Appl. Math. and Comp. Sci., 1994, vol.4, No.4, 605–633

FORMAL DESCRIPTION TECHNIQUE TO SUPPORT LOAD MODELLING FOR INNOVATIVE COMMUNICATION SYSTEMS

JAN MAGOTT*, BERND WOLFINGER*

In order to evaluate the performance of computer and communication systems under realistic boundary conditions, adequate load models are urgently required. It is desirable to derive those models directly from load measurements.

In this paper, Petri net based models for users of innovative communication systems are presented. First, a survey of basic requirements to a formal load description technique is given. A starting point of our technique is an elementary user modeled by a Petri net being an automaton with time factor. Then basic compositions of elementary users such as sequential, alternative, and iterative compositions are introduced. Finally,the parallel compositions of elementary users are considered. In order to define the parallel composition of many users with similar behaviour, equivalence relations between users are defined.

1. Introduction

Though a large subset of existing computers is already embedded into communication systems (local-area, metropolitan-area and/or wide-area networks), research in innovative communication systems has still some considerable significance.

Two tendencies, in particular, are highly important for communication systems: on the one hand, design and realization of high speed networks (Abeysundara and Kamal, 1991) (e.g. as a consequence of enormous progress in optical transmission techniques (Green, 1991), and on the other hand, the trial to establish the so-called service-integration in communication systems (Kuehn, 1989). The main goal of service-integration is to use one and only one communication system to transfer in parallel very different types of information (data, text, voice, high fidelity audio, video) and at the same time taking into account different quality of service (QoS) — requirements for these different types of information (e.g. high reliability in data communication, jitter-free delivery of video frames in video communication). It is typical to use high speed networks to realize service-integration (Holzman-Kaiser *et al.*, 1991; Popescu-Zeletin, 1990).

In order to determine an appropriate architecture and realization for innovative communication systems, knowledge (at least approximate) of applications to be

* Institute of Engineering Cybernetics, Technical University of Wrocław, ul. Wybrzeże Wyspiańskiego 27, 50–370 Wrocław, Poland
* Department of Computer Science, Hamburg University, Vogt-Kölln Strasse 30, 22527 Hamburg, Germany

expected is required. However, most of these applications are presently non-existing, though CCITT recommendation I.121 (1989) introduces a preliminary classification of multimedia applications to be expected with the following classes: interactive services (with subclasses: conversational, messaging and retrieval services) and distribution services (with and without individual user presentation control). Morover, realistic models for communicating applications as a basis for representative load models are indispensable in order to achieve valid performance evaluations for innovative communication systems (cf. Abeysundara and Kamal, 1991; Pujolle, 1991).

Support of realistic load modelling for communication systems represents the primary goal of this paper: first, we want to introduce a general proceeding in modelling communicating applications, and as a support of this proceeding, we will present a formal description technique, which can be used in specifying realistic load models for communication systems (in particular, for high speed networks with service-integration). At the beginning, we summarize the basic requirements to the load description technique (cf. section 2). Section 3 introduces the general proceeding for load modelling as it is proposed by us. Thereafter, our approach to formal load description (based on extended Petri nets) is elaborated for elementary user models (section 4). The description technique is then generalized by introduction of composition methods in section 5 and 6 (e.g. to specify complex user models, which may result from a superposition of different arrival processes). The description technique will be exemplified by means of typical load models as observed in communication systems. Possibilities to reflect the varying level of detail in load models (refinements and abstractions of user models) will be illustrated in section 7. Finally, advantages as well as limitations of our description technique are demonstrated in section 8.

2. Requirements to the Description Technique

In the application of load models for communication systems the following primary objectives exist:

1. Characterize the load generation process for single user or groups of communication system users and, then, use this characterization to compose load models for more complex environments resulting, e.g. by overlapping the traffic generated by a large quantity of individual users (possibly of different type).
2. Characterize the set of users of communication system and, then, use the resulting load model LM_0 in order to
   - derive a simplified load model LM_0' (where LM_0' still possesses some of the basic properties of LM_0 w.r.t. load generation process specified, though LM_0' is more elementary); or
   - create an artificial load in an existing system (e.g. for the purpose of performance measurements for this system).
3. Characterize the set of users of a communication system and, then, use the resulting load model LM_0 in combination with an appropriate system model SM_0, in order to
   - obtain a (sufficiently realistic) performance analysis/prediction for the communication system modeled; or

- transform the load model LM_0 for a given interface I_0, e.g. by means of a simulator, into a load model LM_1 for a different interface I_1, which may be a more system internal interface (cf. problem of transforming a primary load into a secondary load as introduced e.g. in (Wolfinger and Kim, 1990)).

Above mentioned objectives demonstrate that e.g. load modelling in general is dependent on the following characteristics of an innovative communication system:

- the properties of the interface, which is used to access the communication systems services (e.g. very different types of interfaces are possible, such as interprocess communication (IPC)-, Broadband-ISDN (B-ISDN)-, FDDI-interface etc., cf. (Tanenbaum, 1989));
- the type of resource management within the communication system (e.g. rough characterization of resource requirements for data transmission-oriented requests is typically sufficient, whenever resource allocation is more static such as in circuit-switching networks; on the contrary, very precise characterization of resource requirements, whenever allocation is highly dynamic as is often the case in packet- switching networks);
- the types of interactions (direct or indirect) between users or between users and communication system, respectively (e.g. relationship between communicating users in different endsystems; dependency in user behaviour of the communication system's state).

In order to be able to specify realistic load models, therefore, we want the following **requirements** to be satisfied by the formal description technique developed in this paper:

(R1) *Clean separation between model of environment and of system*, respectively.

(R2) Requirements in *characterizing single user behaviour* (where users are considered to be the entities generating requests over time and being part of the environment):

a) characterize arrival process of requests (attributed events generated by users) over time,

b) characterize single requests (their types w.r.t. a given interface, e.g. OPEN, CLOSE, SEND, RECEIVE, and their attributes w.r.t. their resource requirements).

In order to express single user behaviour in a rather flexible way, we would like to have description methods, which — on the basis of an elementary user model — allow us to formulate sequential order of behaviour, choices of behaviour variants and repetitive behaviour.

(R3) Requirements in *characterizing the behaviour of a set of users*, which may be

a) independent on each other ($\longrightarrow$ straight-forward extension) or

b) dependent on each other, e.g.

b1) within the environment (example: users communicating with each other);

b2) in accessing a common interface (example: users having access to a common communication network interface, such as X.25 or S_{2m} interface);

b3) by demanding common system resources (such as transmission lines, buffers, communication processors, etc.).

(R4) *Classifying and expressing similarities of users*, e.g.

a) similarities in the arrival process of requests over time,

b) similarities in types and attributes of requests (leading e.g. to similar resource demands).

Thus, we want to be able to compose complex load generating environments based on elementary user models.

Overall, we would like to have a formal description technique, which allows us to specify (for various system interfaces) load models as largely independent of system models as is adequate. At the same time, we want to be able to express dependencies of users generating the load on system reactions. In addition, we would like to have a straight-forward possibility of load model evaluation (mathematical analysis or simulation) and we want to support different levels of detail in load modelling (e.g. refinement/abstraction in the load-generating environment and in the interface).

Though there exist load description techniques, such as user behaviour graphs (cf. Ferrari, 1984) or queueing networks (cf. Kleinrock, 1976), which satisfy some of our above requirements, we are not aware of any approach, which covers all of these requirements to a sufficient extent. Our approach will be strongly based on a generalized proceeding in load modelling, being the result of earlier work of one of the authors of this paper (cf. Wolfinger and Kim, 1990). The basic assumptions and the steps in load modelling according to this proceeding will be introduced in the next section.

3. A Unified Approach to Description and Modelling of Communication System Loads

We now are going to introduce a new approach to load description and modelling. To begin, let us shortly refine our notion of load as it is used in the following:

Definition. The (offered) load $L = L(E, S, IF, T)$ denotes the total sequence of requests, which is offered by an environment E to a (service-) system S via a well-defined interface IF during the time-interval T. We then call L the load generated by E for S during T. ■

The main purpose of this section is to present a unified description technique which allows us to formulate models of load (mainly for simulation experiments) for different degrees of detail in modelling and for various kinds of system interfaces. In particular, we want to cover load which reflects requirement of communication resources (however, this is no general restriction).

It should be mentioned that we do not want to support modelling users of computer and/or communication systems on such an extremely high level of detail as is the case in some man-machine-communication studies (cf. Kobsa, 1985). We also do not want to solve problems like finding specific distributions for request arrival processes or request parameters.

As boundary conditions for our approach, we try to take into account the requirements to load modelling as introduced in section 2 as well as results of existing measurements (cf. Pawlita, 1988).

We want to present our approach to load modelling based on four main steps, now. The first of these steps is motivated by the fact that load modelling necessarily requires a well-defined interface where load is offered (cf. above definition).

