Participatory Task Modelling: users and developers modelling users’ tasks and domains

Eamonn O’Neill and Peter Johnson
Human-Computer Interaction Group
Department of Computer Science
University of Bath
Bath BA2 7AY
UK
eamonn@cs.bath.ac.uk

ABSTRACT
Participatory Task Modelling (PTM) is an approach to systems development that combines the strengths of the task analysis and participatory design traditions. This paper briefly reviews the strengths and weaknesses of standard participatory design and task analysis methods and presents PTM as a practical approach that integrates the best of both. We describe techniques for the participatory development of models of the users’ current tasks and domain and models of their envisioned tasks in an enhanced domain. The paper concludes with a discussion of a range of tools and representations used in PTM.

Author Keywords
Task model, task analysis, task-based design, participatory design, participatory analysis, participatory task modelling, task representation, user-task elicitation.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces — task analysis, participatory analysis, participatory design; H.1.2 [Information Systems]: User/Machine Systems — human factors; D.2.1 [Software Engineering]: Requirements/Specifications — elicitation methods, methodologies; D.2.2 [Software Engineering]: Design Tools and Techniques — computer-aided software engineering (CASE); General Terms: Design, Human factors; Keywords: participatory analysis.

INTRODUCTION
This paper presents an approach to combining the strengths of task analysis and participatory design in a task-based participatory approach to systems development. Our previous publications [e.g. 19] have addressed the more theoretical aspects of this work, including the concepts of common ground [4] and interaction spaces [20, 21]. This paper focuses on the practical application of our task-based participatory method. We first briefly discuss some of the strengths and weaknesses of existing task analytic and participatory methods and propose an approach to participatory task modelling that integrates the two traditions in a way that allows the strengths of each to address weaknesses of the other. Crucial to our approach, users are involved directly and actively not just as sources of information about themselves, their tasks and their domains, but in their actual modelling. We describe the practical application of this approach, drawing on examples from its use in two software development projects. We present a two-stage iterative process of (1) modelling the users’ current tasks in the situation under analysis and (2) modelling their envisioned tasks in an improved situation. We then discuss the strengths and weaknesses of various tools and representations used in generating these task models.

Our overall approach covers both analysis and design activities. The analysis activities include the participatory development of “current task models” (stage 1). The design activities include the participatory development of “envisioned task models” (stage 2) and also include the participatory development of application prototypes (stage 3). Our approach includes techniques for moving from current task model to envisioned task model, and from the latter to application prototype. This paper focuses on participatory task modelling and so covers stages 1 and 2 and the transition between them. The transition to and process of participatory application design is beyond the scope of this paper. For details of these stages of our approach, see [19, 20].

TASK ANALYSIS
Early task analysis (TA) work in HCI reflected the field’s more general focus on cognitive modelling and quite low level specification and evaluation of interfaces, e.g. Task
ACTION LANGUAGE [22]. Some work at this time [6] presaged later HCI concerns with understanding and designing for users’ higher level work tasks. As this concern has grown, so too has the application and development of TA within HCI. Current TA approaches span a wide range of research and development activities, with different granularities of task considered, from the detailed cognitive features of task performance to the daily workplace tasks which computer systems are designed to support.

Johnson [10] noted that “any TA is comprised of three major activities; first, the collection of data; second, the analysis of that data; and third, the modelling of the task domain” [p.165]. In the application of TA to system development, the activities are extended to include: collecting data on users’ (current) tasks, analysing the data, modelling users’ (current) tasks, deriving design requirements from the models and modelling users’ envisioned tasks with a designed system. TA based approaches to system development have tended to support the involvement of users in the processes of gathering data on their current tasks. This compares favourably with much system development practice in which users often are not involved in these activities.

However, while TA has often involved users in data gathering, the activities of analysing the data and modelling users’ tasks have remained almost exclusively the preserve of the analyst, deliberately excluding the user. TA practitioners have tended to follow the conventional approach of gathering data from subjects, analysing and interpreting the data and modelling the analysts’ interpretations. Even when, as in many cases, iterative validation of the model with users is recommended, task analysts use their own representations rather than involving users in the construction and definition of the representations.

