TM-0677

A Study on Verification of Service Specification in Communication Software Development

by

K. Shibata, Y. Ueda, S. Yuyama
& H. Hasegawa (Oki)

February, 1989

THE TRANSACTIONS OF THE IEICE, VOL. E **71**, NO. **12** DECEMBER **1988**

PAPER (Special Issue on CAS Karuizawa Workshop)

A Study on Verification of Service Specification in Communication Software Development

Kenji SHIBATA†, Yoshihiro UEDA†, Satsuki YUYAMA† *and* Haruo HASEGAWA†, *Members*

SUMMARY This paper gives a verification method of a service specification in a communication system by utilizing Petri net. We define a service specification and clarify the feature of the specification. And we discuss a relation between a service specification and Petri net which represents the specification. Furthermore, this paper describes some kinds of verification for the specification and shows several examples. We are now developing a software support system-EXPRESS (EXPeRt system for ESS). EXPRESS designs automatically the service specification from user's requirements. In EXPRESS, each requirement is converted into ISG (Individual Service Graph) and all ISGs are integrated into TSG (Total Service Graph). TSG is the final service specification. ISG and TSG are described by using Petri net. Petri net is used for modeling a communication protocol and an asynchronous system. A service starts from an idle state and returns to the same idle state in a communication system. This means that a service specification should have a t-invariant. And a t-invariant which can not be decomposed is called a prime service. The followings are needed for verification of the specification: verification of ISG, detection of ISG from TSG and detection of other prime services than ISG. These are verified by utilizing an analysis of Petri net

1. Introduction

In recent years, user's requirements have become various and diversified in a communication system, and it takes many hours of expert engineers to develop communicaion software, especially at the specification phase. It is important to design the specification from user's requirements accurately because the specification is a fundamental document of the next phases in communication software. At present, however, only some expert engineers, who have long been engaged in developing switching software, can specify user's requirements, that is, can understand user's requirements and design the service specification without inconsistency.

We have been developing a support system EXPRESS (EXPeRt system for ESS) which designs automatically the service specification for a communication system[4]. EXPRESS understands user's requirements written in a natural language and designs the specification. Besides, EXPRESS verifies the specification. The service specification has been verified by expert engineers based on many years' experience. But in case that the service specification is designed automatically, it is important that the service specification is verified automatically. This system may decrease the man-hours of engineers. The representation of the service specification holds the key to the method of verifying the specification. In EXPRESS, the service specification is described by using Petri net[3]. Petri net is said to be one of the best candidates that can be used as models of communication protocols. And Petri net is widely used for verification of asynchronous systems. Especially a t-invariant in a Petri net model is said to be effective to analysis of practical systems. A t-invariant is also utilized from the feature of service in order to analyze the service specification in a communication system, and the specification is verified logically and topologically by utilizing an analysis of Petri net[5].

This paper gives an outline of EXPRESS. And it clarifies the feature of Petri net in EXPRESS and the relation between a service specification and Petri net. Then it describes the method of verifying the service specifcation and shows some examples.

2. Outline of EXPRESS

2.1 System Configuration

Figure 1 shows a system configuration of EXPRESS. The system consists of requirement understanding subsystem, specification integrating subsystem, prototyping subsystem and knowledge base.

The functions of them are as follows:

(1) Requirement understanding subsystem

When users input an ambiguous requirement about services of a switching system in a natural language, this subsystem understands the requirement by utilizing the knowledge in the knowledge base. This subsystem transforms each requirement into an individual specification. When the user's requirement is inconsistent with the knowledge base, the inconsistency is resolved interactively.

(2) Specification integrating subsystem

All of the individual specifications are integrated into a final specification by this subsystem. Each of the individual specifications is fragmentary and they may be

Manuscript received July 4, 1988.
Manuscript revised September 5, 1988.

† The authors are with Telecommunications Division, Oki Electric Industry Company, Ltd., Tokyo, 108 Japan.

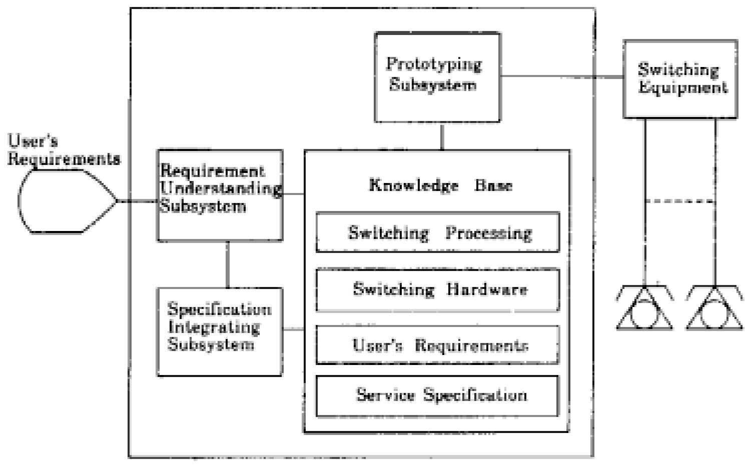


Fig. 1 System configuration.

