Int. J. Human-Computer Studies (1999) **50,** 213–241
Article No. ijhc.1999.0245
Available online at http://www.idealibrary.com on IDEAL

Designing claims for reuse in interactive systems design

A. G. Sutcliffe
Centre for HCI Design, School of Informatics, City University, Northampton Square, London ECIV 0HB, UK. email: a.g.sutcliffe@city.ac.uk

J. M. Carroll
Center for Human-Computer Interaction and Department of Computer Science, 660 McBryde Hall, Virginia Tech, Blacksburg, VA 24061-0106, USA. email: carroll@cs.vt.edu

(Received 9 January 1998 and accepted in revised form 22 November 1998)

Claims have been proposed as a means of expressing HCI knowledge that is associated with a specific artifact and usage context. Claims describe design trade-offs and record HCI knowledge related to a specific design, or artifact, as psychological design rationale. Claims are created in the task-artifact cycle of interactive design and evaluation. Usability evaluation establishes a claim for a specific usage context, but this can restrict subsequent reuse of claims-related knowledge. To widen the scope of reuse the knowledge contained within claims and their associated artifacts has to be classified and generalized. To address this problem a schema and method for classifying claims is introduced. The schema elaborates the description of HCI knowledge in claims and enables reuse by describing the assumptions and dependencies upon which a claim rests. Methods for generalising claims and discovering new claims from existing claims and artifacts were investigated. A factoring method for evolving child claims from parent claims and their usage scenarios is described. This employs a walkthrough technique based on Norman's model of action with questions directed at the contributions a claim makes to usability at different stages in interaction. Factoring promotes evolution of child claims that either address different aspects of task support in the same domain as the parent claim, or development of more general child claims for user-interface design. The relationships between claims are represented in maps to illustrate histories of task-artifacts investigation that lead to claims evolution either via the factoring process or by empirical investigation. The schema and method for claims evolution are illustrated by case studies of claims development in tutoring systems and claims for functional requirements for specification reuse support tools. The paper concludes with a discussion of the contribution that reusable claims can make as a repository of HCI knowledge.

1. Introduction

The context of design in human–computer interaction is one of a perpetual race between knowledge and implementation. For example, direct manipulation interfaces were developed in the mid-1960s (Sutherland, 1963; English, Englebart & Berman, 1967), but scientific analyses of direct manipulation appeared two decades later, indeed, after direct manipulation was already an established engineering practice (Shneiderman, 1982;

1071-5819/99/030213 + 29 $30.00/0

Hutchins, Holland & Norman, 1986). It is a typical pattern in HCI for new ideas to be first codified in exemplary artifacts and only later abstracted into explanations and principles (Carroll & Campbell, 1989).

This somewhat iconoclastic relation between knowledge and implementation has had many impacts on methodological discussions about HCI. For example, it contributed a theory crisis in the 1990s which marginalized traditional cognitive modelling (Carroll & Campbell, 1986; Carroll, 1991; Long & Dowell, 1996). More constructively, it contributed to the growing interest in design rationale as a means of better integrating the development of knowledge representations and the development of interactive systems (Moran & Carroll, 1996). A key aim in HCI design is to bring knowledge from cognitive psychology, and other disciplines such as sociology, to bear upon the design process. The cognitive modelling tradition (e.g. Barnard, 1991) has sought to solve this by development of predictive models so that usability problems can be diagnosed early in the design process. Although cognitive predictive models have been developed, the range of design phenomena that they apply to is narrow, e.g. design of cuts in video clips (May & Barnard, 1995), design of meaningful labels in menus (Kitajima & Polson, 1997). In contrast, claims can cover a wider span of design problems and deliver HCI knowledge to designers in the form of specific design trade-offs. A claim expresses a psychologically based design principle that, although it is less formal than predictive models, does provide knowledge that is more tractable in its expression and, more importantly, related directly to its context of use in a design and scenario of use.

Our work on design rationale has emphasized specifying and observing the usage scenarios that motivate a design project, and articulating the underlying issues and problems in those scenarios. Design issues are schematized as trade-offs, expressed as upsides and downsides associated with given features of an envisioned or extant artifacts as used in a task. For example, contextualized help messages may be easier for a user to interpret and apply in a given situation, but may be less effective in conveying general principles to the user and may be more susceptible to being erroneous [for example, if the context as perceived by the user differs from the context as recognized by the help system (Carroll & Aaronson, 1988)]. The process of making explicit, the implicit trade-offs in a scenario description is called *claims analysis* (Carroll & Kellogg, 1989). Claims express a design treatment with upsides and downside trade-offs that may be expected from implementing the claim; for instance, in a tutoring application which gives feedback on the learner's progress via a goal tree, a claim that 'providing a dynamic fish eye view of a goal hierarchy" may have the upsides of "reducing the complexity of the display and focusing the learner's attention on the part of the hierarchy pertinent to current learning context" but also the downside "that it could conceal parts of the hierarchy of interest to the learner". Options and justifications are presented to the designer, who can then make an informed decision.

In a variety of design projects, drawn chiefly from the domains of tutoring systems and of programming and software design tools, we demonstrated how a scenario-based design rationale of claims can guide design decision-making (e.g. Bellamy & Carroll, 1990; Carroll & Rosson, 1991, 1995; Rosson & Carroll, 1995; Singley & Carroll, 1996). This was achieved by documenting observed or envisioned scenarios as narratives and then analysing the HCI design trade-offs implicit in them. We then used these analyses as a resource to guide design decision making. While these case studies are encouraging,

they raise many methodological issues. For example, are there types of scenarios and claims? How can the identification of scenarios and claims be supported or verified? Can claims analysis provide a bridge between social and behavioural theories and real design reasoning? Some of these issues are addressed in Carroll and Rosson (1991, 1992).

One advantage of claims is that they associate theoretically motivated, and empirically justified knowledge of human computer interaction with a designed artifact and context of use. This makes HCI knowledge available to designers who can recognize the applicability of a claim to a new design context via similarities in user tasks, scenarios or artifacts. Indeed by situating design knowledge in context with a scenario and an example design (the artifact), HCI knowledge may be delivered to the wider community software engineers. A set of methodological issues pertains to the *reuse* of claims analyses. Carroll, Singley and Rosson (1992) showed how design "models" can be construed as abstractions of claims identified in specific scenarios. Specializations of these model-level claims would be reused by designs that instantiate the model. Carroll, Koenemann, Singley and Rosson (1993) demonstrated how claims can be threaded through sets of usage scenarios, and thus how a given claims analysis can be reused and generalized within the scope of a particular design project. This work indicates the kinds of reuse that might be feasible, but does not provide a general framework or a format for organizing claims to support reuse.

Guidelines (e.g. Apple, 1987; ISO 9241, 1997) have a wide range of applicability but their very generality often poses problems of interpretation in context. Furthermore, guidelines rarely promote insight into design problems. Heuristic evaluation (Neilsen, 1993) develops the guideline theme by placing it in a task/usage context for usability assessment; however, heuristic evaluation still requires considerable skill for interpretation of usability problems (Neilsen & Philips, 1993). Another approach has been to develop simple models that may be applied in investigatory methods; for instance, Cognitive Walkthroughs (Wharton, Reiman, Lewis & Polson, 1994). While this approach has gained some acceptance, walkthroughs only provide a general model which requires considerable expertise for interpreting the usability of a design in a specific task context.

Claims offer more targeted delivery of HCI knowledge by virtue of the task-artifact cycle that provides a context. However, the process of generalization from scenarios and specific claims to models is poorly understood. Furthermore, there are different utility trade-offs for knowledge reuse between general but under-specified models (e.g. walkthroughs; Wharton *et al*., 1994) and richer yet domain-specific models (Timmer & Long, 1997). More general models can target a wider range of domains but at a penalty of delivering less knowledge, while the converse is true for more domain-specific models. Since claims have a domain-specific anchor in the artifact context, if the knowledge they contain can be generalized, then insight into the general to domain-specific trade-offs may be gained. The quest to extend the reuse of claims-related knowledge motivated the investigation reported in this paper. We deal with the initial problem in the journey toward effective reuse, namely description and means of design claims for reuse. Problems of reuse library construction, claims retrieval and design by reuse will form the subject matter of subsequent papers.

To facilitate reuse, claims must be formatted and indexed so that retrieval mechanisms can find appropriate claims for a user's new design context. One approach is to adopt

faceted classification schemes from information science that have been successfully applied to indexing reusable software components (Prieto-Diaz, 1991). However, the effectiveness of classification schemes is limited by the consensus on terminology that can be achieved among the indexers and users. Although classification may be a useful starting point, it is unlikely in itself, unless a mutually comprehensible framework for organizing HCI knowledge can be offered to designers and users. The task-artifact cycle suggests a more powerful approach. Claims are associated with usage scenarios that describe either the context from which the claim was originally derived, or the current usage context that a claim embodies in a redesigned artifact. Scenarios enable fragments of claims-related knowledge to be associated with identified cases and classes of users, usage tasks and product features. This provides a rich classification scheme contextualized by artifact examples for interpreting how claims may be reused.

The investigation we report in this paper was motivated by the desire to improve the generality of claims and hence empower reuse of HCI knowledge. First, we have demonstrated that the concept is feasible and has potential utility in the design process, and this is the subject matter of this report. In the longer term, we will have to address the issues of engineering a claims library, such as indexing and classification retrieval and matching claims to a new application context and methods for reuse of claims knowledge in the design process. Furthermore, the utility and reusability of claims as a mechanism for reuse of HCI knowledge would have to be demonstrated by case studies and experiments which show whether application of claims knowledge does indeed improve usability. Within this longer term plan of research, in this paper we focus on the theme of formatting claims for reuse by a combination of classification and contextual description in scenarios. We propose a framework for classifying and describing claims that addresses the scope of potential reuse and helps the future designer-reuser assess the relevance of a claim for a future application.

The users of claims are intended to be software engineers, so another motivation for this research is to spread HCI knowledge beyond the community of human factors specialists; however, effective delivery of design knowledge is a research topic in its own right. To make claims usable by software engineers will require research in explanation, provision of support tools and tutorial environment; for instance, design critics (Fischer, Nakakoji, Otswald, Stahl & Summer, 1993). In the short term, our ambition is to demonstrate the feasibility of claims reuse within a community of HCI-knowledgeable designers. It is important to note that the main thrust of claims reuse is to deliver reusable knowledge that can be applied to designing new user interfaces rather than software reuse. Claims may therefore complement software component reuse (Prieto-Diaz, 1991). The paper is organized as follows. In the next section, we introduce the claims description schema and classification framework. This is followed by designing claims for reuse using the framework. Section 4 then reuses a subset of the claims in a case study example of a new application. The paper concludes with a discussion of the potential of claims as a reusable library of HCI knowledge.

