Identification Criteria in Task Modeling

Josefina Guerrero García, Jean Vanderdonckt, and Christophe Lemaigre

Belgian Laboratory of Computer-Human Interaction (BCHI)
Louvain School of Management (LSM), Université catholique de Louvain (UCL)
Place des Doyens, 1 – B-1348 Louvain-la-Neuve (Belgium)
E-mail: josefina.guerrero@student.uclouvain.be, {jean.vanderdonckt, christophe.lemaigre}@uclouvain.be

Abstract: Task modeling consists of a fundamental activity that initiates user-centered design in user interface development. It is therefore important to reach the best task model possible and that the task modeling activity remains consistent when the task modeler changes. For this purpose, this paper introduces a set of criteria in order to identify tasks during task modeling in an unambiguous way that results into a task model exhibiting desired properties of quality such as completeness, consistency. In addition, starting and stopping criteria provide designers with guidance on when and how to start and finish the task modeling.

Keywords: Process, task identification criteria, task modeling.

1. Introduction and Related Work

Task modeling (Duursma, 1993; Paterno, 1999) is probably one of the most central activities to conduct in order to ensure user-centered design of interactive systems. A task model is supposed to capture most elements describing how a task is carried out by a particular user in a given context of use or in a given scenario (Limbourg, 2003).

In general, the purpose of model-based design, for instance (Calvary, 2003; Limbourg, 2003; Santoro, 2002; Sinnig, 2007; Traetteberg, 1999; Vanderdonckt, 2003), of User Interfaces (UIs) is to identify high-level models which allow designers to capture specifications of interactive applications from a more abstract level than the implementation level at which the future application will be developed. This allows designers to concentrate on important design aspects without being influenced by the implementation constraints. In particular, the task model's goal is to capture specifications of how a task is carried out in a given context of use (Calvary, 2003) (i.e., a triple user-computing platform-physical environment).

Please use the following format when citing this chapter:

García, J.G., Vanderdonckt, J. and Lemaigre, C., 2008, in IFIP International Federation for Information Processing, Volume 272; *Human-Computer Interaction Symposium*; Peter Forbrig, Fabio Paternò, Annelise Mark Pejtersen; (Boston: Springer), pp. 7–20.

A *task* is an activity that should be performed in order to reach a goal. A *goal* (in UI model-based design) is a desired modification of state or an inquiry to obtain information on the current state of an interactive application (Paterno, 1999). One of the advantages of task modeling lies in its characterization of the logical activities that an interactive application must support independently of any underlying technology or implementation. Task modeling has become today a widely recognized activity in the UI development life cycle. Several task models are precisely defined and are adequately made editable through software (Limbourg, 2003): Hierarchical Task Analysis (HTA) is supported by Architect, Méthode Analytique de Description de tâche (MAD) is supported by SUIDT (Aït Ameur, 2003); Goals, Operators, Methods, and Selectors (GOMS) is supported by QGOMS (Beard, 1996), Task Knowledge Structures (TKS) is supported by ADEPT, Groupware Task Analysis (GTA) is supported by EUTERPE (van der Veer, 2000), ConcurTaskTree (CTT) (Paterno, 1999) is supported by CTTE-editor (Santoro, 2002), Diane is supported by the Diane+ editor, ISOLDE is supported by an eponym editor. Despite these recent advances, task modeling still remains a challenging problem for the following reasons:

- Although there is more or less a consensus about its definition, about the information to be captured in a task model, and about the tool usage, yet there exists a significant gap on the means to be used to obtain such a task model.
- In the literature, there is little or no methodological guidance on how to obtain such a task model. When some guidance is provided, it mainly consists of syntactical rules to get a task model without any defect. These rules are completely independent of the domain of human activity. Formal validation of a task model is considered very important (Aït Ameur, 2003), but its external validation (i.e., with respect to the users' needs) is equally important.
- This often results in many variations in the task model obtained in the end: different people may produce different, possibly inconsistent, task models for the same design problem because they do not share the same perception or rules; a same person (e.g., a task analyst, a task modeler) may produce task models with different levels of details depending on the design problem; even more, a same person can produce different task models for the same design problem over time.
- People experience some trouble in identifying the points where to start the task modeling and where to stop it. Until when should we proceed with task modeling such as decomposition and refinement?
- People may diverge on their interpretation of what needs to be captured in a task model and what not, in particular what makes a task and what does not make a task? Several different interpretations of what a task model is and what task modeling should be co-exist without reaching any consensus (Limbourg, 2003).
- This problem is even more acute when task modeling is conducted in the context of a larger design problem such as workflow modeling (Guerrero, 2008):

it is difficult to distinguish what is workflow specific from what is task specific.
- The relationship between a task model and use cases that depict a particular scenario as a task model instance is obvious, but yet hard to obtain (Constantine, 1999), although some method exists that establishes this type of relationship (Santoro, 2002).
- Equally important are the design rationale techniques used to argue and to reason about the task modeling decisions (Lacaze, 2006; MacLean, 1991).