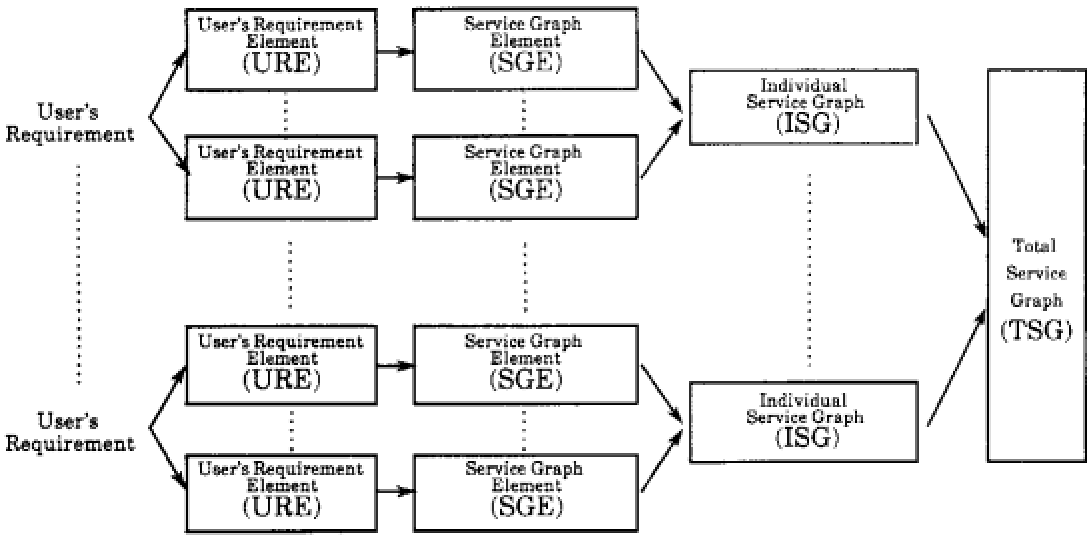

User's Requirement
User's Requirement Element (URE)
Service Graph Element (SGE)
Individual Service Graph (ISG)
Total Service Graph (TSG)

Fig. 2 Design process.

inconsistent with each other. When some inconsistencies are made in this process, they are also resolved interactively.

(3) Prototyping subsystem

This subsystem controls a switching equipment according to the specification stored in the knowledge base and examines whether the specification satisfies user's requirements semantically or not.

(4) Knowledge base

The service specification which was designed and the knowledge required to design the specification are stored in the knowledge base. The knowledge is related to switching processing, switching hardware and user's requirements.

2.2 Specification Expression

Figure 2 shows the process of designing the specification without inconsistency or ambiguity.

The process in which experts design the specification from user's requirements can be considered as the conversion of the service expression representing the user's requirements into that representing the specification. Each of service expressions is explained below. Figure 3 shows an example of user's requirements about PBX (Private Branch eXchange) : a standard call (the caller first goes on-hook) in a station-to-station dialing.

(1) URE (User's Requirement Element)

URE describes the terminal operation which means an operation for a telephone and the terminal state changed by the operation from the beginning to the end of a user's requirement. Figure 4 shows an example of URE. URE expression consists of an 'act' part and a 'sres' part. The former means a terminal operation and the latter a terminal state. Each expression consists of a

1. A caller goes off-hook.
   Then the caller hears a dial tone.
2. The caller dials the number of a callee.
   Then the caller hears a ringback tone.
   The callee is rung.
3. The callee goes off-hook.
   Then the caller and the callee talk to each other.
4. The caller goes on-hook.
   Then the callee hears a busy tone.
5. The callee goes on-hook.
   The busy tone stops.

Fig. 3 User's requirement (station-to-station dialing - the caller first goes on-hook).

```
1.  ure([act(off-hook,[agent(caller)])],
        [sres(hear,[object(caller),influence(dial-tone)])])
2.  ure([act(dial,[agent(caller)])],
        [sres(receive,[object(caller),influence(rbt)]),
         sres(receive,[object(callee),influence(rgt)])])
```

Fig. 4 An example of URE (an internal expression of URE corresponds to 1 and 2 in Fig. 3).

verb and cases of the verb such as an agent.

