

Tamot: Towards a Flexible Task Modeling Tool

Shijian Lu[†], Cécile Paris[†] & Keith Vander Linden[‡]

[†] CSIRO/MIS, Locked Bag 17, North Ryde, NSW 1670, Australia,

[‡] Department of Computer Science, Calvin College, Grand Rapids, MI 49546, USA
{shijian.lu, cecile.paris}@csiro.au, kvlinden@calvin.edu

Abstract

The usefulness of task analysis and task modeling has been widely acknowledged in software design and development. However, the rate of adoption of task analysis and modeling has been painstakingly slow. One of the reasons for this may be the lack of flexible software tool support. In this paper, we present Tamot, a graphical task modeling tool that was designed to include the characteristics that would be desirable in a task-modeling tool. Those characteristics include ease of learning, support for entering multiple tasks quickly, mechanisms to support top-down and bottom-up approaches to task modeling, ease of modification, customisable reporting facilities, and, finally, inter-operability.

1. Introduction

Despite of the recognition of the usefulness of task analysis and modeling in software design processes, the adoption of task analysis and modeling is far from common in the software industry. We believe an important reason for this is the lack of appropriate software tool support. Before the early nineties, task analysts and practitioners mainly relied on pencil and paper or general-purpose tools such as text editor or graphical drawing tools to perform task analysis and modeling. Task modeling tools (prototypes) started to emerge in the mid-nineties, e.g., (Beard *et al.*, 1996). Gradually, more task modeling tools came into existence, e.g., (van Welie *et al.*, 1998; Williams, 2000; Paternò *et al.*, 2001), especially after the CHI 1999 workshop on tool support for task-based UI design (Bomsdorf and Szwillus, 1999), where the need for flexible task modeling tools was clearly recognized. This marks a significant progress in the task analysis and modeling field. A detailed review of task analysis and modeling can be found in (Bentley *et al.*, 2000). In this paper, we present some of the key issues for a useful task modeling tool. This is largely based on observations gained in the process of many iterations of the development of Tamot, our graphical task modeling tool. We also present Tamot and explain how it addresses those issues.

2. Key Characteristics

Any task modeling tool is based around a specific task modeling notation. To maximize the usefulness of task modeling, it would be good if the notation was able to support a number of the stages within the Software Development Life Cycle, as the cost of building task models would then be offset by the use and re-use of the models (Paris *et al.*, 2000). In addition, it is desirable for the notation to be usable and readable by a variety of people. These characteristics are features of the notation itself. There are also a number of desiderata for a tool whose aim is to support the construction of a task model. Based on our development of such a tool and its uses by HCI task analysts in commercial projects, we have observed the need for the following characteristics: learnability, efficiency of multiple task creation, affordance of different working

styles, efficacy of managing change, customisability of report generation, affordance of multiple views with varying degrees of abstraction and interoperability.

Learnability: Learnability is important for any software tool. With the same degree of utility, the easier it is for a user to get started with the tool, the more likely the tool will be adopted. The fact is that modern workers are increasingly overloaded. Most people have to focus on tasks at hand and cannot afford to spend much time to learn new tools. Often, the time span of a commercial project is very short, thus demanding human factor experts to produce task analysis and modelling report quickly.

Efficiency of multiple task creation: High learnability enables new users to get started with a tool easily. To enable realistic model building, a task modelling tool must also provide features that support the efficient creation of multiple tasks. When early versions of Tamot were first tried in commercial projects, task analysts were frustrated with the lack of support for creating multiple tasks. In a real world situation, a task model normally consists of a large number of tasks. Therefore, it is critical to allow users to create a series of tasks easily.

Affordance of different working styles: Users have different working styles. A software tool should not force users to adopt a particular style unnecessarily. For task modeling, there are 3 common approaches, namely, top-down, bottom-up, and a combination of the two. When using a top-down approach, users start from the root of a task model and progressively work their way from higher-level tasks down to lower level tasks. Whereas bottom-up approach is the other way round, i.e., from lower level tasks users work their way upwards to form higher-level tasks.

