Modeling and Simulation of Workflow Processes Using Multi-level Petri Nets

Marina Flores-Badillo, Ernesto López-Mellado, Mayra Padilla-Duarte

CINVESTAV Unidad Guadalajara
Av. Científica 1148, Col. El Bajío, 45015 Zapopan, Jal., México
{mflores, elopez, mpadilla}@gdl.cinvestav.mx

Abstract. A method for modeling workflow processes using a Petri net based multilevel formalism is proposed, which yields modular and hierarchical descriptions of the organization, the processes, resource management, and user interactions. The models allow the synthesis of agent based software in which mobile agents guide the *cases* through the organization units; the software can be interactively executed for simulation purposes or distributed within a computer network for implementing a workflow management system. A tool for editing and simulating the models is presented.

Keywords: Workflow processes, Hierarchical models, Multi-level Petri Nets, Simulation.

1 Introduction

Since the early nineties, Workflow Management Systems (WfMS) has been widely used to improve organizational performance in a broad variety of industries [1]. Its primary characteristic is the process automation involving combination of human and machine-based activities [22], [3].

According to the Workflow Management Coalition (WfMC, an international nonprofit organization responsible for the establishment of standards and some common grounds in the field), a WfMS is: "A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications" [4]. Although the most prevalent use of WfMSs is within the office environment in staff intensive operations such insurance, banking, legal and general administrations, etc., it is also applicable to some classes of industrial and manufacturing applications [5].

Workflow is concerned with the partial or full automation of business processes [4]. Also workflows can be defined as activities involving the coordinated execution of multiple tasks performed by different processing entities (persons or machines) [6].

Workflow management aims to improve process performance and help to achieve business goals with high efficiency [7]. In WfMS design, the modeling framework is a key aspect but, despite the efforts of the WfMC, it lacks a standardized theory that

provides a theoretical background [3]; for this reason there are many different modeling approaches which have their strengths and weaknesses in different aspects [8]. However, Petri Nets (PN) has been used for formal modeling of workflows (first used for Zisman [9] in 1977). Some reasons for adopting PN include its formal semantics, graphical description, and abundance of analysis techniques [10]. According to [11], PN are also a formal technique providing modeling and analysis. Currently PN are widely used for specifying, analyzing and verifying the properties of static workflow models [3], [12], [13], [14], and extended for improve the expressiveness [15], [16].

Although PN provides clear and unambiguous models, when the systems are large and complex the handling and building of ordinary PN models becomes a difficult task. In this paper we propose a multilevel modeling technique based in the n-Level Net System (n-LNS) formalism [17] for addressing this kind of problems.

Using this formalism, in this paper a general technique for conceiving multi-level models is proposed, allowing a systematic construction of the model [18]. A workflow model may be clearly represented using three levels, where the first (highest) level describes the company structure and the possible flows between organizational units; the second level net describes the general behavior of a entity which guides the processes within the company according to the defined workflow process definition, which is represented for a net of the third level; also the third level nets describes the tasks that have to be performed to complete the workflow *case*.

The remainder of this paper is organized as follows. Section II summarizes the n-LNS formalism. In Section III we present a methodology for the modeling of workflows using n-LNS; section IV presents some simulation results and the last section some conclusions and future work.

2 The n-LNS Formalism

The formalism follows the approach of nets within nets introduced by R. Valk in [19], where a two level nested net scheme is presented. An extension to the Valk's technique, called n-LNS, has been proposed [17]; in this section we present an overview of n-LNS; a more accurate definition of the formalism is detailed in [17].

2.1 Definition

An n-LNS model consists mainly of an arbitrary number of nets organized in n levels according to a hierarchy; n depends on the degree of abstraction that is desired in the model. A net may handle as tokens, nets of deeper levels and symbols; the nets of level n permits only symbols as tokens, similarly to CPN (Colored PN). Interactions among nets are declared through symbolic labeling of transitions.

Figure 2 sketches pieces of the components of a 4-LNS model. The level 1 is represented by the net NET_1, the level 2 by the nets $NET_{2,1}$ and $NET_{2,2}$, the nets $NET_{3,1}$, $NET_{3,2}$, $NET_{3,3}$, and $NET_{3,4}$ compose the level 3, and the nets $NET_{4,1}$, $NET_{4,2}$, $NET_{4,3}$ form the level 4.

A net of level *i* is a tuple $NET_i = (typenet_i, \mu_i)$, where is composed by a PN structure, the arcs weight ($\pi((p, t), lab)$ or $\pi((t, p), lab)$) expressed as multi sets of variables and symbols, and a transition labeling function declaring the net interaction. μ_i is the marking function (see fig. 1).

A n-LNS model, called net system, is a n-tuple NS= (NET_1, NET_2, … NET_n) where NET_1 is the highest level net, and $NET_i = \{NET_{i,1}, NET_{i,2}, \ldots, NET_{i,r}\}$ is a set of r nets of level i.

