

# Computer Aided Task Model Acquisition From Heterogeneous Sources

Shijian Lu<sup>†</sup>, Cécile Paris<sup>†</sup> and Keith Vander Linden<sup>‡</sup>

<sup>†</sup> CSIRO/MIS, Locked Bag 17, North Ryde, NSW 1670, Australia,  
{shijian.lu, [cecile.paris](mailto:cecile.paris@cmis.csiro.au)}@cmis.csiro.au

<sup>‡</sup> Department of Computer Science, Calvin College, Grand Rapids, MI 49546, USA,  
kvlinden@calvin.edu

## ABSTRACT

Task models have proven useful in many different areas, including user requirements analysis, user interface design, user documentation generation and usability evaluation. However, building task models from scratch has proven to be a difficult process. In this paper, three different tools are presented for automatically acquiring task models from three heterogeneous information sources, namely from object-oriented UML diagrams, from written task scenarios, and from recordings of user and system interaction events. We then demonstrate how task models obtained from these diverse sources can be merged into a coherent whole.

**Keywords:** HCI, task model acquisition, tool, UML, XML.

## 1. INTRODUCTION

Often, the bottom line of developing software systems is improving productivity. That, in turn, requires that the resulting system support end users in performing their tasks in an efficient manner. This almost always implies that the system needs to go beyond utility, though essential, into the realm of usability. As the computer user population diversifies, usability is becoming the determining factor in a system's success or failure. Therefore, it is no surprise that more and more software companies are putting emphasis not only on utility, but also on usability. As with utility, which has many dimensions<sup>9</sup> such as usefulness, reliability, robustness, extensibility, and efficiency in terms of resource usage, usability covers many aspects of system usage<sup>21</sup> including efficiency (in terms of facilitating task performance), learnability, memorability, flexibility, and user satisfaction. Among them, how well user tasks are facilitated is often by far the most important aspect of software usability.

In order to construct software systems to best support users in their task, we need to have a good understanding of user tasks. That is where task analysis and modelling come in. Increasingly, task modelling is used in different stages of software development life cycle, e.g., in requirement analysis<sup>14</sup> and in design<sup>5</sup>. It is also used for implementation<sup>15</sup> and evaluation<sup>3</sup>. Task models can also be used to drive prototyping of user interfaces<sup>6</sup> or act as communication tools between all participants in the software design process<sup>10, 1</sup>. Properly formalised, task models can be used to generate user interface code<sup>17, 22, 13</sup> and user documentation<sup>20</sup>.

Given that task models are so useful in software development, the question of how they can be produced becomes important because building them by hand from scratch can be difficult and time consuming<sup>11</sup>. It is thus desirable to acquire automatically as much of them as possible from existing information sources.

In this paper, we present 3 different tools that acquire task models automatically from 3 different knowledge sources. The approaches are presented in the following 3 sections. In the first approach, a tool is developed to obtain task models from object-oriented UML diagrams. The second approach uses written task scenarios as a source of information. The third approach provides a way to construct task models from the interface events recorded when monitoring a system's use. While it is unlikely that one could acquire a complete task model from any one given source, there is promise in integrating of pieces of task models from different sources. This is the issue discussed in section 5 where Tamot, an open task modelling environment is introduced. With Tamot, it is possible not only to merge fragments of task model from different sources, but also to add or to modify portions of the task model. The paper concludes in section 6.

## 2. DERIVING TASK MODELS FROM UML DIAGRAMS: UML-TO-TASK TOOL (U2T)

Object oriented (OO) analysis and design is the major development paradigm in software industry. As a result, many OO design models exist. The OO design model normally consists of two parts, one modelling system structure (such as class diagrams), and the other modelling system behaviour (such as use cases, use case diagrams, interaction diagrams and state transition diagrams). While task models represent an overt, user-oriented view of a system's behaviour, system behaviour models represent internal, system-oriented view of this system. The common concern on dynamic system behaviour forms the common semantic ground between task models (TM) and system behaviour models (SBM). Based on detailed analysis of the relationship between the two, we can observe the following<sup>7</sup>:

- Use cases and use case diagrams can be seen as equivalent to composite tasks in a task model;
- The containment of use cases by other user-case diagrams resembles the hierarchical structure in a task model;
- Extend relationships in use case diagrams can be seen as equivalent to task sequence preconditions;
- In terms of scope, task models are a sub-set of interaction diagrams, but they contain considerably more information on user tasks;
- While some task attributes are defined explicitly, others are defined implicitly, and some are not defined at all in system behaviour models.