Several difficulties arise from this approach. Even given an accurate and comprehensive set of gathered data, it is possible that the analysts’ interpretation of the data may be substantially different from an interpretation which users might have made. Therefore, the model derived from the analysts’ interpretation may not accurately reflect the users’ tasks. Indeed, it may prove difficult to validate the accuracy of the task model because, even if accurate, it may be based on an interpretation with which the users are unfamiliar.

Validation of the task model may also prove difficult and time consuming because the modelling notations and media are not familiar to the users. Amending and validating an unsatisfactory task model may also take considerable time and effort because of the need again to move through the cycle of assimilating, assessing and interpreting gathered data, constructing a fresh model and returning to validation. Hence, there is demand for an approach to task analysis and modelling that emphasises rapid iteration around a tight user-developer feedback loop. An effective means of promoting this is to involve users directly through a participatory development process.

PARTICIPATORY DESIGN
Throughout the past 25 years, more and more attention has been placed on encouraging and enabling the active involvement of users with developers in systems development activities. For example, Gronbæk, Grudin, Badker and Bannon [7] state that “cooperative development – full participation by both developers and users – requires rethinking the tools and techniques used in systems development. Motives for doing this range from simple cost-benefit arguments ... to concern for democracy in working life. Our point here is that user involvement is needed to achieve quality: better products, such as computer applications, will result when the developers have a knowledge about users’ practice and future use situations that can only be obtained through cooperation with users” [p.79].

However, despite the frequently declared need to understand the users, their tasks and their situation, much of the published work on user participation has presented techniques and tools for participatory design of new computer systems, with little focus on what, by analogy, may be termed participatory analysis. Hence, a large part of the participatory design (PD) literature describes the use of design mock-ups [e.g. 5] and software prototypes [e.g. 23]. Ehn and Kyng [5] state that “with cardboard mock-ups it’s simple: the purpose is design and the mock-ups are used to evaluate a design, to get ideas for modifications or maybe even radical new designs, and to have a medium for collaborative changes” [p.192; emphasis added].

Muller, Wildman and White [18] present a taxonomy of PD practices which includes many of the prototyping techniques but also includes practices which focus more on analysis than on design, such as Contextual Inquiry [1]. Contextual Inquiry is founded on the premise that “we need to involve users in the design of an appropriate system work model and user interface” [9, p.180]. However, Contextual Inquiry simply uses conventional techniques, such as concurrent protocols, for gathering data about users’ tasks. There is some interpretation and analysis in the dialogue between developer and user as the user describes her tasks: Holtzblatt and Jones [9] claim that “together with users, we reinterpret their experience of work and usability” [p.199]. However, despite the explicit recognition of the need to involve users in the development process, the emphasis in Contextual Inquiry remains on developers’ gathering data on users’ tasks and subsequently, in isolation from the users, modelling the developers’ interpretations of the data.

PUTTING THE PARTICIPATION INTO (TASK) ANALYSIS
Thus, while PD approaches emphasise user involvement in design but often lack adequate attention to upstream analysis work, TA approaches emphasise analysis but typically lack consistent user participation across the range...
of system development activities. It seems, therefore, that a synthesis of TA’s focus on analysis and PD’s emphasis on user participation may prove useful.

The need for participatory analysis, as a complement to participatory design, has been raised in the PD community [e.g. 13, 15]. However, we have seen very few attempts at systematically integrating practices and techniques from participatory and task analytical development with each other and within the overall system development process. Such an integration should foster the participation of users right across the data gathering, analysis and modelling activities of TA and extend the participation of users upstream from envisioning and design activities to contribute to the overall development process.

A rare example of such an attempt is presented by Muller and colleagues [e.g. 16, 17] in their descriptions of CARD (Collaborative Analysis of Requirements and Design). CARD was developed in order to address the problem that “much of participatory work is concerned directly with the process of design. … There was insufficient support for big-picture or work-level analysis, representation, and design” [17, p.142]. In the CARD method, components of users’ tasks are represented on cards. The cards are then sorted and ordered by the participants in the session to describe at a high level the tasks to be supported by the software system. The sorted cards represent an event flow of task components. Brief textual notes are taken, usually on the cards, of requirements that emerge from this manipulation. Lafrenière [14] also presents a card-based task analysis technique that is very similar to CARD.