(2) SGE (Service Graph Element)

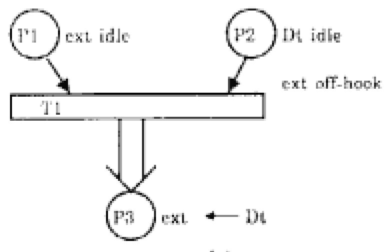
URE is converted into SGE which is described in Petri net. In SGE, a resource such as a telephone or a dial tone is expressed as a token, an action as a transition and a state as a place. Each unit of SGE consists of a present state, an action and a next state. A present state is represented as a set of input places and a next state as a set of output places. A state represents terminal relations each of which means a relation between two terminals. For example, a ringing state represents two relations : a caller is hearing a ringback tone, and a callee receives a bell. A terminal relation is an element of a state. Thus, each place consists of terminal relations. A unit of SGE can be treated as an element of a service graph. Figure 5 shows a unit of SGE. Figure 5 (a) shows an internal expression of SGE and Fig. 5(b) shows a representation of SGE by Petri net. A representation of Petri net in Fig. 5(b) corresponds to a representation of SGE in Fig. 5(a). Places and transitions in this example are as follows :

p1 : place ([rel (token (caller, ext), idle, nil)])
p2 : place ([rel (token (dial-tone, dt), idle, nil)])
p3 : place ([rel (token (caller, ext), receive,
        token (dial-tone, dt))])
t1 : tr (token (caller, ext), off-hook, nil)

The input function values which mean the number of arcs connecting a place with a transition, $\alpha(p, t)$, and the output function values which mean the number of arcs connecting a transition with a place, $\beta(p, t)$, are as

```
sge(spp([place([rel(token(caller, ext), idle, nil)]),
         place([rel(token(dial-tone, dt), idle, nil)])]),
    tr(token(caller, ext), off-hook, nil),
    snp([place([rel(token(caller, ext), receive,
                token(dial-tone, dt))])]))
```

(a)



(b)

Fig. 5(a) An example of SGE (an internal expression of SGE corresponds to 1 in Fig. 4).

(b) Representation of SGE in petri net (this representation corresponds to SGE in Fig. 5(a)).

```
place(Pt1, gpl([rel(token(Ttk1, ext), idle)]),
           spt([Tt4, Tt5]),
           snt([Tt1, Tt2]))
transition(Tt1, Arc1, offhook, transit([Arc1, Arc2]),
                spp([Pt1, Pt6]),
                snp([Pt2]))
arc(Arc1, in(Ttk1), out(Ttk3), str([]))
```

Fig. 6 An example of ISG (an internal expression of ISG).

follows :

$$\alpha(p1, t1)=1,\ \alpha(p2, t1)=1,\ \alpha(p3, t1)=0,$$

$$\beta(p1, t1)-0,\ \beta(p2, t1)=0,\ \beta(p3, t1)=2$$

A present state is expressed as a 'spp' part, an action as a 'tr' part and a next state as a 'snp' part. A terminal relation is expressed as a 'rel' part. An action consists of a token which represents a terminal to be operated, a type of an action and a token which represents a terminal influenced by the action. A type of a token is an element of a token. For example, if a type of a token is 'ext', the token corresponds to a telephone. If a type of token is 'dt', the token corresponds to a dial-tone.

(3) ISG (Individual Service Graph)

ISG is a service specification which is designed by integrating SGEs. A single state corresponds to a place and an action to a transition, and the connection between a place and a transition is clarified. Each ISG consists of places, transitions and arcs. Figure 6 shows an example of ISG. A state is expressed as a 'gpl' part, a connection to transitions as 'spt' and 'snt' part, and a connection to places as 'spp' and 'snp' part. An input token which means a token contained in an input place is expressed as an 'in' part and an output token which means a token contained in an output place as an 'out'

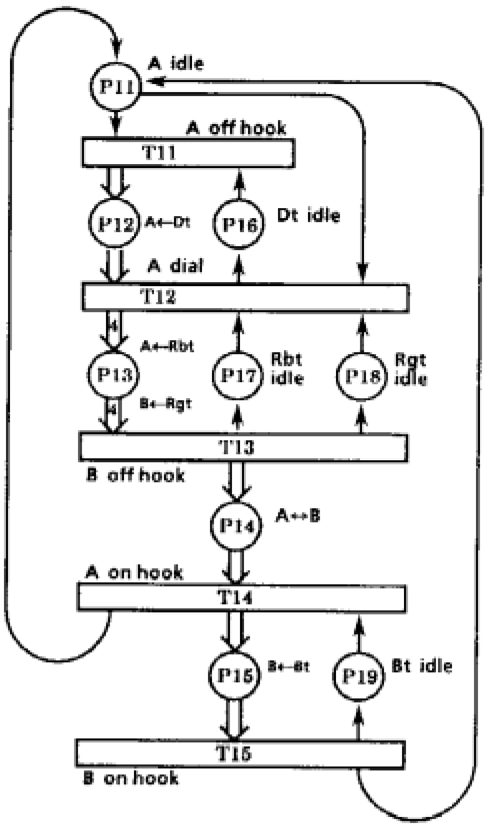


Fig. 7 Representation of ISG in petri net (standard call in a station-to-station dialing).