By taking advantage of these similarities between SBM and TM, we developed a UML diagram to task model (U2T) acquisition tool. U2T, implemented in Rose script, can be seen as an extension of Rational Rose. With U2T, task models can be automatically generated from use cases, use case diagrams and interaction diagrams residing in Rose model. For example, the task model fragment “create new task” shown on the right of Figure 1 was generated from the Tamot Rose interaction diagram shown on the left.

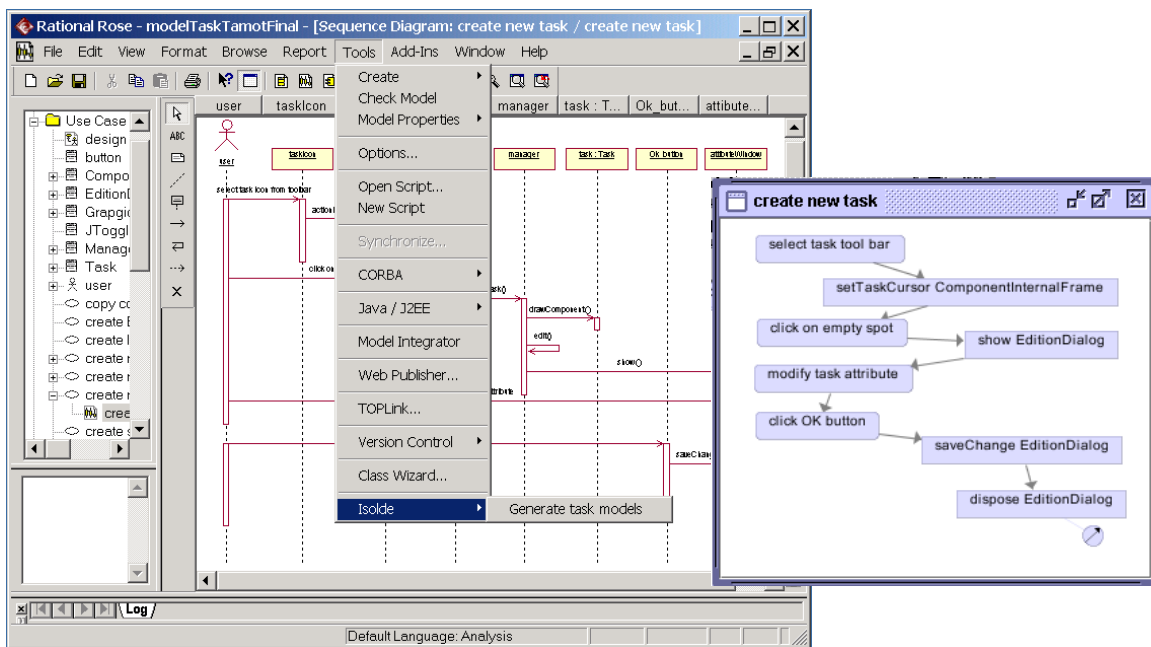
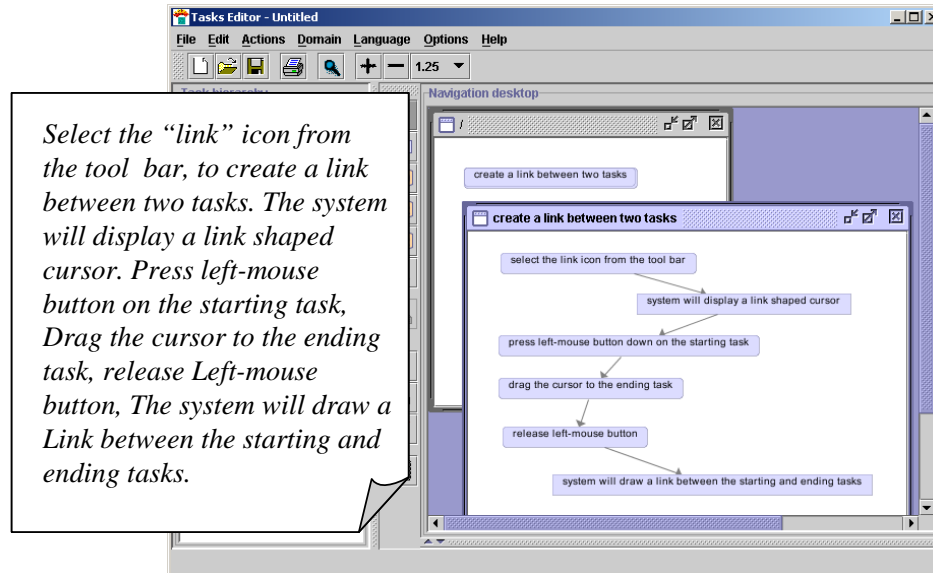


