IEICE TRANS. INF. & SYST., VOL.E78-D, NO.4 APRIL 1995

PAPER

An Analysis of Traceability in Requirements Documents

Kenji TAKAHASHI† *and* Shuichiro YAMAMOTO†, *Members*

SUMMARY We study the correspondence between problem descriptions and requirements specification documents derived from them. Based on the results of this investigation, a model that integrates the problem space and the requirements specification space is developed. This integration is based on a semantic network representation. We also propose a model of the requirements elicitation process that is consistent with our empirical studies of traceability in requirements documents. In this process, analysts derived requirements specifications from incomplete and ambiguous problem descriptions given by customers, identify missing information, completed it, and then decide the system boundaries that define which part of the problem descriptions to implement as the target system. The model can be used to complete problem descriptions given by customers and determine the system boundaries.

key words: *requirements analysis, traceability semantic network, empirical study*

1. Introduction

The requirements elicitation process is the first phase of software development. During this process, analysts learn about the problem space in which target systems should operate based on problem descriptions given by customers. The analysts then determine the system boundaries that define which activities in the problem descriptions should be implemented as facilities of the target systems. However, the problem descriptions are often incomplete and ambiguous, and even customers do not know what facilities the target systems should provide. To compensate for this missing information and to make design decisions, analysts employ their experience and domain knowledge to make assumptions about the problem spaces and the target systems.

These assumptions cause three major problems in the requirements elicitation process: poor traceability, inadequate domain knowledge, and imprecise capturing of customers' needs [1]. Correspondence between the problem descriptions and the requirements specification documents are difficult to trace because assumptions that are never explicitly recorded are added to the requirements specification documents. This causes difficulties in understanding and reviewing the requirements, and in carrying out impact analysis when modifying them. Neither the underlying domain knowledge about the target systems nor their environments are explicitly described in the requirements documents, which makes the requirements elicitation process difficult to support or automate. Lastly, the necessary features that the target systems lack are seldom detected until the systems are implemented, which leads to costly modifications from the requirements documents through to the programs. Analysts typically grope unsystematically for ways to discover all customer needs. A more structured process model is needed to support this process.

To understand and support the analysts' informal assumption-making process, we have analyzed the correspondence between problem descriptions and requirements specification documents produced by professional analysts. This analysis shows how to compensate for the missing information and determine the system boundaries. The analysis also explains the kinds of knowledge used in practice. Based on these results, a model of the requirements elicitation process is proposed to systematically support the process.

2. An Analysis of Traceability in Requirements Documents

2.1 Procedure

To study the requirements analysis process, two professional analysts were asked to produce a requirements specification document from a short problem description and then to explain which part of the problem description each constituent of the documents originated from and why. The analysts were also asked which parts of the specification documents were needed to complete the system requirements and why they were included. The analysts were interviewed and asked to explain their reasoning in detail.

The target system was a small transaction system for managing technical conferences (Fig. A·1). The size of the final program is about 6 KLOC. The problem description was prepared in Japanese. As requirements specification documents, data flow diagrams (DFDs) and data structured diagrams (DSDs) were produced according to the Structured Analysis [2]. Each analyst was responsible for one type of diagram, but they frequently discussed design issues in the process of producing the documents, and reviewed them jointly.

Both the subjects were experienced professional analysts with more than five years experience in software development, including more than a year in Structured Analysis.

Manuscript received April 7, 1994.

† The authors are with NTT Software Laboratories, Musashino-shi, 180 Japan.

2.2 Results

Some constituents of the requirements specification documents (DFDs and DSDs) corresponded to parts of the problem description in a straightforward manner. However, other constituents were added by the analysts based on their assumptions.

Our analysis shows that each bubble (or process) in the DFDs corresponds to a verb phrase in the problem description. Such verb phrases occur in sentences that describe activities to be implemented as facilities of the target system. Some words in the problem description phrase, however, are changed in the DFD process name into words that appropriately represent the process of the computer system. They retain the same meaning however.

Noun phrases related to the verb phrase, such as its objects, also correspond to data flows or data stores in DFDs. The structures of these objects are described in DSDs. For example, "Each session is given a room" is interpreted as a "distribute rooms" process with data flows to and from a "room management file."