Efficacy of managing change: In the process of task analysis and modelling, the task model usually undergoes many iterations before being finalised. It is observed that re-organisation of tasks in the task hierarchy almost always occurs during task model iteration. Re-organisation of task models can be broken down into two types of operations. One is removing existing high level tasks. Another is re-grouping low level tasks to form a new high level tasks.

Customisability of report generation: Task models can be used for many different purposes and it is useful to have reports that present the models. As the reports may be used for different purposes, they should be easily customisable to suit different needs. Ideally, the customisation could be applied to both content and format.

Affordance of multiple views with varying degrees of abstraction: A task model, as a whole, is a coherent entity, and it can be fairly large in real world situations. In order to manipulate such a model effectively, a tool needs to provide appropriate views of the model such that a user can easily get a global view, check on some intermediate level construction, or zoom in on some fine details.

Interoperability: The usefulness of task models could be maximised if a task modeling tool is interoperable with other tools.

3. Tamot

Tamot is a general-purpose task modeling tool. Its task representation is based on the Diane+ formalism (Tarby & Barthet, 1996), which supports task annotations (e.g., repetition and optionality) and procedural relationships (e.g., goals and sequences). Tamot forms part of the Isolde project whose main objective is to generate user on-line documentation (Vander Linden *et al.*, 2000; Paris *et al.*, 1998). Tamot has been used as a task model editor by task analysts, and it has also been shown to be usable by technical writers (Ozkan *et al.*, 1998). In what follows, some

important features of Tamot are discussed. The emphasis is on how these features address the key characteristics discussed in section 2, namely, learnability, efficiency of multiple task creation, affordance of different working styles, efficacy of managing change, customisability of report generation, affordance of multiple views with varying degrees of abstraction and interoperability.

