relate only to objects at one level of abstraction, while other actors (e.g. the Production Engineer) have responsibilities at a number of levels of abstraction. In the case of those human actors who need to move between levels of abstraction, it is important that the ‘invariant structure’ of the interface is maintained through the use of ‘stable’ objects that are manifest, albeit in different forms, at different levels. This allows the user to “cross-reference features in separate

displays” (Rasmussen, 1994, p. 180) or system images. In the case of the Aqualab, the stable objects are aggregated objects, e.g. the *analyser* object. The dashed-line boxes indicate aggregations of objects. These aggregated objects provide the ‘stable’ objects necessary for the user to maintain context as he/she moves between levels of abstraction.

The *analyser*-aggregated object illustrates the links between levels. At the generalized function level an *analyser* is an abstract object that has a name (parameter being measured) and other attributes (e.g. unit) as well as one or more associated results. At the physical process level the *analyser* can be viewed on a couple of sub-levels of abstraction—the analyser as a process viewed as a chemical process (“*test*”), and the same process viewed at the physical level as a process of various fluid flows. The Production Engineer needs to move between these various levels of representation of the same ‘object’. Representing an analyser as a single object in the interface allows the engineer to maintain links back to higher levels of abstraction without getting lost in the details of the physical configuration.

At the physical form level the concept of an *analyser* breaks down. However, if the engineer understands the explicit linkages between the object at different levels (e.g. test → plumbing_process → defining plumbing_configuration state, or detector → reading → calibrator → result) then the transition between levels can be made without dissonance. These links to higher levels of abstraction allows the engineer to reason intentionally from top to bottom about the likely form and behaviour of objects at lower levels of abstraction. For example, if the ‘goal’ of the analyser is to measure pH, certain form and behaviour of the system at lower levels will follow: the Production Technician will need to calibrate the analyser against temperature; a test step will involve the measurement of temperature; a temperature detector will need to be present in the plumbing configuration and so on.

The layering of the object model into levels of abstraction allows us to identify those objects in the system of which the human actor needs to be aware—of the “visibility” (Colman & Leung, 1995) of objects in a particular system image. For example, the *system manager* needs to be aware of those objects depicted at the *general function* level. It is these groupings of objects in relationships at a particular level of abstraction that form the views of the system or the ‘system images’. A number of paper prototypes of system images based on different levels of abstraction were drawn up to gauge customer response which would then form the basis of an interface design.

10. Discussion

In complex systems, the differences between the mental models held by various operators can make analysis of the problem domain a difficult task. The elaboration of intentional models held by different types of user can provide the basis for the definition of a conceptual model that accommodates the differences between users. This accommodation is possible because the starting point for analysis are the models different users hold and use, rather than one ‘correct’ structural model of the system. This approach was particularly suitable in the case of the Aqualab because of the following reasons.

- The Aqualab is a moderately coupled system where autonomous users need to maintain a mental model of the state of the system.

- The job roles associated with the Aqualab problem domain were distinct and easily defined both in terms of goals and prerequisite knowledge. This made the discrimination of various intentional models a useful process in coping with the complexity of the system, and provided a basis for thinking about the system in a structured, layered, manner.
- As a control *and* information system, it was necessary to represent both physical and informational aspects of the system. The elaboration of role-specific intentional models provided a method for defining the various levels of abstraction, and combining them into a consistent taxonomy and conceptual model.
- The Aqualab required a number of human–computer interfaces in order to configure, maintain and use it.

The analysis of intentional models may not be as applicable in systems in cases where:

- Job roles are differentiated by non-knowledge-based criteria, leading to layers of the system that do not reflect varying levels of abstraction.
- The system itself is not multi-layered, for example, a pure information system dealing with information at one level of abstraction.
- The system is not complex, allowing the user to maintain an adequate structural model.
- The system does not require the operator to maintain a mental model of the state of the problem domain or requires minimal interaction, allowing a purely functional interaction.
- The system is tightly coupled and all users have to maintain an accurate structural model of the system in case of non-routine situations.