The analysis also shows which parts of the documents were added to complete the system requirements, what decisions were made to determine the system boundary, and what kinds of knowledge these processes were based on.

Complements

The subject analysts constructed the "current" problem space before some of the activities in it were supported or replaced by the target system's facilities. They completed the problem description by adding missing information about activities, things, and relationships. For example, agents of activities are sometimes omitted in passive sentences in the problem description. Necessary but adjunct activities, such as exception-handling, also tend to be omitted. Missing information falls into the following categories.

- agents of activities occurring in the problem space
  (e.g., applications would be canceled by "applicants")
- activities that need to be carried out in order to fulfill a part of the problem description
  (e.g., applications should be "checked")
- things concerning activities and their inter-relationships
  - sources of activities
  (e.g., "rooms in the room management file" are distributed to sessions)
  - targets of activities
  (e.g., results of the review are sent to the "authors")
  - condition of activities
  (e.g., rooms are assigned according to "time and capacity")
  - results of activities
  (e.g., "accepted papers" results after reviewing them)
  - part-of relationships between things and other things that aggregate related ones
  (e.g., fee is a "part of call-for-participation")
  - kind-of relationships between things and other things that abstract related ones
  (e.g., capacity of a session is a kind of condition to decide rooms)
  - goal-subgoal relationships between activities
  (e.g., "to generate call-for-papers" is a subgoal of "to invite participants")
  - synchronization relationships between activities
  (e.g., "to submit a paper" is asynchronous to "to review a paper")
  - constraints between things
  (e.g., "the number of participants" must not exceed "the capacity")

Determination of system boundaries

A system boundary determines which part of the problem space is implemented as system facilities. Even when the description of the problem space is complete, the system boundary may remain undecided. To determine the boundary, the subjects decided whether the agent of an activity should be the target system or some other agents, such as a user or external system. For example, the review of papers should be done by humans. The management of rooms, however, can be done by the target system.

Mutual complement of requirements specification documents

The subjects detected errors by checking the consistency between each other's documents. They also checked the consistency between the DFDs and DSDs. The same data dependencies were expressed differently in DFDs and DSDs, so errors were detected by comparing the dependencies in the DFDs with those in the DSDs or vice versa.

2.3 Consideration

Further analysis of the rationale for assumptions made by the subjects shows that two kinds of knowledge are used in eliciting requirements: knowledge of the problem domain and system implementation plans.

Problem domain model

The problem domain model represents who does what for what purpose, and what things are in the domain where the target system operates. By using this model, the subjects could infer the information missed out of the problem description and understand the problem space. For example, based on fragments of descriptions about "chairmen," "rooms," and "time," the subjects imagined a scenario of the activities of chairmen, e.g., "chairman arranges a session." For this, the chairman must go to a "room" at a particular "time," before which the chairman must know the room and the time. Based on this scenario, the analysts added a new requirement for a notification facility for rooms and time slots to the requirements specifications.

System implementation plans

The subjects used system implementation plans, that is, templates of processes frequently found in systems for a do-

main, to implement activities in the domain. These plans consisted of one or more abstract activities whose agent is the target system, and the relationships between the activities. Requirements specifications were constructed by using instantiated system implementation plans.

Our analysis reveals the following six plans for implementing activities in the problem space.

- *resource distribution plan* that distributes finite resources according to demands (e.g., rooms are distributed among the sessions).
- *editing and printing plan* that merges different kinds of data into a certain format, such as a table, and prints it out. (e.g., programs of tutorials and technical sessions are merged into a call-for-papers).
- *data storing plan* that stores data to retrieve and use when the information is generated and used asynchronously (e.g., applications are stored because an application is not always received and canceled sequentially).
- *input analyzing plan* that checks external inputs and accumulates them (e.g. application forms are checked before being processing).
- *notification plan* that notifies externals, such as users. (e.g., results of reviews are sent to authors).
- *constraints maintenance plan* that maintains constraints between things (e.g., the number of attendees is managed to keep it within the capacity limit).

3. A Model of the Requirements Elicitation Process

A model of the requirements elicitation process has been developed based on our analysis (Fig. 1). This model consists of a problem space and a system requirements specification space. The problem space represents the space where the problems to be solved occur. When analyzing information systems, such as the technical conference management system used in this analysis, the tasks in work places – that is, who does what for what purpose – are represented in the problem space. The system requirements specification space represents the system requirements specifications of target systems that solve the problems. Different aspects of the system requirements specifications are documented in different kinds of diagrams or formats. In this analysis, the behavioral aspects of the system specification are represented in the DFDs and the structural aspects in the DSDs.