2. Classifying and formatting claims

This section introduces the ontology of HCI knowledge that is anchored in the task artifact cycle (Carroll, Kellogg & Rosson, 1991). Figure 1 schematizes the analysis,

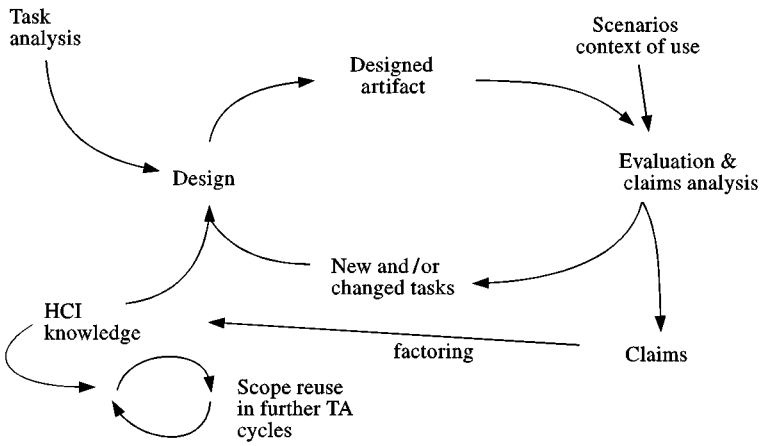

FIGURE 1. Task-artifact cycle showing the derivation of claims from evaluation and empirical studies, and factoring for reuse in related task-artifact contexts.

evaluation and knowledge development activities in the task-artifact cycle: task scenarios that have been empirically evaluated and/or subjected to claims analysis provide requirements for the design of new artifacts. In this figure, the task-artifact cycle has been elaborated to incorporate a reuse sub-cycle that saves claims after analysis/evaluation and adds them to a repository of HCI knowledge.

The major knowledge-related components in the framework are claims, artifacts, scenarios and backing theories. Claims assert trade-offs (upsides and downsides) that pertain to design features. Artifacts are the user-interface design that the claim pertains to. An artifact may be a whole product, or a sub-component thereof. The granularity of a claim is set by its related artifact, i.e. some claims have general scope and describe a complete design, while others may be restricted to a particular feature of the user interface. Scenarios set claims in a specific context of use while backing theories from disciplines such as cognitive science, sociology and anthropology ground and generalize the causal relationships asserted in claims; these disciplinary theories are not themselves elaborated in the ontology.

We distinguish between identified and attested claims. This distinction reflects the status of the claim as HCI knowledge. Claims are initially identified in scenarios envisioned or observed for a given artifact. Identified claims are contextualized by a particular usage scenario; they are testable hypotheses about the usability or functional advantages of the design for that scenario. Identified claims are used to motivate and focus evaluation, analysis and redesign of a situated artifact. Through this process, identified claims may be rejected outright, or they may be empirically and theoretically substantiated and refined. In the latter case, they become attested claims.

We distinguish three kinds of scenarios: problem initiation scenarios, usage scenarios and projected usage scenarios.

Problem initiation scenarios describe the original situation that motivated a cycle of task-artifact investigation; they describe the motivation of investigating a certain type of claim and are the initial problem statement for research. An example is discovery of

a significant usability problem without an obvious solution, that merits further investigation: "the user attempts to reuse the dialogue box but fails to customize the component for the new application. The reason for the user's mistake is not clear".

Usage scenarios describe a sequence of user–system interaction, based on empirical evidence; they illustrate a problem instance in a design, e.g. "the user tries to find the bullet format command in the word processor, but after considerable searching gives up and uses indented paragraphs instead".

Projected usage scenarios describe anticipated interaction with the redesigned artifact; they may be based on the designer's expectations or grounded in actual observation of user interaction with a prototypes, e.g. "having selected the text the user chooses the double columns command and the text is displayed in double-column format. This is easier to use than the previous artifact that displayed text in a single column".

In the co-evolution of tasks and designed artifacts, a problem initiation scenario motivates investigation. These are similar to critical incidents and cause observed usability problems; alternatively, a claims analysis may arise from success stories when users cite the effectiveness of a particular design. Usage scenarios are gathered from empirical studies and help to produce identified claims. Identified claims guide artifact redesign and the projection of new usage scenarios that contextualize the attested claim. With further evaluation the cycle is repeated.

The overall process is composed of three major steps organized in two alternative pathways, as illustrated in Figure 2. First, classification that helps to identify outstanding issues for further investigation. Classification may then be followed by redesign and evaluation of claims to address downsides in the original claim. This step is described in previous reports of task-artifact cycle investigations, and so is not considered further in this paper. Alternatively, the dependencies and design issues identified during classification may lead to new "child" claims and further task-artifact cycles via a factoring process described in Section 3 of this paper. Both of these alternative pathways create families of related claims that may be composed into maps of investigation histories that, *inter alia*, create schemata of related chunks of HCI knowledge. This is described further in Section 4. Factoring claims entails investigating the contribution the claim and its associated artifact make to supporting user–system interaction. This leads to new child claims at a different level of abstraction and hence spawns a new task-artifact cycle. Next we address claims classification and documentation.

2.1. CLAIMS DESCRIPTION SCHEMA

To structure the HCI knowledge in claims for reuse, we propose an explicit claims description schema. The top level of the schema has a number of slots which are filled by free-format narrative, and by a set of terms following the practice of controlled vocabularies of faceted classification. Our starting point was the simple claims format used in many previous publications, i.e. a claim description followed by one or more upsides (benefits) and one or more downsides (disadvantages or potential problems). This kernel schema is expanded as follows.

1. Claim ID: unique identifier for the claim.
2. Title: name give by claim author.

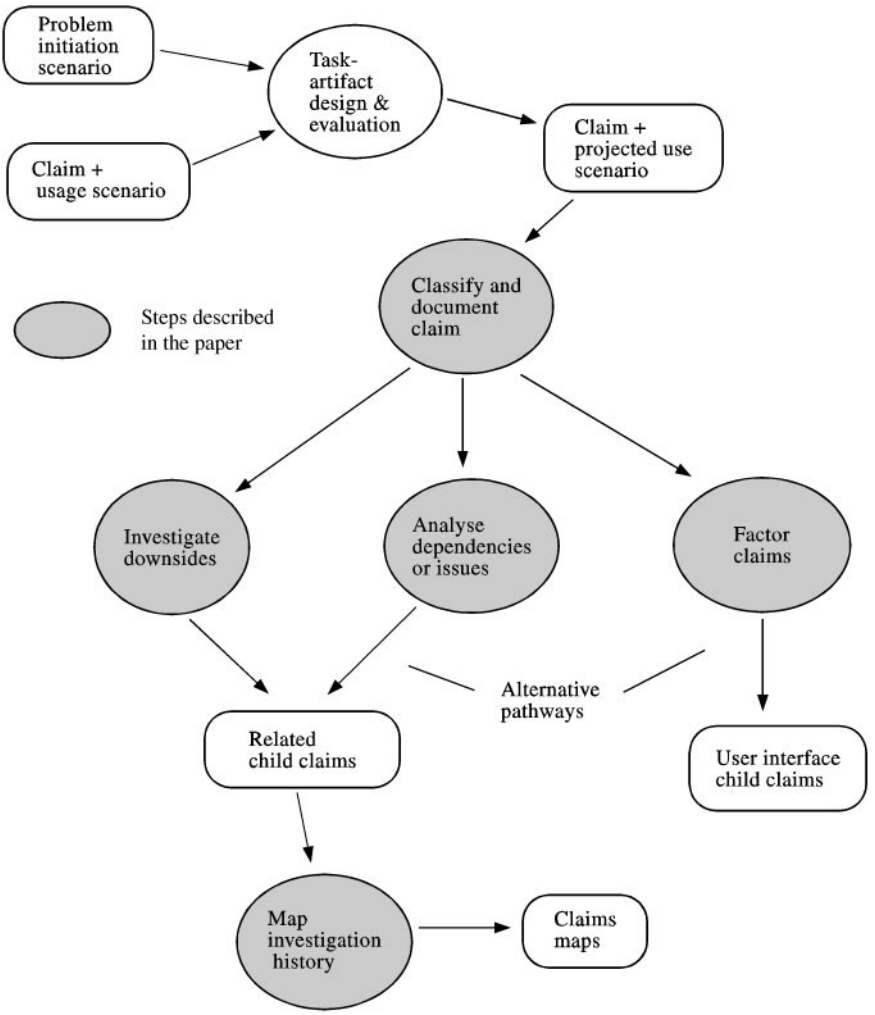

FIGURE 2. Process steps and pathways for claims analysis and evolution.

3. Author: researcher(s) who developed the original claim.
4. Artifact: brief description of the product/application in which the claim originated.
5. Explanation: natural language description of the claim.
6. Upside (s): positive effect of using the claim on a system goal or usability.
7. Downside (s): negative effect of using the claim on a system goal or usability.
8. Scenario: scenario in which the claim was derived or is now used.
9. Effect: desired, and preferably measurable, system goal/usability effect that the implemented claim should achieve.
10. Dependencies: other design problems that have to be solved to achieve the claim's effect.

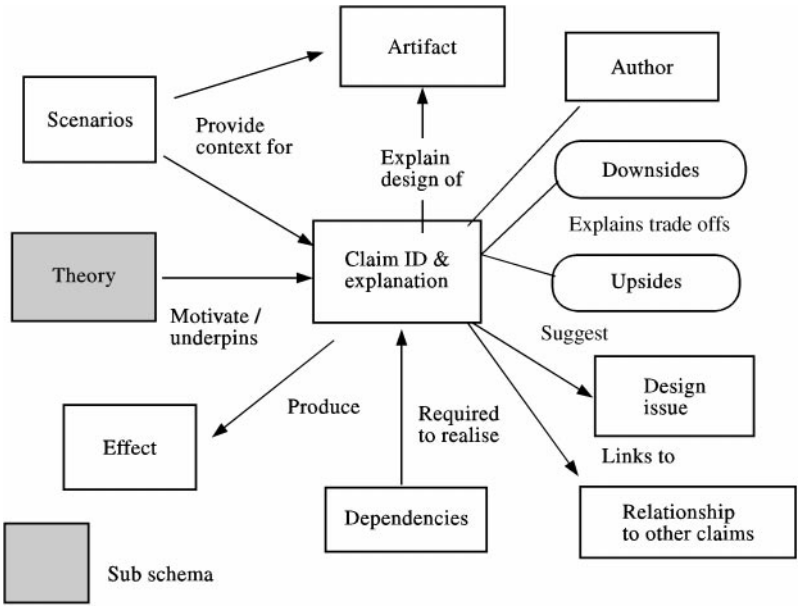

FIGURE 3. Claims description schema components with relationships to show how different arguments can contribute to the classification.