Since there is no apparent need to conduct any research for the model part (the task model has gained today a precise and shared definition) or any development for the tool support (excellent software are publicly available for this purpose, such as CTTE-editor (Paterno, 1999)), we believe that there is still some research to be conducted for improving the methodological guidance for conducting task modeling. This guidance should be of course independent of any task model definition or tool since the task modeling activity should be achieved consistently by any person.

In order to fit this purpose, namely by addressing the aforementioned shortcomings, this paper will in Section 2: define the underlying models, define an expanded task life cycle, and define a set of criteria for identifying a task with respect to other concepts. Section 3 will exemplify a case study based on this methodological guidance and how software that has been developed for this purpose may facilitate applying and structuring this guidance. Section 4 will provide a qualitative cost-benefit analysis of this guidance and present some future avenues of this research.

2. Toward Methodological Guidance for Task Modeling

The next sub-sections will define, respectively, the three dimensions that are typically found in a software development method (Bodart, 1989): the underlying models, the development life cycle, and the method part. These concepts will be exemplified in Section 3.

2.1 Underlying Models

A single model cannot capture all aspects relevant to task modeling. If everything would have been concentrated in such a model, then it would not adhere to the *Principle of Separation of Concerns*. It is therefore necessary to have a family of interconnected models so as to preserve correlativity between models. It is not needed to have all models involved in any development life cycle, but it is desirable to have such a family of models in order to capture all desired aspects.

The Cameleon Reference framework structures the UI development life cycle into four subsequent layers (Calvary, 2003): task and domain models, abstract user

interface (AUI) model, concrete user interface (CUI) model, and final user interface (FUI). In order to properly capture various aspects in respective models, this framework has been expanded (Fig. 1) in order to incorporate the concepts of workflow, process, and resource (Guerrero, 2008). A workflow is decomposed into processes, which are in turn decomposed into tasks. Each task could be supported thanks to a resource model, in which three types of resources can be found: human resources (i.e. a user stereotype), material resources (e.g., hardware, network, machines), and immaterial resources (e.g., software, operating system). More details about the attributes and methods of these classes could be found in (Guerrero, 2008). Fig. 1 only represents the UML class diagram of this metamodel without any attributes or methods. This expanded framework will serve as a reference framework in order to identify concepts that are relevant to these different models, if needed. If a scenario or a use case does not incorporate any element relevant to a particular model, then this model is simply not created.

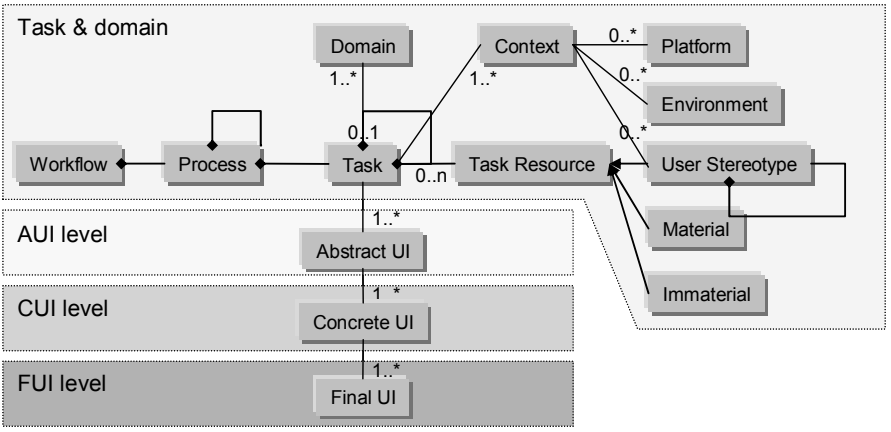

Figure 1. The four levels of the Cameleon Reference framework expanded in this paper expanded from (Calvary, 2003).

