

Multilevel Petri Nets for Modeling and Simulating Organizational Dynamic Behaviour

Aphrodite Tsalgatidou
Panos Louridas
George Fesakis
Thanasis Schizas
University of Athens

The modeling and simulation of the dynamic behavior of organizational systems is essential for understanding their current functionality and for supporting business process analysis and reengineering. The behavior of organizational systems is viewed here in terms of the behavior of their business processes. Multi-level Modified Petri nets are used as the underlying formalism for modeling and simulating organizational systems behavior at various abstraction levels. Each level of abstraction corresponds to one or more levels of the involved part of the organizational hierarchy. The constructed models represent existing or redesigned business processes. Simulation and graphical animation are used for discovering problems in existing business processes as well as in the proposed solutions. The whole approach is supported by a development environment called ERMIS.

KEYWORDS: *animation; business improvement; business process modeling; dynamic modeling; Multi-level Petri Nets; organizational modeling; reengineering; simulation.*

The mission of an organization is the achievement of its business objectives. These objectives are achieved by the execution of the organization's business processes. Hence, modeling and simulation of the dynamic behavior of an organization's business processes is essential for understanding the current functionality of the organization, for discovering existing problems and for proposing and evaluating alternative solutions. These issues are especially important in the context of new process-centered management theories such as Business Process Reengineering (see [Davenport, 1993], [Hammer and Champy, 1993] for an introduction).

A business process (BP) is a set of internal *activities* which are the basic BP building elements and are executed in order to achieve a business objective which is usually to offer the right product or service to a customer with a high degree of performance measured against cost, longevity, service and quality [Jacobson, 1995]. According to [Tsalgatidou & Junginger, 1995] a BP model should encapsulate information related to: (a) *activities*, (b) *resources* assigned to activities, i.e. objects necessary for the execution of activities, like actors, documents, data and so on, (c) *control* of a BP which describes 'when' and 'which' activity will be executed, (d) the *flow* of data in the process and (e) the *organizational structure* which consists of organizational units, people, roles, competence and so on (see also [Starke, 1994] for a more detailed description of the characteristics of BP models). Consequently, BP modeling approaches should enable the modeling of all these types of information while at the same time providing facilities for tracing, simulating and graphical animating the constructed BP models.

Traditional BP modeling approaches either do not pay any attention to the organization's architecture and dynamics or, if they do, they do not provide any means for simulation/ animated evaluation of the constructed models. Moreover, they suffer from increased complexity,

presenting too coarse or too fine detail to people at different levels of the organization. See also [Tsalgatidou & Junginger, 1995] for a discussion of BP modeling approaches.

The approach presented here aims to overcome these problems by accommodating the dynamics of an organization and by deriving executable models amenable for formal as well as graphical dynamic analysis and validation while working at different levels of detail to accommodate the complexity problem. Along these lines, the rest of the paper is organized as follows: the following section presents an overview of the proposed approach for modeling and simulating organizational dynamic behavior within a business improvement framework which aims to identify existing problems and to propose alternative solutions. This is followed by a description of the underlying modeling formalism which is a Multi-level Modified Petri Net (MPN) based on Petri Nets [Murata, 1989]. The process for deriving MPN models is then presented by using an example BP of a telecommunication organization. The formal and executable nature of MPN models enables the employment of formal algorithms as well as simulation, graphical animation and statistical analysis techniques for validation purposes. The whole BP modeling and validation process is supported by a software development environment called ERMIS⁽¹⁾, the functionality of which is demonstrated in the sections related to the derivation of MPN models and their validation. The advantages of this approach are then discussed and compared with related work and finally some conclusions are drawn.

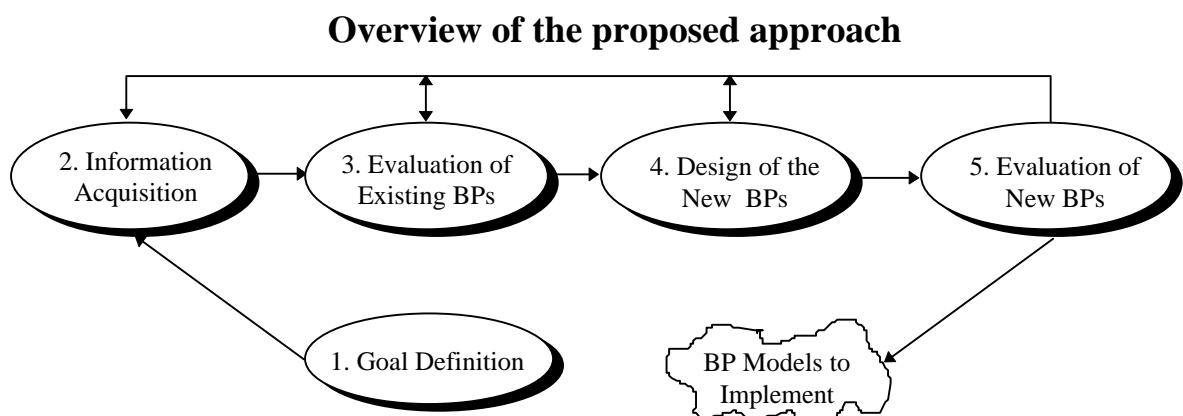


Figure 1. Steps for developing a new BP model

Business process modeling is viewed here as a process taking place in a larger context of a business improvement effort, whereby increased efficiency of some BPs is sought. Figure 1 depicts the steps of the proposed approach, the final result of which is a set of evaluated new BP models which overcome the deficiencies of existing BPs. The derivation of the new BP models is an iterative process which is terminated when the models satisfy the improvement goals. This view of BP modeling is along the lines of the Business Process Management Systems (BPMS)

⁽¹⁾ ERMIS is the Greek God of commerce and of quick and efficient communication. He was quick-witted to always find the right solution at the right time by exploiting the current situation.

approach [Karagiannis, 1995]. Thus, the first step is *Goal Definition* where the goals of the improvement effort are defined (e.g. selection of the BP(s) to be improved, definition of improvement goals, metrics for assessing the improvement result and so on). This is followed by the *Information Acquisition* step where information pertinent to the BPs under study is acquired from all the people involved in these BPs (managers, lower level employees, etc.) and the BP models are gradually constructed. BP modeling is carried out using MPNs which demonstrate the flow of control and data between the involved organizational units, as well as between the various activities and the external environment.

The following step is the *Evaluation of Existing BPs* where the constructed BP models are evaluated in order to reveal the current deficiencies. If some information is missing, the previous step of information acquisition is re-executed until the constructed BP models reflect the current BP situation. The decomposable nature of the underlying MPN formalism provides the means for handling the BP model complexity. Furthermore, the executable and formal nature of MPNs allows for the employment of formal algorithms and simulation, graphical animation and statistical analysis techniques which facilitate the validation of the BP model and the identification of problems in the current situation.

The next step, *Design of New BPs*, aims at designing new BP models which satisfy the improvement goals defined in the beginning and remedy the problems identified in the previous step. Depending on the results and exigencies of this step, return to the information acquisition or evaluation of existing BPs step may be necessary. *Evaluation of new BPs* is the last step of the improvement process, during which the designed new BP models are evaluated (using again simulation, animation or statistical analysis techniques) against the improvement goals. Again, depending on the evaluation results, the previous steps (e.g. information acquisition if some information is missing, evaluation of the BP models, design of other solutions and so on) may be re-executed until the improvement goals are reached. The result of using the proposed approach is a set of Multi-level MPNs which model the selected BPs (existing and proposed ones) at various abstraction levels and which are described in the following section. It is worth to note here that all these steps are carried out by the BP engineer with the help of the various involved people.