11. Issues: design issues influenced by the claim.
12. Theory: underlying theory explicitly referenced by the author (sub-schema).
13. Relationships: inter-claim links that describe how claims evolved during a history of investigation (see Section 4)

All slots are filled with informal, natural language text, the extent of which depends on the depth of analysis provided by the claim's author. The classification schema is extended by hypertext links to contextual material in a variety of media (e.g. screen dumps, video of the artifact), the product itself, relevant design documents, as well as to related claims. Slots 6 and 7 are typically iterated, as two or three upsides and downsides are usually stated. Slot 12 may be filled either by a reference to the underlying theory or it may act as a link to a claims map that describes a series of claims which share a common theoretical base and problem initiation. This slot is expanded in the theory sub-schema. The relationship slot 13 contains a pointer to other claims that are associated with the current claim in terms of their subject matter or the history of investigation. The use of this slot is covered in Section 4. The claims schema components and relationships are illustrated in Figure 3.

Assuming claims are input in their minimal form of a description, scenario of use, upsides and downsides, the process starts with initial documentation and then scoping the artifact's features and the claim's effect by stating the usability goals or expected outcome on improved task performance. Claims are classified by the following steps.

1. Initial documentation.
   - Record the claim's author. Access to the claim's author is vital to get a satisfactory understanding of its origin and motivation.

- Assign a unique identifier and fill in the slots for author, artifact, claim, description, scenario, upsides and downsides.
- Trace the artifact and scenario to provide links to an example of the artifact and/or recordings of its use. Similarly, links to evaluation data from which the scenario was taken should be added.

2. Scoping the effect.
   - Describe the intended effect of implementing the claim. This should be implicit in the upsides of the claim, usage scenario and possibly the background theory. The intended effect is stated at a high level of satisfying end-user goals or improving task performance.
3. Explore dependencies and issues.
   - Trace dependencies implicit in the claim by asking "What features/knowledge/assumptions does implementation of this claim depend on?". For example, implementing a strategy may depend on availability of certain actions, knowledge or system functions.
   - Identify assumptions about the properties of the user necessary for the claim to be effective by asking "What knowledge should the user possess for the interaction (with the design) to be successful?"
   - Describe design issues that are not directly addressed by the claim but are important usability features for delivering the claim's effect. Issues are posed as questions often point to new claims that need to be investigated.
4. Assign theoretical derivation.
   - Specify the theory that motivated the investigation that led to the claim or that may underpin its origins. Generally, theory relates to a whole claim, although the upside and downside slots in a claim schema may have different relationships to backing theory. For instance, a feature posting current learning goals provides feedback and a model to a person pursuing these goals. This benefits learning as described by the theory of Lewis and Anderson (1985).

We regard the linking of claims to specific backing theory as particularly important, and propose a sub-schema for classifying six different kinds of relationships between clauses and backing theory.

(a) Theory prediction—the claim is a direct consequence of some theory's prediction in a design context, e.g. pedagogical theory predicts that active engagement in constructive learning environments should be more effective (Fischer *et al*., 1993)
(b) Grounded theory—indirect consequence, or interpretation of a theory by the designer. Memory schema theory suggests that reuse of design templates should improve developer efficiency and problem solving (Maiden & Sutcliffe, 1994)
(c) Experimentally derived—claims based on results of a controlled experiment; explicit task-related labelling of commands reduces user errors (Franzke, 1995)
(d) Empirically grounded—claims that rest on analysis of observations (quantified or qualitative) with a scoped context; this covers case studies, ecological studies and usability evaluation. Most published studies of claims fall into this category.
(e) Proof of concept—claims that rest on the fact that they can be interpreted and constructed as an artifact that works effectively.

(f) Operationally grounded—the claim has been demonstrated to work in practice by either marker success, process improvement, or quality improvement. The claim may be derived from intuition or by theoretical inspiration; see (a) and (b).

This classification allows us to integrate and differentiate a variety of approaches in HCI and software engineering. For instance, industrial practitioners might create claims based on their experience and product success in the market [i.e. (f)], whereas software engineers may embody a claim for a novel algorithm in an implementation that is demonstrated to run efficiently [see (e)]. One claim may have more than one theoretical justification. Theory is important because it allows authority of the claim to be checked and if necessary questioned. Another benefit of this classification is that claims can be related to certain theoretical assumptions, and this may be useful in their interpretation. For instance, grouping related items in lists may be justified by recourse to a memory theory of natural categories (Rosch, 1985); alternatively a slightly different claim for the same design issue could be linked to script schema theory (Schank, 1982). One claim and artifact order lists by classes, while the other is in sequential or procedural order. Knowledge of the backing theories helps in understanding the claim author's motivation. We acknowledge that theory may be of more interest to academic users and validators of claims libraries than practical benefit to designers who reuse claims; however, we argue that more knowledge is better than less. The designer can always choose to ignore the link.

2.2. CLAIM CLASSIFICATION EXAMPLE

To illustrate usage of schema, the following claim is taken from Singley and Carroll (1996).

- Claim ID: Colour-coded Telegraphic Display.
- Author: Singley, M.; Carroll, J.M.
- Artifact: MoleHill tutor – GoalPoster tool (see Figure 4).
- Explanation: the presentation of individual goals in the window is telegraphic, several words at most. However, the learner can expand any of the telegraphic goals (through a menu selection) to display a fuller explanation of why the goal is worthwhile pursuing or not. Thus the system provides both shorthand feedback on correctness and access to further help.
- Upside: provides persistent feedback on the correctness of actions as well as access to further information.
- Downside: learners must learn the display's feature-language and controls.
- Scenario: the learner is trying to create an UpperCaseTextPane, the system infers that the learner is "searching for relevant classes and methods" and posts this phase in the GoalPoster, the learner selects the posted goal to display an expansion of why the goal should be pursued and how it can be advanced, the learner resumes work navigating to the Window class and causing this sub-goal to be posted in the GoalPoster.
- Effect: improved learning by provision of appropriate feedback.
- Dependencies: tracking user's progress in learning tasks, known goal structure.

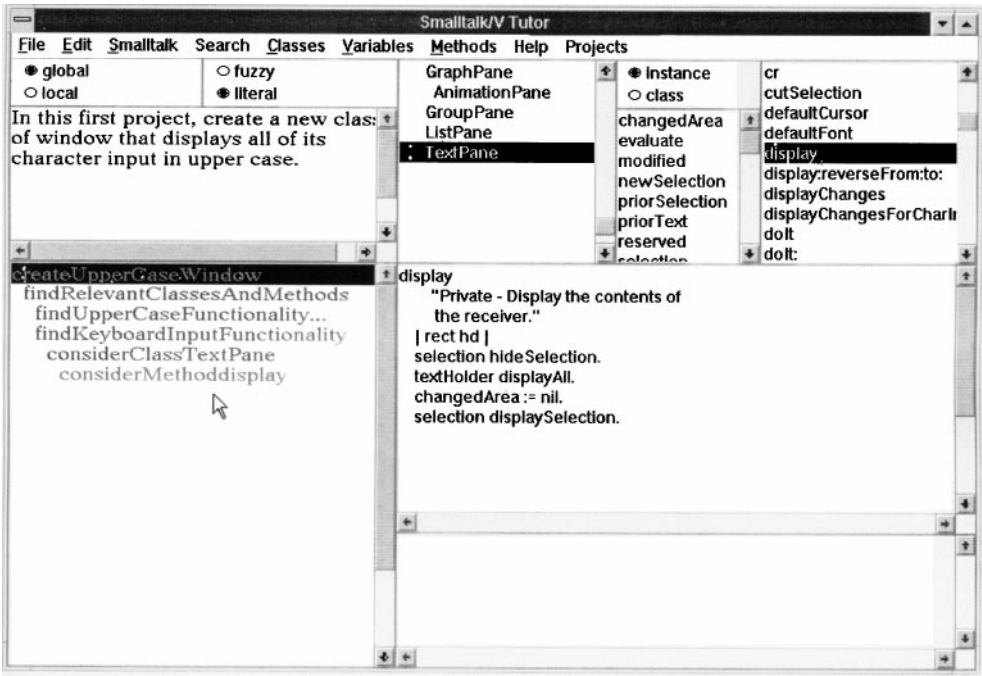

FIGURE 4. Illustration of the Goalposter tool in the MoleHill tutoring system for Smalltalk programming.

- Issues: what sorts of feedback displays are effective and under what circumstances? is hue coding effective for indicating the three-way distinction between achieved sub-goals, apparently correct sub-goals, and apparently incorrect sub-goals?
- Theory: feedback in discovery-based learning (Lewis & Anderson, 1985).

The effect clause describes the identified and quantifiable effect that application of the claim and its accompanying artifact should deliver in terms of usability, learning or task performance. Dependencies list the assumptions made by the designer that need to be fulfilled to enable the artifact's operation and are analysed by asking a question about what knowledge source, information or input the artifact requires to produce the output. Dependencies may also be suggested by examining the interfaces between sub-systems and components in the artifact. The GoalPoster provides a display for the user's progress in the learning task. To do so it depends on acquiring knowledge about which task the user has completed and the learning task being undertaken. Progress events must be detectable and the task needs to have a known structure for feedback progress to be given. The issues slot records design questions that are not directly referred to in the claim but are implicit in the artifact. These may be user-interface features vital for delivering the claims effect, so that in the GoalPoster, the tree structure of the feedback display and the telegraphic colour coding are important, for lower level, design issues. Further advice may be given by attaching sub-claims or design rationale to answer the

questions raised. In this manner, the structuring process encouraged by the claim schema can help to guide the development and articulation of HCI knowledge.

No direct theory attribution was given for this claim by Singley and Carroll (1996); however, from their paper we can infer that the theory grounding falls into (b)—grounded theory in the classification described in Section 2.1, an indirect consequence or interpretation of a theory by the designer. In this case, the initial theory posited that discovery-based "learning by doing" environments were an effective means of learning support for certain tasks. Learning feedback is important to maintain orientation in the learning task, hence the GoalPoster artifact. The initial theory (Lewis & Anderson, 1985) suggests that the artifact should produce the claim upside and effect, i.e. that feedback on correct actions should reinforce learning. The claim, and the artifact design, also results from investigating how such feedback may be delivered by empirical studies on this variant of the GoalPoster tool. The grounding is therefore a combination of indirect theoretical motivation (b) and empirical evidence (d). This is a common pattern in task-artifact cycle investigations. We now turn to the method for factoring claims for further reuse.