The techniques provide a taxonomy of which objects should be visible to operators performing particular roles, and which attributes and behaviours those objects should exhibit. This gives us a basis for the objects in the interface. The dynamics of the interface can also be suggested by the dynamics of the various mental models and metaphors held by users.

This paper has concentrated on the issue of effective problem domain representation. This paper has not attempted to describe, or demonstrate the existence of, intentional models as a cognitive process, other than to note that structural and functional partial models will be tested for consistency against an intentional framework. As such, an intentional model is a *weak* model in that it does not attempt to provide a set of definitive causal relations for a system. The concept of an intentional model as a intentional explanation providing the framework for partial functional and structural models is not itself a structural model. On a methodological level, the discussion has been limited to the description of taxonomies of the problem domain that are consistent with the scope and level of abstraction of typical users performing particular work roles. If the system image is at the appropriate level of abstraction, and consists of objects that match the user's mental model and also behaves in a way consistent with the model, then the user will have an intentional model of how those objects are likely to behave to achieve a goal. For example, if the system image portrays the 'scene' as a chemical laboratory, with objects representing familiar objects in a laboratory, the user is more likely to conclude

they need to conduct a test with this equipment (a tool suggests its use), than if the 'scene' was portrayed at a lower level of abstraction – as a network of pumps, tubes and nodes.

Further work needs to be done to establish the way in which an intentional framework interacts with functional and structural sub-models; that is, how do the goals we have affect the models we construct of the world? To do this we need to elaborate the process of construction of intentional models. We need to elaborate the learning process by which such models are developed and transformed by the user with the acquisition of further knowledge – and how novices and experts differ in their use of intentional models. Once this process of problem domain task acquisition is better understood, further work could also be undertaken to integrate such a process into the overall process of task execution and evaluation described by a framework such as Layered Protocol Theory.

References

- ANNET, J. & DUNCAN, K. (1967). Task analysis and training design. *Occupational Psychology* **41**, 211–221.
- BAYMAN, P. & MAYER, R. (1984). Instructional manipulation of users' mental models for electronic calculators. *International Journal of Man-Machine Studies*, **20**, 189–199.
- BUCCIARELLI, L. L. (1988). An ethnographic perspective on engineering design. *Design Studies*, **9**, 159–168.
- CARD, S., MORAN, T. & NEWELL, A. (1983). *The Psychology of Human Computer Interaction*. Hillsdale, NJ: Erlbaum.
- CARROLL, J. & MACK, R. (1985). Metaphor, computing systems, and active learning. *International Journal of Man-Machine Studies*, **22**, 39–57.
- CARROLL, J., MACK, R. & KELLOG, W. (1988). Interface metaphors and user interface design. In M. HELANDER, Ed. *Handbook of Human-Computer Interaction*, pp. 67–85. Amsterdam: Elsevier Science Publishers.
- CARROLL, J. & OLSON, J. (1988). Mental models in human-computer interaction. In M. HELANDER, Ed. *Handbook of Human-Computer Interaction*, pp. 45–65. Elsevier Science Publishers.
- CHECKLAND, P. (1981). *Systems Thinking, Systems Practice*. New York: Wiley.
- CHECKLAND, P. & SCHOLAS, J. (1990). *Soft Systems Methodology in Action*. Chichester: Wiley.
- COCKBURN, A. (1996). Structuring of use cases with goals. <http://members.aol.com/acockburn/papers/usecases.ht>.
- COLMAN, A. & LEUNG, Y. (1995). Knowledge Analysis of Tasks as a basis for interface design of complex developmental systems. *Australian Journal of Information Systems* **2**, 2–16.
- CUNNINGHAM, W. (1994). How do teams shape objects? *How do objects shape teams*. OOPSLA Workshop #12 Position Paper <http://c2.com/doc/oopsla94.htm>
- DOBSON, J. E., BLYTH, A., CHUDGE, J. & STRENS, R. (1994). The ORDIT approach to organisational requirements. In M. JIROTKA & J. GOGUEN, Eds. *Requirements Engineering – Social and Technical Issues*. London: Academic Press.
- GARLAND, W. J. (1993). Dealing with the dilemma of disparate mental models. In *Proceedings of the Fifth International Conference on Human-Computer Interaction*, Vol. 2, IV: *Help and Learning*, pp. 903–908. London: Elsevier Science Publishers.
- GENTNER, D. & STEVENS, P. Eds. (1983). *Mental Models*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- GIBSON, J. J. (1966). *The Senses Considered as a Perceptual System*. Boston: Houghton-Mifflin.
- GIBSON, J. J. (1979). *The Ecological Approach to Visual Perception*. Boston: Houghton-Mifflin.
- HALASZ, F. G. & MORAN, T. P. (1983). Mental models and problem solving in using a calculator. In *Proceedings of ACM CHI'83 Conference on Human Factors in Computing Systems Cognitive Models*, pp. 212–216, Boston.
- JACOBSON, I. (1992). *Object-Oriented Engineering: A Use-Case Driven Approach*. Reading, MA: Addison-Wesley.