Chin, Rosson and Carroll [3] have also recognized the need to push participatory activities upstream in the development process. They involved users in “participatory claims analysis”. “Claims” are articulations of implicit causal relationships between features in a situation and consequences for users in that situation. Analysis of claims can identify requirements for the system. The media used in this study included scenarios, claims and new features inspired by the claims. Chin et al found that users “were able to participate fully in the initial requirements analysis – not as informants, or subjects of analysis, but as analysts” [p.168; emphasis in original].

**PARTICIPATORY TASK MODELLING**

Our approach to Participatory Task Modelling (PTM) involves users directly in the analysis and modelling of their current tasks and the contexts in which they are performed, and the design and modelling of their envisioned tasks in an improved context. We use the term Participatory Task Modelling, rather than for example Participatory Task Analysis, to reflect that our approach covers both analysis and design activities and to emphasise our concern that users participate directly in the modelling.

In contrast to other approaches in which there is typically a data gathering phase involving users, followed by analysis and modelling phases from which users are excluded, PTM is based on the concept of an integrated data gathering, analysis and modeling session. As these sessions progress, the emerging task models are both the product and the facilitator of the session. In a highly iterative process, developers and users discuss, share and model an emerging understanding of the users’ roles, tasks and domains.

In the first phase of development work, the participants – users and developers – engage jointly in the development of a shared model of the users’ current situation. The model of the users’ current situation developed in this first participatory modelling phase serves as a starting point for the second phase. In this, the participants cooperate in developing a model of an envisioned, enhanced situation. The following subsections present these phases in turn and describe the construction of both current and envisioned task models.

**Modelling the users’ current situation**

It is unwise to begin the integrated modelling and analysis sessions “from cold”. The user has to pick up the skills of modelling while the developer gathers information about a work domain which is wholly new to her and while both user and developer are taking the first steps in working together. This may place too much load on both user and developer. Hence, our recommended approach is to precede modelling with some “bootstrapping” information gathering and introduction to modelling.

In PTM, an understanding is required not only of the tasks but also of the domain in which those tasks are performed. Our task models include not only decompositions of tasks, but representations of the roles in which tasks are performed and of workflows and relationships amongst the roles and tasks. At first, the developers are interested in gaining a preliminary understanding of the work situation and identifying the main roles and tasks. The developers need to come up to speed on knowledge of the domain effectively to communicate with users. The users also need to develop some understanding of the design process, its tools and artefacts and the technical possibilities, in order effectively to communicate with the developers.

Before the initial modeling sessions, semi-structured interviews are conducted with users from throughout the work domain. Other forms of data gathering may also be used, such as observation of the users at work, collection of documentation from the work place, concurrent and retrospective protocols. For a summary of data gathering techniques, see [11]. From this, the developers begin to derive a broad understanding of the users’ current work situation.

As the developers begin to share tentative and partial understandings of elements of the users’ work, pencils and paper – and erasers! – are used to sketch partial models of the users’ tasks. Initial sketching begins during the participatory sessions. The developers then turn these
initial sketches into a first pass at sections of a task model. The developers’ construction of these initial models serves to bootstrap the participatory task modelling work. These early, tentative and partial models introduce the user to the concept and practice of task modelling and provide a basis on which to begin the construction of a more sophisticated, complete and integrated model of the users’ current work situation.

In this first phase of analysis, the participants identify:

- the work situation of concern (i.e. the domain into which a new system is to be introduced);
- actors who currently fulfill roles in the work situation; this provides an indication of the people in the work situation whose cooperation in subsequent development work should be particularly valuable;
- roles within the situation of concern which should potentially be supported by a new system;
- a preliminary description of the primary tasks associated with each role to be supported; this serves as a starting point for the cooperative construction of a model of the tasks performed in the role;
- a preliminary statement of the desired system (without reference to envisioned technology or possible designs) to support the attainment of the identified tasks.