STEP 1: Decomposition (of system and model)

A general study of how to adequately decompose complex structures (or systems) is presented in (Courtois, 1985). Here, we have to decompose some given communication system and its embedding in some environment (comprising e.g. the set of system users) in a way that we exactly decide where to place the boundary (called *demarcation line*) between the system and environment. Moreover, we now suppose that the system S and the environment E, which has been a result of our basic decomposition, interact explicitly by exchanging *requests* (from E to S) and *reactions* (from S to E). For the creation of requests we suppose a set of *users* (being part of E), which may take into account reactions of S in their future behaviour. Reactions are seen as being consequences of earlier requests serviced by S.

With these assumptions, after mapping the system and environment onto corresponding models, we get the following model components as depicted in Fig. 1. One should note here that system decomposition implies a corresponding model decomposition. The semantic of the components of Fig. 1 is such that

- The system model (*m_system*) MS represents e.g. a communication system model.
- The *interfaces* (*IF*) are introduced to have some explicit points, where interactions between the system and environment can take place; an interface specifies exactly, in which way the system and environment can interact (possible interactions); we suppose that the interfaces of an *m_*system are constant over time (static property); each interface is assumed to be associated to exactly one internal component of *m_*system (which would only be seen in a further decomposition of the *m_*system itself); we will also say that an interface represents an *interaction point*, where *m_requests* are accepted (for being serviced) and *m_reactions* (if any exist at all) are handed over to *m_*users (cf. below).
- The environment model (*m_environment*) ME contains only those components which are required to model creation of load over time (for all of the interfaces); the load-creating components are called *m_users* (*m_*u in Fig. 1); *m_*users can be dynamically created over time, they are the only components generating requests; the interface(s) via which an *m_*user interacts with the *m_*system are supposed to be known in advance; the possible behaviour of an *m_*user is restricted by the interface(s) it interacts with; what we model as an *m_*user strongly depends on the degree of detail chosen for modelling the environment (it may even be possible that the total *m_*environment is represented as just one *m_*user).

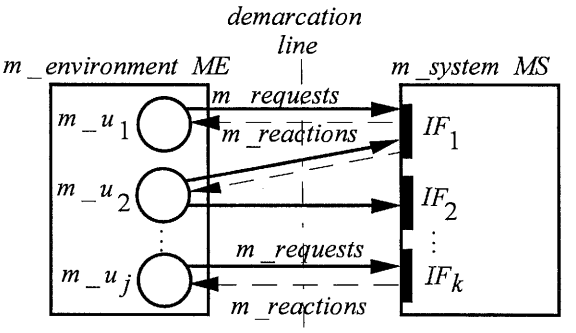


Fig. 1. Model components and interactions important for load modelling.

The prefix "*m_*", used to denote model components, indicates that we describe some part of "model world" as opposed to the existing system and environment being modeled. In the following we will drop this prefix whenever ambiguities can be excluded.

STEP 2. Choice of level of detail in modelling

The purpose of the second main step is to decide what types and which parameters/attributes of requests and reactions, exchanged between ME and MS, we have to take into account (in order to obtain a suitable load model). We first have to decide on the adequate level of detail for MS which then implies the correct level of detail in modelling ME-MS-interactions. Two possibilities for interactions between MS and ME exist: ME can be independent of MS-reactions (e.g. no reactions exist or they are just ignored by users) or reactions of MS influence behaviour of ME (i.e. of the users it comprises). We will cover the second possibility here, because it can be considered as a generalization of the first one. So, the following load modelling approach has to describe the sequences of requests generated by the set of users over time and has to model influence of reactions on user behaviour. Creation of reactions, however, is not part of load modelling as reactions are assumed to be always consequences of earlier requests. We are going to model interactions between MS and ME within two main steps (3 and 4) now.

STEP 3. Analysis and description of interactions being possible (for a given interface)

For each interface IF of MS we have to solve two basic problems: first, to formally describe all types of requests and reactions, which are relevant for IF, and second, to express in advance, which sequences of requests/reactions are possible at all for IF. The second problem can be solved by some (protocol) specification technique (examples, cf. Tanenbaum, 1989). Its purpose is to be able to verify that actual interactions between MS and ME (cf. Step 4) are indeed valid sequences of requests /reactions with respect to the interfaces.

STEP 4. Description of actual interactions between MS and ME (during the interval, model behaviour is observed)

Describing actual interactions between MS and ME means that we basically have to solve the following two main problems:

P1) For any user U we have to describe the stream of requests (possibly as a superposition of s substreams), which U generates over time and passes to MS. Here, for a given interface IF a (sub)stream of requests S_R corresponds to a vector of (time, request)-tupels $S_R = \langle\langle t_1, A_1\rangle, \langle t_2, A_2\rangle, \langle t_3, A_3\rangle, \cdots, \langle t_k, A_k\rangle\rangle$ for some $k \in N$ where $t_i \cong$ instant of A_i generation (thus $\langle t_1, t_2, \cdots, t_k\rangle$) characterize the arrival process for S_R), and $A_i \in \mathcal{A}(IF)$, where $\mathcal{A}(\mathcal{IF})$ denotes the set of requests possible at IF.

P2) We have to describe the total offered load resulting from the superposition of all streams of requests (generated by all users part of ME).

The next sections (4 to 7) will concentrate on solving problems P1) and P2).

4. Formal Description of Elementary User

In the following, we will introduce a load description technique, which will be based on extended Petri nets (in particular timed, stochastic, coloured Petri nets). The use of Petri nets has been motivated e.g. by some of our requirements towards the description technique to be elaborated, such as parallel composition of individual user models, possibilities of refinements, and support of the description technique by tools for (simulative) model evaluation. Based on past experiences (cf. Killat *et al.*, 1988; Magott, 1984; 1985) in using successfully Petri nets for computer and communication system modelling (e.g. to support performance and/or performability analyses), we claim that Petri nets represent the appropriate means in order to cover those requirements. As an alternative, a load description technique trying to satisfy similar requirements could be based on extended finite automata, too; this approach was investigated e.g. in (Wolfinger and Kim, 1990).

An elementary user is expressed by such a Petri net that is an automaton. Hence, for each transition there exist exactly one arc directed to the transition and exactly one arc directed from the transition. At each time instant, one place only is marked. The state of the elementary user is represented by the place where a token is located. If there is a token in the initial place p_i, then the user is idle. The requests can be generated when the token is put in one of active places P_a. If the token is located in one of blocked places P_b, then the user is waiting for system reaction. The token in a terminated place p_t means that further creation of requests is impossible. A brief representation of elementary user is given in Fig. 2. Symbol T with two indices denotes a subset of transitions. The meaning of symbols is given in the following definition.

Definition. An elementary user is an *interpreted Petri net*

$$U = \langle P, f, T, F, M_0, R\rangle$$

where the meaning of the components is as follows.

1. P is the *set of places* such that

$$P = \{p_i\} \cup P_a \cup P_b \cup \{p_t\}$$

where p_i is initial place, P_a – set of active places, P_b – set of blocked places and p_t is terminated place.

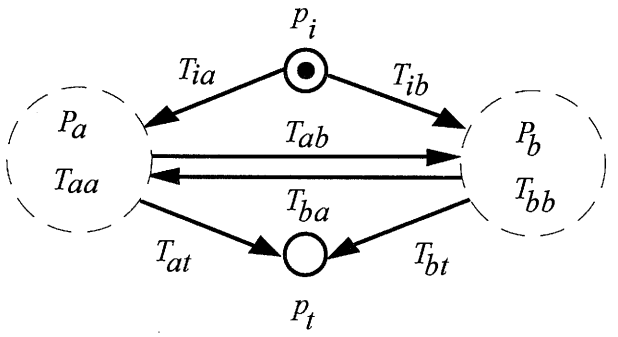


Fig. 2. Model of elementary user.

2. $f : P_a \longrightarrow RT \cup \{\theta\}$ is a *request type function*,

   where RT is a set of types of requests that can be generated by the user U, whereas θ means that no request is generated. The value $f(p_k) \in RT$ is the type of request that is generated when there is a token in the active place p_k. If no request is generated by the user when the place p_k is marked, then $f(p_k) = \theta$.

   The request type $r_i \in RT$ is characterized by a pair $r_i = \langle \Omega_i, \Pi_i \rangle$. The component Ω_i is a space of request attributes and is given by

   $$\Omega_i = A_{i_1} \times A_{i_2} \times \ldots \times A_{i_k} \times \ldots \times A_{i_p}$$

   where A_{i_k} is the set of values for the i_k-th attribute, i_p is the number of attributes for the r_i request type. The component Π_i determines the way how concrete requests are obtained. A concrete request is such a vector from the space Ω_i that values of attributes can be read as input parameters or can be obtained from a probability distribution over Ω_i. The interface used by a user being in a given active place can be one of the attributes too.

3. T is the *set of transitions* such that

   $$T = T_{ia} \cup T_{ib} \cup T_{aa} \cup T_{bb} \cup T_{ab} \cup T_{ba} \cup T_{at} \cup T_{bt}$$

   where the meaning of the subsets is as follows.

   The meaning of indices: i – initial, a – active, b – blocked, t – terminated. For example, T_{ia} is the set of transitions for token flow from the set of initial places (one element p_i only) to the set of active places, whereas T_{aa} is the set of transitions for token flow between active places. The other sets are determined in a similar way.