Analysts elicit requirements specifications by using these two spaces opportunistically. They first read and interpret problem descriptions to construct the problem space by inferring the missing information about activities in the problem description and relating that information to the problem domain model. Next, the analysts recognize activities to which system implementation plans can be applied and supply the missing constituents of those plans to the problem description to finalize the system requirements specification space.

In this model, system requirements specifications are specified by deciding which activities in the problem space are implemented as system facilities. Thus, the system requirements specification space is part of the problem space. The analysts also elaborate system specifications by using complementary aspects of the system requirements specification spaces represented as corresponding diagrams, such as DFDs for the behavioral aspect.

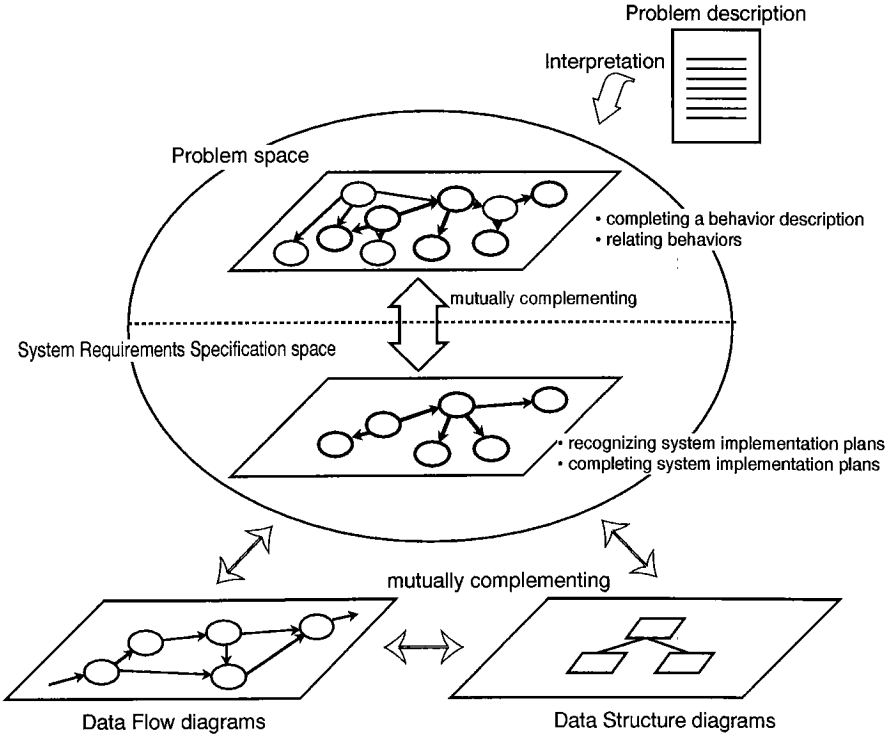

Fig. 1 Overview of the requirements elicitation process model.

Semantic networks are used to represent the problem space and system requirements specification space in a uniform manner. The requirements elicitation process can be modeled as operations on those networks, such as insertion and deletion of nodes and arcs.

3.1 Semantic Network Representation

The requirements elicitation process model is represented by using semantic network representation. This representation is used for two reasons. First, it has been used to represent the intellectual activities of humans in studies of artificial intelligence and natural language processing [3], [4]. Thus, it can be used to represent these activities in the problem space. Next, semantic networks can be used to represent various aspects of system requirements specifications in a uniform manner [5], [6]. Because most requirement specifications documents are represented by network diagrams, such as DFDs and DSDs, these diagrams can be uniformly represented by using the semantic network. This uniform representation lets analysts detect inconsistencies between different aspects of the specifications and to analyze the impact of fixing them.

Problem spaces and system requirements specification spaces can also be integrated by representing both of them in uniform semantic networks. This integration enables each task of the elicitation process to be executed as an operation of the semantic network.