Multi-Level Modified Petri Net Models

Multi-Level Modified Petri Nets (MPNs) provide constructs for accommodating all kinds of BP related information enumerated in the introduction. Thus, BP *activities* are modeled as MPN transitions. BP *control* information, *resources* and *actors* required for the execution of activities are modeled as objects inscribing the MPN places. More specifically, control objects are either signals (representing messages among activities) or events (representing occurrences of incidents) and allow for the representation of control flow in the process. Resource objects are data objects used by the process, such as invoices, etc. and allow for the representation of data

flow in the process. An actor represents a set of duties and responsibilities in the organization or an external participant. Actors are required in the input places of a given transition where the presence of specific human participants is essential for its enactment.

MPNs model BPs at various levels of abstraction. Each abstraction level corresponds to the activities of the organizational units of one or more levels of the organizational hierarchy involved in the modeled processes. Decomposition of MPNs is based on the decomposition of transitions. Transitions that are decomposed are called *compound transitions*, in contrast to *primitive transitions* that do not require decomposition. The highest abstraction level of a MPN (called the top level MPN) contains just one transition which represents the behavior of the organizational unit which is at the highest level of the part of the organizational hierarchy involved in the process under study. Places represent control information and data coming from or going to the BP's external environment. As external environment we consider everything (e.g. a person, another BP, another organizational unit, another organization, etc.) which is external to the BP and it triggers some activities (by providing some information) or receives some output generated by the execution of the activities of the BP.

The top level MPN transition is decomposed into a number of other transitions and places. Each resulting transition may be further decomposed into a number of other transitions and places of lower level MPNs, again representing respectively activities and the information exchanged between them. Sub-activities of a given activity may be performed by organizational units of the same level as those performing the parent activity and/or lower levels organizational units if they are carried out by more specific organizational entities.

By carrying out a series of decompositions, we arrive at a set of MPNs of varying abstraction levels well suited to handle the complexity problems of the BP analysis/ design process: early analysis phases are carried out using high level MPNs, whereas in later phases of the development process lower level MPNs are employed. The mapping between different MPN levels is both formal and direct, ensuring that work on different levels of detail can be carried out unobtrusively.

Formal Definition of Multi-level Modified Petri Nets (MPN)

A Modified Petri Net *MPN* is defined as a tuple

$$MPN = \langle P, T, F, N, \Sigma, script, struct, SubMPNs, M_0 \rangle$$

where:

- P is a set of places.
- T is a set of transitions.
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs connecting places to transitions and transitions to places
- N is a function that maps each arc of F to an integer number denoting the multiplicity of this arc, i.e., the number of tokens transferred across this arc each time the transition at the one end of the arc fires.
- Σ is the underlying structure of an MPN and has the form $\Sigma = \langle Dobj, OrgStr \rangle$.

- *Dobj* is the object domain of the MPN; it is a non-empty set of object types, i.e., a non-empty set of tuples of the form $\langle \text{ObjectName}, \text{SuperObjectName}, \text{Properties} \rangle$. *Properties* is a set of attributes pertaining to the object. A common property owned by all object types is the *Timestamp* property containing its creation time. This renders MPNs capable for temporal modeling; specifically, a transition may introduce delays by appropriate handling of the *Timestamp* property of its output (created by the transition) tokens. This allows for several kinds of statistical analysis of temporal aspects of MPNs (see the section on Model Validation). An object may be either a control object, or a resource object: the two object types differ only in what regards their properties and not in their treatment as tokens in MPNs. For example, resource objects contain a *Permissions* property which is a list of permissions (create, read, modify, delete) associated with actor classes, conveying the notion that specific actor classes are entitled to particular actions on the resource object at hand.
- *OrgStr* represents the part of the structure of the organization that is of interest to us. It has the form: $\text{OrgStr} = \langle \text{OrgEntityStr}, \text{RoleStr}, \text{ActorStr}, \text{EmployeeStr} \rangle$.
 - *OrgEntityStr* is the organization entities hierarchy, i.e. a non-empty set of tuples of the form $\langle \text{OrganisationalEntityName}, \text{SuperEntityName} \rangle$.
 - *RoleStr* is the roles structure, whereby a role we designate a specific set of activities; *RoleStr* is constituted by a non-empty set of tuples of the form $\langle \text{RoleName}, \text{RoleNames}, \text{Activities} \rangle$, with $\text{Activities} \subseteq T$. Conceptually, a role is a related set of roles and activities that are carried out by an actor.
 - *ActorStr* is the actor class hierarchy, i.e. a non-empty set of tuples of the form $\langle \text{ActorName}, \text{SuperActorName}, \text{Roles}, \text{Employees} \rangle$. Conceptually, an actor is responsible for enacting a set of roles. An actor may be external, such as a client, or internal. Since several distinct employees may be apt for performing the same set of roles, several employees may be associated with one internal actor class. External actors are not associated with employees, hence their *Employees* property is empty.
 - *EmployeeStr* is a subset of the organization employees, i.e. a non empty set of tuples of the form $\langle \text{EmployeeName}, \text{OrganisationEntityName}, \text{Diary} \rangle$. An employee has an associated diary with the activities he/she has to perform (which depend on his/her association with actor classes).

A meta-model of *OrgStr* using an Entity-Relationship scheme is shown in figure 2 (for simplicity, hierarchy relationships are omitted from the figure).

- *script* is a mapping from transitions to scripts where a script is the set of steps to be carried out during the execution of a given activity.
- $\text{struct} : P \rightarrow \text{Dobj} \cup \text{ActorStr}$ is a function that maps places to the underlying structure of the MPN, and specifically to object types and actor classes.

- *SubMPNs* is a set of MPNs (which may be empty). For each compound transition in MPN there is a member in *SubMPNs*. The input and output places of compound transitions are shared between MPN and the corresponding member of *SubMPNs*. Specifically, a decomposition mapping from T to $S \in \text{SubMPNs}$ is as follows: $S = \langle P_s, T_s, F_s, N_s, \Sigma_s, \text{script}_s, \text{struct}_s, \text{SubMPNs}_s, M_{s0} \rangle$ is a Modified Petri Net with the following properties: $\forall (p, t) \in F \exists (p, t') \in F_s$ and $N_s(p, t') = N(p, t)$, and $\forall (t, p) \in F \exists (t', p) \in F_s$ and $N_s(t', p) = N(t, p)$. In both cases, p is called an interface place. Moreover, script_s is a mapping from T_s to scripts and $\bigcup_{t' \in T_s} \text{script}_s(t') \equiv \text{script}(t)$, that is, the behavior described by the transitions of S is the behavior described by the compound transition t . Finally, $\Sigma_s = \Sigma$. It is important to note that since S is an MPN it has its own *SubMPNs* set allowing for Multi-level decomposition hierarchies.
- M_0 is the initial marking of the Petri Net.

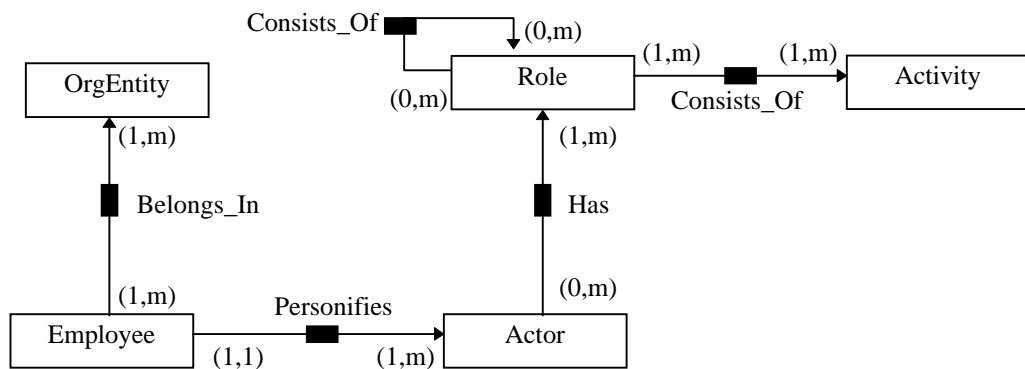


Figure 2. The Organization Structure Meta-Model.