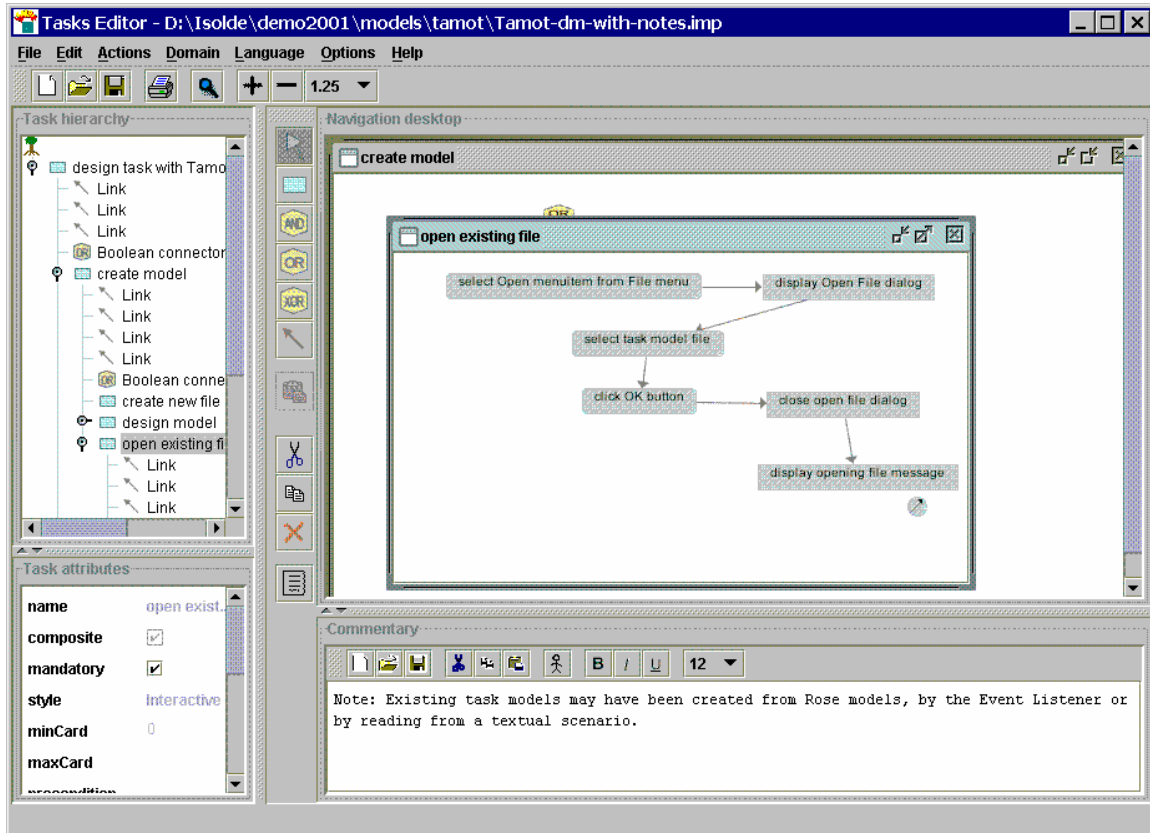


Figure 1. Tamot main window

The main window

In graphically based tools, the conventional way to facilitate ease of use is to allow commands to be activated by selecting icons, dragging and dropping them as appropriate. For instance, in Rational Rose (Rational Rose, 2002), a use case can be easily created by first selecting the “use case” icon from the palette, then putting it in a use case diagram by clicking on the appropriate spot on the workspace. We have used a similar mechanism in Tamot: tasks and boolean connectors (logical operators) can be created by selecting (clicking) the appropriate icon on the palette. Figure 1 shows the Tamot main window. As we can see, there are five parts below the menus and tool bar. On the upper-left, the task hierarchy is displayed with attributes for the selected task shown in the window underneath. On the upper-right, workspaces showing detailed task decompositions at various points in the hierarchy can be displayed. The bottom right window allows a user (e.g., a task analyst or a technical writer) to enter *comments* in plain text or in Rich Text Format about the task being modeled. On the bottom-left, the user can set task attributes for the currently selected task. The palette can be seen vertically in the middle.

To create a task, the user clicks on the task icon in the palette section, and then clicks on a window in the upper-right section. This provides users with an easy way to get started.

Task input window

In addition to the method for creating tasks described in the previous section, new tasks can also be created using the Task Input window (Figure 2). With this window, new tasks can be created by typing descriptive task names and pressing the “Return/Enter” key. This creates a new task automatically and then clears the text in the text-field. Since this method doesn’t specify the task position, the new task will be automatically aligned next to the last task in the selected decomposition window. If the “Link” check box is checked, then the consecutive tasks will be also automatically linked. With this task creation facility, a series of parallel/unlinked, sequenced/linked, or parallel and sequenced interleaved tasks can be easily created. For parallel tasks, the time saving will be proportional to the total number of tasks to be created. For sequenced tasks, the time saving will be even greater, since the user is not obliged to create the sequence links in the conventional way, i.e., selecting the appropriate icon and linking the source and destination tasks (a 3-step operation for creating 1 link).

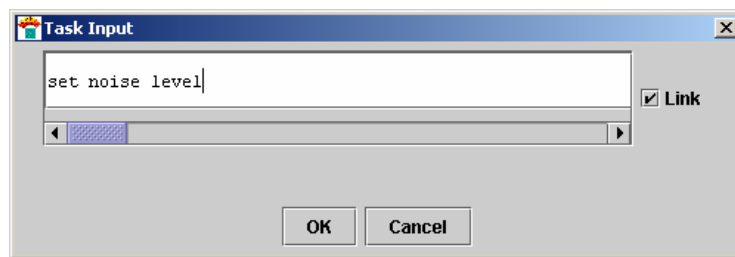


Figure 2. The task input window

Add and remove intermediate levels

In this sub-section, we introduce the *add* and *remove level* functionality and discuss how they contribute to bottom-up model building and the improved efficiency for managing model change.

The *add level* functionality enables a user to create a new high level task from a collection of tasks. In effect, the group of tasks is “demoted” to a lower level in the hierarchy. For instance, it takes two steps to form a mother task Task m from a group of tasks Task 1, Task 2, ..., Task n (Figure 3):

- (1) Selecting tasks Task 1, Task 2, ..., Task n;
- (2) Selecting “Add level” popup menu item, which prompt the user for the new task’s name, here Task m.