This expanded framework allows us to support a total integration of models in the context of model-based UI development: vertical integration is obtained when different models at different levels of abstraction are interconnected and; horizontal integration when different models describing different aspects at the same level of abstraction are interconnected. In our case, horizontal integration is obtained at the first level of Figure 1 since all models are clearly separated and interconnected. Vertical integration is ensured since the cornerstone model, i.e., the task model, initiates the rest of the subsequent levels of abstraction (i.e., AUI, CUI, and FUI). Now that the relationship between a task model and other related models have been outlined, we would like to define the central concept of task in such a way that its identification becomes as precisely as possible. Therefore, we define:

- A *task model* describes the task analyst's view of the end users' interactive tasks while interacting with the system, where a task is any operation unit that is carried out in the same time-space frame with the same set of resources for the same information set.
- A *process model* describes how tasks are arranged in time, space, and resources so as to form basic more elaborate operation units satisfying transitive disclosure (i.e., having a clear entry point, a middle portion, and an exit point). Each process consisting of a number of tasks and a set of conditions that determine the execution order of the tasks, and task relationships.
- A *workflow model* describes the flow of the work inside, outside, and between organizations. In other words, a workflow model is aimed at representing the flow of work inside and outside organizational units in terms of tasks that describe the way humans perform tasks to accomplish a goal.

From the above definitions, we will first deduce an expanded task life cycle in the next sub-section and precise criteria in order to identify what a task is, what a process is, and what a workflow is, depending on the presence of these concepts in a case study, a design problem, or a scenario of interest.

2.2 Expanded Task Life Cycle

Tasks are dynamic entities whose life-cycle can be described with a small quantity of significant states. Changes in those states, along with transitions between those states, are produced by the stimulus. On the one hand, there are some particular states of interest that result from the definition: cancellation, suspend/resume, among others. On the other hand, we have to separate the definition of a task from the means to allocate it to resources and from the states where the task is indeed executed. Each task can therefore benefit from the following actions thus resulting in the life cycle of Figure 2:

- *Cancel*: any task can be cancelled, once started at any moment.
- *Delegated*: any task can be delegated to another resource (e.g., another user stereotype) once it has been allocated or initiated.
- *Finished*: any task is said to be finished when the goal is reached.
- *Undo*: any task can be undone once initiated.
- *Redo*: any task that has been undone can be redone.
- *In course*: any task could be executed.
- *Repeat*: any task that has been accomplished can be repeated as many times as necessary.
- *Review*: any completed task can be reviewed before it is considered finished.

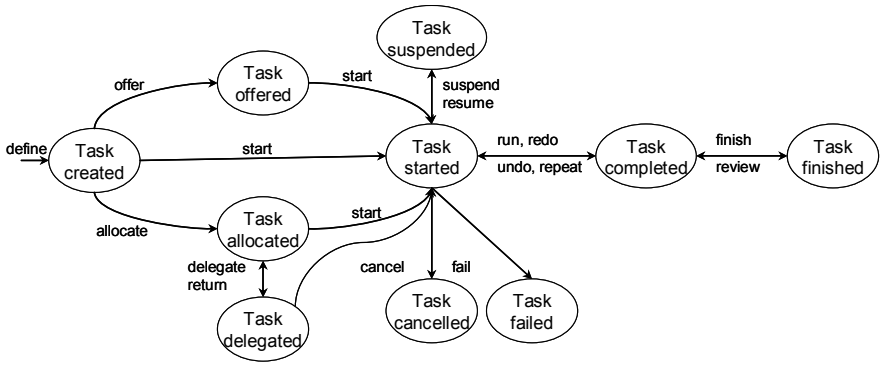

Figure 2. Expanded task life cycle.

2.3 Identification Criteria for a Task, a Process, and a Workflow

Since a task is defined as an operation executed while four dimensions remain constant (i.e., time, space, resources, information), any variation of any of these four dimensions, taken alone or combined, thus generate a potential identification of a new task in the task modeling activity. A task is like "a play", which is decomposed into acts, which are in turn decomposed into scenes (Figure 3a). In a piece, an act is a unit during which time, space, and actors remain constant, even across scenes. Variation to any of these dimensions means that another act has begun. Similarly for a task, any variation of time, space, set of resources or information set will mean a change of task. It is important that identifying a new task is independent of any task decomposition. In the IDA (Interactive Design Approach) method (Bodart, 1989), the decomposition is limited in that it is only possible to decompose into four levels: a project is recursively decomposed into interactive applications, which are in turn decomposed into phases (with one level only), which are in turn decomposed into functions (Figure 3b). In our method, we do want to stay independent of any decomposition (in order to accommodate any task modeling approach). Therefore, we do not limit the number of decomposition levels, as represented by the recursive aggregations in Figure 1 for processes and tasks.