The roles identified by users in the work situation serve as the unit of analysis. Roles roughly correspond to job distinctions in the department, perhaps with several users performing more than one role. Roles may be based on mixtures of physical location, departmental structures and common tasks. The task models represent the work situation in terms of roles, tasks associated with each role and flows of work objects amongst the roles. Tasks are represented in a primarily hierarchical structure of subtasks with notions of sequence, selection and iteration. The tasks performed in each role are analysed and modelled through “vertical scenarios”. If we consider roles listed in a row across the top of our task model, with a task decomposition descending from each, a vertical scenario explores the activities within a particular role, hence developing a vertical section of the overall task model.

Where there is a small number of highly interactive roles in the work situation, one or more of the actors who fulfill each role become involved together in the cooperative construction of a model of the tasks which are performed across the work situation. Each user participant has special insight into the role that she performed and often also provides new and fresh insights into other roles in the work situation.

Where there is a larger number of roles than could be effectively analysed in a single group session or where there is little interaction between roles, representatives of each role become involved in the participatory development of a model of the tasks which are performed in their own role. For example, one session may be devoted to participatory modelling of a technical consultant’s tasks, while another session is devoted to participatory modelling of a receptionist’s tasks. In these cases, further sessions are then required to integrate the individual task models into one comprehensive model and to model the interactions between the roles. The integration process provides a check that users from across the work situation agree that workflows and interactions between roles have been modelled accurately.

Advantages to this approach include producing a single, comprehensive model of the work situation, the users, their tasks, objects and workflows. Additional representations may, if desired, be produced to complement the task models, such as models of work or domain objects, their decompositions and relations.

In the participatory analysis work, the participants build up strong working relationships, the users feel actively involved in the analysis and modelling work and the developers achieve a detailed understanding of the users’ current work situation. The task model provides a resource on which subsequent design work can be based. During the combined analysis and modelling work, the model serves as a focus for the participatory development activities. The users learn and use the concepts of task modelling while the developers learn and use the concepts of the application domain. The notation used in the task modelling is defined and refined cooperatively by the participants as the work progresses. Hence, the user is not confronted with a developer-imposed notation that the user cannot understand and comfortably use. Rather, the user’s understanding of systems development concepts, tools and notations is gradually built up during the work. This facilitates ongoing communication between developer and user and encourages the user to own the model.

**Unifying roles with object flows**

In contrast to the use of “vertical scenarios” in modelling roles, “horizontal scenarios” are used in integrating the roles. A horizontal scenario follows the flow of a work object as it passes through the domain from role to role and from task to task within the roles.

The first phase of task modelling involves developers and users modelling the users’ tasks with the role as the unit of analysis. But again, participation does not mean that everyone does everything. At this stage, only those users who actually fulfill the role should be involved in modelling each role’s tasks. There are several theoretical and practical reasons for this.

A given user will generally know most about the tasks that are performed in the role or roles which that user fulfills in the domain. Other users will generally know less about the day to day performance of that user’s tasks. Similarly, that user’s knowledge of tasks performed in other people’s roles may be partial or inaccurate.
Few users will have an accurate knowledge of the broad sweep of tasks which occur across many roles. This applies no less to managers, whose knowledge of users’ tasks is generally not of the detailed kind which can lead to sound systems design. Relying on managers’ descriptions of users’ tasks is a notorious cause of poorly usable systems.

Attempting to model the whole domain with a group of users is not just inefficient – it can also lead to heated arguments. Not only will users have misunderstandings of what other users do, they will have opinions about what other users should do.

So, in the participatory analysis situation, we work with each user to model his or her own role(s) first. We also model privately with them what they think other users’ tasks are. This provides a cross-check on other analysis work and provides useful insights into the users’ understandings of their overall domain.

Taking the role as the unit of analysis also facilitates the analysis and modelling of arbitrarily large and complex domains by modularising the work for developer and user.

Having developed the parts of an overall model of the users’ domain, it is necessary to integrate the parts into a single model. This integration also helps to check that users from across the domain agree that the tasks and the workflows and interactions between roles are modelled accurately. The result is a comprehensive model of the users’ current roles, tasks and domain.