The components of a model may interact among them through synchronization of transitions. The synchronization mechanism is included in the enabling and firing rules of the transitions; it establishes that two or more transitions labeled with the same symbol must be synchronized. A label may have the attributes ≡, ↓, ↑, which express local, inner, and external synchronization respectively.

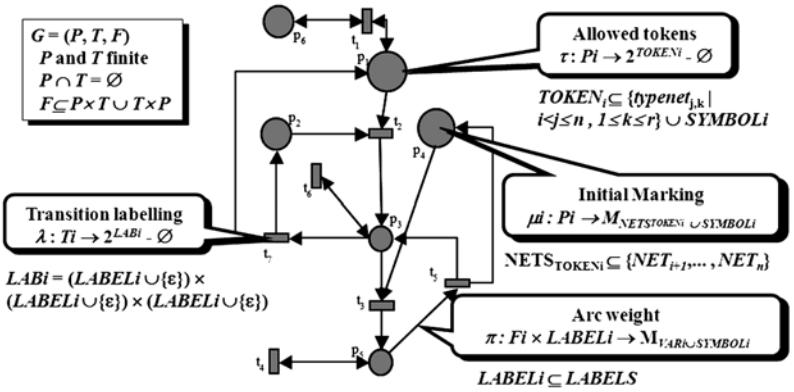

Fig. 1. Example of a Net of Level i, in the n-LNS Formalism, defined as Neti=((typeneti , μi), where typeneti = (G, TOKENi, LABELi, VARi, τ, λ, π)).

2.2 Transition Enabling and Firing

A transition *t* of a net of level *i* NET_i is enabled with respect to a label *lab* if:

-There exists a binding b_t that associates the set of variables appearing in all $\pi((p,t),lab)$.

- It must fulfill that $\forall p \in \bullet t, \pi((p, t), lab)_{<bt>} \subseteq \mu_i(p)$.
  ($<b_t>$ is not necessary when the level net is *n*).

- The conditions of one of the following cases are fulfilled:

Case 1. If there is not attributes then the firing of *t* is autonomously performed.

Case 2. If *lab* has attributes one must consider the combination of the following situations:

{≡} It is required the simultaneous enabling of the transitions labeled with $lab^{\equiv}$ belonging to other nets into the same place *p'* of the next upper level net. The firing of these transitions is simultaneous and all the (locally) synchronized nets remain into *p'*.

{↓}) It is required the enabling of the transitions labeled with $lab^{\uparrow}$ belonging to other lower level nets into $\bullet t$. These transitions fire simultaneously and the lower level nets and symbols declared by $\pi((p, t), lab)_{<bt>}$ are removed.

{↑}) It is required the enabling of at least one of the $t' \in p'\bullet$, labeled with $lab^{\downarrow}$, of the upper level net where the *NETi* is contained. The firing of *t* provokes the transfer of *NETi* and symbols declared into $\pi((p', t'), lab)_{<bt>}$.

The firing of transitions in all level nets modifies the marking by removing $\pi((p, t), lab)_{<bt>}$ in all the input places and adding $\pi((t, p), lab)_{<bt>}$ to the output places.

In fig. 2, NET_1 is synchronized through the transition labeled with $a^{\downarrow}$ with $NET_{2,2}$, $NET_{3,2}$, $NET_{3,4}$ and $NET_{4,2}$ by mean the transitions (locally synchronized) labeled with $a^{\uparrow}$; all these transitions must be enabled to fire. The simultaneous firing of the transitions removes these nets from the input places.

$NET_{2,1}$, $NET_{3,1}$ and $NET_{4,1}$ are synchronized through the transitions labeled with $b^{\downarrow}$, $b^{\equiv}$, $b^{\uparrow}$ respectively; the firing of the transitions changes the marking of $NET_{2,1}$ and $NET_{3,1}$; $NET_{4,1}$ is removed from the place of $NET_{2,1}$.

$NET_{3,3}$ is removed from the input place of $NET_{2,2}$ and $NET_{4,3}$ is removed from $NET_{3,3}$; this interaction is established by $c^{\downarrow}$, $c^{\downarrow\uparrow}$, $c^{\uparrow}$, respectively.