4. F is the *set of arcs* such that

   $$F = F_{ia} \cup F_{ib} \cup F_{aa} \cup F_{bb} \cup F_{ab} \cup F_{ba} \cup F_{at} \cup F_{bt}$$

   where the meaning of the subsets is similar as for the set of transitions (e.g. F_{ia} is the set of arcs for token flow from the initial place p_i to the active places).

The set of arcs satisfies the requirement imposed on automata

$$(\forall\, t \in T)\Big(card({}^{o}t) = card(t^{o}) = 1\Big)$$

where $card({}^{o}t)$, $card(t^{o})$, respectively, are the cardinality of the set of input, output places of the transition t, respectively.

The above subsets of the set F are defined according to the relations:

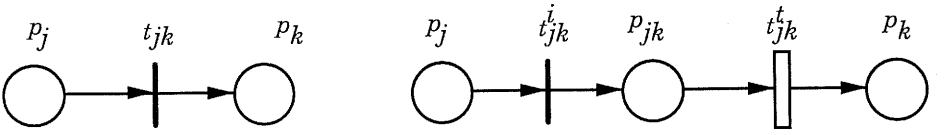
$$F_{ia} \subset \{p_i\} \times T_{ia} \cup T_{ia} \times P_a\,.$$
$$F_{ib} \subset \{p_i\} \times T_{ib} \cup T_{ib} \times P_b$$
$$F_{aa} \subset P_a \times T_{aa} \cup T_{aa} \times P_a$$
$$F_{ab} \subset P_a \times T_{ab} \cup T_{ab} \times P_b$$
$$F_{ba} \subset P_b \times T_{ba} \cup T_{ba} \times P_a$$
$$F_{bb} \subset P_b \times T_{bb} \cup T_{bb} \times P_b$$
$$F_{at} \subset P_a \times T_{at} \cup T_{at} \times \{p_t\}$$
$$F_{bt} \subset P_b \times T_{bt} \cup T_{bt} \times \{p_t\}$$

5. $M_0 : P \longrightarrow \{0, 1\}$ is the *initial marking* such that
   $M_0(p_i) = 1$ for the initial place p_i,
   $M_0(p_j) = 0$ for the other places.
6. R is a *refinement of transitions.* It will be given later, after some additional explanation, required forehand.

Explanation of refinement of transitions. In order to incorporate the choices and time into the model, the previously presented transitions are refined according to the following scheme. The transition t_{jk} (Fig. 3a) can be replaced by the subnet given at Fig. 3b. The transition t^i_{jk} pictured by a bar is called an immediate one. The firing time of such a transition is equal to zero. The transition t^t_{jk} pictured by a rectangle is called a timed one. The firing time of such a transition could be greater than zero.

Refinement of the transition t_{jk} has three forms:

1. t_{jk} is replaced by a subnet containing the immediate transition t^i_{jk} only,
2. t_{jk} is replaced by the timed transition t^t_{jk},
3. t_{jk} is replaced by the subnet given at Fig. 3b where t^i_{jk} is the immediate transition, and t^t_{jk} is the timed one.



a) b)

Fig. 3. Refinement of a transition (t_{jk}).

The *firing time* of the timed transition can be given in three ways:

1. by a positive real number that is the same for each firing of the transition,
2. by a non-negative real number that is read as a separate input parameter for each firing, e.g. sequence of firing times is given by a trace,
3. by a probability distribution of the non-negative random variable.

The timed transition is fired in the following manner. Let the token be located in the place p_{jk} in a moment τ, and let the firing time instance of t^i_{jk} be equal to τ'. Then in the moment $\tau + \tau'$, the token from the place p_{jk} is removed and added to the place p_k.

Now we present the *ways of choosing the transition to fire* when the marked place enables more than one transition.

Let p^o_j denote the set of output places of transition p_j.The choice of transition to fire will be based on "*race semantics*": the transition with the smallest firing time in the set p^o_j is fired. Therefore, if the place p_j enables at least one immediate transition, then the immediate one is fired.

If the place p_j enables more than one immediate transition, then there are two options.

First option. Let p^{o*}_j denote the subset of immediate transitions in the set p^o_j. The transition from the set p^{o*}_j is chosen to fire according to a discrete probability distribution such that

$$\sum_{t^i_{jk} \in p^{o*}_j} p(t^i_{jk}) = 1$$

where $p(t^i_{jk})$ is the probability that the transition t^i_{jk} will be fired if the place p_j is marked. This first option is illustrated in Fig. 4.

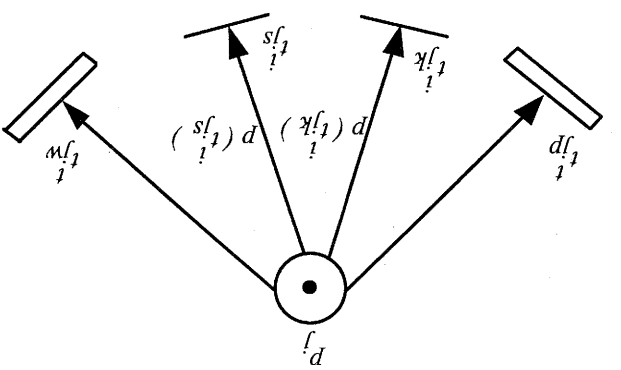


Fig. 4. Choice of transition to fire (given a marked place p_j).

Second option. The transition from the set p^{o*}_j is chosen to fire according to an external event. The result of occurrence of these events is modelled in the following way. The immediate transition $t^i_{jk} \in p^{o*}_j$ is refined by adding a place p^i_{jk} and an

arc $\langle p^i_{jk}, t^i_{jk}\rangle$. Hence, the transition t^i_{jk} is enabled when the places p_j and p^i_{jk} are marked. The token is put to the place p^i_{jk} as a result of occurring of the external event (which is not included in the model). The external event can be, e.g. a reaction of the communication system or a decision of the user. The time instant, when the token is put to the place p^i_{jk} (as a result of the external event occurrence), is given by a real number. We assume that:

1. For each time when the place p_j is marked, one place p^i_{jk} only (from the set of input places of the transitions $t^i_{jk} \in p^{o*}_j$) is marked.
2. The place p^i_{jk} can contain at most one token.

Remark. In this paper, in order to simplify the figures, we will draw neither the place p^i_{jk} nor the arc $\langle p^i_{jk}, t^i_{jk}\rangle$, but we will label the transition t^i_{jk} by a symbol of the external event.

Definition of the elementary user (continued)

Refinement of transitions of the user $U = \langle P, f, T, F, M_0, R\rangle$ is defined for each transition $t_{jk} \in T$ by:

1. definition of the subnet that replaces the transition t_{jk},
2. definition of firing time for timed transition t^t_{jk} which can be given by either:
   a. positive real constant,
   b. sequence of non-negative reals,
   c. non-negative random variable,
3. definition of choice policy for immediate transition t^i_{jk} that is described by either one:
   a. probability $p(t^i_{jk})$,
   b. sequence of non-negative reals expressing time instants, when the token is put to the place p^i_{jk} (input place of the transition t^i_{jk}). ■

Now we present two short examples of representation of communication system user features using the above model.

Example 1. Time — out (Fig. 5).

Let the firing time $\tau(t^t_{bj})$ of the transition t^t_{bj} express a length of a time interval during which the user would have waited for acknowledgement if there had not been a bound for the waiting time. Let the firing time $\tau(t^t_{bk})$ of the transition t^t_{bk} represent a length of a time interval after which the time-out signal is generated. Hence, according to the "race semantics":

1. if $\tau(t^t_{bj}) > \tau(t^t_{bk})$, then the time-out signal is generated,
2. if $\tau(t^t_{bj}) \leq \tau(t^t_{bk})$, then the acknowledgement is received. ■

Example 2. The user is blocked and two events e_1, e_2 are required in any order to become active (Fig. 6).

P_a is the set of active places, P_b is the set of blocked places. ■

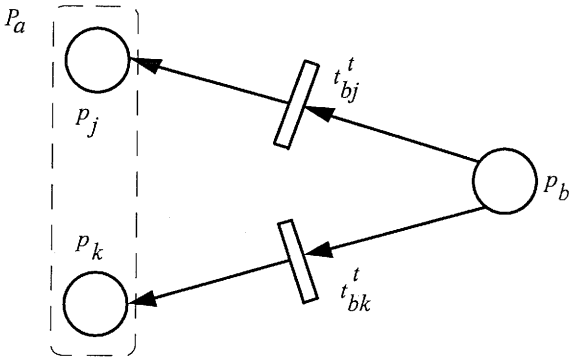


Fig. 5. Choice between transitions in case of time-outs (Example 1).