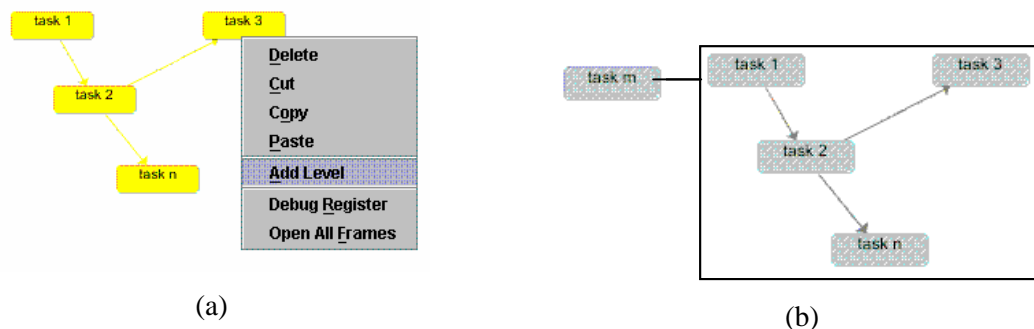


Figure 3. Grouping multiple tasks to form a higher level task with Add Level command

In contrast, achieving the same objective with conventional multiple select/cut/paste, the following steps need to be performed:

- (1) Creating task Task m;
- (2) Selecting tasks Task 1, Task 2, ..., Task n;
- (3) Cutting task Task 1, Task 2, ..., Task n; and
- (4) Pasting the cut tasks into task Task m's decomposition.

The *remove level* functionality allows users to easily remove a composite task. Consequently, all its sub-tasks are “promoted” to a higher level in the hierarchy. For example, Figure 4 (a) shows a simple task model for “counter anti-ship missile attack”. If we want now to remove the higher-level task “set the noise rejection level”, we simply right-mouse click on the task and select the “*Remove level*” menu item. As a result, the “set the noise rejection level” task will be deleted and its sub-task “match altitude dial to threat altitude” will be promoted to its place. Notice that even the task links will be kept intact (Figure 4(b)). Similarly, this objective can be accomplished with conventional multiple select/cut/paste/delete, but this requires greater effort as shown in the following:

- (1) Selecting the sub-task “match altitude dial to threat altitude”;
- (2) Cutting the selected task;
- (3) Pasting the selected task to the same task decomposition as its original mother task;
- (4) Creating links between the pasted task and neighbours of its original mother task;
- (5) Selecting the original mother task and its associated links;
- (6) Deleting the selected components.