Derivation of MPN Models Using ERMIS

In this section we shall provide some of the MPN models derived from a real-world example of a business process concerning the Fault Repair process in the Greek Telecommunications Company (OTE) described in the appendix. The models have been derived using the ERMIS development environment. The figures shown are screen dumps taken from real use instances of ERMIS.

In the first step of the modeling process a top-level MPN is derived; figure 3 shows the top-level MPN for the OTE example.

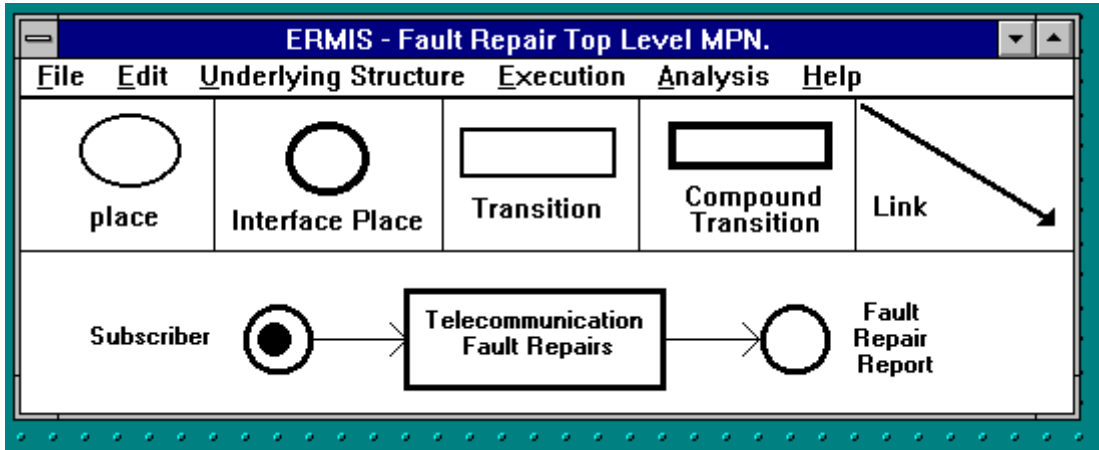


Figure 3. The top-level MPN in the OTE example.

The Telecommunication Fault Repairs compound transition requires decomposition in order to be of use; an MPN resulting from a first decomposition is shown in figure 4.

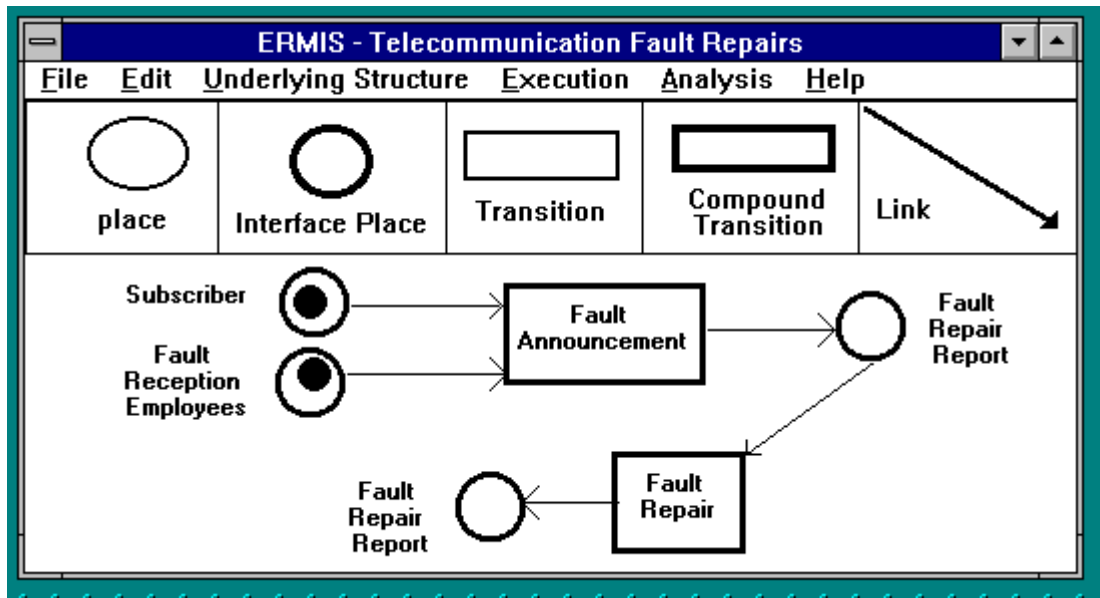


Figure 4. Decomposition of the Telecommunication Fault Repairs compound transition.

The Fault Announcement and Fault Repair compound transitions are further decomposed in the MPNs shown in figures 5, 6 respectively..