3. Developing claims for reuse

To investigate the potential of a claim for application in different usage contexts a walkthrough method is proposed which assesses the generalized contribution a claim and its associated artifact can make to stages in interaction. We describe a method for factoring claims that acts as a "gedanken"† experiment to explore possibilities for evolving further claim schemas from claims already articulated. This builds on Norman's (1986) model of action that has been used to motivate questions for claim identification in scenarios (Carroll & Rosson, 1992). Norman's model specifies categories for interaction; however, it deliberately abstracts away any reference to the presentation aspects of user interfaces. We turn Norman's model around using it to classify artifacts with respect to stages of interaction. The method requires HCI expertise to describe new claims that are indicated by the method, hence our view of the reuse process is that designing claims for populating a reuse library would be carried out by HCI experts, whereas design by reusing claims could be carried out by software engineers and user interface designers who may not be HCI specialists. The model (see Figure 5) provides a description of the cycle of user–system interaction that can be "walked through" by the HCI specialist with question prompts at each stage that indicate the type of new claim that may be discovered at each stage.

The process for claims factoring is dependent on the maturity of the investigation and the level at which the claim is stated. Some claims may be high-level conceptions, for instance, of learning strategy; while others pertain to fine scale features of user interfaces. Furthermore, first-generation identified claims for the problem initiation scenario might comprise a relatively immature insight into a class of design phenomena. Generally, we advise that claims analysis is allowed to progress through several generations before

†gedanken from the German denken = to think, is a means or way of thinking about process or problem through a series of steps to reach a satisfactory resolution.

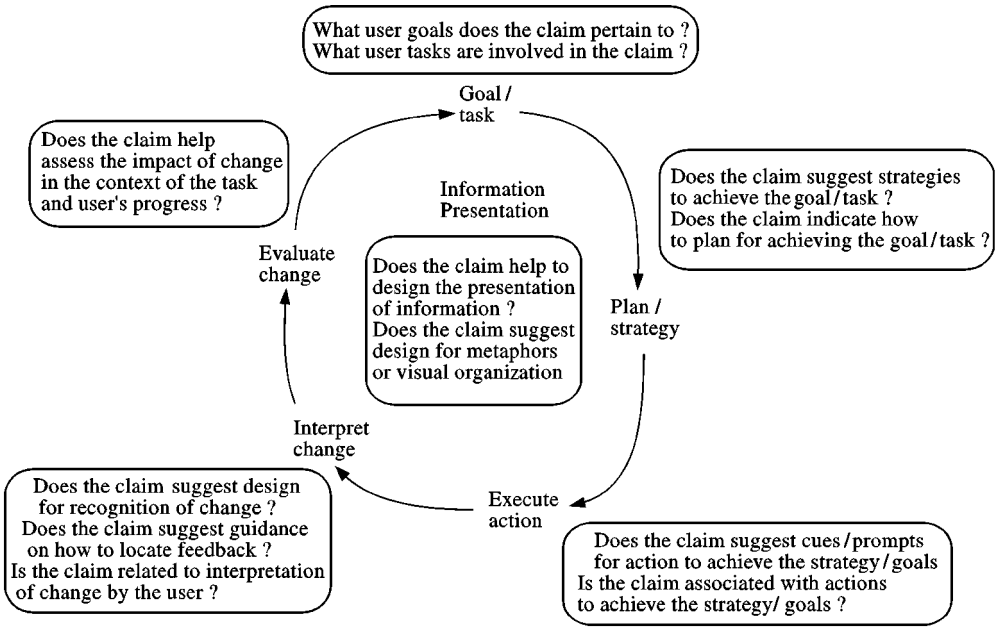

FIGURE 5. The claims factoring walkthrough method, composed of questions attached to stages in the process model of interaction (after Norman, 1986). Norman's stages to recognize and evaluate change have been merged as they tend to implicate the same artifact components.

factoring is attempted, since encoding immature knowledge is likely to lead to incomplete and ambiguous descriptions. However, the act of factoring can itself stimulate further investigation and theory development so we do not wish to preclude the interleaving of factoring and evolving claims.

Claims are factored by using the walkthrough to prompt questions that assess the contributions an artifact and its associated claim could make at different stages of interaction. By definition, claims refer to particular features, but factoring claims may suggest extensions or generalizations of the original claim, new artifacts, descriptions, upsides, downsides, scenarios and so forth. The questions illustrated in Figure 5 are used to step through the task described in the originating claims scenario. Eack task initiates an interaction cycle which should achieve its goals; however, goal related questions may suggest sub-goals in the task structure. The original goal may, for instance, be underspecified, or a different means of supporting the task may become apparent. Questions within each cycle focus more directly on the user-interface components rather than task support functionality. The contribution that the claim and the artifact make in supporting interaction are assessed using the model to structure questions. Hence, if the artifact is primarily intended to support formation and monitoring user's goals, it has a goal task contribution, whereas a display feature is more likely to support interpretation and evaluation of change.

Assessment of one artifact may result in discovering or deriving several new "child" claims with accompanying artifact features from the parent. Taking the colour-coded

telegraphic claim, the walkthrough elaborates the following potential contributions. Using the previously cited scenario, the domain task goal is "the learner is trying to create an UpperCaseTextPane" within which there is a sub-goal that the system supports—"locate appropriate class for reuse" and a learning task goal "explain next goal in the domain task"—the system infers that the learner is "searching for relevant classes and methods" and posts this phase in the GoalPoster, the learner selects the posted goal to display an expansion of why the goal should be pursued". Factoring so far has exposed a complex goal structure in the learning and domain tasks. Note the claim is related to the domain task as well as the learning task, so we may suggest that the goalposter artifact may support (a) software reuse by helping users find reusable components and (b) help programmers learn design by reuse in Smalltalk. Theoretical underpinnings suggest that the claim belongs more closely to the latter. Following the "explain next goal in the domain task" the factoring questions point first to how the artifact supports strategy formation. This is achieved by the goal poster tree that displays a map of the domain tasks for the user to follow, and by highlighting the next goal. Execution is not directly supported in the GoalPoster, as this is carried out in the domain task by selecting the appropriate class. Interpretation is supported by updating the goal tree on successful completion of action; in addition expansion of the telegraphic text promotes understanding about the effect of action. Finally, evaluation is helped by placing the user's progress in the context of the overall domain task by highlighting achieved goal on the tree diagram. Presentation of information indirectly supports most interaction stages, by providing an adaptable presentation that can be expanded to supplement the user's understanding for either planning or interpretation. The walkthrough produced the following list of explanations of the GoalPoster's contribution at different stages of the explain next goal task.

- Artifact: GoalPoster tool.
- Goals/task: learning Smalltalk, explain next goal.
- Planning/strategy: the goal tree and status indicator enable tracking of the user's progress, and hence planning future learning strategy.
- Execution: no direct support, but domain task actions are registered to update the user's progress on completion of learning sub-tasks.
- Interpretation: labels on the tree display help monitoring progress in a learning task; expansion of the telegraphic text display helps users understand the learning goals in more depth.
- Evaluation: providing a goal tree of the learning task helps the user understand how much of the task has been completed, the next steps to achieve and where to backtrack to, if necessary.
- Presentation: telegraphic text is a mnemonic cue for the goals, yet economical in screen real estate; colour coding is used to segment the goal tree into sub-structures and indicate the user's current position. Expansion of telegraphic text provides an adaptable presentation for more detail.

The effect of the walkthrough is to assess the claim's contributions and derive new contexts for reuse. New claims arise from this process either within the same domain as new ways of supporting task sub-goals are discovered or more general claims may evolve

that relate to user interaction support rather than a task. The new claims are documented following the guidelines and schema elaborated in Section 2.1. The walkthrough is suitable for derivation of child claims for user-interface artifacts from claims which have a wider, more functional scope. Claims that identify functions in their upsides often rely on complex artifacts to deliver their functionality. The walkthrough investigates how different aspects of the user interface contribute to delivering the functionality. Consequently, child claims may not require all components or the original artifact, so the feature most closely implicated in fulfilling the new claim's effect are investigated. Sometimes such features may not be easy to identify.

For high-level claims, the feature may be synonymous with the product, but in many cases the issues clause may refer to design of a discrete UI component, hinting at a new claim-artifact combination. Indeed, the issues clause in the original claim often gives a valuable pointer towards factoring analysis. Note the component may not be visible at the user interface, e.g. a dialogue strategy. This process may iterate as child claims are derived from the parent, although claims that are already associated with UI components may not be amenable to further factoring. Once the factoring is complete the scenario and feature descriptions are revisited to document the scope of the claim's future applicability. To illustrate the process, the GoalPoster artifact and its associated claims are evolved into a new task artifact combination.

Notice that the colour-coded telegraphic claim was motivated by support for learning; however, the factoring indicates evolution towards a new artifact with, perhaps, a more general applicability as a progress monitoring tool that may be suitable for a wide variety of user interfaces. The GoalPoster tool constitutes a generalizable feedback display that monitors user progress in a task and then provides feedback for interpretation and evaluation. Furthermore, the colour-coded telegraphic display provides a novel design concept for presentation. Classification and factoring proceed hand in hand to produce new claims, as follows.

Progress Tree Display claim

- Author: Sutcliffe A.G.
- Parent claim: colour-coded telegraphic display.
- Explanation: users need to track their progress through a complex task and may lose their place. This hinders task completion as they lose the thread of the goals they have achieved and have to back track. A progress indicator display of a goal tree prevents them from losing their place.
- Upsides: provides persistent feedback on status of user action within a tree structured task as well as access to further information; supports effective work by relieving user's working memory from progress tracking.
- Downsides: users must learn the tree structure display and goal labels; the current location may be difficult to find in massive trees.
- Artifact: Tree monitor tool (link to artifact).
- Projected use scenario: the user interface presents a tree structure of the user's task with a status indicator showing the current sub-goal. The user is distracted by a colleague asking questions and forgets

where he was in the task. The user looks at the progress tree display and recognizes where he was in the task; however, the next step cannot be remembered, so the user expands the currently highlighted telegraphic goal through a menu selection. The system provides a longer description of the task step as well as shorthand feedback on current status, the user remembers the task and continues.

- Effects: improved provision of appropriate feedback on task progress.
- Dependencies: detecting the user's progress in a task, known goal structure.
- Issues: what sort of feedback displays are effective and under what circumstances? Is hue coding effective for indicating the three-way distinction between achieved sub-goals, apparently correct sub-goals, and apparently incorrect sub-goals? Is a tree structure the most effective means of visualizing progress in a task?.
- Theory: Observability principle (Thimbleby, 1990; see also ISO 9241 1997, part 10).