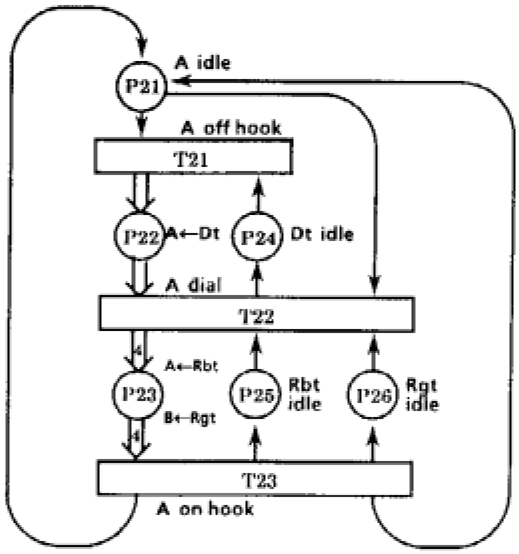


Fig. 8 Representation of ISG in petri net (a caller goes on-hook while hearing a ringback tone).

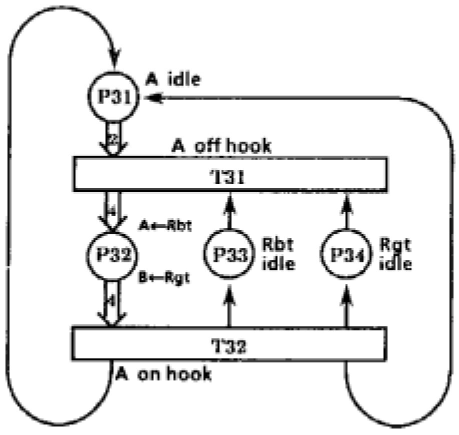


Fig. 9 Representation of ISG in petri net (a caller goes on-hook while hearing a ringback tone in hotline).

part. An attribute of transition, which points out a token that enables the transition to fire, is expressed as an 'atr' part. Figure 7, Fig. 8 and Fig. 9 show a case that a standard call in a station-to-station dialing, a case that a caller goes on-hook while hearing a ringback tone and a case that a caller goes on-hook while hearing a ringback tone in hotline, respectively.

(4) TSG (Total Service Graph)

TSG is a final service specification which is designed by integrating all of fragmentary ISGs. The expression of TSG is as same as that of ISG. Figure 10 shows TSG which is created from 3 ISGs.

2.3 Process of Designing Service Specification

In this section, we describe the process of designing the service specification. This process consists of 4 phases; User's requirement-URE conversion, URE-SGE conversion, SGE-ISG conversion and ISG-TSG conversion. Each conversion is explained below:

(1) User's requirement-URE

Each sentence of user's requirements is converted into URE. This process consists of morphological analysis, syntactic analysis and semantic analysis. In this process, the system finds syntactic errors which may be included in a sentence and deletes them. URE is represented in a unified clause such as shown in Fig. 4.

(2) URE-SGE

URE is converted into SGE in accordance with a sequence of each clause. In this process, a transition is made from an action of each clause, tokens are made from resources in the clause, input places are made from the set of present places and output places are made from the states changed as a result of the action. In case that the token in the input place does not appear in any output place and vice versa, the system generates an idle place which means an idle state of the token.

For each clause, this conversion consists of 4 phases explained below. When this conversion is carried out for all clauses, conversion of URE into SGE is completed.

① Create Tokens Each token is created from each object that appears in URE, unless the token has been already created. This object represents a resource such as a telephone and a dial tone.

② Create Transitions A transition is associated with an action in URE. A transition consists of an action and a terminal which is operated. A token which corresponds to the terminal is selected from tokens created in step ①.

③ Create Input Places An input place is created from a set of output places created from a previous clause of URE. If there does not exist any place that contains tokens which are created in step ① in the set, the system generates an idle place which means an idle state.

④ Create Output Places An output place is created from a changed state after an action in URE. If there exists a token which appears in an input place and not in an output place, the system generates an idle place where a token exists.

(3) SGE-ISG

SGE is converted into ISG which is described in Petri net. This conversion is made by integrating all units of SGE. Places, transitions and arcs in each unit of SGE are integrated with those of other units of SGE, respectively.

(4) ISG-TSG

All ISGs each of which is fragmentary are integrated into TSG.

This conversion consists of 3 phases as follows:

① Search a place in TSG which corresponds to a place in ISG. A place in ISG corresponds to a place in TSG if the elements of each place are the same. The elements are the same if the types of tokens are the same and the types of relations are the same. If there exists in TSG any place that corresponds to the place in ISG, go to ②. If not, add the place to TSG.