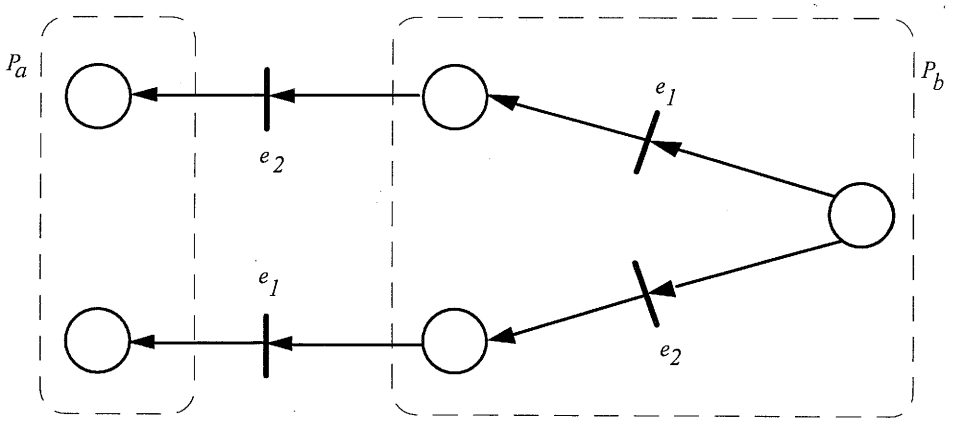


Fig. 6. Modelling of blocking situations (Example 2: user blocked by having to wait for two events e_1, e_2).

5. Basic Compositions of Users

In this section, we present the following composition operations that can be applied on elementary users:

1. sequential composition,
2. alternative composition,
3. iterative composition.

5.1. Sequential Composition

In communication systems we often observe the situation that one behaviour pattern of a user is followed by another behaviour pattern, e.g. editing and thereafter compilation. Hence, a sequential composition of users is required.

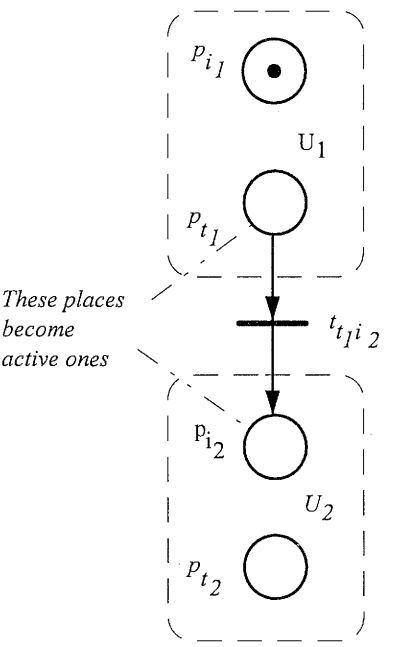


Fig. 7. Illustration of sequential composition of users U_1 and U_2.

Definition. Let us consider two elementary users $U_1 = \langle P_1, f_1, T_1, F_1, M_{0_1}, R_1 \rangle$ and $U_2 = \langle P_2, f_2, T_2, F_2, M_{0_2}, R_2 \rangle$ with the following requirements being satisfied: $P_1 \cap P_2 = \emptyset$, $T_1 \cap T_2 = \emptyset$.

A user U being the sequential composition of the users U_1, U_2 is defined as

$$U = \langle P, f, T, F, M_0, R \rangle$$

where the meaning of the components is given below. (In this definition, index 1 refers to U_1 and index 2 to U_2).

The set of places is defined by equation

$$P = \{p_i\} \cup P_a \cup P_b \cup \{p_t\}$$

where

$$p_i = p_{i_1}, \qquad P_a = P_{a_1} \cup P_{a_2} \cup \{p_{t_1}\} \cup \{p_{i_2}\}$$

$$P_b = P_{b_1} \cup P_{b_2}, \qquad p_t = p_{t_2}$$

Let $f|X_1$ be a restriction of the function $f : X \longrightarrow Y$ to the subset $X_1 \subset X$. For the request type function, the following holds:

$$f|P_{a_1} = f_1, \qquad f|P_{a_2} = f_2$$

$$f(p_{t_1}) = \theta, \qquad f(p_{i_2}) = \theta$$

The set of transitions T is defined as follows:

$$T = T_{ia} \cup T_{ib} \cup T_{aa} \cup T_{ab} \cup T_{ba} \cup T_{bb} \cup T_{at} \cup T_{bt}$$

where

$$T_{ia} = T_{ia_1}, \qquad T_{ib} = T_{ib_1}$$

$$T_{aa} = T_{aa_1} \cup T_{aa_2} \cup T_{at_1} \cup T_{ia_2} \cup \{t_{t_1 i_2}\}$$

($t_{t_1 i_2}$ is an additional transition between the users U_1, U_2 as shown at Fig. 7)

$$T_{ab} = T_{ab_1} \cup T_{ab_2} \cup T_{ib_2}, \qquad T_{ba} = T_{ba_1} \cup T_{ba_2} \cup T_{bt_1}$$

$$T_{bb} = T_{bb_1} \cup T_{bb_2}, \qquad T_{at} = T_{at_2}, \qquad T_{bt} = T_{bt_2}$$

For the set F of arcs the following holds:

$$F_{ia} = F_{ia_1}, \qquad F_{ib} = F_{ib_1}$$

$$F_{aa} = F_{aa_1} \cup F_{aa_2} \cup F_{at_1} \cup F_{ia_2} \cup \left\{\langle p_{t_1}, t_{t_1 i_2}\rangle, \langle t_{t_1 i_2}, p_{i_2}\rangle\right\}$$

$$F_{ab} = F_{ab_1} \cup F_{ab_2} \cup F_{ib_2}, \qquad F_{ba} = F_{ba_1} \cup F_{ba_2} \cup F_{bt_1}$$

$$F_{bb} = F_{bb_1} \cup F_{bb_2}, \qquad F_{at} = F_{at_2}, \qquad F_{bt} = F_{bt_2}$$

The initial marking M_0 is such that $M_0(p_{i_1}) = 1$, $M_0(p) = 0$ for the other places.

The refinement R is given by:

$$R|T_1 = R_1, \qquad R|T_2 = R_2$$

The transition $t_{t_1 i_2}$ is refined to the immediate transition $t^i_{t_1 i_2}$. The transition $t^i_{t_1 i_2}$ is the only output transition of the place p_{t_1}. Hence,there is no conflict when the token is located in p_{t_1}. ■

5.2. Alternative Composition

In communication systems, a user often behaves according to one of several possible behaviour patterns, e.g. file transfer and electronic mail. Therefore, an alternative composition is required.

Figure 8 presents the alternative composition. The formal definition is of similar form as the definition of sequential composition and thus omitted.

5.3. Iterative Composition

A user often repeats a sequence of activities, e.g. editing and compilation could be done many times. This leads us to an iterative composition.

Figure 9 illustrates the iterative composition. Places p_i, p_t are added because of compositionality requirements.Again, the formal definition is analogous to that for sequential composition.

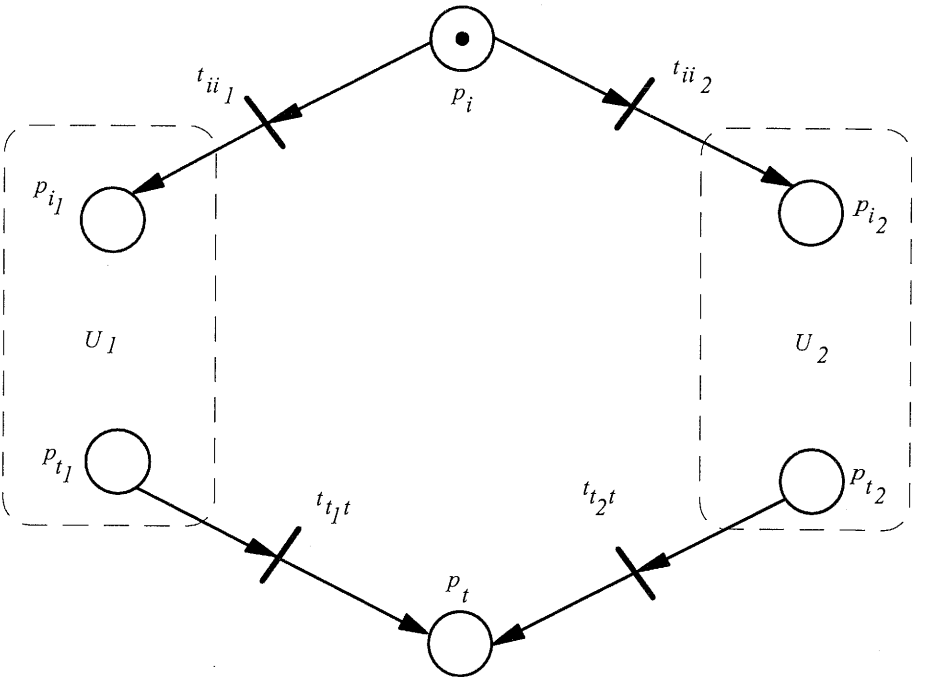


Fig. 8. Illustration of alternative composition of users U_1 and U_2.

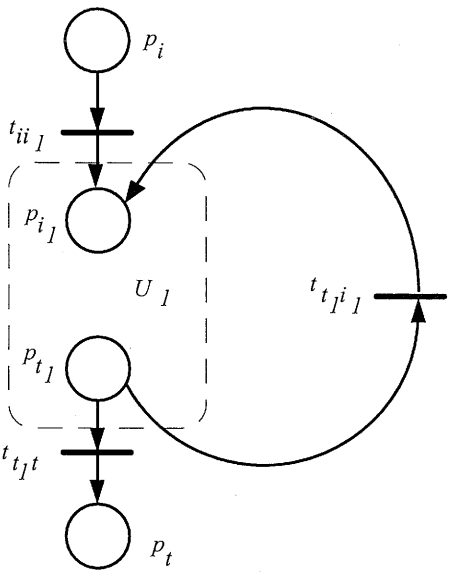


Fig. 9. Illustration of iterative composition of a user U_1.