Figure 1. Example for U2T





**Figure 3.** T2T Extract Task Models from Task scenarios

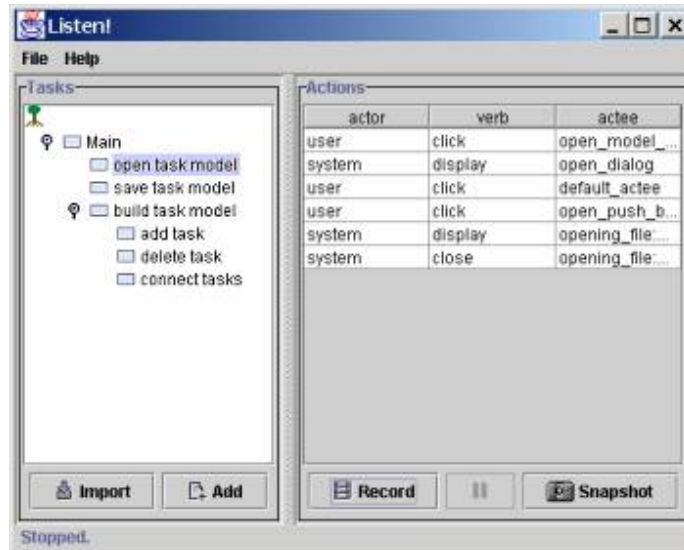
#### 4. ACQUIRING A MODEL BY RECORDING USER ACTIONS: USER INTERACTION RECORDER (UIR)

Sometimes, interactive system prototypes exist at the time when task models are constructed. If this is the case, automatically capturing user system interactions provides a good way to flesh out high-level tasks with a sequence of elementary tasks. That is the purpose of our user Interaction Recorder (UIR) tool. The UIR makes use of SUN Microsystems Java Accessibility Utility<sup>16</sup> to extract GUI objects, such as Menu, Menu Item, Button, Dialog-box, Combo box, and Text Field, etc, and to capture user interaction events associated with those GUI objects, such as, mouse in, mouse out, mouse clicked, mouse double clicked, key pressed, key released, and key typed, etc. Obviously, not all these events are relevant to task models, as some are too low-level. For instance, mouse in or out of a GUI object and key pressed or released are generally considered too low-level to be included in a task model of the form we are considering here. We thus have built into UIR heuristics to filter them out.

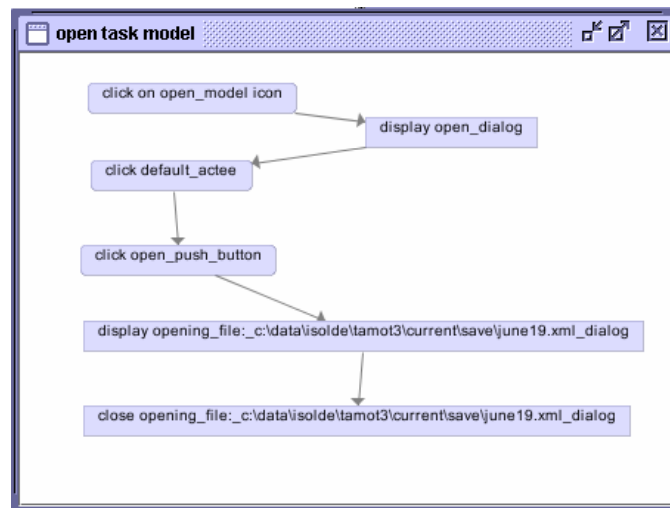
The UIR can be configured to record user interaction with any Java applications. Once configured, the UIR runs parallel with the acting application. Figure (a) shows the user interface of the UIR. The left panel of UIR window displays the task hierarchy, which can be built manually or imported from other sources. Instantiating a higher level task is a three step process:

- (1) Select – the user selects the high level task to be instantiated, e.g., “open task model”
- (2) Press – the user presses the “Record” button, which causes the UIR window to disappear and the system to begin recording actual interactions between the user and the acting application
- (3) Demonstrate – the user interacts with the application, e.g., when the user selects “open model” icon, the system records an “open dialog box” task; when the user clicks on a file name and clicks Open push button, system records an “open file dialog” task and then a “close file dialog” task once the file is opened.
- (4) Stop - A session of recording is finished when the user presses the “Stop”(II) button

A list of recorded tasks is displayed in the right-hand side panel. Figure 4 (b) shows the task decomposition built in this way: the “open task model” task.