Representation of problem spaces

A problem space consists of activities and their inter-relationships. Based on the concept of case grammar [7], an activity is represented by an action and the things corresponding to its cases: namely, an agent, objects, targets, results, and conditions. In a sentence, the verb phrase basically correspond to an action and the noun phrases to the cases. There are three kinds of relationships between things: kind-of, part-of, and constraint relationships; and two kinds of relationships between activities: asynchronous and goal-subgoal relationships. Based on this analysis, semantic networks that represent the problem space should have the eleven kinds of arcs listed below (also Fig. 2 and Fig. A·2). These arcs and cases can be naturally enhanced by creating or modifying them to fit the problem domains [8].

AGENT: relationship between an action and one of its agents
OBJ: relationship between an action and one of its objects
SOURCE: relationship between an action and the source of an object
TO: relationship between an action and one of its targets
CAUSE: relationship between an action and one of its results
CONDITION: relationship between an action and one of its conditions
GOAL: relationship between an activity and one of the goal activities to be accomplished through the activity
ASYNCHRO: relationship between an activity and a thing that is used asynchronously by others
PART_OF: relationship between a thing and one of its constituents
KIND_OF: relationship between two things of the same type
CONSTRAINT: relationship between a thing and something that constrains it

Representation of system requirements specification spaces

The system requirements specification space consists of activities and things extracted from the problem space: activities to be implemented as facilities of the target system and things related to the activities. Each aspect of the system requirements specification space is represented in a different kind of diagram. The space consists of the following:

- Actions to be implemented,
- Agents of the actions above and externals to the system,
- Things having a relationship except an AGENT relationship with the actions above,
- Things having a PART_OF relationship with the things above, and
- Relationships among these actions and things. (OBJ relationships fall into one of two categories depending on

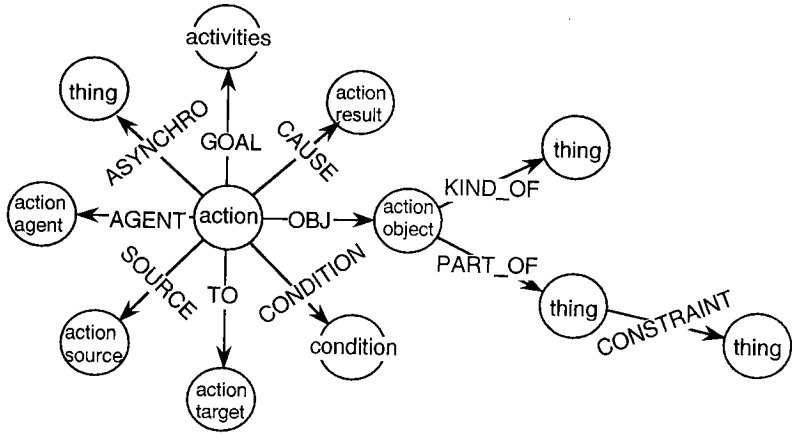

Fig. 2 Semantic network representation.

whether the object is an input or an output of the system. An I_OBJ relationship is for an input and indicates that the object already exists. An O_OBJ relationship is for an output and indicates that the object results from the action.)

Figure A·2 shows a part of the semantic network representation, which corresponds to the third paragraph of the problem description shown in Fig. A·1.

Representation of DFDs

The semantic network representation of a data flow diagram is extracted from the system requirements specification space. This is done by removing the intermediate constituents of data structures not appearing in the DFD (Fig. A·3). This semantic network representation corresponds to a DFD as follows:

- An action and things that have I_OBJ or O_OBJ relationships with the action correspond to a process
- Things that have I_OBJ, CONDITION, or SOURCE relationships with actions correspond to input data flows.
- Things that have O_OBJ or CAUSE relationships with actions correspond to output data flows. Things that have PART_OF relationships with these things correspond to input data flows.
- Constituents of output flows correspond to those of input flows or data stores.
- Constituents of input flows correspond to those of output flows or data stores.
- Things that have an ASYNCHRO relationship with actions correspond to data stores.

Representation of DSDs

The semantic network representation of a data structure diagram is also extracted from the system requirements specification space. It represents the part-of relationships between things in the space (Fig. A·4). In the network, however, DSD modifiers, such as selection, are not currently represented because part-of relationships are sufficient to represent the system's functionality. Correspondences between the semantic network representation and a DSD are as follows:

- Things that have no PART-OF relationship with other things correspond to a root node of a data structure
- Things that have PART-OF relationships with these things above correspond to constituents of that data structure.