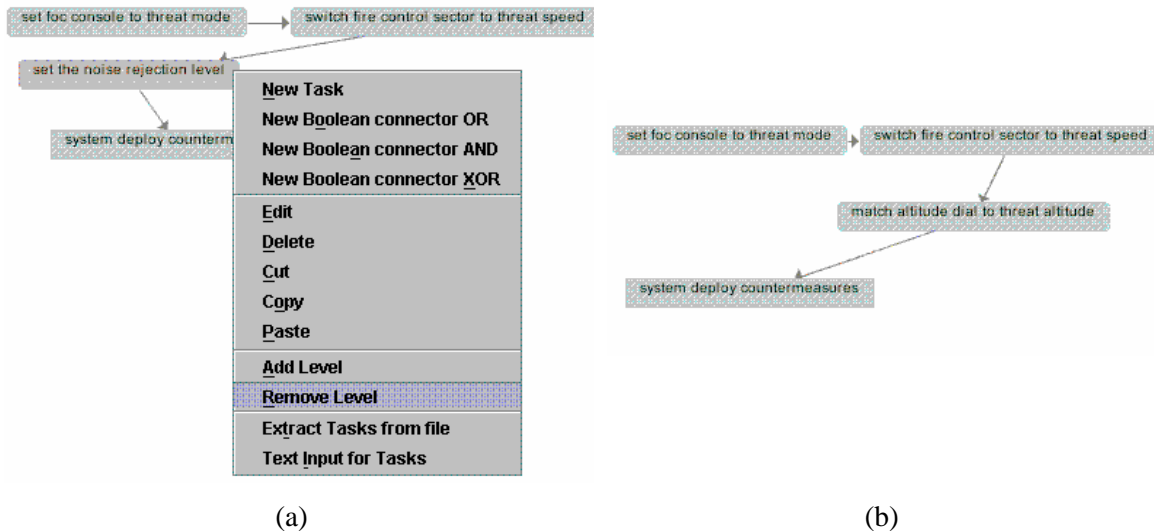


Figure 4. Remove higher level tasks with the Remove level command

It is probably worth noting that conceptually, *remove level* is the reverse of *add level*. However, in terms of implementation, *remove level* could be far more complicated than *add level*. The challenge comes from how to automatically work out the starting and ending points when a task decomposition contains multiple interwoven tasks. In the current state of Tamot, some simple heuristics are employed while allowing users to intervene in complicated situations.

Add and remove level functionality supports two common types of operations in managing model change, namely, re-grouping some tasks to form a new high level task and removing some existing composite tasks. In terms of supporting different working styles, *add level* supports building task models bottom-up. Tamot also supports the top-down approach. The user creates a high level task and then adds sub-task into its decomposition.

Reporting facility

In Tamot, the user has full control as to what task attributes to include in the generated report (i.e., the content). This is achieved with the dialog box shown in Figure 5 (a). Task attributes are included or excluded by selecting or deselecting in the dialog box. This prevents reports that contain irrelevant information. The report is generated using XSL (XML stylesheet language) in the form of HTML. Currently, the format of the report cannot be customised. The report, starting with a table of contents (task names listed in alphabetical order), lists all tasks with selected attributes and its graphical decomposition (Figure 5 (b)).

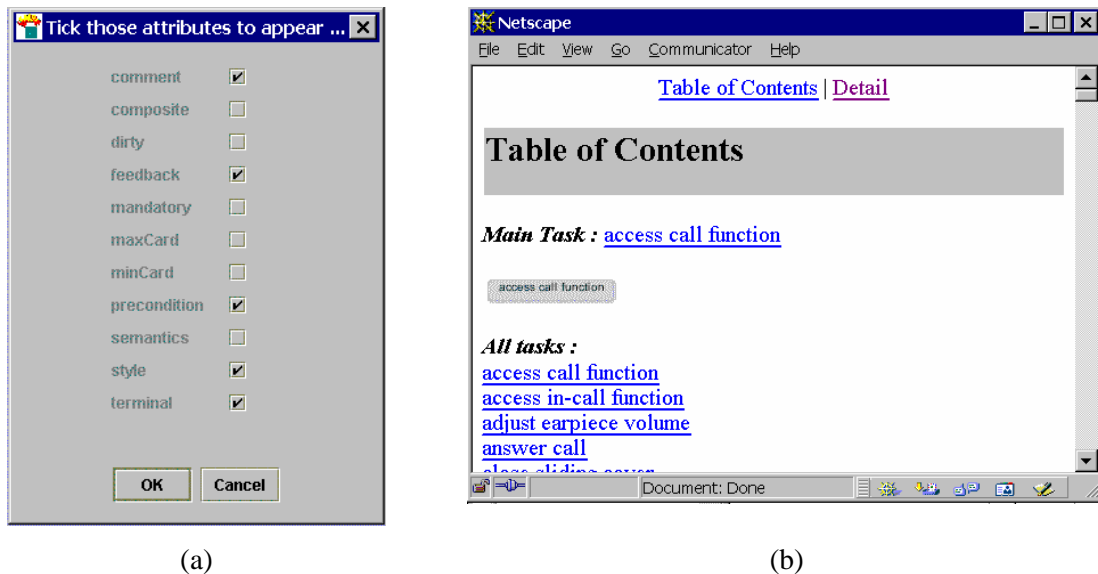


Figure 5. Customize task model report

Interoperability

Tamot uses XML as its internal representation. Tamot is, therefore, able to cooperate with a set of external tools, including task model acquisition tools. In our environment (Paris *et al.*, 2001), these tools currently include U2T (Lu *et al.*, 1999), a tool for acquiring task models from UML diagrams; T2T (Brasser and Vander Linden, 2002), a tool for extracting a task model from task scenarios; and UIR (Paris *et al.*, 2001), a tool for building task models from user system interactions. Tamot thus provides a consolidation platform for integrating task models from heterogeneous sources. In addition, Tamot can interoperate with tools that make use of task models. In our work, we currently have two such tools: ITG (instructional text generator), a tool that generates user manuals (Vander Linden *et al.*, 2000) and T2U (task model to UML model transformer), a tool that generates system behavior models in Rational Rose (Lu *et al.*, 2002).