Remark. When the initial and terminated phases of the user activity can be neglected (because of long term activity of an iterative user), then the transitions t_{ii_1}, $t_{t_1i_1}$, t_{t_1t} can be omitted and the places p_{i_1}, p_{t1} can be merged into one place.

6. Parallel Compositions

First, we consider the parallel compositions of two elementary users, and then the parallel composition of many elementary users of the same type.

6.1. Parallel Compositions of Two Users

When studying the parallel composition of two users, we want to distinguish two cases:

1. users without interactions expressed in the load model,
2. users with interactions.

6.1.1. Parallel Compositions of Users Without Interactions

In communication systems, sometimes users are combined into a set of correlated users, and in other cases, they create a new user with common (for the analysed set of users) initialization and common termination. This leads us to the following compositions: *addition* and *fork-join*.

Definition. The result of addition composition of two elementary users $U_1 = \langle P_1, f_1, T_1, F_1, M_{0_1}, R_1\rangle$, $U_2 = \langle P_2, f_2, T_2, F_2, M_{0_2}, R_2\rangle$ without interactions, under assumptions $P_1 \cap P_2 = \emptyset$, $T_1 \cap T_2 = \emptyset$, is the user

$$U = \langle P, f, T, F, M_0, R\rangle$$

where

$$P = P_1 \cup P_2, \quad T = T_1 \cup T_2, \quad f|P_{a_1} = f_1, \quad f|P_{a_2} = f_2, \quad F = F_1 \cup F_2$$

$M_0(p_{i_1}) = M_0(p_{i_2}) = 1$ and $M_0(p) = 0$ for the other places

$$R|T_1 = R_1, \qquad R|T_2 = R_2$$

■

The user obtained by the composition has two marked initial places and two terminated places. A brief illustration is given in Fig. 10. The resulting user can be considered as a set of two users.

Example 3. The user U_1 sends a sequence of messages to user U_2, and the communication is supported by a time-out mechanism.

```
while true do
   begin
     message production;
     repeat
       message sending;
       wait for (time-out or ack.)
     until there is ack.
           during interval
   end
```

```
while true do
   begin
     message receiving;
     ack. sending;
     message consumption
   end
```

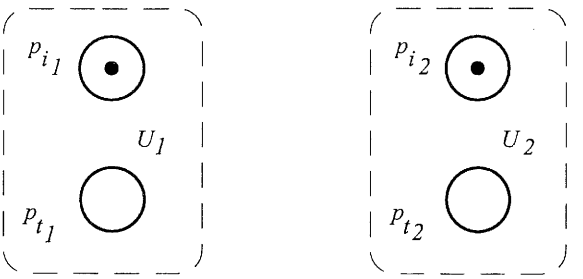


Fig. 10. Illustration of addition composition of users U_1 and U_2 without interactions.

Assumption: Acknowledgements are sent correctly.

The addition composition of users U_1, U_2 is represented in Fig. 11.

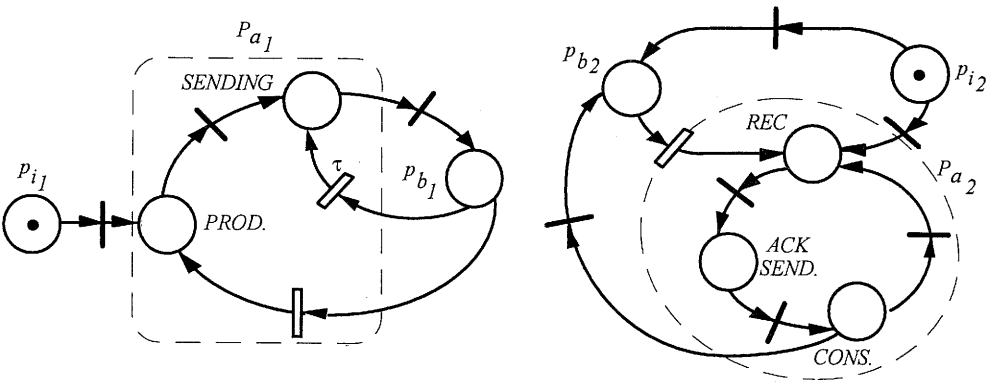


Fig. 11. Addition composition of communicating users U_1 and U_2 (Example 3).

The users interact with each other indirectly through the system. ■

For *fork-join composition*, a brief representation is given in Fig. 12. (The obvious formal definition is omitted).

6.1.2. Parallel Compositions of Users with Interactions

Because of different forms of parallel composition, it is impossible to give a formal definition covering the general case. Let us consider an example of parallel composition of two users with interactions.

Example 4. Two users send video and voice: U_1 – video, U_2 – voice. They are initialized simultaneously. In order to start a transmission, they both have to get access to the transmission media. After they have got access to the media, the transmission can be started. Both kinds of transmission are performed separately (using different interfaces). The user obtained by parallel composition of the users is terminated when both users are terminated. The users are represented in Fig. 13 and a result of parallel composition of the users is given in Fig. 14.

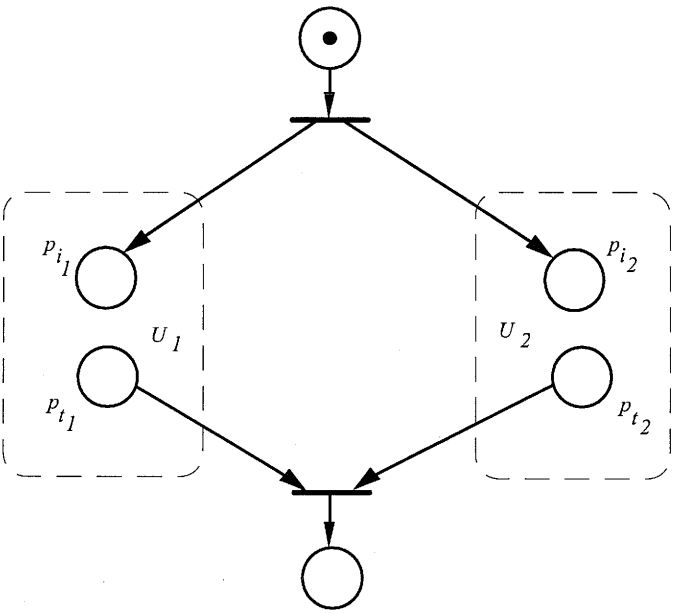
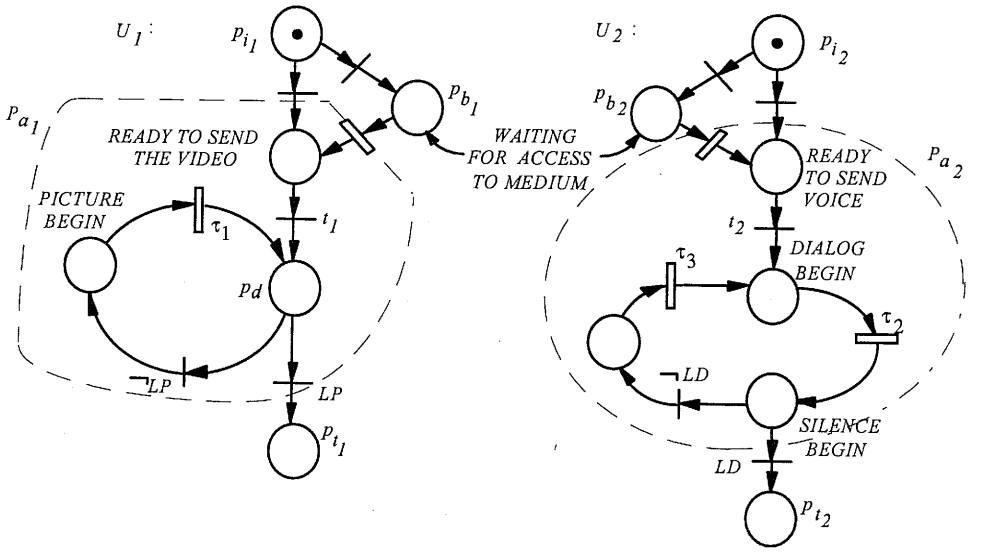


Fig. 12. Fork-join composition of users U_1 and U_2 without interactions.



τ_1 : interpicture time
LP : event of completing the video transmission

τ_2 : dialog time
τ_3 : interdialog silence time
LD : event of completing the voice transmission

Fig. 13. Examples of two users U_1 (sending video) and U_2 (sending voice) (Example 4).