② Check the transition whose input place is the place in ISG with that in TSG. A transition in TSG corresponds to a transition in ISG if the types of actions are the same and the types of terminals are the same. And check the output place of the transition in ISG with that in TSG. If there exist both a transition and a place in ISG which correspond to those in TSG, go to ③. If not, add the transition and the place to TSG.

③ Do nothing when unification between ISG and TSG is completed. Return to ①.

3. Basic Terminology[1)-(3),(7),(8)]

This section presents the terminology and some results.

[Definition 1] Petri net is a quadruple $C=(P, T, \alpha, \beta)$, where:

(1) P is a finite and non-empty set of places.

(2) T is a finite and non-empty set of transitions.

(3) $P \cap T = \phi$

(4) α: $P \times T \rightarrow N$, is an input function.

(5) β: $T \times P \rightarrow N$, is an output function.

(N represents a set of non-negative integers: 0, 1, 2, …)

We can represent Petri net by bipartite graph. Places are represented by circles and transitions by rectangles. Places and transitions are connected by oriented arcs. The input function values are associated to arcs connecting a place with a transition. The output function values are associated to arcs connecting a transition with a place.

[Definition 2] We adopt the following notations:
For $t \in T$, ${}^{\circ}t = \{p \in P | \alpha(p, t) \neq 0\}$ and $t^{\circ} = \{p \in P | \beta(p, t) \neq 0\}$. We call ${}^{\circ}t(t^{\circ})$ a set of input (output) places of t. Let ${}^{\circ}T = \{{}^{\circ}t\}$, $T^{\circ} = \{t^{\circ}\}$.

[Definition 3] The matrix of Petri net is described below. We define D^- and D^+ which represent and input function and an output function of transitions, respectively. Each matrix consists of m rows which correspond to transitions and n columns which correspond to places. A component of j-th row and i-th column of D^- means a multiple degree of arcs which go from i-th place to j-th transition, and a component of j-th row and i-th column of D^+ means a multiple degree of arcs which go from j-th transition to i-th place. When n-dimension vector μ, which is called marking, represents the number of tokens which exist in each place, the j-th transition T_j fires if the following expression is realized.

$$\mu \geqq e[j] \cdot D \qquad (1)$$

$e[j]$ is a m-dimension vector in which the j-th component is 1 and other components are all 0. When T_j fires in the marking μ_0, the new marking is as follows.

$$\mu = \mu_0 + e[j] \cdot D \qquad (2)$$

where $D = D^+ - D^-$

Therefore, when a firing sequence $\sigma = T_{j_1}, T_{j_2}, \cdots, T_{j_k}$ fires, we have a new marking.

$$\mu = \mu_0 + (e[j_1] + e[j_2] + \cdots + e[j_k]) \cdot D$$
$$= \mu_0 + f(\sigma) \cdot D$$

where $f(\sigma) = e[j_1] + e[j_2] + \cdots + e[j_k]$

[Definition 4] A vector I is a t-invariant of C if $I \cdot D = 0$.

[Definition 5] A marked Petri net is a pair (C, μ_0) where C is Petri net and μ_0 is an initial marking. A place is said to be k-bounded if the number of tokens in that place cannot exceed an integer k. A marked Petri net is said to be k-bounded if all places are k-bounded. A marked Petri net is said to be live if for every transition t and every marking μ, there exists a firing sequence from μ containing t.

[Definition 6] T-base of Petri net satisfies the next properties:

(1) It is a set of t-invariants.

(2) T-invariants in t-base are linearly independent of each other.

(3) All t-invariants of Petri net are linear combination of elements of t-base.

[Definition 7] Petri net is said to be consistent if there exists a t-invariant I whose components are all non-negative.

A minimal consistent net satisfies the next properties:

(1) A net is consistent.

(2) Any subset of transition does not satisfy property (1).

[Definition 8] Petri net is said to be t-complete if ${}^{\circ}T = T^{\circ}$. A t-complete element is a subnet which is determined by X, if X satisfies $X \subseteq T$ and ${}^{\circ}X = X^{\circ}$.

[Definition 9] Petri net is said to be rigidly connected if it satisfies the next properties:

(1) Any place p has one input transition and one output transition at least.

(2) Any subnet, that is made from the subset of transitions, does not satisfy property (1).

[Definition 10] A minimal consistent net has only one t-invariant and the components of the t-invariant are all non-negative. Structurally it satisfies $P = {}^{\circ}T = T^{\circ}$.

4. Expression of Service Specification

Petri net has been used in order to model and analyze dynamic systems, and the feature of Petri net is to simplify verification and analysis. Petri net is considered to be effective to specification representation because the specification can be easily added or modified by describing it in Petri net.

We discuss the feature of the representation used in this system. And we clarify the feature of Petri net in EXPRESS and the relation between a service specification and Petri net which represents the specification.