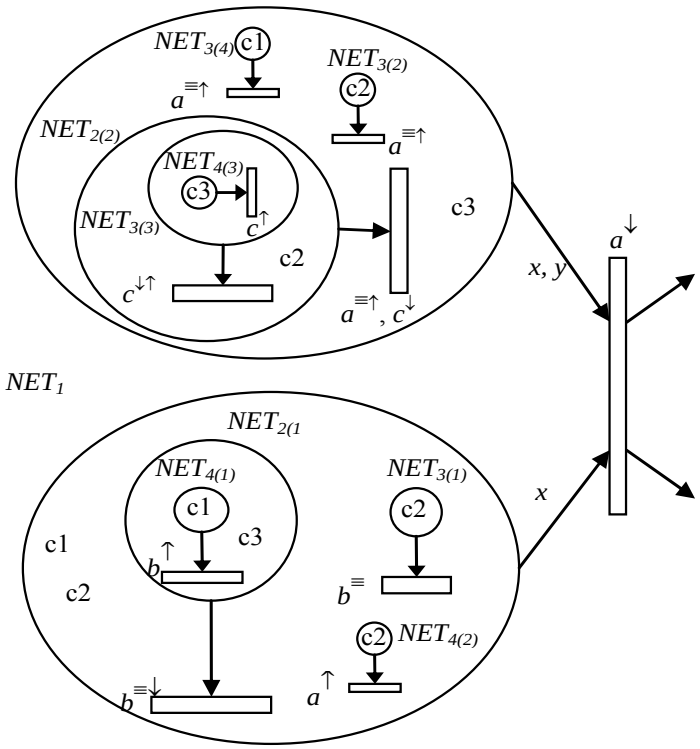

Fig. 2. Piece of a 4-LNS

3 Workflow Modeling

The use of n-LNS induces a modular and hierarchical modeling methodology allowing describing separately the environment and the behavior of all the involved components in a workflow problem, and then, integrated such models into a global one through transition synchronization.

3.1 General Strategy

As mentioned before, the workflow problem can be clearly described using only 3 levels of the n-LNS (see fig. 3).

The highest level net (1) describes the organization structure, i.e. the different *organizational units* involved in the solution of the workflow problem (*case*) and the relations (flow of information, tasks or documents) among them.

The second level net describes the general behavior of an *entity* (software agent) which guides a process (current *case*) within the organizational units and executes the *tasks* within the departments expressed in the *WF-process definition net* (both types of nets modeled as nets of level 3 and described later); this entity net will be a token-net for the net of level 1 [18].

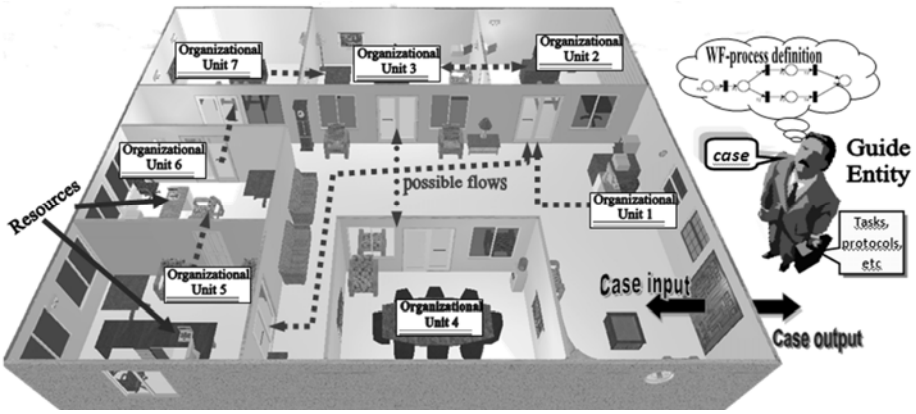

Fig. 3. General modeling strategy

3.2 Modeling Technique

For the sake of readability the proposed method is illustrated through a case study dealing with claim processes in an insurance company.

Consider the following description: "Define the WF for the claim processes in an insurance company in which a customer claims the insurance policy of a personal property (real state, car, life insurance). The company must receive the claim, request

personal data from the customer (insurance policy number, etc.), and verify the insurance validity, payments, and beneficiaries. It must do the adjustment of real damages, validate the *case*, calculate the corresponding assessment, do the necessary payments to the customer if the complain is valid, or inform in case that the process has some invalid data".

Now we are going to describe the different nets that compose the model at the three levels.

Level 1 Net

This net describes the general structure of the company and the possible workflow between every pair of *organizational units* or departments. First, the different departments in which some tasks are executed are identified; then one must consider all the possible flows. The departments are represented by places and the flows by transitions (see Fig. 4); the firing of a transition means that the next task will be performed in the organizational unit that represents the output place of the fired transition. Furthermore, places for representing resources may be added in a classical structure allocation-release.

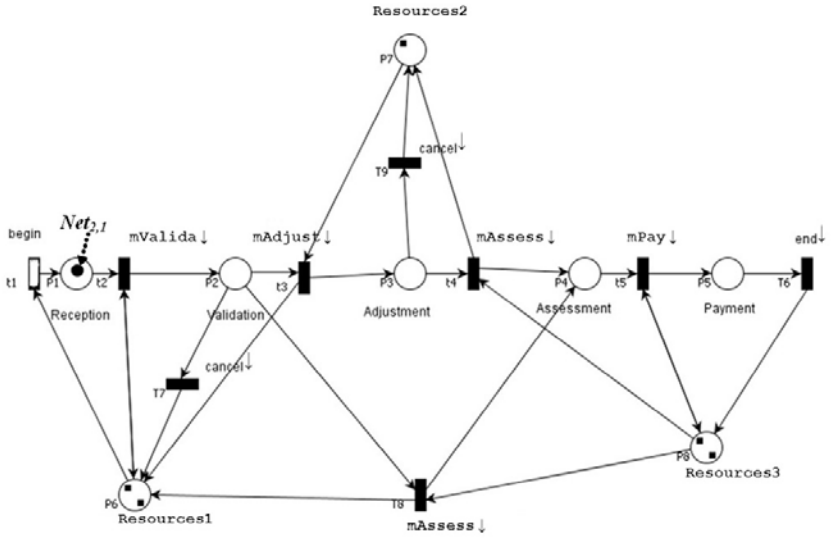

Fig. 4. $Net_{1,1}$. Flow through the organizational units (departments).