Navigation facility

As Shown in Figure 1, Tamot provides 4 ways to navigate through a task model, i.e., the Task hierarchy, the navigation desktop, zoom, and search. The task hierarchy presents a containment (folder like) view of the task model, which enables users to inspect containment of tasks at any level of the hierarchy by either opening or closing composite tasks. Detailed decompositions for composite tasks can only be viewed in the navigation desktop, and can be activated by double clicking the task icon either in the Task hierarchy or on the navigation desktop. Therefore, from the root of a model, it is easy to drill down into the hierarchy. However, it is not as easy to navigate up the hierarchy. In the Navigation desktop, each internal window contains one task

decomposition. If a composite task contains too many sub-tasks to be accommodated within one viewable frame in the default resolution, zoom can be used to contract the model in adjustable magnitudes. The search functionality supports locating tasks whose names contain the string being searched.

In summary, Tamot provides reasonable support in:

- random access of tasks (search);
- from high level drilling down to details;
- overview of a composite task (zoom).

And limited support in

- global overview of the whole task model, which requires the users to integrate two sources of information: the task hierarchy and navigation desktop;
- navigation from low level to high level tasks.

3 Conclusion

Despite of the growing awareness of the usefulness of task analysis and modeling in software development, task analysts have largely been relying on either pencil/paper or rudimentary task modeling prototypes to perform their work. The time for usable tools to support task analysis and modeling is long over due. In this paper, drawing on our experience in developing Tamot, a graphical task modeling tool, and on user feedback, we have discussed some aspects which are of critical importance to the usability of a task modeling tool. We also introduced Tamot*, focusing on how it addresses these issues and identifying areas that require further development. One such an area is providing an adjustable global overview. One possible solution would be to have a window displaying the whole task model with full details while allowing individual composite tasks to be either open or closed as desired.

The significance of our work is in two areas: (1) We hope that providing a task modeling tool with improved support for common tasks in task modeling will promote task analysis and model building, which, in turn, will lead to improved usability in developed software systems. (2) By identifying some significant usability issues for task modeling tools and how Tamot addresses them, whether reasonably or otherwise, we hope to enable the construction of better task modeling tools in the future.

4 Acknowledgments

The authors thank the past and present members of the ISOLDE team, including Sandrine Balbo, Todd Bentley, Nathalie Colineau, Thomas Lo, Nadine Ozkan, Maryline Specht, Robert Tot, Jean-Claude Tarby, and Laurent Chretienneau. This work has been supported by the Office of Naval Research (ONR) grant N00014-96-0465, CSIRO and Calvin College.

5 References

- Beard, D., Smith, D. and Danelsbeck, K. (1996), QGOMS: A direct-manipulation tool for simple GOMS models. *In the Proceedings of CHI 96 companion on Human factors in computing systems: common ground*. Vancouver Canada. 25:26.
- Bentley, T., Balbo, S., Paris, C., Tarby, J., and Johnston, L. (2000), Task modelling review of methods, tools, and other. *CSIRO/MIS Report Number: 2000/155*, Sydney.

* Tamot is freely available from the Tamot home page (Tamot, 2002).