4.1 The Feature of Service Specification

Required conditions for a service specification in switching software are as follows:

(a) All transitions should be able to occur logically in the service specification.

(b) All states that are described in the service specification should be able to exist.

(c) All resources should return to the initial state.

(d) There exists one service at least.

We can say that the service specification is integration of services which users require. A service is defined as follows.

[Definition of Service] The fireable vector $f(\sigma)$ is defined when $f(\sigma)$ satisfies the following conditions.

(a) $f(\sigma)$ is a t-invariant.

(b) Among the firing sequences of the transitions which correspond to $f(\sigma)$, there exists at least one firing sequence which satisfies Eq. (1) in any marking.

When $f(\sigma)$ is a fireable vector, any marking μ can reach from μ itself by firing in sequence the transitions which correspond to $f(\sigma)$. If the tokens exist in only the places which means idle states, $f(\sigma)$ represents one service. In other words, the loop from an idle state to the idle state means a service. The service which is represented by the fireable vector may be linear combination of independent services which can not be made from other services. We call such an independent service a prime service.

4.2 Correspondence between Specification Representation and Petri Net

It is important to investigate the feature of a service specification (ISG, TSG) for analysis of the specification. The features of Petri net which represents the service specification are as follows:

(a) Petri net is live because all transitions may occur.

(b) Petri net is strictly conservative because all tokens represent the resources.

(c) Petri net is t-complete.

(d) In a matrix D, there should exist one t-invariant or more. This means that the next expression should be satisfied.

$$\mathrm{rank}(D) \leqq m \quad 1$$

where m is the number of rows in matrix D.

We discuss further investigations of service specifications and point out that ISG and TSG belong to a sub-class of Petri net.

ISG corresponds to one service. If any transition is removed from Petri net which represents a service, the service will not be realized. Therefore, Petri net which represents ISG is rigidly connected. If the number of firing times of input transitions is equal to that of output transitions for every place in Petri net, there exists t-invariants in the Petri net[1].

Petri net which represents ISG is strictly conservative as shown in Fig. 7. Consequently ISG has necessarily a t-invariant. Furthermore, all the elements of the t-invariant are non-negative because a transition means action in ISG, and Petri net which is created from the transitions decided by the t-invariant is consistent. And the net is a minimal consistent net because the net which represents ISG is rigidly connected as mentioned above. As an independent t-invariant of a minimal consistent net is unique[6], one independent t-invariant exists in ISG. This t-invariant is the vector which represents a prime service.

TSG is designed by integrating ISGs. TSG is t-complete and is a set of minimal t-complete elements each of which represents ISG.

5. Verification of Service Specification

5.1 What Verification is Needed?

User's requirements may be frequently modified and added in a switching system. There is a possibility that the specification includes some inconsistencies, because user's requirements have already included some inconsistencies or they can be made in the process of designing the service specification.

We consider what kind of verification is needed in the process of designing the service specification.
The followings are needed.

(a) ISG is designed correctly.
(b) TSG contains every required service.
(c) No service beyond user's requirement is included in TSG.
(d) The service specification which is designed by adding a partial service is correct. A partial service means a service which is decided by a firing sequence of transitions from one state to the other, both of which are not idle states.

We should examine the followings to verify (a) to (d)

① ISG is a minimal consistent net.
② Every required service belongs to a set of prime services which are obtained from TSG.
③ Any prime service obtained from TSG is included in prime services which mean all ISGs.
④ There exists a minimal t-complete element which includes an idle state and all transitions contained in the added service.

5.2 Example of Specification Verification

(1) Verification of ISG

The matrix D of ISG in Fig. 7 is as follows:

$$D=\begin{array}{c|ccccccccc} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} & P_{19} \\ \hline T_{11} & -1 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ T_{12} & -1 & -2 & 4 & 0 & 0 & 1 & -1 & -1 & 0 \\ T_{13} & 0 & 0 & -4 & 2 & 0 & 0 & 1 & 1 & 0 \\ T_{14} & 1 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & -1 \\ T_{15} & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 \end{array}$$

There exists a linearly independent t-invariant of the ISG because the next equation is satisfied.

$$m-\text{rank}(D)=1$$

The next vector is the t-invariant obtained from the ISG.