**Modelling the envisioned situation**

The model of users’ current tasks developed in the first PTM phase serves as a starting point for the second phase. The participants cooperate in amending and refining the model of current user tasks to synthesise a model of envisioned user tasks. The users contribute to this work both knowledge of their current tasks and their wishes for enhancements to those tasks. The developers contribute their knowledge of computer based support for tasks.

Our approach proposes a direct route across the “design chasm” from the task model of the users’ current work situation (TM1) to a task model of an envisioned work situation (TM2). This route involves (i) mapping work flows in TM1, (ii) identifying key tasks in the current situation which require support in an envisioned system, (iii) mapping envisioned work flows through the key tasks, (iv) extending this mapping into an envisioned task model (TM2).

The primary criteria for identifying key tasks that should be carried forward from the current situation to the new design are (a) tasks necessary to goal achievement; i.e. tasks without which the system’s job could not be done; and (b) the user’s (mental) model of her tasks. In general, workflow or “housekeeping” tasks are not considered as key tasks for this purpose. These are, for example, simple physical tasks of moving a work object from one location to another. Workflow and housekeeping tasks for the envisioned system are identified while extending the work flow-key task mapping into the envisioned task model.

Hence, an envisioned task model, again covering the overall work situation and including roles, tasks and workflows, is derived in further participatory task modelling sessions. The second participatory analysis phase delivers a model of envisioned user tasks in an enhanced domain.

Development of TM2 from a first cut of key tasks and workflows may be conducted with the same media, tools and participants as the construction of TM1. Again, the user is directly and actively involved in the modelling activity. In this phase, however, she is involved in participatory design rather than participatory analysis – contributing ideas of what should work well, what should not appear in the new work situation and why. Again, tasks performed in particular roles are modelled in participatory sessions with users who work in those roles. When the constituent parts of a comprehensive model of the work situation have been produced, users are brought together in a larger group to integrate the parts into a whole. In this way, the second PTM phase delivers a comprehensive model of envisioned user roles, tasks and domain.

**TOOLS AND REPRESENTATIONS**

The PTM approach provides recommendations of which activities to perform, in which order and with which tools, but allows participants to refine any and all of these as required. A clear example of this is the use of task modelling notations. Those we have used are based on previous work on TKS [12] and EuroHelp [2]. However, as an integral part of the participatory development work, the notations are refined and agreed by the participants as the task modelling activities progress. This helps to promote the users’ understanding and feelings of ownership of the task models. Similarly, the “steps” of the PTM approach are applied, ignored, refined and reapplied on a contingent basis. Hence, the participatory development method is itself cooperatively developed.

**Representing tasks, users and domains**

The task models may use various notations, representations and tools but are commonly represented through a graphical notation. The model of the users’ work situation begins with a rough schematic representation of the users’ physical working environment. The sketch includes schematic representations of people who work in the environment, their relative working locations (represented by their relative placement in the sketch) and flows (represented by arrowed lines) of work objects, documents and so on, amongst the people.

The task models are developed from this base. The people identified in the work situation are formalised as roles. Again, the notational device for a role may vary with the medium used for the representation. In a typical graphical
representation, a “head-and-shoulders” icon or “stick figure” denotes a role. In each case, the graphical role notation is labelled with a textual title, for example “Receptionist”.

Tasks associated with a role are denoted by nodes or boxes connected to the role by arcs or lines. Similarly, subtask boxes are connected by lines to the task into which they are composed. Task decompositions are represented below their associated role with leaf nodes at the bottom, furthest from the role symbol. Task and subtask boxes contain a brief textual description; for example, “Record ID”. The connecting lines also are usually labeled

Some boxes are used to represent conditional tests which the user applies to determine which actions to carry out next. To represent this, a box contains the conditional test (e.g. “Item in database?”), lines from the box are labelled with the range of possible responses (e.g. “Branch YES”, “Branch NO” and “Branch DON’T KNOW”) and the box to which each line runs contains the task that is carried out following each response. Each of these boxes in turn may have subtasks. Another specialised notation covers cases where a task (or subtask) takes different forms depending upon the object on which the task is performed. In these cases, the line between the main task or subtask box and the variants is labelled with “Variant” and the name of the object on which the specific task variant is performed.