(a)

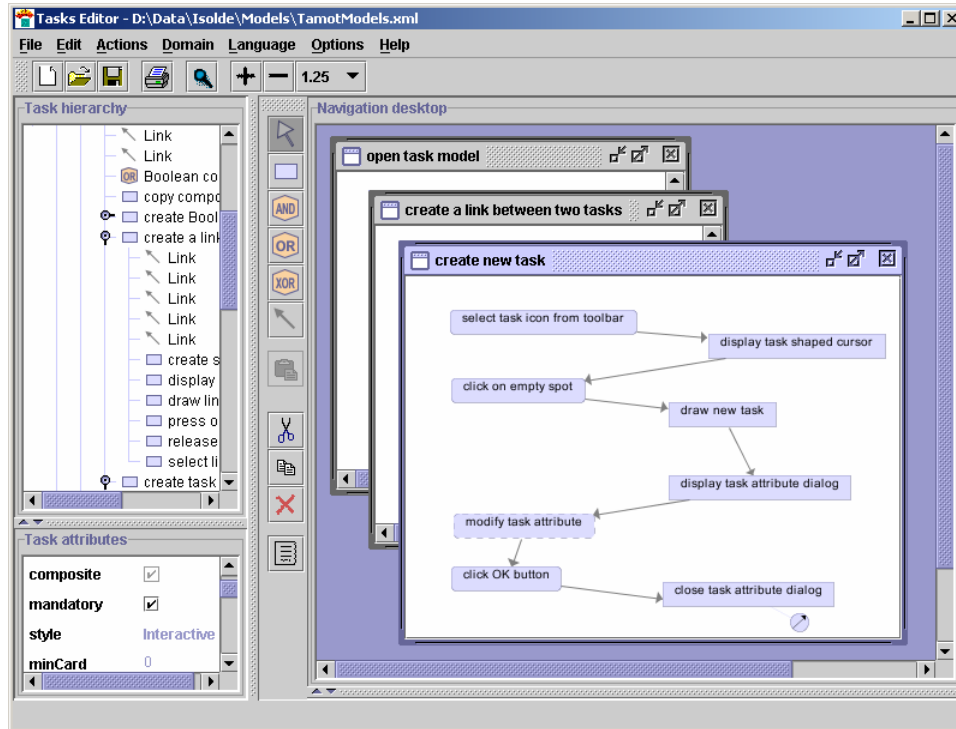


(b)

**Figure 4.** R2T Extract Task Models from User System Interaction

## 5. INTEGRATION

It would be rare for a task model derived from any single given source discussed in sections 2, 3, or 4 to be complete. The result would almost certainly be a task model fragment of some sort. For example, a fragment might contain some high level tasks (obtained from UML use cases for example) or a set of low-level tasks (e.g., obtained through UIR). Even when tasks from all three sources are extracted and consolidated, there might still be missing or inaccurate task information. Thus, we need a platform where task model fragments can be integrated and extended manually if necessary. In this context, we introduce Tamot (Task Modelling Editing Tool), a software tool for supporting task model construction and editing<sup>8</sup>.



**Figure 4.** The main window for Tamot

Tamot is a graphical task model editor. It uses the Diane+ notation<sup>19</sup> and stores the models internally in an XML specified format. The task model acquisition tools, namely, U2T, T2T and UIR, all use the same XML representation for task model fragments, and thus Tamot is able to integrate the model fragments. In addition, Tamot provides a rich set of functionalities which are geared to support task modelling with maximum usability. For instance, Tamot supports conventional copy/cut/paste functionality and short cuts for creating a set of parallel as well as sequenced tasks. Some common operations in task modelling are also made easy such as adding or removing intermediate-level tasks.

Generally, it does not matter from where the partial task models are derived, as long as they conform to the XML DTD defined for Tamot. Any such conforming task model can be easily integrated with the facility to copy/paste across different task models. For instance, to integrate the resulting task model fragments from section 2, 3, and 4, we could first load the resulting task model from section 1 and copy all the tasks need to be incorporated using multiple selections. Then, one can load the resulting task model from section 2 and paste the one from section 1 into the appropriate place. Similarly, task model fragments from section 3 can be incorporated. Figure 5 shows the Tamot main screen with the integrated task model from section 2, 3, and 4.

In addition, the task model fragments may require manual editing. For example, the second task name in section 2 should be “display task shaped cursor” rather than “setTaskCursor componentInternalFrame”. Similarly, the third task name in section 4 should be “(user) click task model file” rather than “(user) click default\_actee”. Tamot provides manual editing features to fix this sort of problem which can be employed either before or after the integration of the model fragments.