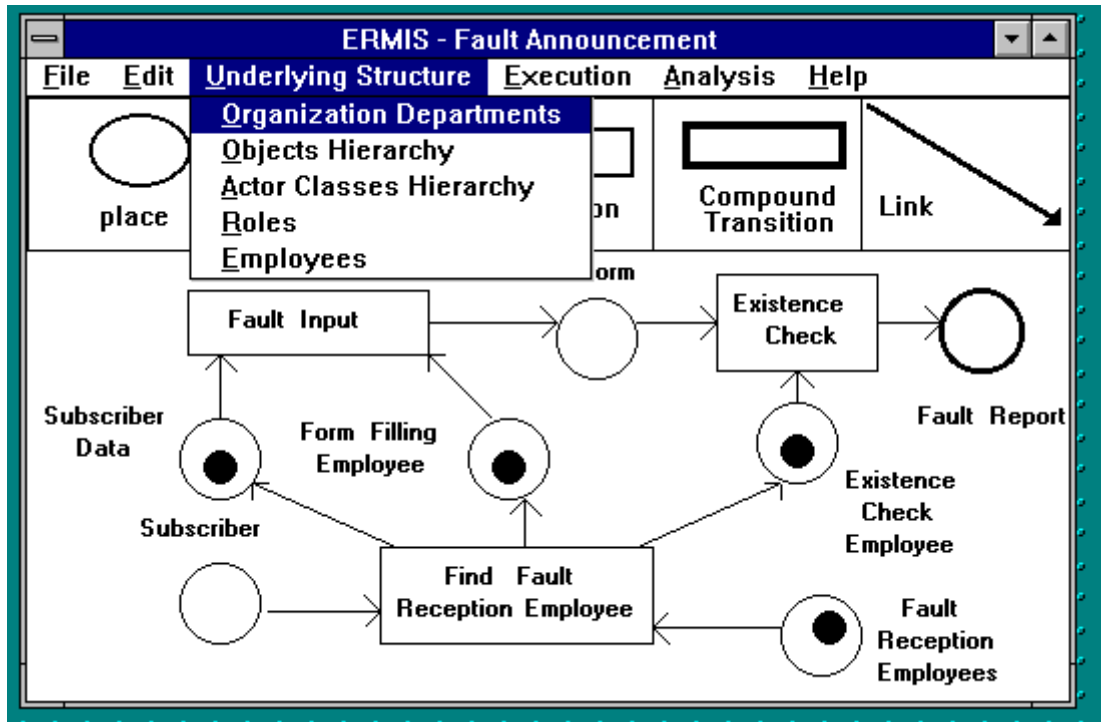


Figure 5. Analysis of the Fault Announcement compound transition

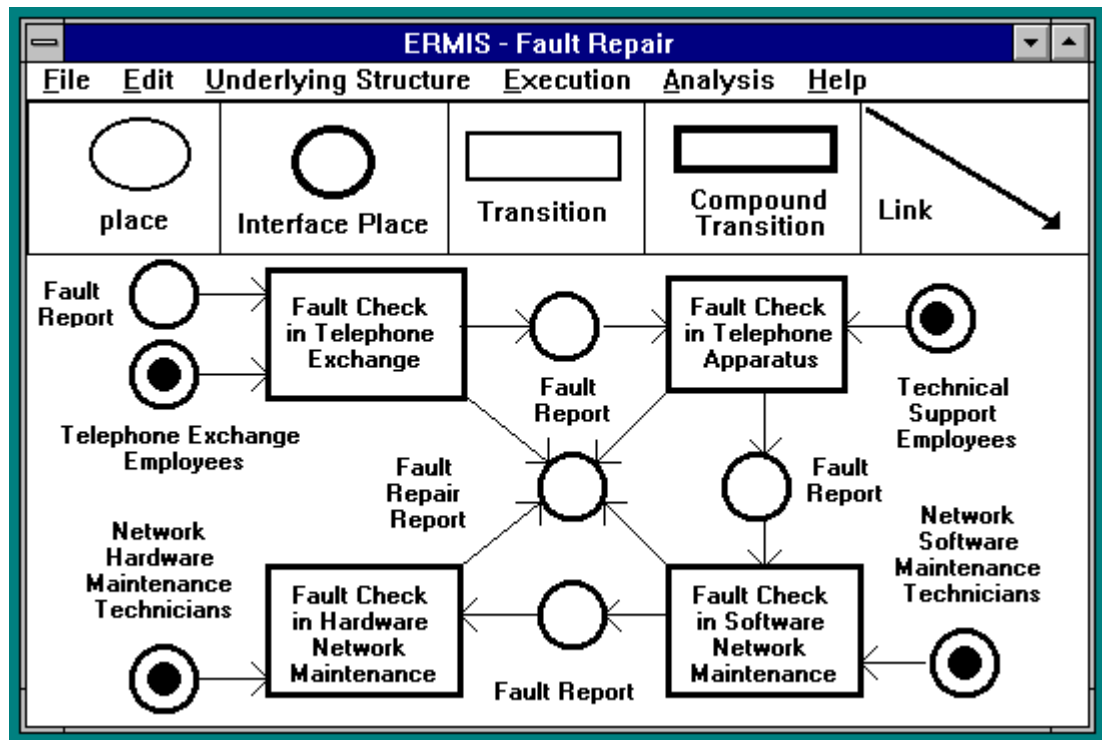


Figure 6. Analysis of the Fault Repair compound transition.

In figure 5, all transitions are primitive since they represent activities in sufficient level of detail. In figure 6, all transitions require further decomposition. For economy of space, not all lower level MPNs are shown here; the MPN referring to the Fault Check in Telephone Exchange compound transition is given in figure 11 in the Model Validation section.

The process of MPN derivation includes the modeling of the organization hierarchy (shown in figure 7) and the creation of the actor class hierarchy (part of which is shown in figure 8).

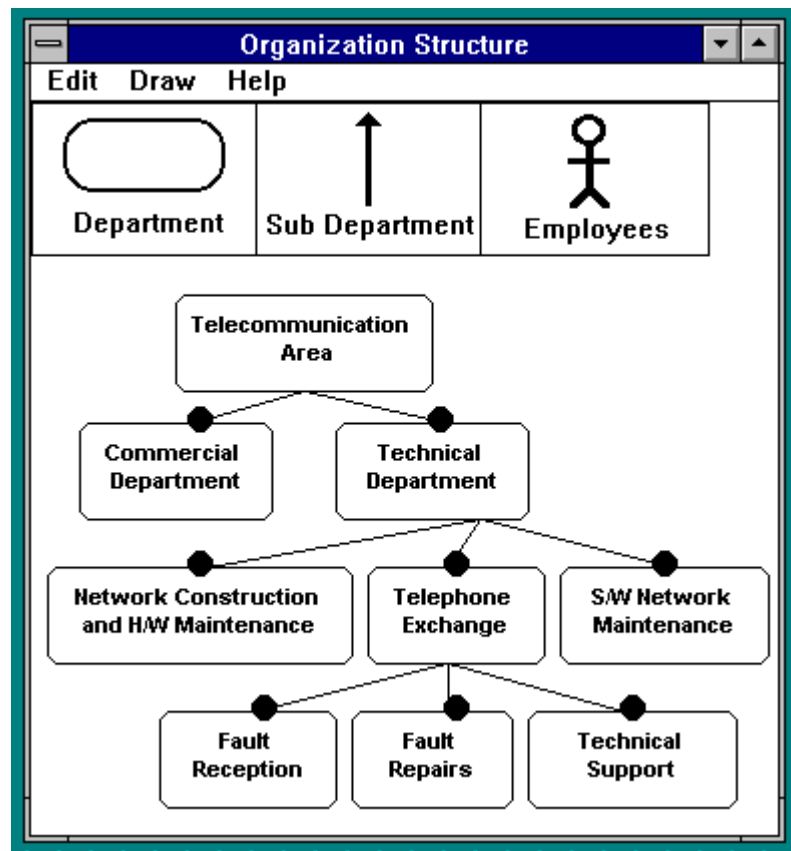


Figure 7. The OTE departments concerning the Fault Repair process.

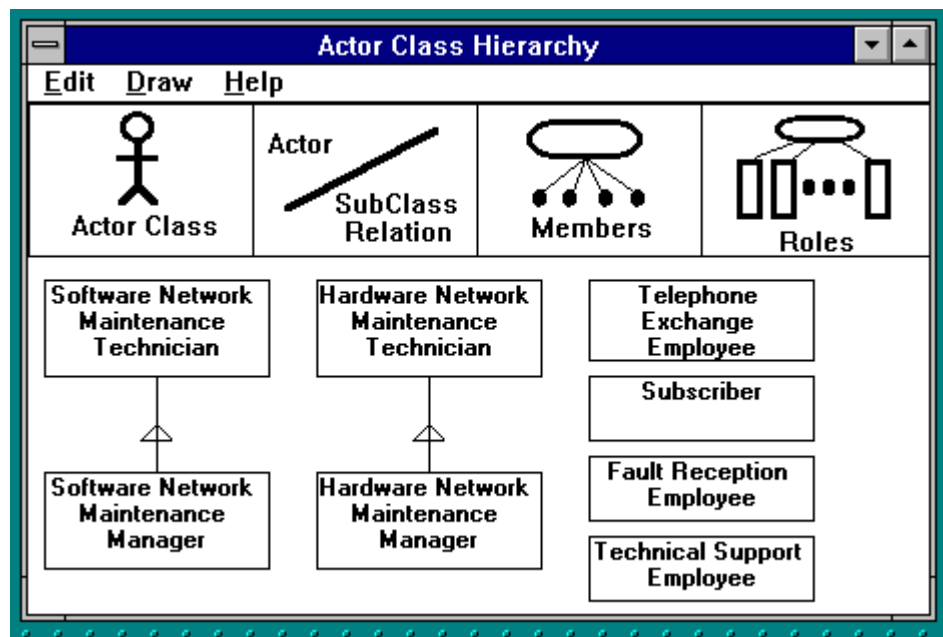


Figure 8. The actors hierarchy of Fault Repair process.