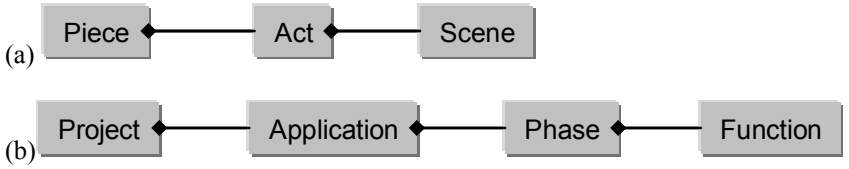

Figure 3. Some decomposition: (a) a play; (b) a project into sub-units.

From the definition of a task, we deduce the following identification criteria:

1. *Change of space* (or change of location): when the scenario indicates a change of location of the operations, a change of task may occur. Therefore, any scenario fragment like "in the headquarters, the worker does …, then in the local agency, the worker does…" indicated a change of space, therefore a change of task. The main location where the task is carried out takes the advantage. In case of collaborative or cooperative tasks, the main location is considered to detect whether there is any change of location.
2. *Change of resource*: when the scenario suggests that new or different resources are exploited, a change of task may occur. We distinguish three categories of resources from Fig. 1:
   a. *Change of resource of type "User stereotype"*: when another user stereotype appears in the scenario may indicate that there is a change of task. For example, "a clerk does …, then an employee files the results of …". The two different names for two different users indicate a change of task. This reasoning always forces identifying who is the main responsible person for carrying out a task. Again, in case of collaborate or cooperative tasks, the main user stereotype involved in the task is selected.
   b. *Change of resource of type "material"*: when another material resource appears in the scenario, a change of task may occur. For example, "a clerk enters the customer's data on a PocketPC, and then takes a picture with a mobile phone camera" indicates two tasks resulting from the usage of two different resources, here a PocketPC and a mobile phone. This should not be confused with a task that is performed on different computing platforms, like in the context of a multi-device UIs (Santoro, 2002). Thus, any significant change of software and/or hardware resource may indicate that there is a change of task.
   c. *Change of resource of type "immaterial"*: when another immaterial resource appears in the scenario, a change of task may occur. For example, "a network administrator uses a specific software to check network status; s/he uses another software to update the computers of the network". The two different types of software involved indicate a change of task.
3. *Change of time*: when the scenario indicates a different time period in which the task is performed, a change of task may occur. We differentiate four criteria resulting from any potential change of time:
   a. *Existence of an interruption*: when the task is interrupted by an event that changes the time period. For instance, "an employee registers every incoming complaint. After registration a form is sent to the customer who returns the form within two weeks". A task can be interrupted for many reasons, such as an error, an external event, a dynamic task.
   b. *Existence of a waiting point*: when in the development of a task there is a moment where it is necessary to wait that something occurs for continuing, a change of task may occur. We have two types of a waiting points:
      i. *Waiting point of type "decision"*: when a determination arrives at after consideration of a question, a change of task may occur. For example, "after the preparation of a flight plan, the pilot will take the decision to fly". A decision can be made by a human agent, a system

agent or in a mixed-initiative way. In any case, there could be as many different tasks as there are different alternatives coming out the decision. In this way a decision could result into two tasks after the decision or multiple tasks (like in a "case of").

ii. *Waiting point of type "accumulation"*: when there is necessary to create a waiting list for some information, a change of task may occur. For instance, "due to a car accident, more complaints arrived yesterday at the insurance agency and the employee had to register all incoming complaints to send as a group to directors". The accumulation may be related to documents (or messages, or data) or to processes (e.g., a repetition of tasks).

c. *Permanence of execution unit*: when the task execution depends of the results of at least two previous asynchronous tasks. For instance, "the results of an insurance complaint are delivered to the client when the complaint manager provides whether the complaint applies or not and when the evaluator provides the estimated cost".

d. *Periodicity of execution*: when there are different periodicities established to execute tasks, then a change of task may occur. For example, "every Monday the employee does a backup of the information". This criteria is often detected when one can determine that the frequency of one task is different from the frequency of another. For instance, a backup (automatic) task could be incremental every day and full each Friday. In this case, we separate two tasks (incremental vs total backup) because their frequencies are different: every day vs. every Friday.