3.2 Representation of Requirements Knowledge

The knowledge analysts used to elicit requirements is represented by the semantic networks. There are two kinds of knowledge: problem domain models and system implementation plans. Problem domain models represent knowledge about activities and things in a problem domain. These models can be represented by a semantic network and problem spaces are a part of them. System implementation plans are also represented as a set of "abstracted" activities and related things in these networks. Names of nodes in the semantic network are abstracted to be independent of problem domains. System requirements specification spaces consist of instantiated plans.

3.3 Requirements Elicitation Process Model

The tasks in the requirements elicitation process are represented as operations in the semantic networks. There are three tasks in this process: inferring missing information from problem descriptions, determining the system boundaries, and validating consistency among system requirements specifications.

Inferring missing information from problem descriptions

In the problem space, each activity is first completed by identifying essential cases not described in the problem description and filling them out. Next, relationships between activities are formed, such as goal-subgoal and synchronization relationships. These operations are based on the problem domain space model. System implementation plans are also used to complete the problem space. They are identified in the problem space, and any missing constituents are added to the semantic network.

Determining system boundaries

After completing the problem spaces, the analysts determine which part of the spaces should be implemented as system facilities (Fig. A·2). There are two steps in this process: selecting candidates from the entire problem space, and then determining which activities to implement based on these candidates.

In the first step, the analysts identify a minimal set of candidate activities that are explicitly described in problem descriptions, and extend the minimal set by adding candidates to make use of information created by the set. They first focus on the goal-subgoal structure of activities in the problem space. Each leaf node, that is, a node in the lowest layer of the structure, corresponds to a system implementation plan. If an activity in the problem description is detailed enough to correspond to a system implementation plan, it is selected as an implementation candidate. Otherwise, the analysts must refine the activities so they are detailed enough to correspond to the plans. The analysts also select as additional candidates those activities that use the information resulting from activities already selected as candidates and activities that are under leaf nodes of activities in the problem description.

For example, the activities, "make a presentation" and "distribute rooms," are described in the problem description (Fig. 3). The latter corresponds to the resource distribution plan and creates information about "time" and "room." An activity, "be sent a notification about the room and time," is a leaf node of the goal, "make a presentation," and uses the room and time information. Thus, this activity is implemented as a system facility, even if it is not directly referred to in the problem description. An analyst compensates for

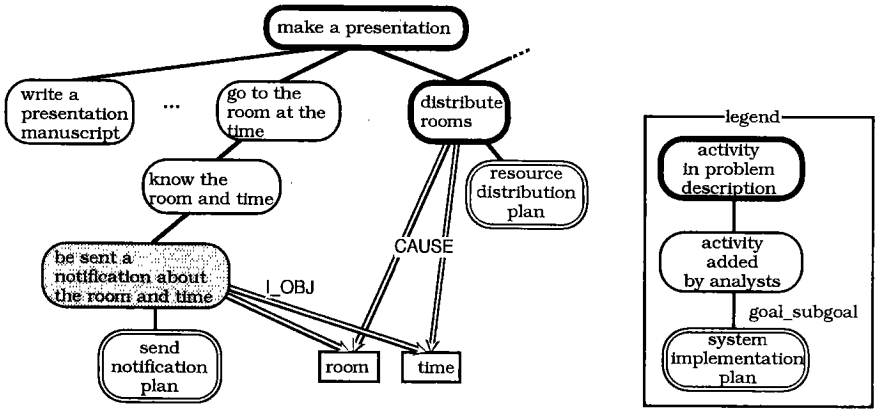

Fig. 3 Determining system boundaries.