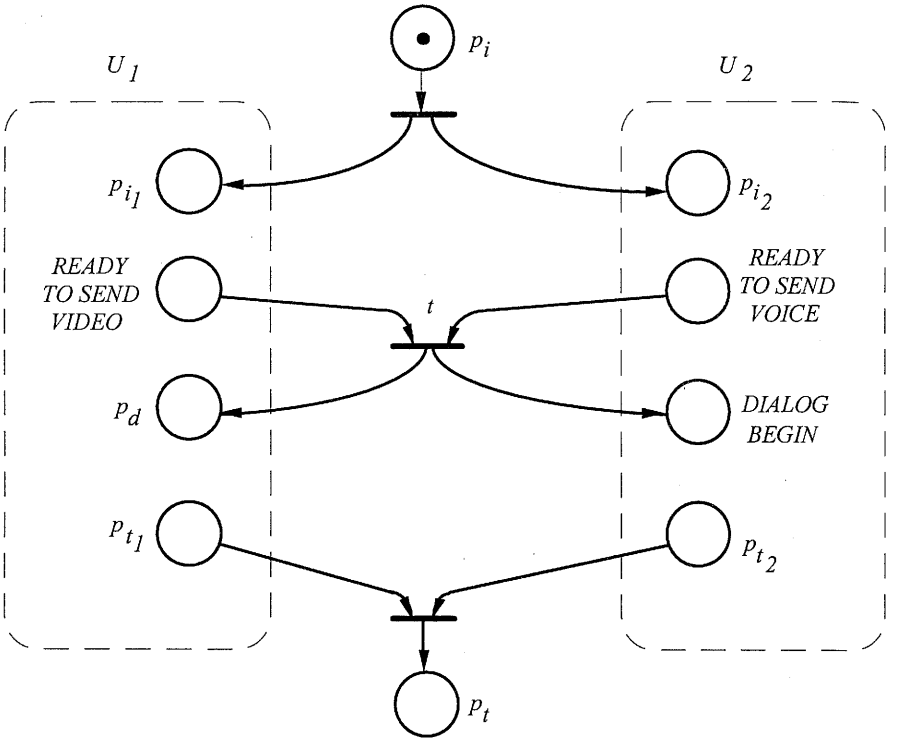


Fig. 14. Parallel composition of users U_1 and U_2 as represented in Fig. 13 (Example 4).

Except the transitions typical for fork-join composition, there exists a transition t to express the required synchronization, when both users have obtained access to the transmission media. Hence, direct interaction between the users is expressed in the load model. ■

The example could suggest that compositions of CCS (Milner, 1980) are sufficient to define the parallel compositions of users. The example given in the next subsection shows that this it is not true in general.

6.2. Parallel Compositions of Many Users of the Same Type

In order to simplify the models of complex users (by not dealing separately with similar users), we will introduce equivalence relations between users.

Equivalence relations between elementary users

Definition. Two elementary users $U_1 = \langle P_1, f_1, T_1, F_1, M_{0_1}, R_1 \rangle$, $U_2 = \langle P_2, f_2, T_2, F_2, M_{0_2}, R_2 \rangle$ are E_1-equivalent if:

1. $P_1 = P_2$
2. $T_1 = T_2$
3. $F_1 = F_2$
4. $M_{0_1} = M_{0_2}$

i.e. their Petri nets are identical. ■

The definition can be extended using a notion of isomorphism of nets, but it will not be done in this paper.

For example, typical telephone users are described by the same Petri net. The users that are E_1-equivalent can be interpreted as the users of the same type.

Definition. The elementary users U_1, U_2 are E_2-equivalent if:

1. they are E_1-equivalent,
2. $f_1 = f_2$

i.e. for the same places, they generate requests of the same type. ■

The users that are E_2-equivalent could be characterized by different refinements. For example, let us analyse telephone users that generate the following request types: a) talkspurt begin (when talkspurt period is started), b) silence begin (when silence period is started). Telephone users being E_2-equivalent generate the same request types, but they could be characterized by different talkspurt periods or different silence periods during their conversation.

Definition. The elementary users U_1, U_2 are E_3-equivalent if:

1. they are E_1-equivalent,
2. $R_1|(T_{aa} \cup T_{ab} \cup T_{at}) = R_2|(T_{aa} \cup T_{ab} \cup T_{at})$

i.e. their refinement is equal in their system-independent part. ■

Let us consider telephone users that are E_3-equivalent but not E_2-equivalent. They are characterized by equal durations of talkspurt periods and by equal durations of silence periods. However, they should differ in request types, for example their requests could concern different lines because they are located e.g. in different countries. We assume that E_3-equivalent users have identical refinement for transitions describing the activity of users only. We do not require that e.g. waiting time by users being blocked for system reaction is equal (transitions from the set T_{ba}).

Definition. The elementary users U_1, U_2 are E_4-equivalent if:

1. they are E_2-equivalent,
2. they are E_3-equivalent. ■

E_4-equivalent users generate the same request types for the same places, and the same interarrival times between pairs of request types generated for the same pairs of places.

A hierarchy between the relations is expressed in Fig. 15. Relations E_2, E_3 are not comparable.

In order to simplify the presentation, we suppose that we consider one of two cases:

1. there are relations E_1, E_2, E_4 only,
2. there are relations E_1, E_3, E_4 only.

Hence, in the sequel we use notation $E_{2(3)}$ which means E_2 or E_3.

Let us consider a set $\tilde{U}$ of users. The relation E_i ($i \in \{1, 2, 3, 4\}$) determines a partitioning of the set $\tilde{U}$ into equivalence classes.

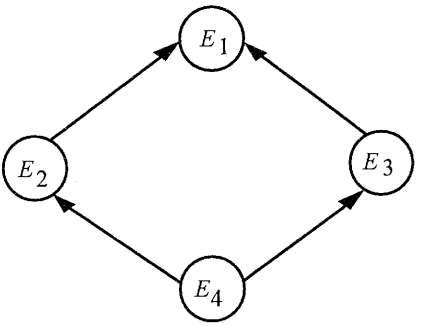


Fig. 15. Hierarchy resulting from equivalence relations E_i ($i \in \{1, 2, 3, 4\}$).

6.2.1. Parallel Compositions of Users Without Interactions

Now we consider the *addition composition.*

Let us analyse the equivalence class U^* resulting of such users from the set $\tilde{U}$ that are E_1-equivalent. The users $U_k \in U^*$ are described by an identical Petri net $N = \langle P, T, F, M_0 \rangle$. Such users (nets) can be folded: one upon another, and if it is necessary, they can be distinguished by colours. Hence, one coloured Petri net (Jensen, 1983; 1991) only is required to represent a result of parallel composition of users. We assign one unique colour to all users from one equivalence class of relation E_4. The collection of colours creates a colour class. (The terminology used in our paper is based on (Chiola *et al.*, 1991)). The colour class is attached to each place and each transition. The colour class of the place determines the different colours of tokens that are allowed in that place, whereas the colour class of the transition represents the different possibilities for firing these transitions. In the coloured Petri net obtained as a result of addition composition of users without interactions, an arc between the place and the transition is labeled with an identity function mapping the colour class of a transition on the same colour class of the place. When firing a transition, one colour is chosen in the transition colour class and the same colour is assigned to input and output arcs. In order to fire the transition, there should be at least one token of chosen colour in each input place. If the marking satisfies the firing condition, one coloured token is removed from each input place and one token of the same colour is put to each output place.

The hierarchy of relations E_i ($i \in \{1, 2, 3, 4\}$) is a foundation for hierarchy in colourization. The users that are E_1-equivalent are modeled by one coloured Petri net. The colour class C of this net is partitioned into static subclasses. A static subclass D_i contains colours that are used to colour the users that are $E_{2(3)}$-equivalent. An element $c \in D_i$ is a common colour of the users that are E_4-equivalent.

Now we define addition composition of many E_1-equivalent users using the above hierarchy in colourization.

Definition. Let U^* be the set of such elementary users from the set $\tilde{U}$ that are E_1-equivalent and described by a Petri net $N = \langle P, T, F, M_0 \rangle$. The set U^* is partitioned according to the relation $E_{2(3)}$ into a family of subsets $S^* = \{S_1, ..., S_i, ..., S_n\}$.

The users from the set S_i are characterized in the following way:

1. If the family S^* is obtained by E_2, then the set S_i is distinguished by the request type function f_i such that
$$(\forall U_k \in S_i)(f_k = f_i),$$
2. If the family S^* is obtained by E_3, then the set S_i is distinguished by the refinement R_i such that
$$(\forall U_k \in S_i)(R_k = R_i)$$

Each subset $S_i \in S^*$ is partitioned according to the relation E_4 for the family $S_i^* = \{S_{i1}, ..., S_{ij}, ..., S_{in_i}\}$. The number of E_4-equivalent users in the class S_{ij} is given by $card(S_{ij})$.

The users from the set S_{ij} are characterized by the function f_{ij} and refinement R_{ij} such that

$$(\forall\, U_k \in S_{ij})(f_k = f_{ij} \wedge R_k = R_{ij})$$

The user obtained by addition composition of the set U^* of users without interactions is an *interpreted coloured Petri net*

$$U = \langle P, f, T, C, F, \bar{M}_0, R\rangle$$

where $\langle P, T, F\rangle$ is an unmarked Petri net resulting of the net $N = \langle P, T, F, M_0\rangle$, the latter describing the users that are E_1-equivalent; for transitions $t \in T$, an infinite server semantics is assumed (unbounded number of simultaneous firings of the transiton can take place at a given time instant), C – colour class that is the colour domain for all transitions and all places.