$$p=(1\ 1\ 1\ 1\ 1)$$

Consequently, the ISG is designed correctly because

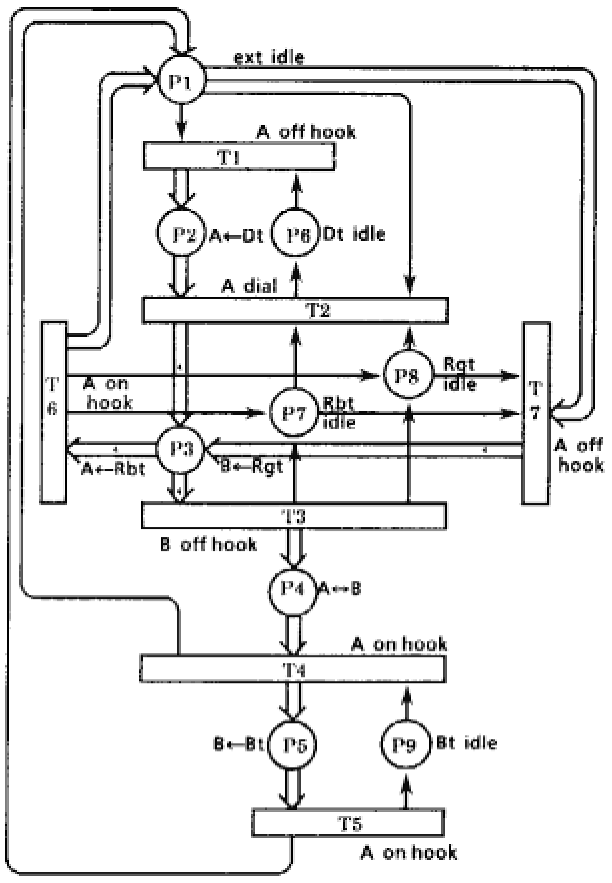


Fig. 10 Representation of TSG in petri net (this TSG is created from 3 ISGs which are shown in Fig. 7, Fig. 8, and Fig. 9).

the Petri net is a minimal consistent.

(2) Conclusion of all prime services of ISG

We can get all prime services by using a method of the t-base construction[2]. The next vectors are the minimal t-invariants which represent prime services obtained from TSG in Fig. 10.

$$s=(1\ 1\ 1\ 1\ 1\ 0\ 0)$$

$$t=(1\ 1\ 0\ 0\ 0\ 1\ 0)$$

$$u=(0\ 0\ 0\ 0\ 0\ 1\ 1)$$

$$v=(0\ 0\ 1\ 1\ 1\ 0\ 1)$$

For example, the vector s means firing transitions: ($t1$, $t2$, $t3$, $t4$, $t5$) but it does not decide a firing sequence. The vector s, t and u mean ISG in Fig. 7, Fig. 8 and Fig. 9, respectively. Therefore, we can verify that TSG includes 3 required services.

(3) Detection of services beyond user's requirement

We have the 4 vectors which represent prime services in TSG. The vector v means a service beyond user's requirement. This new service means a firing sequence ($t7$, $t3$, $t4$, $t5$). If new services are detected, the

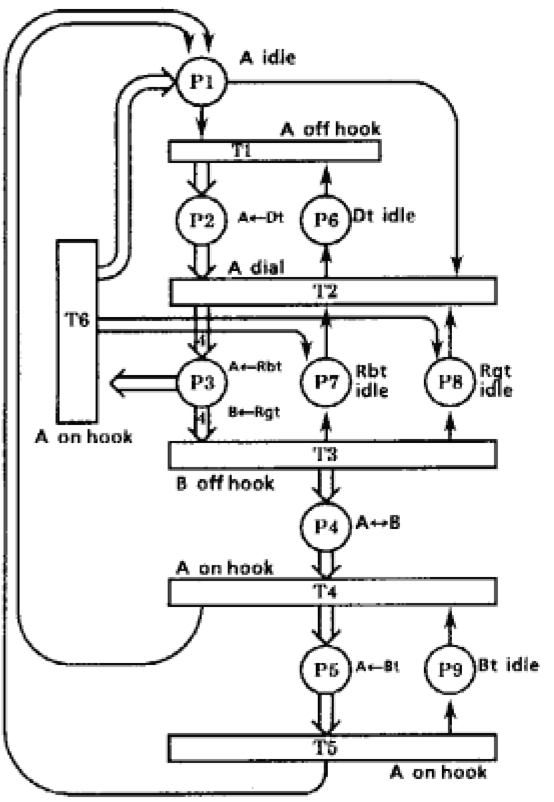


Fig. 11 Representation of TSG in petri net (this TSG is created by adding a partial service to ISG shown in Fig. 7).

system informs users of the service, and ask users whether they are allowed or not.

(4) Verification of a partial service

We assume that a partial service includes a set of transitions: $T_k=\{t_1, t_2, \cdots, t_k\}$. The minimal t-complete element, which contains T_k, can be obtained as follows. Let $T=T_k$. Add to T a transition which has an input place which belongs to T° and not to ${}^{\circ}T$ or T. Obtain ${}^{\circ}T$ and T° on modified T. When $T^{\circ}={}^{\circ}T$, a set which is meant by T and ${}^{\circ}T$ is a minimal t-complete element. In this process, if ${}^{\circ}T=T^{\circ}$ for a net without T_k, remove the transition from T and choose another transition.