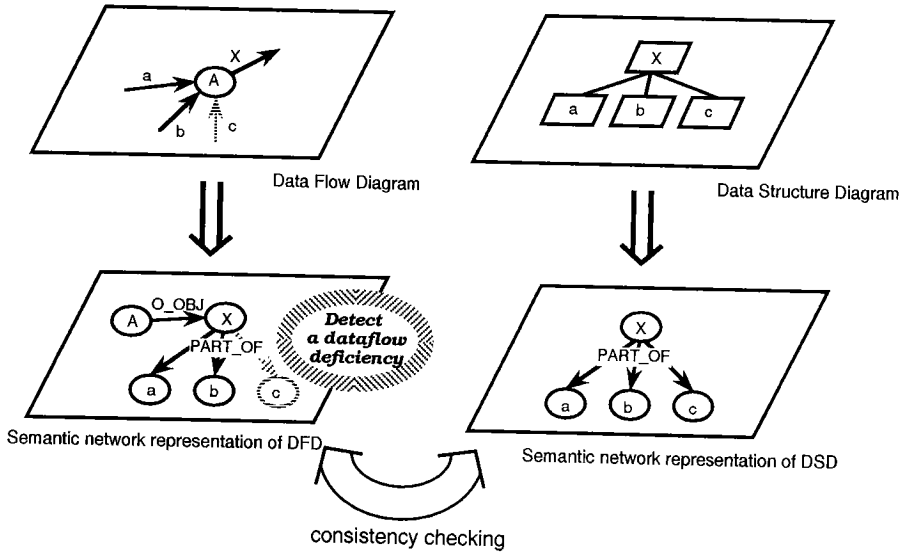

Fig. 4 Consistency checking between DFDs and DSDs.

these intermediate activities – such as "know the room and time" and "go to the room at the time" – to elicit this facility in his mind.

Next, according to constraints (such as performance and cost) and user intentions, the analysts determine which of the candidate activities are to be implemented as system facilities. This process has yet to be closely investigated.

Validating consistencies among the system requirements specifications

System requirements specifications are elaborated through reviews from various aspects represented in different diagrams. These diagrams are complementary to each other, representing the same or related information. For example, DFDs and DSDs represent the same data dependencies in different ways. By representing these shared or related pieces of information in a uniform semantic network and comparing them, analysts can check the validity of consistencies between the diagrams.

For instance, errors in the diagrams can be detected by comparing the data dependencies represented in the DFD and DSD (Fig. 4). As a sanity check, the analysts use a heuristic concerning data dependencies of DFDs; "constituents of output data flows of a process should be included in input data flows; conversely those of input data flows should be included in output data flows." According to this heuristic, this DFD states that X consists of a and b, whereas the DSD says that X consists of a, b, and c. In this way, a possible deficiency of data flow c is found by comparing these two representations. This heuristic is too restrictive, though, in the sense that it detects dataflows that are not necessarily incorrect. It is true that input and output dataflow constituents do not always match (e.g. those of a filtering process). However, unmatched constituents are usually few and, if any, analysts have only to check the validity of a few detected unmatched constituents, which reduces the analysts' works and ensures the validity of the requirements specification diagrams. In addition, detected relationships between unmatched constituents are utilized in creating different kinds of programs, such as entity-relationship diagrams (ERDs).

Further validation of unmatched dataflows could be

done by a consistency check between DFDs and ERDs, since ERDs define the cardinality of input and output dataflows and their semantic relationships.

4. Related Work

Other researchers are working on semantic network representations of system requirements specifications. By using a semantic network, they can uniformly represent various aspects of system requirements specifications, such as the behavioral and structural aspects. ARIES [9], TINA [5], and PRISMA [6], are three systems of this kind. CARD [8] generates requirements specifications from pseudo-natural language by using case grammar. These systems mainly deal with modeling system requirements rather than eliciting requirements.

Leite et al. [10] have developed the Viewpoint Resolution Method for validating and constructing a problem space. This method, however, does not provide any means to derive system requirements specifications. Goal-directed concept acquisition [11] deals with problem spaces and system requirements specification spaces and accounts for our findings. This acquisition method focuses on deriving formal requirements specifications from given goals. On the other hand, our model focuses on exploring and finding missing goals, and representing them in relatively informal diagrams. Formalism in the goal-directed acquisition method may provide a good basis for formalizing our model.

Maiden and Sutcliffe [12] have proposed a model that consists of a taxonomy of goal-types and heuristics to identify each goal-type. These goal-types and heuristics match our model well and will be useful in guiding the process of deriving the goal-hierarchy of our requirements semantic network.

5. Conclusion

By analyzing the correspondences between a problem description and requirements specification documents, we have developed a model of the requirements elicitation process based on semantic networks. This model integrates the problem space and the system requirements specification space. This integration helps analysts complete the problem space and determine system boundaries.