**Tools**

Tools that we have used for producing the task models include cards, CASE tools and whiteboards. Here, we describe some of the strengths and weaknesses of these tools.

**Cards**

We have applied a card-based PTM technique with mixed success. We constructed a model of the user’s current tasks through user-developer cooperation using card-based “analysis components”. Generic analysis components took the form of blank cards of different shapes and colours representing generic tasks and subtasks, roles, actions or leaf nodes in a task decomposition, and objects. In addition, participants could introduce other analysis components for the purposes of a particular analysis.

The analysis sessions proceeded through manipulation of the analysis components on an “analysis surface” that took the form of a large (A1 size) sheet of white card around which the participants sat. The participants cooperated in manipulating, amending and introducing analysis components to construct a model of the user’s tasks. On being introduced to the model, generic analysis components were turned into specific analysis components by annotating them.

Construction of a model of the user’s current work began with the introduction of a role card. This card was annotated with the role whose tasks were to be modelled. Cards representing the primary goals associated with that role were produced and placed alongside the role card. Analysis proceeded by selecting each primary goal in turn and modelling a decomposition of the tasks involved in achieving that goal.

The proposed advantages of the use of analysis component cards to support PTM were:

- direct manipulation of the cards by users should encourage a sense of direct participation in the analysis work;
- the use of familiar stationery should prove unintimidating to users from most backgrounds;
- mobility and ease of manipulation of the cards should facilitate early attempts at construction of the model when multiple, tentative attempts are often required;
- the initially unstructured and informal nature of the modelling tools should facilitate the participatory development and application of a task modelling notation that all participants understand and own;
- the modelling surface and materials should provide a shared interaction space for user-developer collaboration.

However, there were substantial problems with the use of cards. Even a partial model of a small subset of the user’s tasks quickly covers a large area. The A1 size analysis surface can be inadequate to contain a decomposition of even one primary task, while the analysis surface is unwieldy and not readily portable or storable. The problem of limited space in which to develop the model exacerbates other difficulties in using the cards. There are major difficulties in denoting relations between task model components. Placing task cards in a horizontal row indicated that the tasks were at a similar level of the decomposition. A problem with this is that the necessary rows easily exceeded the width of the analysis surface.

A more abstract, representational issue is the need to represent relations such as sequence, selection and iteration amongst task components. A bracket notation was devised to represent an optional task, but this did not explicitly represent the conditions under which the task should and should not be performed. A lack of expressiveness in the modelling tools led to conditions being discussed by the participants but not being represented in the external model.

The graphical notation described above was used to represent hierarchy in the task decomposition. Our first versions involved drawing a line with a marker between the bottom of a task card and the tops of its respective subtask cards. This ran into problems due to the dynamic and iterative nature of the analysis work. Participants would decide that a task belonged somewhere else in the evolving model and, when the card was moved, the now redundant marked line remained on the analysis surface.

Markers were abandoned as a tool for representing the hierarchy relations, in favour of thin, coloured tape. It was
hoped that the tape would support the dynamic development of the model because it could be stuck down, removed and replaced at will. In practice, use of the tape suffered from the main problem which dogged use of the analysis component cards. Both cards and tape were too time consuming, distracting and cumbersome to prepare and to use. Cards had to be selected, carefully inscribed with marker, placed on the analysis surface, as often as not in a location which was already occupied by another card. Whole sections of the model often had to be moved, card at a time, when even a minor change was made to the represented task structure. Anything more than minor changes to the labelling of a card required the rewriting of the entire card. An appropriate length of tape had to be unrolled, cut and stuck in place. Whilst the tape had the advantage over marker that it was erasable and movable, this was not particularly convenient, still less when several components of the model were moved simultaneously.