Moreover, roles are defined and ascribed to transitions as shown in figure 9 (corresponding to the MPN model of figure 11). Actors are associated to roles using the facilities provided by ERMIS (as shown in figure 8).

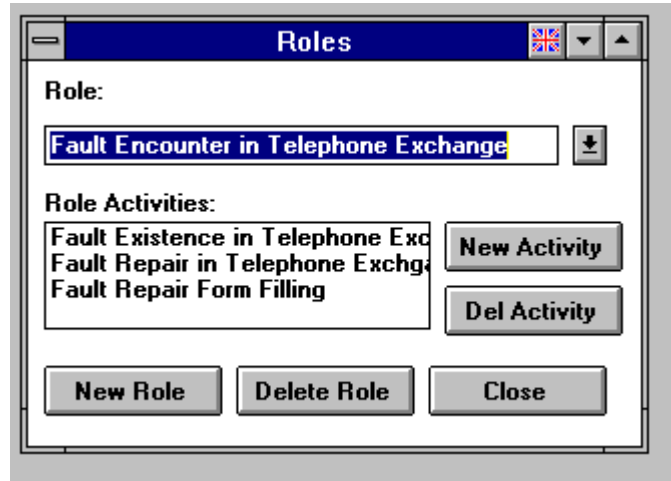


Figure 9. The role definition form

Also, the resource and control objects hierarchy is developed; figure 10 shows the resource object hierarchy in ERMIS for the OTE example.

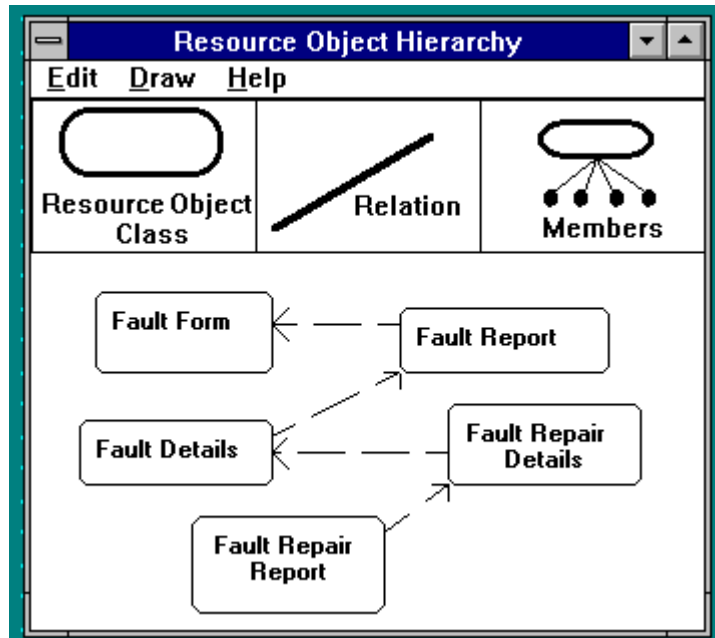


Figure 10. The resource object hierarchy.

During decomposition, a special case, referring to connections between actors and transitions deserves separate mention: a place containing internal actors is connected to an actor selection transition, which describes the selection method for the actors (transition Find Fault Reception Employee in figure 5). The distributed selection of appropriate actors is an advantage of this

formulation. In essence, during execution (see the next section), when an actor selection transition is firing, it finds actors to enable the actor claiming activity. In this case, ERMIS updates the *Diary* property of the selected actor.

Model Validation

The process of deriving a model is an activity that involves a great deal of interaction between the model developers and the people who are the experts of the real world situation. The derived model should be validated in order to ensure its consistency, completeness and accuracy and in order to ensure that it reflects the real world situation and satisfies the users' needs. We distinguish two kinds of validation of a derived model: static validation and dynamic validation.

Static Validation

Static validation refers to the study of the derived models using specific algorithms and analysis approaches (not simulation). In the case of ERMIS, MPNs comprise the underlying formalism. The firing rule of MPNs is the same as in the case of conventional Petri Nets; therefore the relevant research that has been carried out for generic Petri Nets applies equally well (this was one of the requirements of our approach). Specifically, a static analysis of the derived MPN can examine such aspects as its behavioral (marking dependent, i.e. reachability, boundedness, liveness, persistence, synchronic distance, fairness etc.) and structural (marking independent, i.e. structural liveness, structural boundedness, repetitiveness, consistency, etc.) properties. Analysis of such MPN properties, can be implemented using general techniques such as coverability tree production or matrix-equation approach [Murata, 1989] and is supported by ERMIS.

Dynamic Validation

Dynamic validation refers to the study of the derived models by way of their dynamic behavior. Simulation of the model specification is one approach for dynamic validation. When simulation is combined with graphical animation, the dynamic behavior of the modeled system or process is clearly demonstrated and validation is greatly facilitated. Moreover, simulation, combined with graphical animation, supports and simplifies the communication between the BP engineers and the people involved in the actual BP. Otherwise, it is very difficult to notice and localize errors that cause unexpected or undesirable behavior by just looking at the simulation code or static numerical output [Verbraeck, 1993].

In the case of ERMIS, simulation is based on the formal definition of MPNs, whereas animation is based on their graphical representation. Places may contain resource, control or signal objects or actors, whose flow in the process is animated. Furthermore, each place computes several statistics such as the minimum, maximum, and average number of tokens and the minimum, maximum and average time between token arrivals. Transitions hold statistics

about the number of firings, the minimum, maximum and average firing time, the time between successive firings, etc. Since all tokens are time-stamped, the temporal aspects of the process may be studied.

The simulation/ animation may be carried out automatically, by giving an initial marking and observing the evolution of the process in the MPN at hand; the result is a visually pleasing animation sequence. Alternatively, simulation/ animation may also be user-driven: the user may specify breakpoints where simulation should be halted by giving conditions on the contents of places or the state of transitions. Simulation may also be halted after a certain number of transition firings, or after a certain time limit. In any case, if the set of enabled transitions is empty the simulation will stop. In a halted/ stopped simulation the user can examine places contents, modify certain tokens and restart the simulation; details for specific transitions may also be accessed and/or modified. Figure 11 shows a snapshot from a simulation sequence which started with the marking shown in figure 5, followed by the firing of transitions Fault Input, Existence Check and Fault Existence in Telephone Exchange.