Towards Requirements Elicitation Support System

Based on this requirements elicitation process model, two facilities of requirements elicitation support systems are proposed. Each is based on the uniform semantic network representation. The first is for mutual translation and consistency checking between the problem space and system requirements specification space, and among the various aspects of the requirements specification space.

The second type of support involves facilities using domain knowledge and system implementation plans. Some examples are facilities for completing activities and their relationships by using domain knowledge, for completing system requirements by using system implementation plans, for suggesting goal-subgoal trees and system implementation plans corresponding to given activities, and for checking the feasibility of system requirements specifications according to estimated performance and costs.

In order to develop this support system, further investigation of analysts' activities is needed. In the model, analysts are assumed to thoroughly know a problem domain space. In practice, however, the analysts often encounter novel domains. In such a case, the analysts can elicit requirements in novel domains based on similar cases they have experienced. Case-based reasoning/learning [13] in the requirements elicitation process should be analyzed to solve the knowledge acquisition problem in developing knowledge-based support facilities.

In our analysis and in the majority of real projects, the problem description is given to analysts in natural language. Because automated interpretation and translation are not yet feasible, humans should interpret the description and build a problem space [14]. The semantic network representation, however, is too complex for analysts and customers to handle. We need formal but sophisticated media, such as VRDL [15], for better communication between customers and analysts.

For simplicity, our model does not include a structured representation of the DFDs. Structures of DFDs could be represented by introducing structured representation to the system requirements specification spaces.

Acknowledgement

From NTT Software Laboratories, we thank Sadahiro Isoda, Shigeki Goto and Masaki Itoh for their advice and support in conducting the research and Atusko Oka for her support in carrying out the analysis. We also thank Colin Potts and Jeffrey Donnel of the Georgia Institute of Technology for their valuable comments in developing this idea.

References

[1] Lubars, M., Potts, C. and Richer, C., "A review of the state of the practice in requirements modeling," *Proc. the first International Symposium on Requirements Engineering* (RE '93) (San Diego, California), pp. 2-14, 1993.

[2] Yourdon, E., *Modern Structured Analysis*, Yourdon Press, 1989.

[3] Woods, W., "What's in a link: foundations for semantic networks," *Studies in cognitive science*, pp. 35-82, Academic Press, 1975.

[4] Sowa, J. F. (ed), *Principles of semantic network*, Morgan Kaufmann, 1991.

[5] Jordan, K. and Davis, A., "Requirements engineering metamodel: an integrated view of requirements," *Proc. COMPSAC '91* (Tokyo, Japan), pp. 472-478, 1991.

[6] Niskier, C., Maibaum, T. and Schwabe, D., "A look through PRISMA: towards pluralistic knowledge-based environments for software specification acquisition," *Proc. 5th International Workshop on Software Specification and Design* (IWSSD-5), pp. 128-136, 1989.

[7] Fillmore, C. J., "Lexical entries for verbs," *Foundation of lan-*

guage, vol. 4, no. 4, pp. 373-393, 1968.

[8] Ohnishi, A. and Agusa, K., "CARD: A software requirements definition environment," *Proc. 1st Int. Symp. Requirements Engineering* (San Diego, California), pp. 90-93, 1993.

[9] Johnson, W. L. and Feather, M., "Using evolution transformations to construct specifications," *Automated software design*, pp. 65-91, AAAI Press, 1991.

[10] Leite, J. and Freeman, P. A., "Requirements validation through viewpoint resolution," *IEEE Trans. Software Eng.*, vol. 17, no. 12, pp. 1253-1269, 1991.

[11] Dardenne, A., Fickas, S. and Lamsweerde, A.V., "Goal-directed concept acquisition in requirements elicitation," *Proc. IWSSD-6* (Como, Italy), pp. 14-21, 1991.

[12] Sutcliffe, A. G. and Maiden, N. A. M., "Bridging the Requirements Gap: Policies, Goals and Domains," *Proc. of IWSSD-7* (Redondo Beach, California), IEEE Comp. Soc. Press, pp. 52-55, 1993.

[13] IEEE Expert: special issue on case-based reasoning, Oct. 1992.