Hence, the mechanics of using the modelling tools interfered with the processes of analysis and modelling. Interaction between the participants was broken and erratic. As a fresh point was made or an old one reviewed, the developer would often stop the discussion while he busied himself manipulating cards and scissors and markers and tape. Flowing interaction and analysis of the user’s work situation was common only during post hoc reviews of a section of the model. Even during these periods, there were interruptions when an issue raised required an amendment to the model. It is of course inevitable in analysis and modelling work that a model must be revised as the analysis progresses. The point here is that the tools for that revision failed to support smooth transitions between analysing and modelling. From this account, it may be seen that the intended benefits of analysis component cards were largely unrealised. A sense of direct collaborative participation by the user in the development work was not imbued through use of the cards.

**CASE tools**

Using CASE tools in the standard software engineering paradigm has several disadvantages from a PTM perspective. First, the computer-based presentation and use of a tool may be intimidating to some participants. Secondly, the typical CASE tool is designed for individual use. Thirdly, the typical computer monitor is designed for individual use and does not provide an adequate public display [8] or interaction space [20, 21] for collaborative activities such as PTM.

These difficulties were ameliorated in our approach by a combination of CASE tools and paper-based media. A developer entered the current version of the task model into an application that allows the storage, editing and graphical display of linked data. Before each PTM session, a large chart of this current version was printed and placed on a wall, and the participants worked on it. Amendments and additions were annotated on the wall chart using pencils. After each session, one of the developers updated the electronic version of the model to reflect the manuscript annotations. In this way, a single comprehensive task model was developed and stored.

In practice, pencil-and-wallchart has proved to be a much more efficient medium for task modelling than the analysis component cards. The mechanics of using the modelling tools do not interfere with the processes of analysis and modelling. Interaction between the participants is dynamic and largely uninterrupted by the modelling activities. As a fresh point is made or an old one reviewed, participants can quickly amend the developing models with pencil and eraser.

The use of a CASE tool has several benefits. The electronic version of the task model provides relatively secure storage of the model and facilitates the production of printed charts in a variety of sizes for individual and group use. It may allow the automated generation of related models, such as a model of relations amongst workflow objects. The electronic version also facilitates portability, distribution and reporting.

However, there is also an inconvenient feature of the method described here. During the sessions, markups are made using pen or pencil on the printed wall chart but the original printed version of the model cannot be erased by the participants. Hence, by the end of a session, the model is often cluttered with markups. This may make the model quite difficult to interpret. Participants generally see a clean printed copy of the most up-to-date version of the model only at the beginning of the session following the session in which that version was developed.

**Whiteboards and smartboards**

Another useful medium for modelling the users’ current and envisioned work situations is the whiteboard. One advantage that the whiteboard based models have over the models in the CASE tool/wallchart approach is ease of amendment. During the PTM sessions, amendments and additions to the task models are made using whiteboard markers and erasers. This is fast, reduces ambiguity by immediately erasing unwanted clutter from the modelling space and ensures that participants are always working with the most up-to-date version of the model. Using a whiteboard, new or modified notations and conventions can be quickly agreed, established and mixed, facilitating the participatory development process.

At the end of each modelling session, the whiteboard is generally left unerased until the next session. The difficulty of producing multiple copies of the model (in contrast to the electronic version in a CASE tool) makes the model vulnerable to loss. However, technologies such as Mimio (see www.mimio.com) allow the model to be copied into electronic form.

Another disadvantage of the traditional use of whiteboard and markers is that, apart from manual copies, previous
versions of the model are not recoverable. Also, models on whiteboards are less portable than either electronic models or models on wallcharts, which may be rolled up and displayed in a new location more easily than a large whiteboard could be moved around. Again, technologies such as Mimio overcome these problems.

CONCLUSION
The PTM approach described in this paper makes several contributions. It offers practitioners tools and techniques to encourage direct user participation in the upstream system development activities of task analysis and modelling and in envisioned task design. It emphasizes that task models should include not just tasks but also roles, objects and workflows in the users’ domain. Finally, it encourages further research and practice in combining the strengths of the participatory design and task analysis traditions.

ACKNOWLEDGMENTS
Thanks to George Coulouris, Mick Smith and Hilary Johnson.

REFERENCES