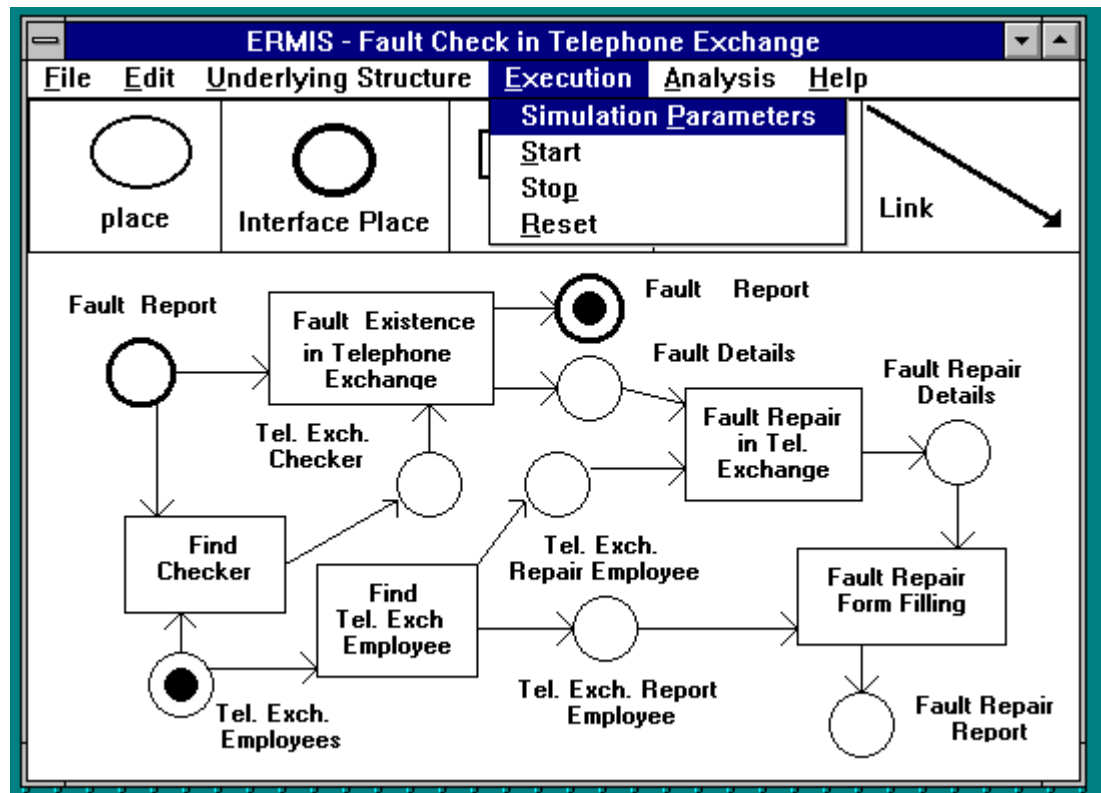


Figure 11. A snapshot from a simulation sequence in the OTE example

In figures 12, 13 the place (respectively transition) form, whereby the user may examine and/or modify place (transition) contents is shown.

Place

Place Name: Token Class: Tokens:

Number Of Tokens:

Current: Minimum: Mean: Maximum:

Time Between Token Arrivals:

Minimum: Mean: Maximum:

Actor Place Interface Place

Figure 12. The place form.

Transition

Transition Name: Compound

Number Of Firings:

Time Between Firings:

Minimum: Mean: Maximum:

Firing Duration:

Minimum: Mean: Maximum:

Figure 13. The transition form.

The simulation/ animation capabilities of ERMIS can be used to discover bottlenecks, deadlocks and critical paths and to detect errors in the proposed solution before proceeding to its implementation. The user can analyze several “what-if” scenarios, define markings and observe the results (step 3 in figure 1). For example, the time response of an activity can be measured against the addition or deletion of a certain number of corresponding actors, while at the same time observing the side-effects on the other activities. In the OTE example, the techniques described above may be used for validating the modeled information and for assessing the quality of the offered services. They can also be used to estimate the mean execution time of the process, the number of faults which have been reported, repaired, which are still on hold and so on, every certain period of time (one hour, two hours, 24 hours and so on). Taking into account the results of simulation and of the statistical analysis as well as the cost of employment of the

existing and new staff, the missing revenues lost due to delayed repair of faults and other factors related to the offered service, the BP may need to be redesigned in order to be improved. Redesign of the BP (step 4 in figure 1) may result in omitting some unnecessary tasks, recruiting new staff, and so on, depending on the type of the detected problems and depending on the results of the cost/benefit analysis.

Before proceeding to the actual implementation of the redesigned process, animation and simulation can be used again to help in evaluating the viability of the proposed solution and in ensuring that this satisfies the reengineering goals (step 5 in figure 1). It is worth to note here that, the people involved in the actual BP, participate in the simulation and graphical animation for the evaluation of the proposed solution. The multi-level nature of MPNs enables the participation of the involved people, only in the evaluation of the MPNs at the level of detail which is within their knowledge and interest and for which they are authorized to comment or take decisions upon the proposed changes. Implementation of the proposed solution can only start if the evaluation results are the expected ones. In any other case, the solution has to be redesigned by making new changes or by collecting more information about the process, analyzing it, redesigning and so on. This means that the modeling steps depicted in figure 1 are repetitively executed until a satisfactory solution is proposed.

Advantages of the Approach & Related Work

The proposed approach views the dynamic behavior of an organization in terms of the dynamic behavior of its BPs. Multi-level Modified Petri Nets are used for the modeling of BPs at various levels of abstraction. The way transitions are decomposed depicts the decomposition of a BP to its sub-processes, activities, sub-activities, tasks and sub-tasks and demonstrates the control and data flow between the different organizational units involved in the BP. Furthermore, the formal decomposition semantics ensure traceability between low level specifications and higher level concerns. Also, the formal, executable nature of the Petri Net model, combined with the modeling of the BP at different abstraction levels, facilitates validation, as the BP engineer may on the one hand employ formal validation algorithms and on the other demonstrate to each individual (involved in the BP) the part of the model which is within his/her own interests and which can be evaluated by him/her.

Simulation and graphical animation are used for analyzing the existing situation as well as for evaluating alternative solutions. Thus, people responsible for taking decisions about changing parts of the BP inspect MPNs at the higher abstraction levels, while people executing the everyday activities check only the part of the model which concerns their work. Furthermore, the type of information demonstrated to each user is different. Executive managers, for example, formulate the business objectives. In order to do so, they need an overview of existing BPs; they also need information like costs, quality of the offered services, execution time and so on. This

type of information does not concern employees at the lower levels of the organizational hierarchy, therefore, information related to their tasks is the only information which is presented to and evaluated by lower level employees.

The formality of the MPNs may be considered as a disadvantage of the approach, which may cause some problems until the BP analysts and designers are used to their use. However, our view is that the tools offered by the ERMIS environment for the construction of the model, its analysis and validation can overcome this disadvantage.

Numerous approaches have been proposed in the literature for the modeling of dynamic properties and behavior of systems. In this section we shall concern ourselves with the ones most relevant to our own approach, establishing the main differences and attempting a brief comparison.

Two schemes that have been developed for modeling the dynamic properties of information systems using variations of Colored Petri Nets (CPNs) are LOOPN [Lakos & Keen, 1993] and LOOPN++ [Keen & Lakos, 1994]. Both adopt an object-oriented philosophy, modeling CPN elements as objects, although some important differences exist. LOOPN uses two object hierarchies, one for token types and one for subnets. Modularization of CPNs is supported via subnets and the usual desirable properties of object-orientation, polymorphism, inheritance and encapsulation add to the model's usability. LOOPN++ extends LOOPN by collapsing the two class hierarchies into one and introducing a more integrated object-oriented language for CPN modeling. Unfortunately, both models fail to take into account important factors in organization modeling such as organization structure and activities enactment by actors; they are oriented towards information systems and not BP modeling. The two approaches are comparable to RBNs [Tsalgatidou et al., 1994] which are considered later in this section.

Design/CPN [Pinci & Shapiro, 1990] is a commercial tool for systems development based on Hierarchical Colored Petri Nets (HCPNs). The tool allows graphical specification of the envisaged system and offers graphical simulation and statistical analysis capabilities; moreover, the resulting models may be automatically translated to executable code, and SADT and HOOD specifications may be imported and automatically translated to HCPN models. Leveling and modularization of models is supported by the underlying formalism, HCPN by substitution transitions (where a transition is replaced by a detailed description) and fusion places (where multiple places are conceptually unified into a single place). Design/CPN is a mature software product supporting cradle-to-grave systems development; however, it suffers from the same defects mentioned above in the context of LOOPN/++. Specifically, its support for organization process modeling is wanting due to the lack of any representation of the organization structure and actor participation. Moreover, it does not offer the usual advantages associated with more object-oriented modeling approaches.