The tokens in this net are nets of level 2; for example in place *P1* of the model of Fig.4, there is a token $Net_{2,1}$ which will be detailed later. If needed, add transitions for allowing the cancelation of *cases* (see *T7* and *T9* for Fig. 4). Once this net is defined one must add synchronization attributes to the transitions. Also it is recommended to add an initial transition (for the beginning of the *case*) and a final transition (for the termination of the *case*). All transitions should include the *internal (↓) synchronization attributes*; it implies that the token-nets of level 2 must have some

transitions labeled using the *same symbols*, with the external (↑) synchronization attributes.

The *case* begins with the firing of the transition *t1*, and ends with the firing of the transition *t6* according to fig. 4. Finally define $Net_{1,1}$ = (G, $TOKEN_{1,1}$, $LABEL_{1,1}$, $VAR_{1,1}$, τ, λ, χ, π). We can also include in this model another kind of information, for example, the available resources for each department. This can be achieved adding places to the obtained net (like the labeled as *ResourceI* in fig. 4). The allowed tokens for this net are symbols for the places which represent resources, and a net of level 2 ($Net_{2,1}$) for the places which model *organizational units*.

Level 2 Net

Define $Net_{2,1}$ = (G, $TOKEN_{2,1}$, $LABEL_{2,1}$, $VAR_{2,1}$, τ, λ, χ, π), which models the net that handles the *case*. This net must also include as marking the level 3 nets describing all the *tasks* the agent is able to perform during a *WF-process execution.*

Figure 5 shows the $Net_{2,1}$ for this case study. A net $Net_{3,1}$ *i*s included as token in place *P2*, and the rest of the third level nets ($Net_{3,2}$, $Net_{3,3}$, etc.) are added as tokens for *P1*. *T1* represents the beginning of the execution of the task in the plan in *P2*, and *T2* represents the ending of execution of such a task. For synchronizing the beginning and ending of the tasks execution it is necessary to add labels as showed in fig. 5; the labels $B.Claim\_register^{\uparrow\downarrow}$ in *T1* and $E.Claim\_register^{\uparrow\downarrow}$ in *T2*; both transitions must be synchronized internally (↓). Finally, the transition *T3* is used to synchronize externally the displacement of this net within the organization with the net of level 1 (labels $mValida^{\uparrow\downarrow}$, $mAdjust^{\uparrow\downarrow}$, etc.).

Level 3 nets

• *WF-Process definition net.* In general it is a net having a simple structure, which describes the sequences of tasks an displacement operations required to accomplish a WF-process; every stage in the plan is specified as place representing the tasks or the displacement operations; two transitions (input and output) represent the beginning/ending of a task; they are synchronized with *T1* and *T2* of the net $Net_{2,1}$. The transitions representing displacements must be synchronized with *T3*, also of $Net_{2,1}$. All the transitions must be externally↑ synchronized.

It is also necessary to add a place for the beginning of the *case* and another one for the ending of the *case* (as *p0* and *p18* respectively in fig. 6). Finally define $Net_{3,1}$ =(G, $TOKEN_{3,1}$,$LABEL_{3,1}$,$VAR_{3,1}$, τ, λ, χ, π); an example of this net is showed in Fig.6.

• *Involved tasks.* These nets describe in detail the operations involved in the execution of the tasks. In general they are nets having a simple state machine structure. We can obtain each net identifying the operations required for each task and its precedence order; with this information model each net adding also a transition for the external↑ synchronization of the beginning and another one for the ending of the task execution (see the labels *B.Claim_register*↑ in T0 and *E.Claim_register*↑ in T5 in Fig. 7a). For each task *i* define the nets $Net_{3,i}$ = (G, $TOKEN_{3,i}$, $LABEL_{3,i}$,$VAR_{3,i}$, τ, λ, χ, π), for *i=2,3,….r, r* is the number of tasks plus 1. Fig. 7 shows the models of two tasks.

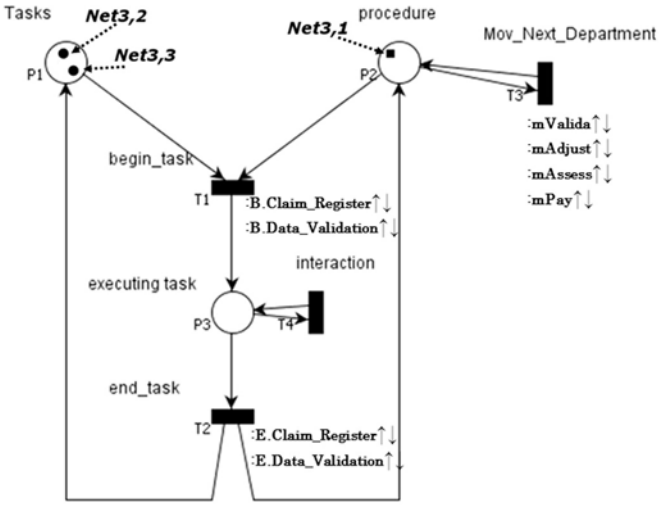

Fig. 5. $Net_{2,1}$ level 2 net for the case study.