This claim has evolved from its original motivating example into a more general and widely applicable claim for user interfaces of any system with hierarchical, deterministic tasks. Note that the newly identified claim includes a projected use scenario that represents the designer's expectation for how the child claim and its associated artifact will function; it may also be viewed as a problem initiation scenario that summarizes the need for further investigation in a new task-artifact context. The issues clause indicates further directions in which the claim might evolve, for instance, there are issues of colour coding the display for structure and indicating status; design of telegraphic, yet mnemonic goal names; and visualization of the goal tree itself.

Factoring may be carried out by the original claim author; alternatively, a designer may carry out the factoring while searching the library for suitable claims for a new application. The heuristics and analysis stages interact, as investigation with the interaction model often leads to further factoring of the claim or elaboration of the original claim description.

4. Relating families of claims

As claims accrue by the process of factoring and task-artifact cycle investigations, tracing the claims inter-relationships is useful to help designers to understand the history of investigation and the thread of theory-based empirical investigation on which they are grounded. The relationship between a claim, its derivation history and background theory can be understood from a map of the investigation history that gave rise to the claim. New claims may arise by one of three routes: first by factoring, secondly by research to address the downsides in the parent claim, and finally by investigating design issues in the parent. Whatever the derivation, claim maps illustrate the motivation for each investigation that led to a claim and show associations between related claims. Figure 6 illustrates the point with a map of claims produced by the downside route.

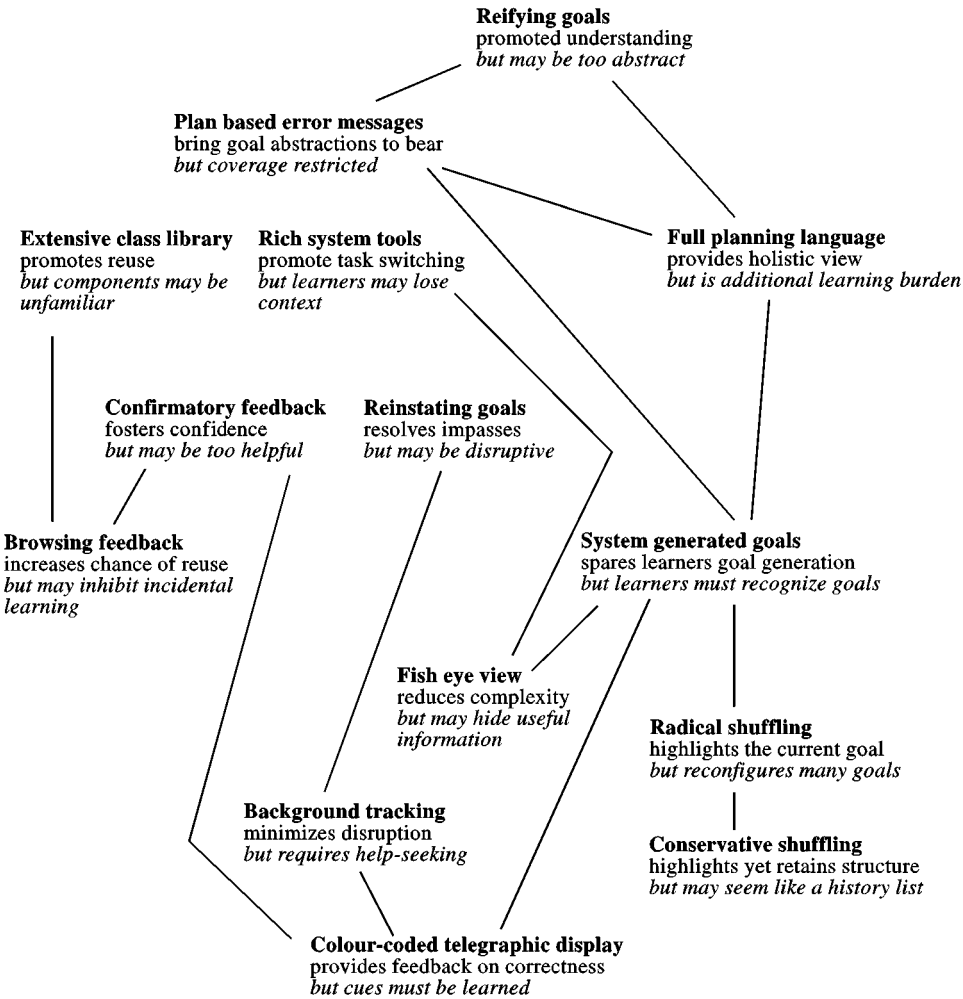

FIGURE 6. Map illustrating the relationships arising from investigation into Smalltalk programming support tools (after Singley & Carroll, 1996). Inter-claim links are represented in the relationship slot of the schema, described in section 2.1.

The colour-coded telegraphic display claim arose from research into the problem of feedback provision in an active tutoring environment for Smalltalk programming, the MoleHill tutor (Singley & Carroll, 1996). Subsequent investigations were motivated to solve downsides of the initial, identified claim. Three parents are linked to the colour-coded telegraphic claim. First, the system generated goals describe display design trade-offs for making the user's learning goals clearer. This claim can be traced back to full planning languages which described the trade-offs for explicitly expressing a plan of what the user was to learn, and plan-based error messages that describe giving users error messages when they make mistakes and showing where they have had mistakes in the context of the learning plan. In this manner the parentage of a claim can be traced

back through a history of investigation to help understand the motivation for a claim and the evidence upon which it rests. The second parent, background tracking, is motivated by a pedagogical concern to provide feedback on the correctness of the user's actions, but it does so in a separate window to minimize disrupting the user's task. The motivation for the third parent, confirmatory feedback, can be traced back to support for browsing and retrieving reusable components from the Smalltalk class library. The design gave feedback on the relevance of a class for the user's current problem and confirmatory feedback helps guiding the user's choice of components but it may overdo the effect and inhibit exploratory learning.

The colour-coded telegraphic claim was motivated by a display design problem, namely how to give feedback that is clear, updateable but not too intrusive, while supporting two different parts of a tutoring environment. The claims map makes reference to a substantial investigation history which is anchored by the theoretical motivation of minimalist instruction (Carroll, 1990).

Describing this claim illustrates some of the problems in designing HCI knowledge for reuse. Claims are anchored in a history of investigation that shaped design solutions for a particular problem. However, the solution involved several components and may recruit different theories and claims during the investigation. For example, the fisheye view claim rests on research by Furnas (1986) motivated by display design rather than learning support systems. If claims can be exported from one task-artifact cycle to another, this raises the question about which "seeding artifacts" are exported and how claims are attached to them.

Our advice is to first export the original artifact with the child claim, but reference the contribution of relevant artifact components in the claim explanation, dependencies and issues. In this manner, one artifact may serve as an example for several claims. This is acceptable when the motivation for developing a claims repository is to reuse HCI knowledge alone. However, when software as well as HCI knowledge reuse is intended, artifacts for child claims will have to be designed *de novo*. The relationship between claims and software artifacts depends on the artifact's software architecture. For instance, in a modular design it may be possible to separate code from a parent artifact that relates to a child claim. When this is not possible, a new artifact may have to be designed or the scope of application of the new claim to the original artifact should be made clear in accompanying documentation. Taking the GoalPoster example, the child claim uses most of the parent artifact's components, i.e. a mechanism for detecting user actions, display of a goal tree and an updating component that highlights the next goal to complete in a task sequence. Hence, the artifact re-design may only require making the software more reusable, for instance, by configuration facilities to customize the action detection mechanism, and load different task hierarchies. To illustrate the need for more radical re-design, if factoring were taken further to evolve a child claim that addressed the usability of the expandable telegraphic display alone, then a display artifact would be designed with functionality limited to expanding telegraphic text in goal trees.

We have already suggested that the colour-coded telegraphic display could be reused more generally for display artifact design. However, this claim contains implicit information that is not explained, for instance, the role of colour coding is used to indicate the user's current location in a goal tree. Linking to a colour image of the artifact may bring this information to the user's attention; but a safer approach is to illustrate and

document implicit knowledge as dependencies and design issues. Alternatively, we may wish to factor the claim by documenting its (and the artifact's) contribution at different stages of interaction, e.g. colour coding, alphanumeric mnemonic coding, tree structure feedback indicators, all contribute to interpretation and evaluation in progress monitoring. This approach was elaborated in Section 3. This leads to evolving claims at different levels of abstraction such as UI functions and task/domain support that we have addressed in other aspects of our work (Sutcliffe & Carroll, 1998).

The design change motivated by a claim should only impact the usability consequences (upside/downsides) that are enumerated in the claim, so an important question for a reused claim is the extent to which it, and the associated design features, will account for the possible usability problems in a new task and context. This can only be thoroughly answered by further empirical testing. In the colour-coded telegraphic display claim the goal tree, colour or adjustable telegraphic feedback may all contribute equally, but there is no evidence that all these design features are vital for solving the usability problems of misunderstood feedback, although that is just one component of the solution to the learning problem that motivated the original investigation.

The problem focus and implicit design issues need to be addressed when factoring the raw material of claims, especially as the feature implicated in the claim depends on the author's viewpoint. In this case, the claim was motivated by investigations into learning promoted by the MoleHill tutoring system, hence the feature identified is a high-level component—the GoalPoster tool which keeps the student informed of their progress through goals in a Smalltalk programming task. The claims map can be used to trace claims from one task-artifact context to another enabling traceability back to a claim's original motivations as well as providing the designer with a view of the maturity of investigations within any one cycle.

5. Designing claims for reuse

In this section, we illustrate the use of the claims classification schema and analysis method in two different contexts. One owes its heritage to tutoring system research, the other concerns Computer-Aided Software Engineering (CASE) tools.

5.1. LEARNING-RELATED CLAIMS

This claim lies at an early stage in the lineage of tutoring system research by Singley and Carroll (1996) motivated by interest in minimalism and guided discovery learning (Carroll, 1990; Papert, 1980). The identified claim was extracted from evaluation of the PROUST intelligent tutoring system (Johnson, 1985).

- Claim ID: plan-based error messages claim.
- Author: Singley, M.K.; Carroll J.M.
- Artifact: PROUST Intelligent Tutoring System (Johnson, 1985).
- Description: system-generated error messages couched in terms of goals and plans.
- Upsides: brings appropriate planning abstractions to bear when they are needed.

spares the learner the burden of generating explicit goal and plan descriptions.