STATEMATE [Harel et al., 1990] is a "working environment for the development of complex reactive systems". STATEMATE provides the means for modeling the structure (i.e. the decomposition into components), the functionality (i.e. the activities hierarchy) and the

behavior (i.e. the control of the activity hierarchy) of systems employing three orthogonal and complementary visual formalisms. The most interesting among them, statecharts [Harel, 1988], is used for modeling the dynamic behavior of systems and presents some analogies with hierarchical Petri Nets. STATEMATE is able to carry out static and dynamic checks of the constructed models, perform simulations and produce prototype executable code in C or Ada. STATEMATE, however, presents the same limitations as the other approaches described above as to what regards its support for modeling organization behavior, since it is oriented towards real time computer embedded systems. Furthermore, although static and dynamic checks can be carried out, this is no match to the wealth of properties checking that can be carried out in a Petri Net model, where related research has been continuously carried out for a time span of more than three decades; this is especially true in the case of dynamic checking where STATEMATE employs a brute-force approach.

LEU [Dinkoff et al., 1994] is a workflow management environment that can be used for BP modeling. LEU presents considerable affinities with the work described in the present paper; specifically, it allows for organization modeling, activity modeling using a variant of high-level Petri Nets called FUNSOFT nets [Grunn & Jegelka, 1992] and data modeling using an extended Entity-Relationship scheme. One main difference between our approach and LEU is that LEU allows only data objects to be the tokens exchanged in a BP representation. Although data flow is indisputably important in modeling BPs, control flow (via signals and events) is also crucial; our approach accommodates this exigency. A second major difference is that LEU models actors as external to the FUNSOFT nets used for process modeling; the only relationship being the responsibility of an actor to enact some activities in a FUNSOFT net. In our approach, places in Petri Nets can be inscribed with actors which are treated as first-class objects like resource and data objects, making the integration among organization models and process models smooth and tightly coupled. Since places can also be inscribed with resource objects and control objects, a desirable integration between control flow, data flow and organization models is attained. Moreover, actor selection transitions provide a distributed and localized control of the human resources in the organization which is missing in LEU.

The approach described in the context of RBNs (Rule-Based Petri Nets) [Tsalgatidou et al., 1994] and the VENUS supporting environment can be considered as orthogonal to the one presented here. RBNs are not suitable for modeling BPs since they lack support for organization models and they operate in a lower level of detail, but are used for the detailed modeling of information systems using an Object-oriented Rule-based Multi-level Petri Net formalism. RBNs can be used after BP modeling has been carried out with MPNs, in order to derive a concrete description of the information system required to support the processes modeled.

Another similar approach is the one presented in [Ferscha, 1994], where Generalized Stochastic Petri Nets (GSPNs) are used for modeling and analyzing business workflows. However, the approach lacks important features, like hierarchies of decomposed GSPNs and explicit representation of data for modeling the data flow in the process. The first deficiency is

especially important since no handling of the complexity problem of modeling real world processes is provided; this makes the employment of the approach laborious in real world contexts. Moreover, the temporal capabilities of GSPNs are restricted since they are able to represent only immediate and stochastic timed transitions; in MPNs, temporal modeling is accommodated by appropriate handling of the *Timestamp* property in transitions thus permitting representation of any kind of delay, temporal constraints, etc.

The present work can also be compared with work related to executable model specifications and to the use of simulation for improving the organizational efficiency. For example, PAISLey [Zave, 1991] is a language designed for real-time distributed systems and uses an operational approach for building a requirements specification, emphasizing the definition of processes as the building blocks of the system. Our approach could be compared to Zave's approach in the sense that they both realize the necessity of executable model specifications in the line of thought of Kemmener [Kemmener, 1985], where it is stated that executable requirements specifications must be tested early in the development process to ensure that the specified system can be implemented and operate in an effective and efficient manner. Furthermore, simulation and animation have been used by many other researchers as a technique for model validation and as an approach to improve communication between problem solvers and problem owners, see for example the work in [deVreede et al., 1993], [Grunn & Jegelka, 19992], [Karagiannis, 1995], [Verbraeck & deVreede, 1993], [Tsalgatidou et al., 1994].

Conclusions

Organizations have usually very complex processes, therefore the formalism used for the organization's business processes should support operational refinement in order to facilitate BP analysis and redesign. This is offered by the formalism presented in this paper, in the form of Multi-level Modified Petri Nets. MPNs are executable and can be simulated, graphically animated and validated in different abstraction levels. The ERMIS environment provides support for all these tasks by offering tools for MPN construction, analysis and validation. People involved in the BP from various levels of the respective organizational hierarchy can easily participate in the evaluation process by watching the graphical execution of the part of the model which is relevant to their tasks. Thus, the proposed approach can be used for understanding the current functionality of an organization and for redesigning BPs; moreover, the existence of formal, executable models of the organization's BPs facilitates adaptability to changes in the external environment by quickly demonstrating their effects on the internal workings of the organization.

MPNs are able to represent BPs at a sufficient level of detail; it would be desirable to couple MPN BPs representations with a lower level formalism allowing for the detailed specification and design of the information systems implementing them. Such a formalism is offered by the orthogonal approach described in the context of RBNs [Tsalgatidou et al., 1994]; work is

currently under way for an integration of ERMIS with VENUS so as to create an integrated both up and low stream business/ systems modeling environment.

Our future work focuses on the relationship of our work and research in workflow management systems supporting BPs in a real world organizational context. In particular, MPN models constructed and validated in ERMIS represent the workflow and actor participation and enactment; workflow management systems are able to implement such representations in a working environment. A future environment comprising ERMIS and an operational workflow management system is under study, since the conceptual infrastructure is already present in the form of actors, activities, diaries, etc. This would offer the advantages of workflow automation coupled with the analysis and validation advantages gained from the formal aspects of Petri Nets, lack of which is a major deficiency in present commercial workflow management tools.

Appendix

Description of the *Fault Repair* business process of the Greek Telecom-munications Company (OTE)

The organizational structure involved in the Fault Repair process is shown in figure 3. When a telecommunication fault occurs, a subscriber announces this fault by making a phone call to the FAULT RECEPTION organizational unit and the whole business process dealing with the repairing of faults starts. The subscriber just reports the symptoms of the fault and then it is the business process' responsibility to discover the type and the source of the fault and to repair it.

There are three types of faults: faults related to the telephone exchange, faults related to the telecommunication network and faults related to the telephone apparatus. In general, telephone exchange faults are repaired by the TECHNICAL SUPPORT sector of the respective TELEPHONE EXCHANGE department; network faults are repaired by the SOFTWARE NETWORK MAINTENANCE department, unless they are faults of the network wiring (e.g. an excavator has cut a network wire) in which case they are repaired by the NETWORK CONSTRUCTION AND HARDWARE MAINTENANCE department.

A 'fault announcement' is received by an employee of the FAULT RECEPTION sector of the respective TELEPHONE EXCHANGE department who notes the symptoms of the reported fault and searches the already announced faults in order to find out if this fault has been already reported. Then the employee sends a 'fault report' to the respective TELEPHONE EXCHANGE department. In this department, an authorized employee checks if the reported fault is due to a fault of the telephone exchange, in which case it is repaired by this department and a 'fault repair report' is sent back to the FAULT RECEPTION sector. If the fault is not due to a telephone exchange fault, the FAULT RECEPTION sector sends the 'fault report' to the respective TECHNICAL SUPPORT sector of the TELEPHONE EXCHANGE department, in order to discover if the fault is due to the telephone network, or due to the telephone apparatus of the subscriber. If there is a telephone apparatus fault, this is repaired by responsible employees of the TECHNICAL SUPPORT sector. If it is a fault of the network then, this is not repaired here. If the fault was a telephone apparatus fault, a 'fault repair report' is sent back to the FAULT RECEPTION sector, containing information about the repaired fault, else the 'fault report' is sent back.