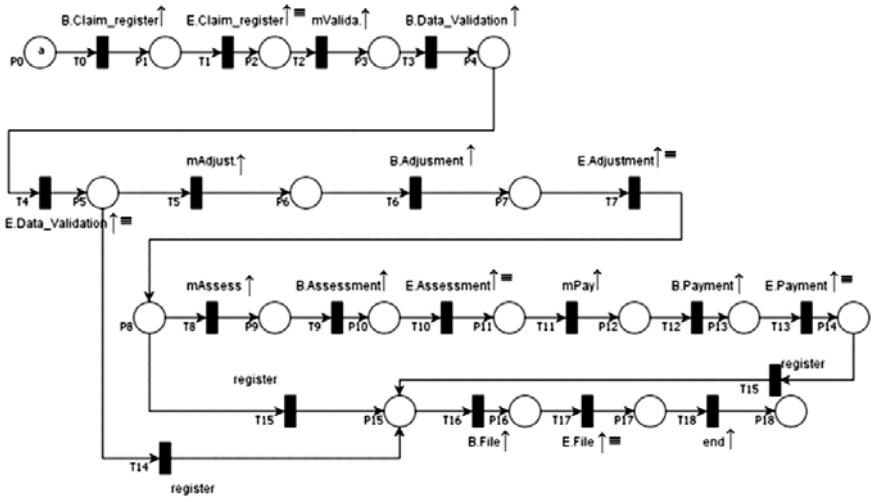

Fig. 6. $Net_{3,1}$ defining a simple *WF-Process*

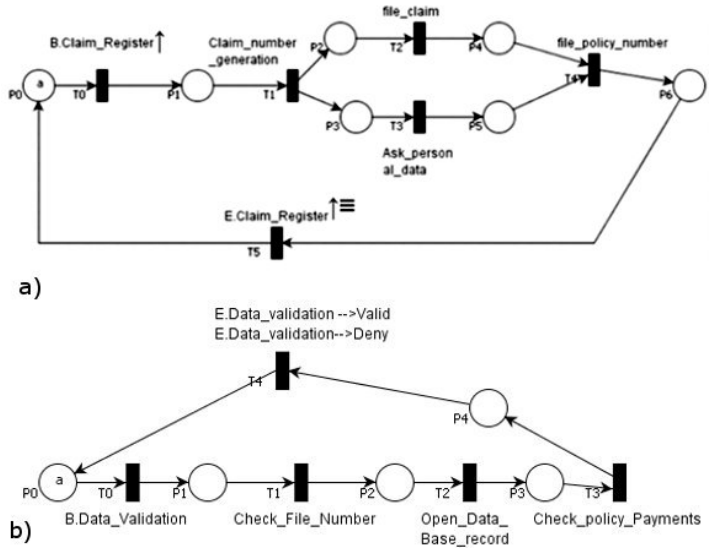

Fig. 7. Level 3 nets modeling two tasks: a) $Net_{3,2}$, task 1: *Claim_Register* task, b) $Net_{3,3}$, task 2: Data_Validation task.

4 Simulation of Workflow Processes

Although the model construction is easily performed by following the proposed modeling methodology, the size of models becomes large as the complexity and quantity of activities in the system grow. In the absence of analytical procedures for verifying the correct functioning of the resulting models, interactive simulation appears as a useful and suitable solution for validating the obtained models. This task is enlightened by the use of a simulation software tool.

In this section we describe the simulation procedure for n-LNS models and the tool for editing and executing such models.

4.1 Model Execution

The simulation of n-LNS models consist in the interactive execution of the models according to the rules for enabling and firing transitions. We are going to briefly describe a procedure, usually named *token player*, which implements such rules.

The n-LNS models, edited through a visual interface, are coded into a XML file according to the standard format Petri net markup language (PNML) [20]; such file is called *Data model*. Besides the PN structure, other information is stored in the Data model, namely the level of the net, transition labels and their attributes, weight

functions, marking, and Java code associated to transitions for software generation purposes.

From the Data model, a set of objects representing every Net_{ij} are built. Every object evolves as the corresponding Net_{ij} by verifying the transition enabling conditions and the transition firing mechanism. Both enabling and firing procedures involve interactions among Net_{ij}-objects for checking synchronizations declared on the labels.

4.2 n-LNS Simulation Tool

The simulation of the case of study has been performed through the execution of the 3-level net model described above. This task was possible with the help of MASGAS, a software tool that allows the visual edition and the interactive execution of multi level net models expressed in n-LNS.