4. *Change of nature*: when the scenario represents a change of category, a change of task may occur. A task may have any of the following nature: manual, automated, interactive or mechanical. Any change of this nature may indicate a change of task. For instance, "first a secretary types a letter in the computer (interactive), after a printer prints the text (automatic) and finally the manager signs the letter (manual)".

When doing task modeling, it is important to decide how far the decomposition of tasks is to proceed. This depends of course of the context and purpose of task modeling; however some stopping criteria based on the task life cycle (Fig. 2) are:

1. For *horizontal stopping*: when the task is finished, or the task is canceled or the task failed.
2. For *vertical stopping*: when a task can be performed in a simple and well-determined way (i.e. the task cannot be decomposed in sub-tasks), when the task is executed by a software system and we do not intend to replace this system with anything else (Duursma, 1993).

Table 1 lists a set of parameters to identify a workflow and a process following the methodology applied in the task identification. Notice that our method assumes that a textual description of the problem has been gathered by any method, for instance, interviews, and the author assumes this information as complete. The way this information is collected and its completeness in not in the scope of this

paper, so, as to compare our method with other task analysis method. The above guidelines covers the challenges described in Section 1.

Table 1. Identification criteria

	Time	Space (location)	Resource	Type
Workflow	Series of time periods	Different locations; same organization	Same or different groups of resources	-
Process	Series of time periods	Different locations	Within groups, group as a whole, or among groups	Primary (production), secondary (support), or tertiary (managerial)
Task	Same time period	Same location	One or two types of resources	User, interactive, system, abstract, or machine task

3. Feasibility study

The literature demonstrated that there is a rich set of task modeling approaches and practices, for instance (Limbourg, 2003; Paterno, 1999; Rosson, 2007; Sinnig, 2007; Traetteberg, 1999; van der Veer, 2000). The set of identification criteria that was defined in the previous section is not intended to replace any of these approaches and practices. Rather, the identification criteria are interpreted as a complementary tool that disambiguates the tasks involved in task modeling. These criteria do not change the task analysis activity per se, but, again, attempts to facilitate the identification of tasks.

In order to demonstrate that this approach based on identification criteria is feasible, we have conducted a series of case studies, some of which being reported in (FlowiXML, 2008). Empirical evidence of method's worth should be one of the first requirements for its acceptance. For space reasons we present a simplified version of the case study related to an insurance company.

3.1 Case Study

An insurance company offers insurances of different types as: medical, life, and accidents. Each type of insurance is processed in different section of the company, in different way, and by different resources. To process claims that result from car accidents, the company uses the following procedure:

- Every claim is registered by a desk clerk. He introduces the customer's full name, policy number, and the problem's details in the insurance company system.
- After the registration, an employee checks the insurance's information as the status, the insurance's coverage, and the damage history.
- Also, he phones to the garage to get the cost of the damages.
- After executing these tasks a decision is made by the manager, with two possible outcomes, positive or negative.
- If the decision is positive, then the insurance company will pay. An accountant handles the payment.
- The secretary of the insurance company sends a letter to the customer. Also, a letter will be sent to the customer when the decision is negative explaining the causes of the rejection.

From the text, we identify that each type of insurance is a process; all of them form a workflow. In the particular case of the accident insurance process, it is possible to classify principal tasks, add an identifier and a name. We can specify if the task needs that a previous one is finished, make a brief description of the task, list the identification criteria and identify the nature of the task. The previous steps to classify tasks are supported by a task classifier, software tool specialized (Fig. 4).