Network faults are subsequently forwarded to the SOFTWARE NETWORK MAINTENANCE department, where information about the whole telephone network and about which part of the network services each subscriber is held. The responsible employees of this department, by using the received 'fault report' from the FAULT RECEPTION sector, inspect the status of the telephone network, technicians are sent to repair the discovered fault and the information held about the network status is updated. In case that the technicians discover that it's a network wiring fault (e.g. some wires have been cut by accident), they just report to their

manager and a 'fault report' is sent to the department of NETWORK CONSTRUCTION AND HARDWARE MAINTENANCE as this is responsible for repairing such faults. If the fault is here, a 'fault repair report' is filled by the manager and is sent back to the FAULT RECEPTION sector. After faults to the network wiring are repaired by the technicians, a 'fault repair report' is filled by their manager and is sent back to the SOFTWARE NETWORK MAINTENANCE department, in order to update the information held about the network. Finally, the 'fault repair report' is sent back to the FAULT RECEPTION sector.

References

- Davenport, TH. (1993). Process Innovation Reengineering Work through Information Technology. Harvard Business School Press, Boston, Massachusetts, 1993.
- deVreede, G.J. & Bots, P.W.G. & Verbraeck, A. (1993). Simulation as an Approach to Improve Coordination within Service Organisations. In Verbraeck, A. & Kerckoffs, E.J. (eds.). Proceedings of the 1993 European Simulation Symposium (ESS93), Society for Computer Simulation, San Diego, 1993, pp. 41-46.
- Dinkhoff, G., Gruhn, V., Saalman, A. & Zielonka, M. (1994). Business Process Modeling in the Workflow Management Environment Leu. In Loucopoulos, P. (ed.). ER-94: Proceedings of the 13th International Conference on the Entity-Relationship Approach, Manchester, 13-16 Dec. 94, Springer-Verlag, pp. 46-63.
- Ferscha, A. (1994). Qualitative and Quantitative Analysis of Business Workflows using Generalized Stochastic Petri nets. In Chroust, G. & Benezur, A. (eds.). Proceedings of Workflow Management - Challenges, Paradigms and Products (CON '94), Wien, 1994, pp. 222-234.
- Grunn, V. & Jegelka, R. (1992). An Evaluation of FUNSOFT Nets. In: Derniame, J.-C. (ed.). Software Process Technology - Proceedings of the 2nd European Software Process Modelling Workshop, Trondheim, Norway, September, 1992, appeared as Lecture Notes in Computer Science, 635, Springer Verlag, pp. 194-214.
- Hammer, M. & Champy, J. (1993). Reengineering the Corporation A Manifesto for Business Revolution. HarperBusiness, 1993.
- Harel, D. (1988). On Visual Formalisms. Communications of the ACM, Vol. 31, No. 5, May 1988, pp. 514- 529.
- Harel, D., Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Shtull-Trauring, A. & Trakhtenbrot, M. (1990). STATEMATE: A Working Environment for the Development of Complex Reactive Systems. IEEE Transactions on Software Engineering, Vol. 16, No.4, April 1990.
- Jacobson, I., Ericsson, M. & Jacobson, A. (1995). The Object Advantage: Business Process Reengineering with Object Technology. Addison-Wesley Publishing Company, 1995.
- Karagiannis, D. (1995). Business Process Management Systems (BPMS). ACM SIGOIS Bulletin: Special Issue on Business Process Reengineering, Vol. 16, No. 1, August 1995, pp. 10-13.
- Keen, C & Lakos, C. (1994) Information Systems Modelling using LOOPN++, an Object Petri Net Scheme. In Verbraeck, A., Sol, H.G. & Bots, P.W.G. (eds.). Proceedings of the 4th International Working Conference on Dynamic Modelling of Information Systems (DYNMOD-IV), 28-30 Sept. 1994, Noordwijkerhout, The Netherlands, Delft University Press, pp. 31-52.
- Kemmener, R.A. (1985). Testing Formal Specifications to Detect Design Errors. IEEE Transactions on Software Engineering, vol. 11, no. 1, January 1985, pp. 32-43.

Lakos, C.A. & Keen, C.D. (1993). Modeling a door controller protocol in LOOPN. In Magnuson, B., Meyer, M. & Perrot, J.-F. (eds.). Technology of Object-Oriented Languages and Systems TOOLS 10, Proceedings of the tenth International Conference TOOLS Europe '93, Versailles, France, 1993, pp. 31-43..

Marca, D.A. & McGowan, C.L. (1988). SADT. McGraw-Hill, New York, 1988.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, 77 (4), pp. 541-580.

Pinci, V.O. & Shapiro, R.M (1990) An Integrated Software Development Methodology Based on Hierarchical Colored Petri Nets. G. Rozenberg (ed.). Advances in Petri Nets 1990, Lecture Notes in Computer Science, Springer Verlag, 1990.

Starke, G. (1994). Business Models and their Description. In Chroust, G. & Benezur, A. (eds.). Proceedings of Workflow Management - Challenges, Paradigms and Products (CON '94), Wien, 1994, pp. 134-147.

Tsalgatidou, A., Gouscos, D. & Halatsis, C. (1994). Dynamic Process Modelling Through Multi-level RBNs. In Verbraeck, A., Sol, H.G. & Bots, P.W.G. (eds.). Proceedings of the 4th International Working Conference on Dynamic Modelling of Information Systems (DYNMOD-IV), 28-30 Sept. 1994, Noordwijkerhout, The Netherlands, Delft University Press, pp. 327-441.

Tsalgatidou, A. & Junginger, S. (1995). Modelling in the Re-engineering Process. ACM SIGOIS Bulletin: Special Issue on Business Process Reengineering, Vol. 16, No. 1, August 1995, pp. 17-24.

Verbraeck, A. & deVreede, G.J. (1993). Animation as a Communication Vehicle in Simulation Studies. In Pave, A. (ed.). Modelling and Simulation ESM93, Society for Computer Simulation, San Diego, 1993, pp. 670-674.

Zave, P. (1991]. An Insider's Evaluation of PAISLey. IEEE Transactions on Software Engineering, vol. 17, no. 3, March 1991, pp. 212-225.

ADDRESSES: Aphrodite Tsalgaidou, Department of Informatics, Panepistimiopolis, TYPA Buildings, University of Athens, 157 71 Athens, Greece; telephone: +30-1-7217941; fax: +30-1-7219561; e-mail: afrodite@di.uoa.gr. Panos Louridas, Department of Computation, UMIST, Manchester, M60, 1QD, UK; Department of Informatics, Panepistimiopolis, TYPA Buildings, University of Athens, 157 71 Athens, Greece; telephone: +30-1-7217941; fax: +30-1-7219561; e-mail: panos@sna.co.umist.ac.uk, louridas@di.uoa.gr. George Fesakis, Department of Informatics, Panepistimiopolis, TYPA Buildings, University of Athens, 157 71 Athens, Greece; telephone: +30-1-7217941; fax: +30-1-7219561; e-mail: george@di.uoa.gr. Thanasis Schizas, Department of Informatics, Panepistimiopolis, TYPA Buildings, University of Athens, 157 71 Athens, Greece; telephone: +30-1-7217941; fax: +30-1-7219561; e-mail: thanasis@di.uoa.gr