C is partitioned into n (n is the number of sets in the family S^*) static subclasses D_i ($i \in \{1, 2, ..., n\}$) e.g.

$$C = \bigcup_{i=1}^{n} D_i$$

$$\Big(\forall\, i \in \{1, 2, ..., n\}\Big)(D_i \neq \emptyset)$$

$$\Big(\forall\, i,\ j \in \{1, 2, ..., n\}\Big)(i \neq j \Longrightarrow D_i \cap D_j = \emptyset)$$

The number of colours in the static subclass D_i is equal to n_i (n_i is the number of sets in the family S_i^*). Therefore, the number of colours in the class C is equal to

$$\sum_{i=1}^{n} n_i$$

Colour function is the identity function for all arcs.

Let S^+ be the family of subsets S_{ij} obtained from the set U^* of E_1-equivalent users according to the relation E_4. There exists a one-to-one mapping g such that

$$g : S^+ \longrightarrow C$$

i.e. the users from one equivalence class of relation E_4 are of the same unique colour.

The request type function f is defined as:

$$f : P_a \times C \longrightarrow \bigcup\{R_{ij} \ : \ S_{ij} \in S^+\} \cup \{\theta\}$$

where P_a is the set of active places $\forall U_k \in U^*$, R_{ij} is the set of request types $\forall U_k \in S_{ij}$, and this function is such that

$$(\forall p_k \in P_a)\Big(f(p_k, c_l) = f_{ij}(p_k)\Big)$$

where $c_l = g(S_{ij})$, f_{ij} is the request type function $\forall U_k \in S_{ij}$.

Let $\forall U_k \in U^*$ the initial marking M_0 be such that $M_0(p_k) = 1$, $M_0(p_n) = 0$ for the other places.

The initial marking $\bar{M}_0$ of the user U obtained by addition composition of the users from the set U^* is given by:

$$\bar{M}_0 : P \times C \longrightarrow \{0, 1, ...\}$$

and it is such that

$$\bar{M}_0(p_k, c_l) = card(S_{ij})$$

where $\bar{M}_0(p_k, c_l)$ is the number of tokens of colour c_l in the place p_k, $c_l = g(S_{ij})$, and additionally, $\bar{M}_0(p_n, c_l) = 0$ for all places $p_n \neq p_k$.

The refinement R is defined analogously as the request type function is determined. ■

In a similar way we can define the *fork-join composition.*

6.2.2. Parallel Compositions of Users With Interactions

In this case, a general formal definition cannot be given, because of the possible variety of interactions between users. In order to illustrate how complicated the parallel composition could be, we present a specific interaction.

Example 5. Let U_i be a telephone user with behaviour as described in Fig. 16. The meaning of symbols in this figure is as follows.

Request types: SB – silence begin; TB – talkspurt begin; HU – hang up; C – call.

Probabilities: q – probability of continuation of the talk; r – probability that the next activity of a user with marked HU place will be initiating a new call (not being called).

External events: S – successful call attempt; R – resignation from further call attempts; N – next call attempt.

Firing times: τ_1 – total duration of one call attempt; τ_2 – duration of silence period after successful call attempt, and before first talkspurt period; τ_3 – duration of talkspurt period; τ_4 – duration of silence period; τ_5 – duration of hanging up period; τ_6 – duration of time after hanging up and before the calling; τ_7 – duration of time after hanging up and before being called.

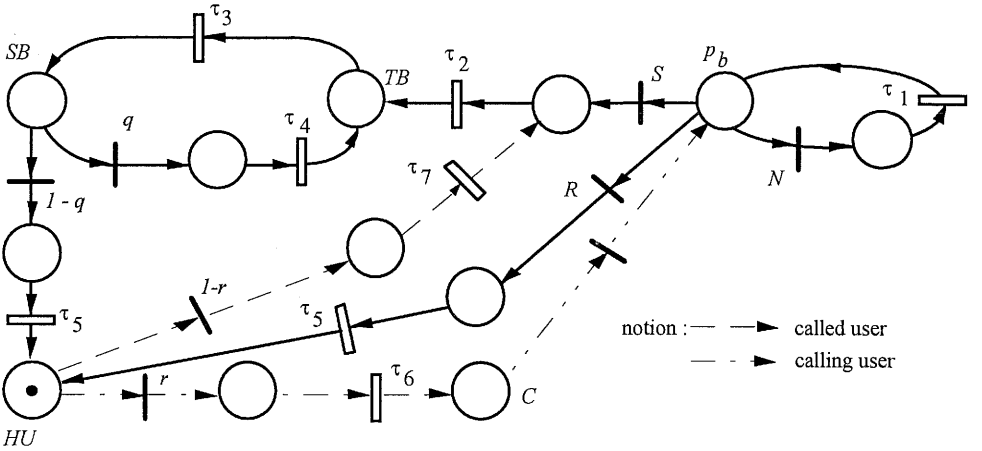


Fig. 16. Model for a telephone user (Example 5).

Let the *addition composition* be performed on the set U^* of users that are the same as the user U_i. We want to model the pairs of users that are talking together at a given time instant.

The coloured Petri net representing a complex user is illustrated in Fig. 17. U^* is the set of users identified by colours, e.g. telephone number corresponds to one colour. U^* is treated as the colour class. All users have identical behaviour pattern. Hence, it is not required to use colours in order to distingish their performance characteristics. The colour domains are assigned to each place and each transition. The colour domains have two forms: U^* or U^{*2} (i.e. Cartesian product $U^* \times U^*$). E.g., if there is the token with colour $\langle U_i, U_j \rangle$ in the place with request type TB being generated (colour domain for this place is U^{*2}), this means that in the talk between the users U_i and U_j (U_i is calling) the talkspurt has begun.

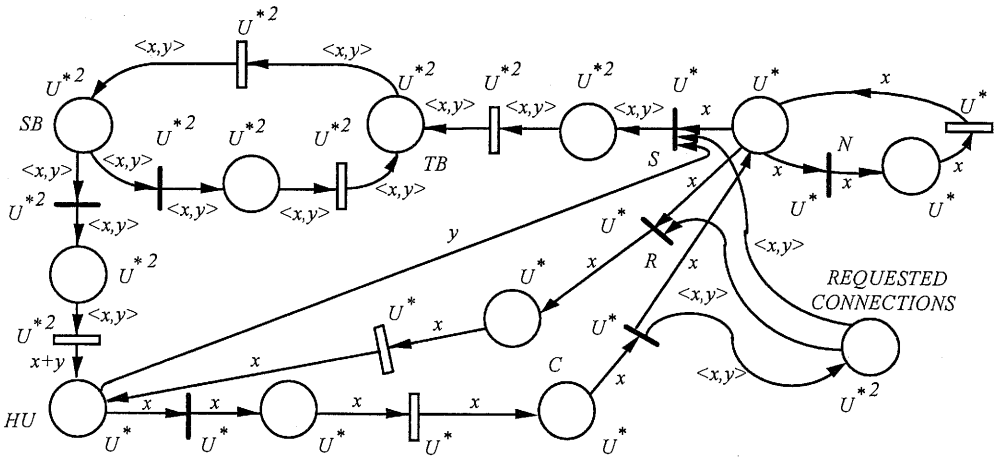


Fig. 17. Addition composition modelling a pair of communicating telephone users (Example 5).

An additional place "Requested connections" contains pairs of users $\langle U_i, U_j \rangle$ with the property that user U_i is calling user U_j. In inscriptions on arcs (colour functions), x is a variable, whose value is the colour of the calling user and y is a variable, whose value is the colour of the called one. ■

In the example, after parallel composition has been performed, not only there is an additional place but the model of single user is changed (a path from the place with request type HU is directed to the transition – not to the place). This shows the difficulties in finding a general definition which would cover all kinds of interactions.

7. Refinement and Abstraction

When we apply the description technique introduced in sections 4 to 6, the result will be a *concrete load model*, corresponding to some specific level of detail in modelling the environment observed. As in modelling studies, the variation in the level of detail for a given model is quite a common task, we would like to have some support of this task inherent in the description technique. An example for such a refinement would be the need for a refined load model as a consequence of a refined system model.

So, let us look at the kinds of refinements typically desirable, when modelling communication system load:

(1) *a set of lower level users may become visible* (with new types of users), e.g. when increasing the granuality of users in modelling a fixed communication system environment;

(2) *additional request types* may have to be considered (e.g. when modelling in a more detailed way the interactions between users and communication system);

(3) *additional request type attributes* for a given request type may have to be considered (e.g. more detailed modelling of communication resource requirements);

(4) *attribute values for requests* may have to be reflected more precisely (e.g. in using a greater granuality in the set of possible values);

(5) *values* may have to be determined in a *less stochastic* (e.g. in a deterministic) way for request interarrival times or attribute values.