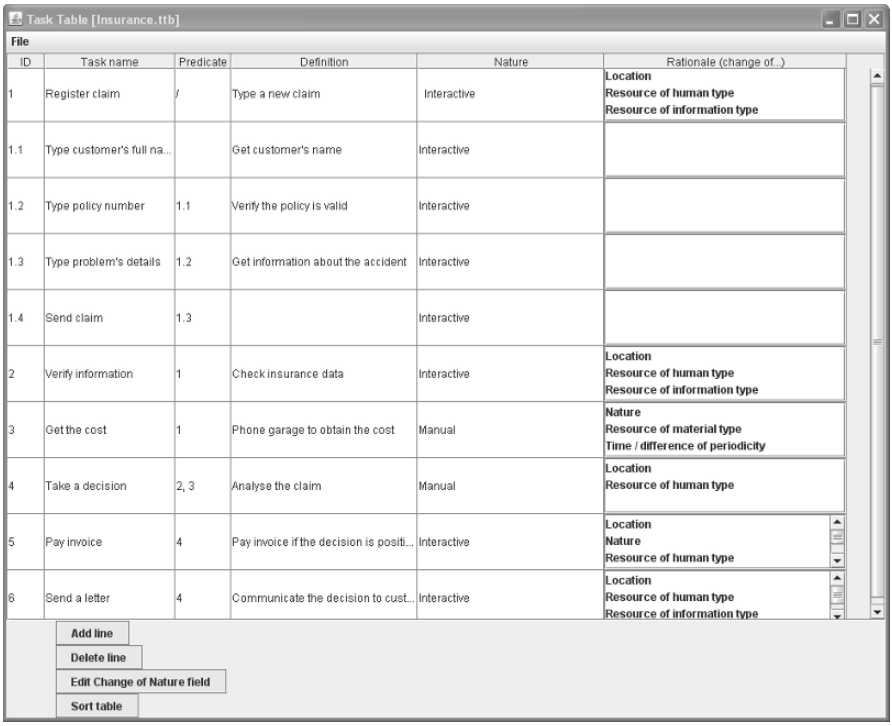

Figure 4. Tool to classifier tasks.

As we observe, a task can be decomposed in sub-tasks, for instance Task 1 *Register claim* is decomposed in sub-tasks 1.1 *Type customer's full name*, 1.2 *Type policy number*, 1.3 *Type problem's details*, and 1.4 *Send claim*. Even though task 2 and task 3 are executed by the same resource, they present a change of resource type (immaterial and material), a change of nature (interactive and manual), and a change of time. In order to execute task 4, task 2 and task 3 should be finished; also there is a change of space and resource. If the result of task 4 is positive, then task 5 could be executed with a change of space, resource of type user stereotype, immaterial, and exist a waiting point of type decision. Task 6 will be executed after task 4, it is necessary a waiting point of type decision, a change of space, and a change of resource.

After the identification and classification of tasks, processes and workflow, we can represent all these components in a workflow editor tool (Figure 5). The workflow describes as the work in organization flows by defining models of: process (what to do?) and tasks (how to do it?). For each process a task model can be specified to describe in detail how the task is performed.

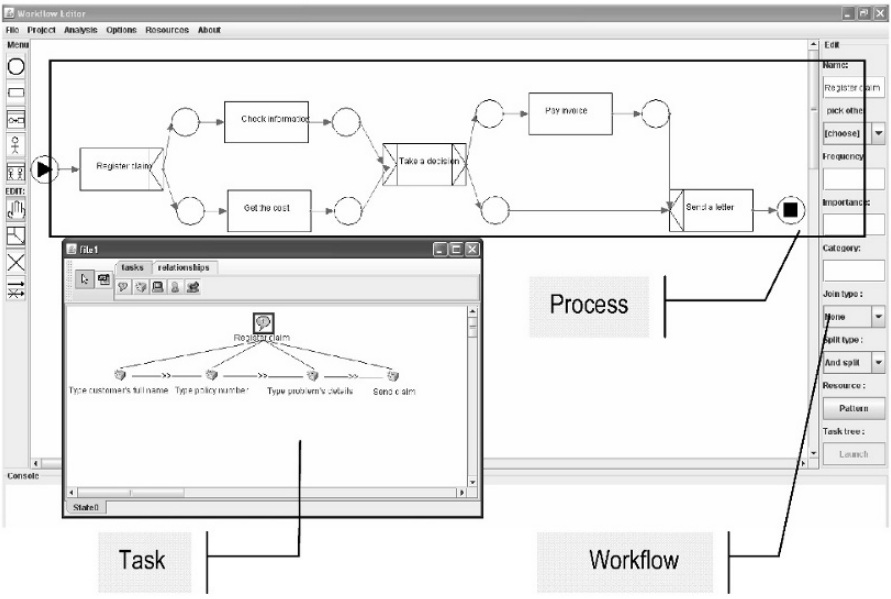

Figure 5. Workflow model.