The tool provides facilities for the interactive execution of the model: for a current marking the system indicates, through the highlighting of transitions, which transitions are enabled with respect to which label; then the user selects the transition to fire. After the firing the new markings of the involved nets are reached and displayed on the corresponding windows.

Based on n-LNS, MASGAS provides a complete syntactic control over the different hierarchical levels of the net system; this avoids nesting of upper level nets allowed by other net within nets simulators [21].

Below we are including several views of the edited model. Every net is built in a single window and it can be saved and updated for model adjustments.

In Fig. 4 it is showed the net which describes the company structure where the workflow tasks will be performed (net of level 1). Notice that the transition *t1* is enabled (label *begin*); when it is fired, the given *case* begins and the net that guides such *case* is generated (as the token in P1 also in Fig. 4).

In Fig. 5 the $Net_{2,1}$, described in section 3, is showed; the edited net shows the defined initial marking. At this situation the entity is ready to perform the initial task in the pertinent department ($Net_{1,1}$ in Fig. 4, where this net is contained). When transition *t1* of $Net_{2,1}$ is fired, it means that the net contained in its place *p2* (net $Net_{3,1}$ in Fig. 6) indicates that a particular task (one net as $Net_{3,2}$ in Fig. 7) need to be performed and will be selected (thanks to transitions synchronization) from the available tasks described by token-nets in place *p1*. The firing of *B.Claim_register* produces the evolution of this net indicated in Fig. 8.

Figure 6 shows the edited model for the *Workflow Process definition Net* $Net_{3,1}$; when transition *t0* is fired (*B.Claim_register* $\uparrow$ label), that indicates the beginning of the execution of the corresponding task (*Claim_register* task in Fig. 7a, in which *t0* of $Net_{3,2}$ is fired with respect also to the label *B.Claim_register* $\uparrow$; the firing is performed at the same time that *t0* from the net $Net_{3,1}$ of Fig. 6). The nets $Net_{3,1}$ and $Net_{3,2}$ are released (returned to the places *P1* and *P2* respectively) after the firing of the transition *T2* of $Net_{2,1}$. When the given task needs to be performed in a different department than the current one, the transition *Move_Next_Department* is used (Fig. 9, label *mValida* in T2), and indicates that the entity migrated to another department within the organization (as indicated in Fig. 10, label *mValida*).

Figure 11 shows the evolution of some nets of the system after several transitions firings.

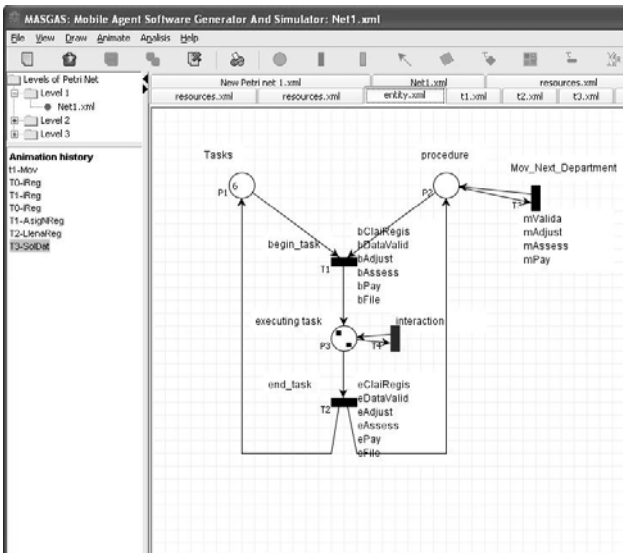

Fig. 8. Level 2 net (Mobile entity Net), after the firing of the transition *T1*: a token net of place *P1* and the token net of place *P2* are placed into *P3*

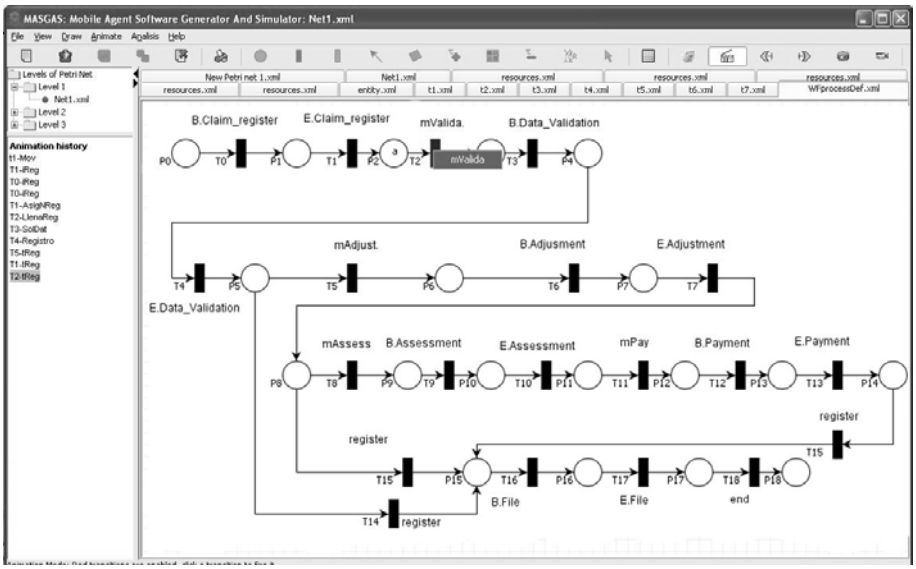

Fig. 9. $Net_{3,1}$; T2 (label *mValida)* is enabled: the entity need to migrate to the *Validation Department* (*P2* in $Net_{1,1}$).