We consider the addition of a partial service from a ringing state to an idle state, to the standard call of a station-to-station dialing service. In TSG of Fig. 11, the transition $t6$ is included in the added partial service. The minimal t-complete element including the transition $t6$ becomes the next subset.

$$T=\{t1, t2, t6\},\quad T^{\circ}={}^{\circ}T=\{p1, p2, p3, p6, p7, p8\}$$

Consequently, the added service is proved to be correct.

6. Conclusion

This paper has described design of a service specification and a method of verifying the specification in a communication system.

Each of user's requirements is converted into ISG which represents an individual service and all ISGs are integrated into TSG which represents a service specification. A service specification consists of prime services which can't be made from other services, and a vector which represents a prime service corresponds to a t-invariant in ISG described in Petri net. We consider what kind of verification is needed for a service specification and examine a method of verification. A service specification is verified by utilizing an analysis of Petri net. At present, EXPRESS designs service specifications for certain services from user's requirements and verifies service specification. It is confirmed that Petri net is applicable to the expression for a service specification. Moreover, we are going to apply a method of verification to many services and evaluate it.

Acknowledgement

The authors thank all the members of the EXPRESS project for useful discussions. They would like to be thankful to Dr. R. Hasegawa of ICOT (Institute for New Generation Computer Technology) for many helpful suggestions.

References

(1) K. Onaga and Q.-W. Ge: "Structural analysis of T-invariant of petri nets", Trans. IEICE, **J70-A**, 2, pp. 185-194 (Feb. 1987).
(2) Q.-W. Ge, T. Tanida and K. Onaga: "Construction of a t-base and design of a periodic firing sequence of a petri nets", Proc. of the 8th Mathematical Programming Symposium, pp. 51-57 (Nov. 1987).
(3) J. L. Peterson: "Petri net theory and the modeling of systems", Prentice-Hall (1981).
(4) K. Shibata, W. Tanaka and H. Hasegawa: "Support system for specification phase of communication software", GLOBECOM '87, pp. 646-650 (Nov. 1987).
(5) K. Shibata, Y. Ueda, S. Yuyama, W. Tanaka and H. Hasegawa: "A logical method of verifying design specification in a communication system", IEICE Technical Report, **IN87** 75 (Nov. 1987).
(6) H. Hasegawa, W. Tanaka and K. Shibata: "Analysis of design specification in a communication system by means of petri nets", Proc. of 11th Computer Software and Applications Conference, pp. 701-706 (Oct. 1987).
(7) J. Sifakis: "Use of petri nets for performance evaluation", 3rd Int. Symp. Measuring, Modelling and Evaluating Computer Systems, Beilner and Gelenbe, pp. 75-95 (1977).
(8) J. Martinez and M. Silva: "A simple and fast algorithm to obtain all invariants of a generalized petri net", Second European Workshop on Application and Theory of Petri Nets, pp. 301-310 (Oct. 1981).

Kenji Shibata was born in Saitama, Japan, on March 2, 1959. He received the B. E. and M. E. degrees in electrical engineering from Chuo University, Tokyo, Japan, in 1982 and 1984, respectively. Since he joined Oki Electric Industry Co., Ltd. in 1984, he has been engaged in developing a support system for communication software development. He is interested in the application of knowledge engineering. Mr. Shibata is a member of the Information Processing Society of Japan.

Yoshihiro Ueda was born in Kumamoto, Japan, on November 2, 1961. He received the B. S. degree in mathematics from Kagoshima University in 1986. He joined Oki Electric Industry Co., Ltd. in 1986, and he has been engaged in the development of communication software. He is presently interested in linear algebra and he makes a research on algebraic analysis at the specification phase of software development. Mr. Ueda is a member of the Information Processing Society of Japan.

Satsuki Yuyama was born Kanagawa, Japan, on June 3, 1962. She received the B. A. degree in library and information science from University of Library and Information Science in 1985. She joined the Telecommunications Division, Oki Electric Industry Co., Ltd in 1985. Since she has been engaged in development of communication software and support system for communication software development. Ms. Yuyama is a member of the Information Processing Society of Japan.

Haruo Hasegawa was born in Hyogo, Japan, on February 27, 1950. He received the B. E. and M. E. degrees in electronic engineering from the University of Tokyo in 1972 and in 1974, respectively. Since joining Oki Electric Industry Co., Ltd. in 1974, he made a research on subscriber circuits in a digital switching system and developed communication controllers in a message switching system. From 1981 to 1986 he was engaged in the development of PBX software. His current fields of interest are support technology for specification phase of communication software development. Mr. Hasegawa is a member of the Information Processing Society of Japan, Japan Society for Software Science and Technology, and the Society of Instrument and Control Engineers.