3.2 Validation

To validate this approach, we are working on the theoretical and empirical considerations in order to address the internal and external validations respectively. The

empirical validation is being conducted with multiple case studies research available on www.usixml.org and in (Vanderdonckt, 2005)

- **Bank credit for a car**. In this work we will focus just in analyzing the workflow of how to ask a credit to buy a car. We consider this service interesting as it involves at least three different organizations (the bank, car agencies, car buyer-seller company), related in a strategic join venture alliance. Car agencies and the car seller-buyer agency are benefit from the bank credits.
- **Organization of a Triathlon**. The organization of this event is wide and full of flows of information. In short, people wanting to organize an event have to contact different companies and fulfill the needs of the athletes and the spectators.
- **Order personalized compression stockings over Internet**. The case study is situated in the phlebology domain. It deals wit an Internet order system, allowing the ordering of personalized support stockings. The main idea of t is system is to calculate a 3D model of the customer's legs from a series of digital pictures taken from his/her legs. This model will be sent, coupled with a specific order, via Internet to the manufacturing department.

So far, we have conducted some informal observations of how people use the same set of criteria for supporting task modeling, but these observations only indicate some properties:

- The interpretation of the identification criteria is usually perceived as straightforward by people who have used it because the criteria have been defined very distinctively from each other ; in addition, the fact that a task should, in principle, keep constant the space, the time, and the resources is an easy-to-remember and fast-to-apply procedure.
- The set of criteria may lead to different designs for the same task to model. This does not mean that that the resulting designs are inconsistent with each other, but simply that the designer has chosen a particular design alternative by prioritizing the identification criteria. We observed however that designers who are using the same criteria with the same priorities tend to reach a similar design. This should be made clear to designers because some of them reported to us that they thought they should reach a single design if they use the same set of criteria.
- The usage of criteria reinforces the need for recording the design rationale as recommended in (Lacaze, 2006; MacLean, 1991).
- The usage of identification criteria also permits to separate a task from its sub-tasks. If one decides that this action should become a high-level task, then the corresponding criteria should be used for this purpose; when one decides that this action should become a low-level task (e.g., a function, a leaf node in the task model, or a sub-task), then other criteria are exploited to justify this design decision.

4. Conclusion

In this paper, we have introduced a set of precise criteria that can be used in order to identify a task in a textual scenario and to distinguish a task from other concepts like process and workflow which are located at another level in the hierarchy, but at the same level of abstraction. The main advantage of these criteria is that they can be used for any task modeling activity, whatever the task model notation or method used. The second main advantage is that a convergence across designers can be observed when the same textual scenario is given to different persons, thus increasing the internal consistency of the resulting task model. The same advantage is propagated to other models at the same level of abstraction. For this purpose, the Cameleon Reference framework (Calvary, 2003) has been expanded in order to illustrate vertical and horizontal integrations.

In order to support this activity in scenario-based design, a piece of software has been implemented in Java 1.5 that enables designers to conduct the modeling approach that is compatible with this expanded framework and by applying the set of identification criteria in a systematic way. Each time a task has been properly identified, i.e., with at least one identification criteria (multiple criteria could be used to identify the same task), it is then subject to deep task modeling, in connection with the other aspects such as process and workflow. The software then automatically generates a report that can be later used to justify aspects of task modeling within a process, or process modeling within a workflow.

Work in progress includes the evaluation of task analysis methods compare with the one presented in this paper. Dealing with user errors or problems during interaction will be examined to determine their impact in the method. Finally, further empirical evaluation will be conducted confronting two groups of task analysts, the first group, designing a solution without using the guidelines and the second using them. Thus, these results will contribute to the credibility of the proposal.

Acknowledgments. We gratefully acknowledge the support of the SIMILAR network of excellence (http://www.similar.cc), the European research task force creating human-machine interfaces similar to human-human communication of the European Sixth Framework Programme (FP6-2002-IST1-507609) and the CONACYT program (www.conacyt.mx) supported by the Mexican government. We also greatly thank the anonymous reviewers for their constructive feedback that was helpful for improving this manuscript.

References

Aït Ameur, Y., Baron, M., and Girard, P.: Formal Validation of HCI User Tasks. In: Proc. of the Int. Conf. on Software Engineering Research and Practice SERP'2003 (Las Vegas, June 23-26, 2003), pp. 732-738, CSREA Press (2003).