Our load description technique offers the following support to cover these refinements:

- cf.(1): this refinement is straight-forward by specifying Petri nets for lower level users;
- cf.(2): colourization can be used to refine request types or additional places may be introduced to generate additional requests (perhaps of new types), though the number of users is kept constant;
- cf.(3)/(4): these refinements can be achieved by modifying the co-domain of request type function;
- cf.(5): random variables have to be replaced by traces to produce more deterministic user behaviour.

Evidently, there exist limits of refinement in our load modelling approach. These limits are given e.g. by our assumptions (cf. section 2). An additional important

limitation results from our restriction to characterize user behaviour just by modelling request preparation/generation and waiting of users for system reactions (without considering social, economic, institutional and other relationships between users).

Besides refinement of a given load model, abstraction may become necessary and therefore, will have to be supported by a description technique. Abstraction in this context means, that we would like to aggregate a given set of users to a smaller number. The equivalence relations E_1, E_2, E_3, E_4 (as introduced in section 6) enable us to support such aggregation by combining a number of E_i- equivalent users, $i \in \{1, 2, 3, 4\}$, into one user in order to get a more compact description of a communication system environment. Simulation experiments and in some cases mathematical analyses can be used to obtain the performance characteristics of aggregated users (e.g. characteristics such as sequence of requests resulting from overlying the request streams of a large number of individual users, which — as a result of abstraction — become indistinguishable).

8. Possibilities and Limitations in Applying the Description Technique

The final goal of a formal description technique is to support a specification of concrete models. So, which are the basic possibilities of producing such load models based on our description technique?

a. Specification of *elementary users*:

Elementary users (cf. section 4) can be considered as basic building blocks in describing a given communication system environment according to our approach, in particular elementary users

- allow us to express the behaviour of a user as observed at a given interface (sequence of requests and precise characterization of requests with respect to resource demands); this makes user models "portable" to some extent (i.e. they may be combined with systems models of different communication systems, provided these systems have the same interfaces, which is quite common in innovative communication systems);
- are sequential (the reason for this being that an elementary user may model a person working sequential in general or it may reflect a communicating sequential process);
- give us a considerable flexibility to get descriptions of more complex behaviour of users (directly based on elementary user models), e.g. by characterizing partial behaviour and composing such submodels to get description of the overall behaviour (by means of sequential composition and choice between behaviour variants); it is evident that such compositions are important, e.g. for describing multimedia-applications;
- give us the advantage that dependencies of users on system reactions are modeled explicitly (which takes into account blocking states for users and allows one to clearly separate phases in user behaviour, where time intervals between request generation are system-independent and those phases, where these times

depend on delays occuring before system reactions); the reader should note, that in communication system modelling, system dependencies of users are highly relevant, in general, because of system bottlenecks or dependencies on the communication partner implying temporary blocking situations for users (e.g. blocking as a consequence of a synchronous SEND request, of a closed window in window based flow control (Tanenbaum, 1989 etc.).

b. Specification of a *set of users*:

Parallel compositions, such as introduced in section 6, give us the possibility of aggregating e.g. independent users. Again, it is evident that aggregation of users is relevant, in particular, for service-integrated networks, where users of quite different types coexist in the same communication system environment (e.g. voice, text, and data communication users).

c. *Refinements* in the model of environment:

As discussed in section 7, refinement is possible even to the extent that the resulting load models may be used directly in simulation experiments (as, in this approach, load models can be tailored to be sufficiently close to reality for performance evaluation studies).

d. *Measurements* in load models to support performance evaluation:

As we want to apply load models primarily within performance evaluation studies, let us characterize the kind of measurements which are directly supported by our description method:

- Measurements can be embedded in the user model per se (e.g. delays between token arrivals in different places, corresponding to duration of sessions, think times of users, request interarrival times, etc.).
- Events can be observed at system interface(s), so one could measure e.g.
  - the number of events in an observed interval (corresponding e.g. to offered load and/or system throughput),
  - times between corresponding events of the same or different types (corresponding e.g. to reaction time of system, delay of data units in the communication system, etc.).

Limitations — besides the assumptions of section 3 evidently having to be fullfilled — concern the following:

- models exclusively reflect request generation of users (i.e. user behaviour is mainly described with respect to resource demands and some communication aspects of users);
- during blocking states, no preparation of new requests is allowed (which limits the possibilities of specifying request interarrival times); this limitation corresponds to the assumption that preparation time for the requests is supposed to be independent of durations of blocking situations.

In general, we can make use of the expressive power of the class of coloured Petri nets with time factor, which we applied as the basis of our description method.

As stated earlier, the goal of this paper has been to present a methodology for load modelling and a description technique to specify load models. Therefore, no complex load models have been introduced, though some elementary examples have been used to illustrate how models can be specified in principle. Future work will focus on the usage of the description technique in defining concrete load models (including their parametrization) and evaluating them according to the possibilities of model applications as already summarized in section 2.

We hope that the proceeding and the description technique for load modelling introduced in this paper will turn out to be useful in solving the important problems of specifying load models for innovative communication systems (in particular for high-speed networks serving multimedia-applications).

Acknowledgements

The authors would like to thank Prof. Dr. M. Jantzen and Prof. Dr. F. Vogt for some valuable discussions and hints to revise an earlier draft of this paper. Some support of J.J. Kim to this work is appreciated too.

References

Abeysundara B.W. and Kamal A.E. (1991): *High-speed local area networks and their performance. A survey.* — ACM Computing Surveys, v.23, pp.221–264.

CCITT: Recommendation I.121 (1989): *On the Broadband Aspects of ISDN.* — CCITT Blue Book, v.III, Fascicle III.7, Geneva.

Chiola G., Dutheillet D., Franceschinis G. and Haddad S. (1991): *Stochastic well-formed coloured nets and multiprocessor applications.* In: Jensen K., Rozenberg G. (Eds.). — High-Level Petri Nets, Theory and Application, Springer-Verlag, Berlin Heidelberg, pp.504–530.

Courtois P. (1985): *On time and space decomposition of complex structures.* — C. ACM, v.28, No.6, pp.590–603.

Ferrari D. (1984): *On the foundations of artificial workload design.* — Proc. ACM SIGMETRICS Conf. *Measurement and Modelling*, Cambridge, USA, pp.8–14.

Green P.E. (1991): *Exploiting photonic technology for gigabit computer networks.* In: Danthine A., Spaniol O. (Eds.). — Proc. IFIP Conf. *High Speed Networking*, Berlin, Germany, pp.3–14.

Holzman-Kaiser U., Moeller E., Schurmann G. and Weiss K.H. (1991): *A guide for advanced broadband multimedia applications.* In: Telekommunikation und Multimediale Anwendungen der Informatik. — Informatik-Fachberichte, v.293, Springer-Verlag, Berlin Heidelberg.

Jensen K. (1983): *High-level Petri nets.* In: Pagnoni A., Rozenberg G. (Eds.), Applications and Theory of Petri Nets. — Informatik-Fachberichte, v.66, Springer-Verlag, Berlin Heidelberg, pp.166–180.

Jensen K. (1991): *Coloured Petri nets: A high level language for system design and analysis.* In: Jensen K., Rozenberg G. (Eds.). — High-Level Petri Nets, Theory and Application, Springer-Verlag, Berlin Heidelberg, pp.44–119.

Killat U., Muscate H.A. and Wolfinger B. (1988): *A performance analysis of the IEEE 802.5 token ring protocol.* — Philips J. Research, v.43, No.5/6, pp.532–553.

Kleinrock L. (1976): *Queueing Systems.* — Computer Applications, John Wiley and Sons, New York, v.II.

Kobsa A. (1985): *Modelling of user in dialog systems.* — Informatik-Fachberichte, v.115, Springer-Verlag, (in German).

Kuehn P. (1989): *From ISDN to IBCN (Integrated Broadband Communication Network).* In: Ritter G.X. (Ed.). — Proc. IFIP 11th World Computer Congress, Amsterdam, North-Holland, pp.479–486.

Magott J. (1984): *Performance evaluation of concurrent systems using Petri nets.* — Information Processing Letters, v.18, No.1, pp.7–13.

Magott J. (1985): *Performance evaluation of systems of cyclic sequential processes with mutual exclusion using Petri nets.* — Information Processing Letters, v.21, pp.229–232.

Milner R. (1980): *A calculus of communicating systems.* — Lecture Notes in Computer Science, v.92, Springer-Verlag, Berlin.

Pawlita P.F. (1988): *Two decades of data traffic measurements: A survey of published results, experience and applicability.* — Proc. 12th Int. Telecommunication Conference, pp.230–238.

Popescu-Zeletin R. (1990): *From broadband ISDN to multimedia computer networks.* — Computer Networks and ISDN Systems, v.18, 1989/90, pp.47–54.

Pujolle G. (1991) (Ed.): *High-Capacity Local and Metropolitan Area Networks.* — Architecture and Performance Issues. Springer-Verlag, Berlin.

Tanenbaum A. (1989): *Computer Networks.* — Englewood Cliffs: Prentice-Hall.

Wolfinger B. and Kim J. (1990): *Load measurements as a basis for modelling the load of innovative communication systems with service-integration.* — Proc. 2nd IEEE Workshop on *Future Trends of Distributed Computing Systems*, Kairo, Egypt, pp.14–21.

Received: March 30, 1994
Revised: August 28, 1994