- Downsides: learners may develop superficial understanding when errors are not made.
  goal-plan dialogues are restricted to errors and are not under learner control.
- Scenario: the user is learning Pascal by going through exercises. The system spots an error and provides advice in terms of the goals and plans that should solve the problem.
- Effect: improved learning of programming plans and procedures.
- Dependencies: detection and diagnosis of user errors, model of user errors.
- Issues: reactive learning, tutor initiative, contextual advice.
- Theory: indirect theory grounding (Papert, 1980; Lewis &Anderson, 1985).

Identifying the key artifact feature in this claim is difficult. At a high level, the artifact provides a capability for monitoring and interpreting both what people are doing and their underlying intent, plus generating explanatory messages that help learning by advising on solutions to system-detected user errors. The concrete feature would be the specific software components that might implement this capability. The overall effect of the claim depends on several system features and picking out which of these are vital to the claim depends on the author's knowledge of the artifact. Walkthrough analysis indicates the following possible evolutionary paths for the claim.

- Artifact: PROUST tutor/advisor.
- Goals/tasks: learning LISP, mentoring and correcting errors in a learning task.
- Planning/strategy: detects an error, provides an appropriate plan and advice to repair the error.
- Execution: provides advice text for correction, linked to original error.
- Interpretation: system detection of errors alerts users to learning difficulties.
- Evaluation: providing advice to correct the error with an explanation of the mistake helps the user to comprehend the problem.
- Presentation: advice is provided with links to the example program code so the locus of the mistake can be identified.

The walkthrough analysis points out how different artifact features make contributions to the overall effect of the claim. Each process stage could be factored into more general claims which apply to systems beyond tutoring systems, such as error detection monitors for safety critical tasks.

5.2. CASE TOOLS CLAIM

This claim relates to the Smalltalk object-oriented program development environment and more specifically to how the class library supports software development. As class libraries are supported by most object oriented languages and their development environments, this claim could be applicable to all OO CASE tools.

- Claim ID: Class Library claim.
- Author: Rosson, M.B.; Carroll, J.M.; Bellamy, R.K.E.

- Artifact: Smalltalk programming environment, class library browser.
- Description: Smalltalk's extensive library of classes and methods.
- Upside: encourages programmers to reuse standard decomposition in designing new functionality.
- Downsides: programmers must familiarize themselves with the existence and functional roles of many components; but programmers may search the library ineffectively; and may make non-optimal choices among candidates classes.
- Scenario: the user began to search the more than 100 TexPane methods from the top down, and was soon modifying, testing and ultimately rejecting the first few methods on the list that seemed relevant from their names alone. She finally did find the method she needed, buried in the middle of the list …
- Effect: increased programmer productivity from reuse.
- Dependencies: development of a reuse library.
- Issues: library composition, indexing, search facilities.
- Theory: IR searching strategies (Marchionini, 1995), analogical problem solving (Gentner, 1983) indirectly grounded theory.
- Relationship: Analogy-based matching, matching to generic models.

This claim has three downsides which point to different problems, so there may be a case for factoring sub-claims related to Class Library search and explanation facilities. The claim is also of very high level (e.g. reuse problems in general) and the downsides suggest further problems to be investigated, such as why programmers make sub-optimal choices. As a result of factoring, claims may be placed in a derivation map which grows as further investigation takes place and sub-claims are proposed to resolve problems posed by the downsides of super-claims. The effect statement depends on the researcher's viewpoint. The one stated above is a high-level software engineering concern; however, from an HCI viewpoint, the effect might be to promote learning of the class library or to make problem solving more effective by reusing appropriate examples. The relationship slot points to two claims which are functionally related to the problem of OO class reuse, i.e. the problem of retrieval that can be solved by analogy based matching and reuse of generic models which come from the claims family illustrated in Figure 6.

Walkthrough analysis produces the following elaborations.

- Artifact: Class Library browser.
- Goals/tasks: retrieving appropriate classes, browsing library to explore contents.
- Planning/strategy: search for specific class by a query language; browse search for a class using library tree structure.
- Execution: retrieve class.
- Interpretation: N/A.
- Evaluation: possible extension to provide explanation of class functionality and application context.
- Presentation: the class is presented with links to an example program so that its context of use can be understood.

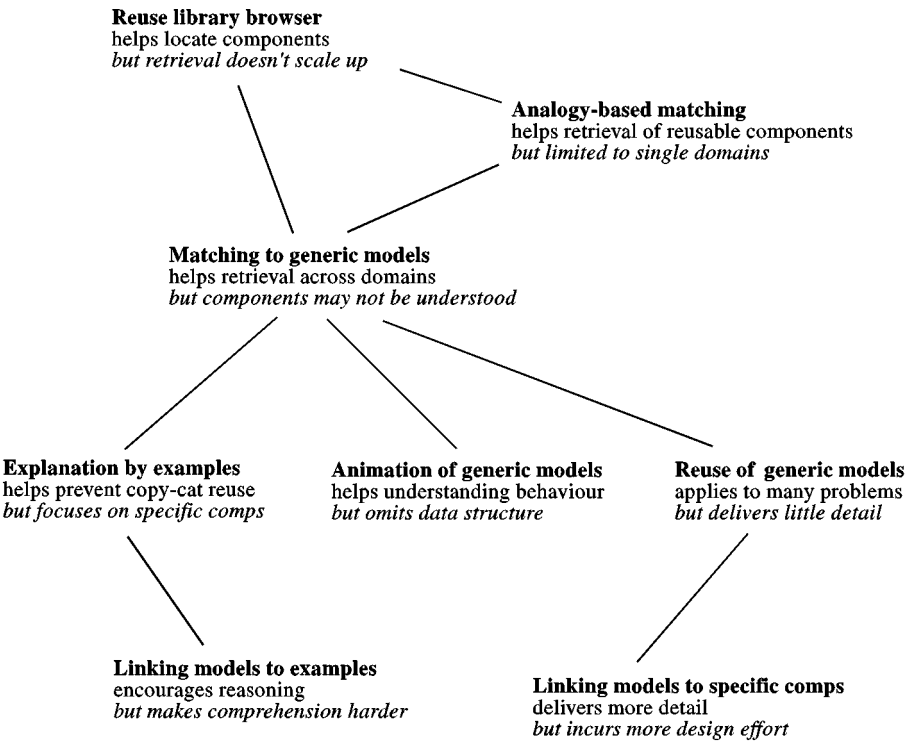

FIGURE 7. Claims map for investigations into reuse support tools.

The walkthrough suggests that further functionality not present in the original library browser could be designed, such as different types of search mechanism and critiquing facilities to help explain reusable components. In this case, other claims-related knowledge is recruited from our previous work on design of reuse support environments (Sutcliffe & Maiden, 1994; Maiden & Sutcliffe, 1994).

A claims map illustrated in Figure 7 shows the relationship between the reuse library browser and claims that started with the problem of promoting reuse of software component, with upsides of improving developer efficiency and downsides of increasing errors by "copy cat reuse". Attempts to solve the matcher problem started by use of analogical matching between an original and new application (Maiden & Sutcliffe, 1992) but this was limited by the need to model two domains in detail, so that downside was addressed in a new matching mechanism using generic models (Maiden & Sutcliffe, 1996). This posed further problems of how generic models could be reused which were answered by developing artifacts (and new claims) for explanation, animation and linking generic models to specific examples (Sutcliffe & Maiden, 1994, 1998). This evolved into the reuse critic which tried to mitigate the "copy cat" reuse by critiquing designs for inconsistencies.

The Smalltalk browser claim (see Figure 7) can be linked into this thread of investigation that tried to address the downside of long search times with browsers as class

libraries scale up. This downside was addressed by developing a search tool that used structure matching (Maiden & Sutcliffe, 1996), an efficient constraint-based search engine for retrieving analogically related artifacts from reuse repositories. However, to make structure matching work a set of generic abstractions of applications had to be proposed. Abstract models were difficult for designers to understand, leading to the problem of "copy-cat" reuse where the generic model was used with any customization leading to errors. This downside was addressed by developing explanation tools that animated the generic models and linked them to specific, but related, software components. This example shows how a claims map can help reuse by giving the context of a task artifact investigation and how such a summary can make the results of previous work more explicit. Reformatting using the claims schema enables the contributions to be assessed and integrated with related work, in this case Carroll and Rosson's (1991) investigations into reuse support tools. Furthermore, factoring the Browser claim by the interaction model analysis suggests opportunities for exploitation of the reuse support tool artifacts by generating new claims that may be applied in different domains, such as information retrieval. Factoring posed questions about how key features of an artifact may contribute to the original claims as well as pointing towards contributions that lower-level components of the artifact may make to improving usability in information retrieval. Other families of claims are described in Sutcliffe and Carroll (1998) where the problem of indexing claims for reuse is also addressed.

The experience of using the method demonstrated that claims can be discovered by the walkthrough process; however, specification of new claims does require HCI knowledge to help interpret the method and claims description schema. We envisage that this part of the reuse cycle, designing claims for reuse, will be an expert role. Design by reuse, which is not covered in this paper, remains to be tested (but see Sutcliffe & Carroll, 1998).

6. Discussion

This work focuses on codifying, sharpening, and applying the concepts-in-action that already typify practice; i.e. the arguments developed for design decisions, either within the design process or as design memoir reflections on that process, and scenarios embodying user concerns and requirements. We seek to refine these concepts-in-action, to make them better practical methods and better abstractions for action science. The task-artifact cycle reifies the background for practical activity in the field, which sets an appropriate target for scientific abstractions and a context for developing methodology. The yield is a potentially thorough, systematic and generalizable design representation, and a framework for design analysis that is also richly contextualized. Psychological design rationale reifies the implicit claims embodied in the use of artifacts, and hence provides a conceptual framework for distinguishing between designers' intentions and design achievements. It not only offers a means of more systematically producing design memoirs, and furthermore, of producing them in advance of producing the design, but also specifies the conceptual elements of a causal theory of artifacts in use. The yield is an explicit quasi-detective design argument detailing what was done and why, while grounding the design in a framework of prior artifacts and embodied principles.

We argue that the framework for documenting psychological design rationale will become a more effective way for promoting the wide application of HCI knowledge because it provides a rich description that links a reified design context on hand—the task and artifact—with theoretical motivations and a research history. Furthermore, documenting the dependencies and effects makes the designers' (or researchers') assumptions explicit. This paper's contribution is a modest start on the difficult problem of designing claims for reuse. Essentially, our suggestion is to apply an extended form of cognitive walkthrough to inductive claims analysis. This "tool for thought" may stimulate new claims and task-artifact cycles; however, it is still dependent on the designer's knowledge for interpretation of applicability in a new context. In our future work, we will expand our design for reuse framework by positing generic models of task contexts that can be used to index claims and their associated artifacts. For that purpose, we will investigate the applicability of the generic models of application classes and tasks proposed by Sutcliffe and Maiden (1994). This approach may enable claims and accompanying artifacts to be indexed to generic patterns in the object-oriented sense (e.g. Gomaa, 1995; Coad, North *et al*., 1995) thereby extending the scope of their reuse.