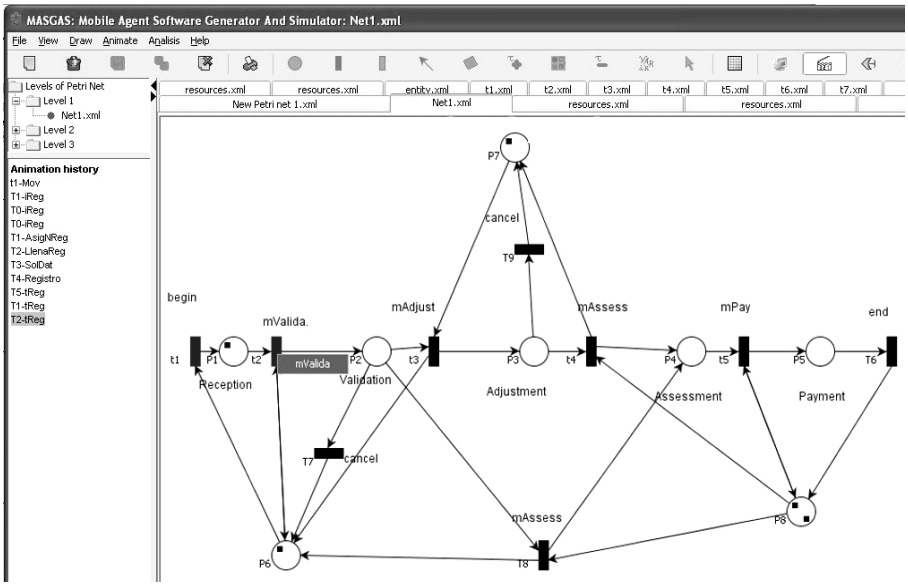

Fig. 10. $Net_{1,1}$ (Organizational Unit Net): t2 is enabled with respect to *mValida*

6 Conclusions

Modeling and simulation play an important role during the earliest stages of the developing life-cycle of workflow management systems. This work presented a modeling methodology for the specification of workflow processes. The use of n-LNS allows modular and hierarchical descriptions of the workflow management system. The models can be separately defined and then related through transition labeling; furthermore, the models may be easily updated because of the modularity and flexibility of the complete model. This feature allows addressing large systems exhibiting more complex behaviors than that obtained using "flat" PN.

Additionally, the model components lead to interactive software modules that implement a distributed agent-based management system. In fact, the net of level 2 describes the general behavior of a mobile agent that evolves through the computer network, and executes the tasks specified in the WF-process net.

Current research addresses the problem of fault tolerance for inter-organizational workflow management in which mobile agents move through internet.