- JIROTKA, M. & GROUGEN, J. (1994). *Requirements Engineering—Social and Technical Issues*. London: Academic Press.
- JOHNSON, P. (1989). In D. DIAPER, Ed. *Supporting System Design by Analysing Current Task Knowledge*, pp. 160–184.
- JOHNSON-LAIRD, P. N. (1983). *Mental Models*. Cambridge: Cambridge University press.
- LAUREL, B. (1993). *Computers as Theatre*. Reading, MA: Addison-Wesley.
- NORMAN, D. A. (1983). Some observations on mental models. In D. GENTNER & A. STEVENS Eds. *Mental Models*, pp. 7–15. Hillsdale, NJ: Erlbaum.
- NORMAN, D. A. (1984). Stages and levels in human-computer interaction. *International Journal of Man-Machine Studies*, **21**, 365–375.
- NORMAN, D. A. (1988). *The Psychology of Everyday Things*, New York: Basic Books.
- PREECE, J. (1984). *Human-Computer Interaction*. Reading, MA: Addison-Wesley.
- RASMUSSEN, J. (1986). *Information Processing and Human-Machine Interaction—an Approach to Cognitive Engineering*. New York: Elsevier Science Publishing.
- RASMUSSEN, J. (1990). Mental models and the control of action in complex environments. In *Mental Models and Human Computer Interaction 1*, pp. 41–72. ACKERMANN, D. & TAUBER, M. J. eds. Amsterdam: North-Holland, Elsevier Science Publishers.
- RASMUSSEN, J., PEJTERSEN, A. M. & GOODSTEIN, L. P. (1994). *Cognitive Systems Engineering*. New York: Wiley.
- RUMBAUGH, J. et al. (1991). *Object-oriented Modelling and Design*. Englewood Cliffs, NJ: Prentice-Hall.
- SEARLE, J. (1984). *Minds, Brains and Science*. Cambridge, MA: Harvard University Press.
- STAGGERS, N. & NORCIO, A. F. (1983). *Mental models: Concepts for human-computer interaction research: International Journal of Man-Machine Studies* **38**, 587–605.
- TAYLOR, J. (1994). Computer managed assessment: a cognitive science perspective. *Proceedings of the 5th International Conference on Computer Managed Learning*, (unpublished), Melbourne.
- TAYLOR, M. M. (1988). Layered protocols for computer-human dialog I and II. *International Journal of Man-Machine Studies*, **28**, 175–257.
- TAYLOR, M. M., FARRELL, P. S. E., HOLLANDS, J. G. & GAMBLE, H. D. (1998). Perceptual control and layered protocols in interface design I. *International Journal of Human-Computer Studies*, **50**, 489–521.
- VAN DER VEER, G., WIJK, R. & FELT, M. (1990). Metaphors and metacommunication in mental models. In P. FALZON, Ed. *Cognitive Ergonomics: Understanding, Learning and Designing Human-Computer Interaction*, New York: pp. 133–150. Academic Press.
- WIRFS-BROCK, R., WILKERSON, B. & WIENER, L. (1990). *Designing Object-Oriented Software*. Englewood Cliffs, NJ: Prentice-Hall.