## 6. CONCLUSION

Task models are useful in interactive system design. However, their use in practice is severely hampered by the lack of, on the one hand, appropriate tool support, and, on the other hand, proper integration mechanisms. In this paper, we explored approaches to alleviate the effort in task model building by reusing information from various sources. Three information sources, namely, UML design models, task scenarios, and user system interaction, are identified in such a context. By analysing key characteristics of each individual source, tools are built to facilitate the task model acquisition process. Finally, we demonstrated how task model fragments from heterogeneous sources could be fully integrated.

## ACKNOWLEDGMENTS

The authors wish to thank the Isolde team, including Nathalie Colineau, Sandrine Balbo, Thomas Lo, Michael Brassier, Nadine Ozkan, and Jean-Claude Tarby. We also acknowledge the support of ONR (grant #N00014-99-0906), CSIRO and Calvin College.

## REFERENCES

1. Balbo, S. and Lindley, C. Adaptation of a task analysis methodology to the design of a decision support system. In *Proceedings of Interact'97*, Sydney, Australia, 1997.
2. Brassier, M. and Vander Linden, K. Automatically Elicitation Task Models from Written Task Descriptions. To appear in *Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces (CADUI'2002)*, Université de Valenciennes, France, 2002.
3. Card, S.K., Moran, T.P. and Newell, A. *The psychology of human computer interaction*. Lawrence Erlbaum Associations, 1993.
4. Fellbaum, C., editor. *WordNet: An Electronic Lexical Database*, MIT Press.
5. Hix, D. and Hartson, R. *Developing user interfaces, ensuring usability through product and process*. Wiley, 1993.
6. Johnson, P. Johnson, H. and Wilson, S. Rapid prototyping of user interfaces driven by task models. In *Scenario based design: envisioning work and technology in system development* (ed. J.M Carroll), John Wiley, New York, USA, 1995.
7. Lu, S., Paris, C. & Vander Linden, K. : 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.
8. Lu, S., Paris, C. & Vander Linden, K.: Tamot: Towards a flexible task modelling tool. *To appear in proceedings of Human Factor 2002 (HF'02)*, Melbourne, Australia, 2002.
9. Meyer, B. *Object-Oriented software construction*. Second edition, 1997, Prentice-Hall PTR.
10. O'Neill, E.J. Task model support for cooperative analysis. In *Proceedings of CHI'96*, 1996.
11. (Paris and Vander Linden, 1996a) Paris, C. and Vander Linden K. An overview of on-line documentation and CASE tool: Isolde, Report on task 1 and 3. *Technical report ITRI-95-16*, ITRI, University of Brighton, UK, 1996.
12. Paterno, F. and Mancini, C. Developing task models from informal scenarios. In *Proceedings of CHI '99 - May 15-20, 1999, Pittsburgh, USA*.
13. Puerta, A.R. (1997) A Model-Based Interface Development Environment. *IEEE Software*, 14(4), July/August, 41:47.
14. Sebillotte, S. Methodology guide to task analysis with the goal of extracting relevant characteristics for interfaces. *Technical report for Esprit project P6593*, INRIA, Rocquencourt, France, 1995.
15. Smith, M.J and O'Neill, J.E. Beyond task analysis: Exploiting task models in application implementation. In *Proceedings of CHI'96*. (1996)
16. Sun Microsystems. Java Event Listener and Java Monkey, <http://java.sun.com>. 2002
17. Szekely, P. Luo, P. and Neches, R. Beyond Interface builders: Model-based interface tools. In *Proceedings of InterCHI'93*, ACM, 1993.
18. Tam, R. C., Mailsby, D., & Peuerta, A. R. U-TEL: A Tool for Eliciting User Task Models from Domain Experts. *IUI 98: International Conference on Intelligent User Interfaces*. (pp. 77-80). IUI. San Francisco, USA

19. Tarby, J-C and Barthet, M-F. 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, 5-7 June, 1996. J. Vanderdonckt (Ed.), Presses Universitaires de Namur, Namur, 1996, pp. 95-119.
20. 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.
21. Wilson, D. The usability engineering framework for product design and evaluation. *Handbook of the Human Computer Interaction*, second completely revised edition. M Helander, T.K. landauer, and P. Prabhn (eds), 1997, Elsevier Science B.V.
22. Wilson, S. and Johnson, P. Bridging the generation gap: from work tasks to user interface design. In *Proceedings of CADUI'96*, 1996.