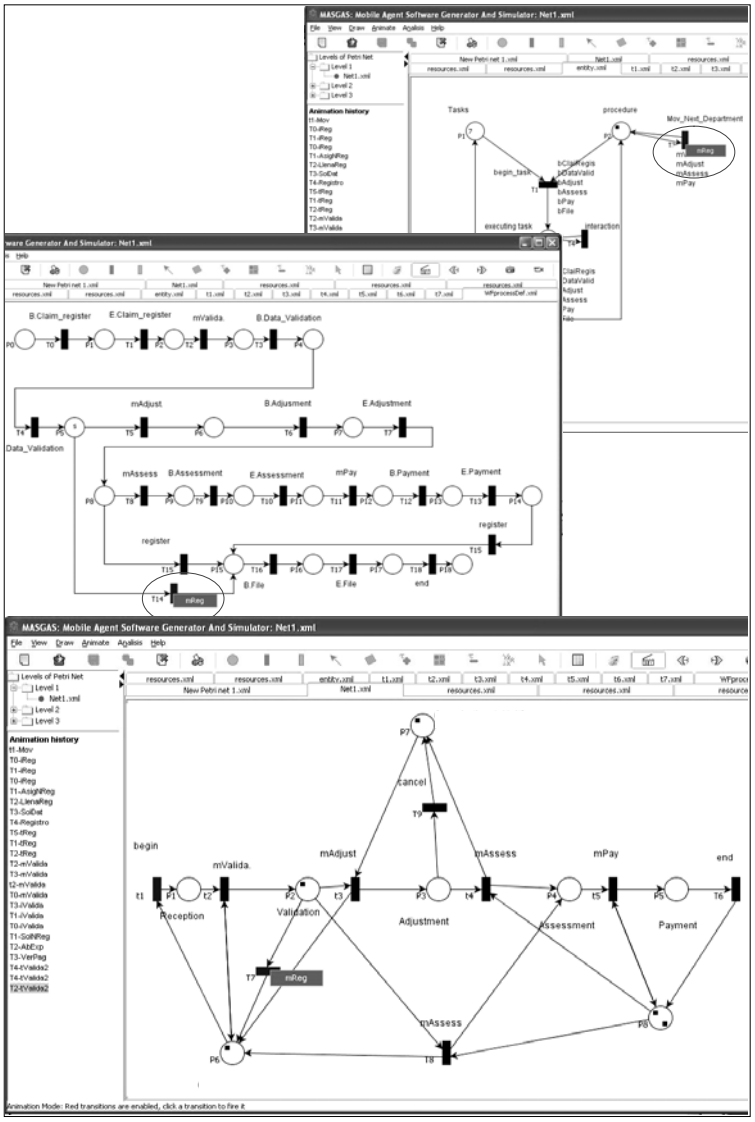

Fig. 11. Marking on several nets after the firing of a transitions sequence

References

1. Shi, M.L., Yang, G., Xiang, Y., Wu, S.: Workflow Management Systems: A Survey. Proc. IEEE International Conference on Communication Technology, ICCT'98, S33.05.01-s33.05.06, Beijing, China (1998).

2. van der Aalst, W.M.P., Hee, K.: Workflow Management: Models, Methods and Systems. London, MIT Press. (2002).
3. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers v8 i1. 21-66 (1998).
4. Workflow Management Coalition: Workflow Management Coalition – Terminology & Glossary. Technical Report. Document Number WFMC-TC-1011 (1999).
5. Hollingsworth, D.: Workflow Reference Model. Technical Report. The Workflow Management Coalition, Document Number WFMC-TC-1003 (1995)
6. Rusinkiewicz, M., Sheth, A.: Specification and Execution of Transactional Workflows. In W. Kim, editor, Modern Database Systems: The Object Model, Interoperability and Beyond, pp. 592-620, ACM Press, New York, NY (1995).
7. Reijers, H.A., van der Aalst, W.M.P.: The Effectiveness of Workflow Management Systems: Predictions and Lessons Learned. International Journal of Information Management v25 i5. 458-472 (2005)
8. Lu, R., Sadiq, S.: A Survey of Comparative Business Process Modeling Approaches. BIS 2007, 82-94 (2007)
9. Zisman, M.D.: Representation, Specification and Automation of Office Procedures. PhD Thesis. Wharton School of Business, University of Pennsylvania (1977).
10. van der Aalst, W.M.P.: Three Good reasons for Using a Petri-net-based Workflow Management System. In Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), Cambridge, Massachusetts (1996).
11. Salimifard, K., Wright, M.: Petri Net-based Modeling of Workflow Systems: An Overview. European Journal of Operational Research, Vol. 134, pp. 664-676 (2001).
12. van der Aalst, W.M.P., Anyanwu, K.: Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. In Proceedings of the Second International Conference on Telecommunications and Electronic Commerce (ICTEC'99), pp. 141-157, Nashville, Tennessee (1999).
13. Adam, N.R., Atluri, V., Huang, W.K.: Modeling and Analysis of Workflows Using Petri Nets. Journal of Intelligent Information Systems. Volume 10, Issue 2, pp. 131-158 (1998).
14. Li, J.Q., Fan, Y.S., Zhou, M.C.: Performance Modeling and Analysis of Workflow. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, Vol. 34, No. 2 (2004).
15. Eshuis, R., Dehnert, J.: Reactive Petri Nets for Workflow Modeling. In W.M.P. van der Aalst and E. Best, editors, Applications and Theory of Petri Nets 2003, volume 2679 of Lecture Notes in Computer Science, pages 295-314, Springer Verlag, Berlin (2003).
16. van der Aalst, W.M.P., Hofstede, A.: YAWL: Yet Another Workflow Language. Information Systems. V30 i4. 245-275. (2005)
17. Sánchez-Herrera, R., López-Mellado, E.: Modular and Hierarchical Modeling of Interactive Mobile Agents. IEEE International Conference on Systems, Man and Cybernetics, Vol. 2, 10-13. pp. 1740-1745 vol.2. (2004)
18. Flores-Badillo, M.: Mobile Agent Systems Based Workflow Coordination. MSc. Thesis, Cinvestav Guadalajara Mexico. August (2006).
19. Valk, R.: Petri Nets as Token Objets And introduction to Elementary Object Nets. Int. Conf. on application and theory of Petri nets, pp1-25, Springer-Verlag (1998)
20. Weber, M., Kindler, E.: The Petri Net Markup Language. Lecture Notes in Computer Science, Springer Berlin, Volume 2472/2003, pp 124-144 (2003).
21. Kummer, O., Wienberg, F., Duvingneau M.: Renew - user guide Documentation. University of Hamburg, Department for Informatics, Theoretical Foundations Group, release 2.1, (2006).