Beard, D., Smith, D., and Danelsbeck, K.: QGOMS: A direct-manipulation tool for simple GOMS models. In: Proc. of ACM Conf. on Human factors in Computing Systems CHI'96 (Vancouver, April 14-18, 1996), pp. 25-26, ACM Press, New York (1996).

Bodart, F., and Pigneur, Y.: Conception assistée des systèmes d'information : modèles, méthode, outils, Dunod, Paris (1989).

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces, Interacting with Computers, 15(3), pp. 289-308 (June 2003).

Constantine, L.L., an Lockwood, L.A.D.: Use cases in task modeling and user interface design. In: Proc. of ACM Conf. on Human Factors in Computing Systems CHI'99 (Pittsburgh, May 15-20, 1999), p. 352, ACM Press, New York (1999).

Duursma, C.: Task Model definition and Task Analysis process, ESPRIT Project P5248 KADS-II KADS-II/M5/VUB/RR/004/1.1c, Vrije Universiteit Brussel, Brussels (1993).

Guerrero, J., Vanderdonckt, J.: FlowiXML: a Step towards Designing Workflow Management Systems, Journal of Web Engineering, 4(2), pp. 163-182 (2008).

Lacaze, X., Palanque, P., Barboni, E., Bastide, R., and Navarre, D.: From Dream to Realitiy: Specificities of Interactive Systems Development with respect to Rationale Management. In: A.H. Dutoit, R. McCall, I. Mistrik, B. Paech (Eds.), Rationale Management in Software Engineering, pp. 155-172, Springer, Heidelberg (2006).

Limbourg, Q., and Vanderdonckt, J.: Comparing Task Models for User Interface Design. In: D. Diaper, N. Stanton, N. (Eds.), The Handbook of Task Analysis for Human-Computer Interaction, pp. 135-154, Lawrence Erlbaum Associates, Mahwah (2003).

MacLean, A., Young, R.M., Bellotti, V., and Moran, T.: Questions, Options and Criteria: elements of design space analysis, Journal on Human Computer Interaction, 6(3-4), pp. 201-250 (1991).

Paterno, F., and Mancini, C.: Developing task models from informal scenarios. In: Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'99 (Pittsburgh, May 15-20, 1999), ACM Press, New York (1999).

Rosson, M.B., and Carroll, J. M.: Scenario-based Design. In: Sears, A., Jacko, J.A. (Eds.), The human-computer interaction handbook: fundamentals, evolving technologies, and emerging applications, CRC Press (2007).

Santoro, C., Mori, G., and Paterno, F.: Ctte: Support for developing and analyzing task models for interactive system design, IEEE Transactions on Software Engineering, 28(9), pp. 797-813 (September 2002).

Sinnig, D., Chalin, P., and Khendek, F.: Towards a Common Semantic Foundation for Use Cases and Task Models. In: Proc. of the 1st Int. Workshop on Formal Methods for Interactive Systems FMIS'2006, Electronic Notes in Theoretical Computer Science, Vol. 183, pp. 73-88 (11 July 2007).

Trætteberg, H.: Modelling Work: Workflow and Task Modelling. In: Proc. of 3rd Int. Conf. on Computer-Aided Design of User Interfaces CADUI'1999 (Louvain-la-Neuve, October 21-23, 1999), pp. 275-280, Kluwer Academics Publishers, Dordrecht (1999).

Vanderdonckt, J., Furtado, E., Furtado, V., Limbourg, Q., Silva, W., Rodrigues, D., and Taddeo, L.: Multi-model and Multi-level Development of User Interfaces. In: A. Seffah, H. Javahery, (Eds.), Multiple User Interfaces - Cross-Platform Applications and Context-Aware Interfaces, pp. 193-216, John Wiley, New York (November 2003).

Vanderdonckt, J.: A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In: Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, June 13-17, 2005), Lecture Notes in Computer Science, Vol. 3520, pp. 16-31, Springer, Heidelberg (2005).

van der Veer, G., van Welie, M.: Task based groupware design: putting theory into practice. In: Proc. of the ACM Conf. on Designing Interactive Systems: Processes, Practices, Methods, Techniques DIS'2000 (New York, August 17-19, 2000), pp. 326–337, ACM Press, New York (2000).

Website FlowiXML. Available via UsiXML. http://www.usixml.org/index.php?mod=pages&id=40. Accessed April 14th, 2008.