Claims as a means of reuse will have to face the problems encountered in software component reuse, such as obtaining a critical mass of claims to persuade designers to buy into the process of design by reuse, the not-invented-here syndrome, need for management incentives and legal and copyright issues (see Arnold & Frakes, 1992). Many of these problems require managerial solutions which are well-known, e.g. resourcing reuse library development, and providing incentives for sharing reusuable knowledge. Other problems are how to understand and interpret reusable knowledge (Maiden & Sutcliffe, 1992). Here, claims have an advantage in being a structured natural language description which facilitates communication between users and designers from different communities. The penalty in natural language is the potential for misinterpretation; however, claims situate understanding in the context of a scenario and the designed artifact to anchor interpretation with examples.

Clearly, our proposal for claims indexing and factoring raises the need for tool support for a claims/artifact reuse library. This in turn necessitates a mechanism for retrieving claims as the library scales up. The classification framework we have adopted has proved moderately successful in software engineering reuse libraries employing conventional search mechanisms (Sorumgard, Sindre & Stokke, 1993). Part of our future research agenda will be to address tool support by extending previous work on requirements document management (Carroll, Alpert, Karat, Van Deusen & Rosson, 1994) to create a hypertext system to link claims by the multiple access paths described in the schema. More advanced search facilities will be researched by applying structure matching (Maiden & Sutcliffe, 1996, 1998) to retrieve claims according to a set of interlinked schema facets that map to a claims description, thereby efficiently narrowing the search. Clearly, there will be trade-offs between richly linked hypermedia libraries of claims that may provide more leads to design knowledge for solving a particular problem; on the other hand, more links increases complexity and may make the system unusable for designers who want quick answers. Here an efficient search mechanism can help. Indeed, we already have claims that we can apply to this problem (Sutcliffe & Carroll, 1998). In this related work, we have proposed a means of indexing claims to libraries of generic

models that provide a novel access path to a claims library, i.e. claims are linked to classes of applications. Indexing to generic models also necessities generalizing the claim and its scenario to increase the scope of the claims potential reuse.

Empowering reuse of HCI knowledge by claims may augment the contribution to interactive system design of theory-based models of users and interaction (see ICS: Barnard, 1991; PUMS: Young & Howes, 1996). While theory-based models contribute fundamental understanding about the cognition of human–computer interaction, their design advice addresses only small fragments of interaction. Such models have been criticized as recruiting HCI knowledge too slowly for the rapidly changing demands of HCI designers (Long & Dowell, 1996). We do not envisage claims superseding theoretically grounded modelling apparatus to HCI; on the contrary, claims provide a conduit for wider utilization and dissemination of such knowledge. Take the label following effect discovered by Kitajima and Polso (1997). This can be recapitulated as a claim that "command and object labels should be visible and relevant to the user task" and illustrated by selected parts of the Cricketgraph artifact that actually do conform to the claim. This claim is strongly grounded in the LICAI predictive theory that can explain its upsides and downsides, such as users are strongly cued by labels for task actions they expect to do; but they might find locating labels more difficult when many similar labels are present. Derivation of such a claim shows a strong link between theoretical research in display-based cognition and the task-artifact cycle.

However, the original restriction we observed on the granularity of phenomena addressed by cognitive theory can be addressed by claims. HCI knowledge that advises how to support users' tasks or learning must, by the granularity of the phenomena being addressed, have a looser connection with predictive theory. Here claims encapsulate knowledge with the artifact and scenario as context and the framework makes the link to backing theories of various types explicit. The route for validation of this knowledge is inherent in the task artifact cycle; and moreover, the route to recruiting new knowledge from a variety of scientific approaches is provided by the theory classification.

Claims investigations have tended to be driven from investigations of usability or learning problems, probably because scientific investigation is driven by problems that need solutions or because evaluation detects problems that give rise to new claims in subsequent (re)designs. The factoring method makes a break with this tradition by discovering claims that relate to design features that promote good usability, but it feeds on claims that described large-scale artifacts and discovers claims for the user-interface components. One evolution of our method is to look at how the upsides of claims may be improved and factored to provide more knowledge about which design features work effectively and why. The subject matter of claims will always be influenced by their authors.

Over time, we envisage the claims library will be built by aggregation of experience from different domains and at different levels of abstraction, i.e. different tasks and different aspects of user interaction. The factoring method and claims description schema we have reported in this paper enables sharing and structuring claims-related knowledge and a mechanism for discovering new claims at the user-interface level. The experience of using the method has uncovered a distinction between claims that related more strongly to the application functionality and user's task, while the claims evolved by the method relate to more general user-interface features. We are in the process of investigating how

task and functionally related claims can be indexed using abstract models of applications (see Sutcliffe & Carroll, 1998). Different indexing mechanisms and reuse pathways may be necessary for different types of claims.

The other perspective we considered in the introduction to this paper was HCI knowledge as guidelines and principles. The task artifact cycle and the claims classification provide a more powerful vehicle for delivering effective HCI knowledge because the designer receives not only a psychologically based design rationale but also contextual information from the scenario and artifact. This paper has advanced that by embellishing the design information with dependencies and issues, and more importantly, placing a single claim in a web of interconnected HCI knowledge. The framework gives a common classification scheme and a process for factoring psychological design rationale enabling a wider scope of reuse and a conduit for recruiting HCI knowledge from many authors. The proposal in this paper represents the feasibility stage of a claim; our next step is to develop the repository artifact upon which the meta-claims for reuse of HCI knowledge will be based.

The authors wish to thank Sean Arthur, Mary-Beth Rosson and John Dowell for comment on drafts of this paper. This work was carried out while AGS was on sabbatical at Virginia Tech.

References

ANDERSON, J. R., BOYLE, C. F., FARRELL, R. & REISTER, B. J. (1989). Cognitive principles in the design of computer tutors. In P. MORRIS, Ed. *Modelling Cognition*, pp. 93–134. New York: Wiley.

APPLE (1987). *Human Interface Guidelines*: *the Apple Desktop Metaphor*. Reading, MA: Addision–Wesley.

ARNOLD, R. & FRAKES, W. (1992). Software reuse and re-engineering. In Arnod R., Ed, *IEEE Tutorial on Software Re-engineering*. Los Alamitos, CA: IEEE Press.

BARNARD P. J. (1991). Bridging between the basic theories and the artifacts of human computer interaction. In J. M. CARROLL, Ed. *Designing Interaction*: *Psychology at the Human Computer Interface*, Cambridge: Cambridge University Press, pp. 103–127.

BELLAMY, R. K. E. & CARROLL, J. M. (1990). Redesign by design. In D. DIAPER, D. GILMORE, G. COCKTON & B. SHACKEL, eds. *Proceedings of 3rd IFIP Conference on Human–Computer Interaction Interact 90*. 27–31 August. Cambridge.

BELLAMY, R. K. E. & CARROLL, J. M. (1992). Structuring the programmer's task. *International Journal of Man-Machine Studies*, **37,** 503–527.

CARROLL, J. M. (1990). *The Nurnberg Funnel*: *Designing Minimalist Instruction for Practical Computer Skill*. Cambridge, MA: MIT Press.

CARROLL, J. M. ed. (1991). *Designing Interaction*: *Psychology as the Human–Computer Interface*. New York: Cambridge University Press.

CARROLL, J. M. ed. (1995). *Scenario-Based Design*: *Envisioning Work and Technology in System Development*. New York: Wiley.

CARROLL, J. M. & AARONSON, A. P. (1988). Learning by doing with simulated intelligent help. *Communications of the Association for Computing Machinery*, **31,** 1064–1079.

CARROLL, J. M. ALPERT, S. R., KARAT, J., VAN DEUSEN, M. D. & ROSSON, M. B. (1994). Raison d'Etre: embodying design history and rationale in hypermedia folklore, an experiment in reflective design practice. *Library Hi-Tech*, **12,** 59–70 & 81.

CARROLL, J. M. & CAMPBELL, R. L. (1986). Softening up hard science: reply to Newell and Card. *Human–Computer Interaction*, **2,** 227–249.

CARROLL, J. M. & CAMPBELL, R. L. (1989). Artifacts as psychological theories: the case of human–computer interaction. *Behaviour and Information Technology*, **8,** 247–256.

CARROLL, J. M. & KELLOGG, W. A. (1989). Artifact as theory nexus: hermeneutics meets theory-based design. In T. BICE & C. H. LEWIS, Eds. *Proceedings of CHI89*: *Human Factors in Computing Systems*, Austin, TX, 30 April–4, May pp. 7–14. New York: ACM.

CARROLL, J. M., KELLOGG, W. A. & ROSSON, M. B. (1991). The task-artifact cycle. In J. M. CARROLL, Ed. *Designing Interaction*: *Psychology at the Human–Computer Interface*, pp. 74–102. New York: Cambridge University Press.

CARROLL, J. M., KOENEMANN-BELLIVEAU, ROSSON, M. B. & SINGLEY, M. K. (1993). Critical incidents and critical threads in empirical usability evaluation. In J. ALTY, D. DIAPER & S. P. GUEST, eds. *People and Computers VIII*, *Proceedings of the HCI'93 Conference*, pp. 279–292. Cambridge: Cambridge University Press.

CARROLL, J. M. & ROSSON, M. B. (1991). Deliberated evolution: stalking the View Matcher in design space. *Human–Computer Interaction*, **6,** 281–318.

CARROLL, J. M. & ROSSON, M. B. (1992). Getting around the task-artifact framework: how to make claims and design by scenario. *ACM Transactions on Information Systems*, **10,** 181–212.

CARROLL, J. M. & ROSSON, M. B. (1995). Managing evaluation goals for training. *Communications of the ACM*, **38,** 40–48.

CARROLL, J. M., ROSSON, M. B., CHIN, G. & KOENEMANN, J. (1997). Requirements development: stages of opportunity for collaborative needs discovery. *Proceedings of the 2nd ACM Symposium on Designing Interactive Systems*, Amsterdam.

CARROLL, J. M., SINGLEY, M. K. & ROSSON, M. B. (1992). Integrating theory development with design evaluation. *Behaviour and Information Technology*, **11,** 247–255.