- Bomsdorf, B. and Szwillus, G. (1999). Tool Support for Task-Based User Interface Design. *CHI '99 Extended abstracts: The CHI Is the Limit*, ACM Press, Pittsburgh.
- Brassier, M. and Vander Linden, K., (2002) Automatically Eliciting Task Models from Written Task Descriptions. *In Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces (CADUI'2002)*, (15-17 May, Université de Valenciennes, France), p. 83-90.
- Lu, S., Paris, C. and Vander Linden, K. (1999): Towards the automatic generation of task models from object oriented diagrams. *In Engineering for Human-Computer Interaction, Stephane Chatty and Prasun Dewan (Eds)*, Kluwer Academic Publishers, Boston, 1999. pp 169 – 189.
- Lu, S., Paris, C. and Vander Linden, K., (2002): Generating UML diagrams from task models, *CSIRO/MIS Report Number: 2002/101*, Sydney.
- Ozkan, N., Paris, C. and Balbo, S. (1998). Understanding a Task Model: An Experiment. *In People and Computers XII, Proceedings of Human-Computer Interaction 1998 (HCI'98)*, H. Johnson. K. Nigay and C. Roast (Eds), Springer, 123:138.
- Paternò, F., Mori, G. and Galimberti, R. (2001) CTTE: An Environment for Analysis and Development of Task Models of Cooperative Applications, *In ACM Proceedings of SIGCHI'2001*, March 31-April 5, Seattle, WA. (Extended Abstracts). 21:22.
- Paris, C. Balbo, S. and Ozkan, N. (2000) Novel Uses of Task Models: Two Case Studies. *In Cognitive Task Analysis*, Jan Maarten Schraagen, Susan Chipman and Valerie Shalin (Editors), Laurence Erlbaum Associates, New Jersey, 2000. pp 261 – 274
- Paris, C., Ozkan, N. and Bonifacio, F. (1998). Novel Help for On-Line Help. *In Proceedings of ACM SIGDOC'98 (The Sixteen Annual International Conference on Computer Documentation)*. Quebec City, Canada. 70:79.
- Paris, C., Tarby, J. and Vander Linden, K. (2001). A Flexible Environment for Building Task Models. *In Proceedings of the ICM-HCI 2001*, Lille, France, 313:330.
- Rational Rose, (2002), <http://www.rational.com/products/rose/index.jsp>
- Tamot, (2002), <http://www.cmis.csiro.au/iit/Projects/Isolde/Tamot/Index.asp>
- Tarby, J-C and Barthet, M-F. (1996) the Diane+ method. *In Computer-Aided Design of User Interfaces, Proceedings of the Second International Workshop on Computer-Aided Design of User Interfaces (CADUI'96)*. Namur, Belgium. J. Vanderdonckt (Ed.), Presses Universitaires de Namur, Namur, pp. 95-119.
- van Welie, M., van der Veer, G. C. and Eliens, A. (1998). EUTERPE - Tool support for analyzing co-operative environments, *In the Proceedings of the Ninth European Conference on Cognitive Ergonomics*, Limerick, Ireland, 25:30.
- Vander Linden, K., Paris, C. and Lu, S. (2000) Where Do Instructions Come From? Knowledge Acquisition and Specification for Instructional Text. *In IMPACTS in Natural Language Generation: NLG Between Technology and Applications*, Schloss Dagstuhl, Germany, July 26-28. Becker, T. & Busemann, S. (Eds). DFKI report D-00-01, 1:10.
- Williams, K., (2000) An Automated Aid for Modeling Human-Computer Interaction. *In Cognitive Task Analysis*, Jan Maarten Schraagen, Susan Chipman and Valerie Shalin (Editors), Laurence Erlbaum Associates, New Jersey, 2000. 165:180.

BIOGRAPHY

Shijian Lu is a senior research scientist in CSIRO Mathematical and Information Sciences, Australia. He holds a BSc degree from Shandong Institute of Mining and Technology, China and PhD degree from Leeds University, UK. His research interests include Human-Computer Interaction, usability and software engineering.

Cecile Paris is a Principal Senior Research Scientist and R&D Group Leader at CSIRO/Mathematical and Information Sciences. Paris received a BA in Computer Science from the University of California, Berkeley, and an MS and a PhD in Computer Science from Columbia University (New York). Her research is primarily in Language Technology and multimedia presentation, User Modelling and Human-Computer Interaction (especially Intelligent User Interfaces), but also involves authoring tools and intelligent tutoring systems. She is currently the chair of CHISIG.

Keith Vander Linden is an associate professor in Computer Science at Calvin College in Grand Rapids, MI, USA. He has a PhD in Computer Science and Cognitive Science from the University of Colorado, Boulder and has research interests in Language Technology, Human-Computer Interaction, and Software Engineering. He is currently on sabbatical with Siemens Dematic.