CARROLL, J. M., SMITH-KERKER, P. S., FORD, J. R. & MAZUR-RIMETZ, S. A. (1987/1988). The minimal manual. *Human–Computer Interaction*, **3,** 123–153.

COAD, P., NORTH, D. *et al.* (1995). *Object Models*: *Strategies*, *Patterns and Applications*. Englewood Cliffs, NJ: Prentice-Hall.

ENGLISH, W. K., ENGLEBART, D. C. & BERMAN, M. L. (1967), Display based techniques for text manipulation, *IEEE Transactions on Human Factors in Electronics*, **HFE-8,** 5–15.

FISCHER, G., NAKAKOJI, K., OTSWALD, J., STAHL, G. & SUMMER, T. (1993) Embedding computer-based critics in the contexts of design. In S. ASHLUND, K. MULLET, A. HENDERSON, E. HOLLNAGEL & T. WHITE, Eds. *Proceedings of INTERCHI '93*, pp. 157–163, New York: ACM Press.

FRANZKE, M. (1995). Turning research into practice: characteristics of displayed based interaction. In *Proceedings of CHI'95*: *Human Factors in Computing Systems*, pp. 421–428. ACM Press.

FURNAS, G. W. (1986). Generalized fisheye views. In M. MANTEI & P. ORBETON, Eds. *Proceedings of CHIO'86*: *Human Factors in Computing Systems*, pp. 13–23. New York: ACM.

GENTNER, D. (1983). Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, **5,** 121–152.

GENTNER, D. STEVENS, A. L., Eds. (1983). *Mental Models*. Hillsdale, NJ: Erlbaum.

GOMAA, H. (1995). Reusable software requirements and architectures for families of systems, *Journal of Systems and Software*, **28,** 189–202.

HUTCHINS, E. L., HOLLAN, J. D. & NORMAN, D. A. (1986). Direct manipulation interfaces, In D. A. NORMAN, & DRAPER, S. W., eds. *User Centred Systems Design*: *New Perspectives on Human–Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Assocs.

ISO 9241 (1997). *Ergonomic requirements for office systems Visual display terminals*. Parts 10, 11, 16 International Standards, parts 1–9, 12–15, 17, Draft Standards; International Standards Organisation, Switzerland, Available from National Standards Organisations.

JOHNSON, W. L. (1985). *Intention-based diagnosis of errors in novice programs*. Unpublished doctoral dissertation. Yale University, New Haven, CT.

KITAJIMA, M. & POLSON, P. G. (1997). A comprehension based model of exploration. *Human Computer Interaction*, **12,** 345–390.

LEWIS, M. L. & ANDERSON, J. R. (1985). Discrimination of operator schemata in problem-solving: learning from examples. *Cognitive Psychology*, **17,** 26–65.

LONG, J. B. & DOWELL, J. (1989). Conceptions for the discipline of HCI: craft, applied science, and engineering. In A. SUTCLIFFE & L. MACAULAY, eds. *People and Computers V*, *Proceedings of*

the 5th Conference of the BCS HCI SIG. Nottingham, 5–8 September. Cambridge: Cambridge University Press.

LONG, J. B. & DOWELL, J. (1996). Cognitive Engineering and Human Computer Interaction. *The Psychologist*, **9,** 313–317.

MAIDEN, N. A. M. & SUTCLIFFE, A. G. (1992) Exploiting reusable specification through analogy. *Communications of the ACM*, **35,** 55–64.

MAIDEN, N. A. M. & SUTCLIFEE, A. G. (1994). Requirements critiquing using domain abstractionss. In J. SIDDIQI, Ed. *Proceedings of International Conference on Requirements Engineering*. Colorado Springs, USA, pp. 184–193. New York: IEEE Press.

MAIDEN, N. A. M. & SUTCLIFEE, A. G. (1996). Analogical retrieval in reuse-oriented requirements engineering. *Software Engineering Journal*, **11,** 281–292.

MARCHIONNI, G. (1995). *Information Seeking in Electronic Environments*. Cambridge: Cambridge University Press.

MAY, J. & BARNARD, P. (1995). Cinematograpy and Interface Design. In K. NORDBYN, P. H. HELMERSEN, D. J. GILMORE and S. A. ARNESEN, Eds. pp. 26–31. *Proceedings of 5th IFIP TC 13 International Conference on Human–Computer Interaction*, Lillehamer, 27–29 June, London: Chapman & Hall.

MORAN, T. P. (1983). Getting into the system: external–internal task mapping analysis. *Proceedings of CHI'83 Conference*, pp. 45–49.

MORAN, T. P. & CARROLL, J. M. Eds. (1996). *Design Rationale*: *Concepts*, *Methods and Techniques*. Hillsdale, NJ: Erlbaum.

NEILSEN, J. (1993). *Usability Engineering*, New York: Academic Press.

NEILSEN, J. & PHILIPS, V. L. (1993). Estimating the relative usability of two interfaces: heuristic, formal and empirical methods compared. In *Proceedings of INTERCHI-93*, pp. 214–221. New York: ACM Press.

NORMAN, D. A. (1986). Cognitive engineering. In D. A. NORMAN & S. W. DRAPER, eds. *User Centered System Design*: *New Perspectives on Human–Computer interaction*. Hillsdale, NJ: Erlbaum.

PAPERT, S. (1980). *Mindstorms*: *Children*, *Computers*, *and Powerful Ideas*. New York: Basic Books.

PRIETO-DIAZ, R. (1991). Implementing faceted classification for software reuse. *Communications of the ACM*, **34,** 88–97.

REIMAN, J., YOUNG, R. M. & HOWES, A. (1996). A dual space model of iteratively deepening exploratory learning. *International Journal of Human Computer Studies*, **44,** 743–775.

ROSCH, E. (1985). Prototype classification and logical classification: the two systems. In E. K. SCHOLNICK, eds. *New Trends in Conceptual Representation*: *Challenges to Piaget's Theory*. Hillsdale, NJ: Erlbaum.

ROSCH, E., MERVIS, C. B., GRAY, W., JOHNSON, D. & BOYES-BRAEM, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, **7,** 573–605.

ROSSON, M. B. & CARROLL, J. M. (1995). Narrowing the specification-implementation gap in scenario-based design. In J. M. CARROLL, Ed. *Scenario-Based Design*: *Envisioning Work and Technology in System Development*. pp. 247–278. New York: Wiley.

ROSSON, M. B. & CARROLL, J. M. (1996). The reuse of uses in Smalltalk programming. *ACM Transactions on Computer-Human Interaction*, **3,** 219–253.

SCHANK, R. C. (1982), *Dynamic Memory*: *A Theory of Reminding and Learning in Computers and People*. Cambridge: Cambridge University Press.

SHNEIDERMAN, B. (1982). The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology*, **1,** 237–256.

SINGLEY, M. K. & CARROLL, J. M. (1996). Synthesis by analysis: five modes of reasoning that guide design. In T. P. MORAN & J. M. CARROLL, eds. *Design Rationale*: *Concepts*, *Techniques*, *and Use*, pp. 241–265. Mahwah, NJ: Erlbaum.

SMITH, R. B. (1987). The Alternate Reality Kit: an example of the tension between literalism and magic. In J. M. CARROLL & P. P. TINNER, eds. *Proceedings of CHI + GI'87*: *Conference on Human Factors in Computing Systems and Graphical Interfaces*. Toronto April 5–9. New York: ACM.

SORUMGARD, L. S., SINDRE, G. & STOKKE, F. (1993). Experiences from application of a faceted classification scheme. In R. PRIETO-DIAZ & W. B. FRAKES, Eds. *Advances in Software Reuse, Proceedings of the 2nd International Workshop on Software Reuse*, pp. 116–122. Los Alamitos: IEEE Press.

SUTCLIFFE, A. G. (1996). User-Centered Safety critical design. In J. DOBSON Ed. *Proceedings of CSR Conference' Human Factors in Safety Critical Systems*. Burgstock, Switzerland: Chapman & Hall.

SUTCLIFFE, A. G. (1997). *Using domain knowledge in interactive system design*. Technical Report 97/12, Centre for HCI Design, School of Informatics, City University, London, UK.

SUTCLIFFE, A. G., BENNETT, I., DOUBLEDAY, A. & RYAN, M. (1995). Designing query support for multiple databases. In D. GILMORE, Ed. *Proceedings of INTERACT-95*. Lillehammer, Norway.

SUTCLIFFE, A. G. & CARROLL, J. M. (1998). Generalising claims and reuse of HCI knowledge. In *People and Computers XIII, Proceedings of the BCS-HCI Conference*. Sheffield. Berlin: Springer.

SUTCLIFFE, A. G. & ENNIS, M., (1998). Towards a cognitive theory of Information Retrieval. *Interacting with Computers* **10,** 323–351.

SUTCLIFFE, A. G. & MAIDEN, N. A. M. (1994). Domain modelling for reuse. In W. B. FRAKES, Ed. *Proceedings of 3rd International Conference on Software reusability*. pp. 169–173, Silver Springer, MD: IEEE Computer Society Press.

SUTCLIFFE, A. G. & MAIDEN, N. A. M. (1998). The domain theory for requirements engineering. *IEEE Transactions on Software Engineering*, **24,** 174–196.

SUTCLIFFE, A. G. & PATEL, U. (1996). 3D or not 3D: is it nobler in the mind? In M. A. SASSE, R. J. CUNNINGHAM & R. WINDER, eds. *People and Computers XI, Proceedings of HCI-96*. pp. 79–93. Berlin: Springer.

SUTHERLAND, I. E. (1963). Sketchpad: a man–machine graphical communication system. *Proceedings of the Spring Joint Computer Conference*, vol. 23, pp. 329–346, Monvale. NJ: AFIPS Press.

THIMBLEBY, H. (1990). *User Interface Design*. New York/Reading, MA: ACM Press/Addison–Wesley.

TIMMER, P. & LONG, J. (1997). Separating user knowledge of domain and device: A framework. In H. THIMBLEBY, B. O'CONAILL & P. THOMAS, Eds. *People and Computers XII, Proceedings of the British Computer Society HCI-SIG Conference HCI'97*, pp. 379–396. Berlin: Springer.

YOUNG, R. M. & BLANDFORD, A. E. (1994). *The state of PUMs: a revised manifesto*. AMODEUS Report UM/WP26, June.

WHARTON, C., REIMAN, J., LEWIS, C. & POLSON, P. (1994). The cognitive walkthrough method: a practitioner's guide. In J. NEILSEN & R. L. MACK, Eds. *Usability Inspection Methods*, pp. 105–140. New York: Wiley.

Paper accepted for publication by Editor B. R. Gaines