[14] Reubenstein, H. B. and Waters, R.C., "The Requirements Apprentice: Automated Assistance for Requirements Acquisition," *IEEE Trans. Software Eng.*, vol. 17, no. 3, pp. 226-240, Mar. 1991.

[15] Ohnishi, A., "A Visual Software Requirements Definition Method," *Proc. 1st Int. Conf. Requirements Engineering* (Colorado Springs, Colorado), IEEE Comp. Soc. Press, pp.194-201, 1994.

Appendix

Technical conference management system

Design a system that manages a technical conference.

The technical conference consists of technical sessions and tutorials. The tutorials are held on the first two days and the technical sessions on the other days. There are two kinds of technical sessions; paper sessions and panel sessions.

A program committee reviews papers, those selects some to be presented, and assigns those selected to appropriate sessions. It also plans several tutorials and holds them for two days. Each tutorial is given a title, a lecturer, a room, and its capacity, and a time. Each session is given a title, chairman, room, time, and papers.

The conference is announced two months before being held to invite attendees of technical sessions and tutorials.

The program committee decides on the fees. All fees are paid through a bank. There are discounts for applicants who pay more than one month before, students, and members of related associations. When applications are canceled, the amount remaining after the cancellation fee is subtracted is returned to the applicant.

Fig. A·1 Problem description of a technical conference management system.

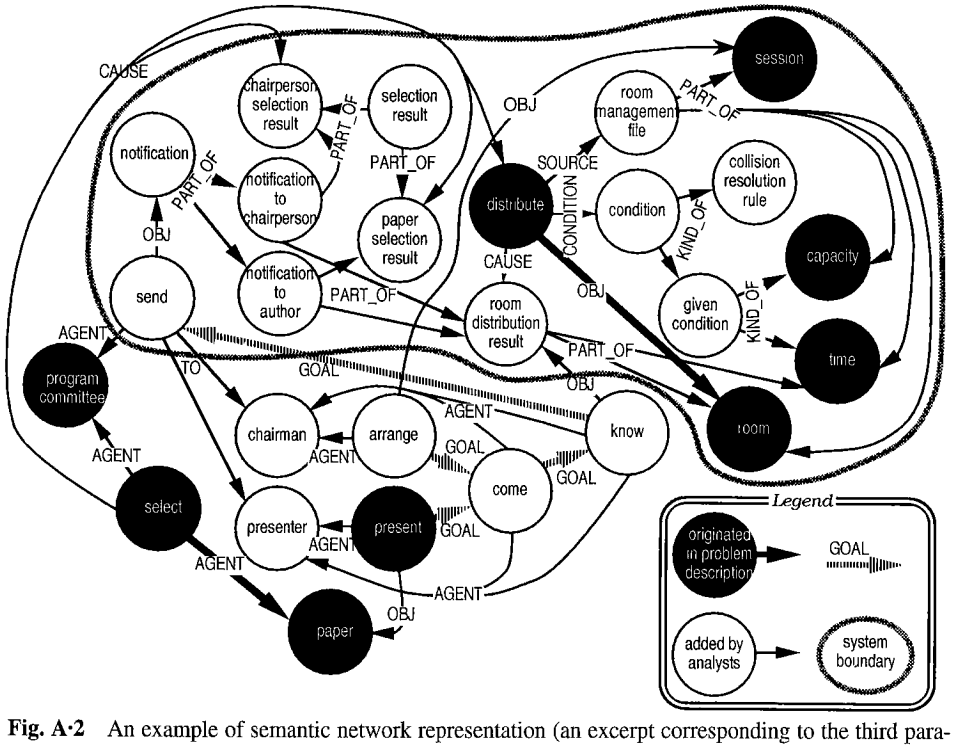

Fig. A·2 An example of semantic network representation (an excerpt corresponding to the third paragraph of Fig. A·1).

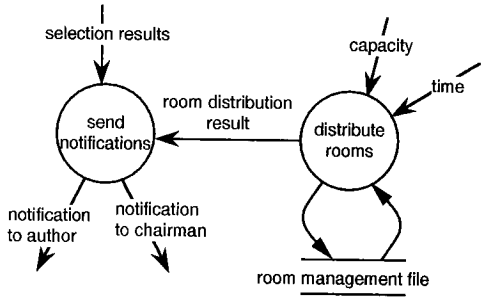

Fig. A·3 Resulting DFD.

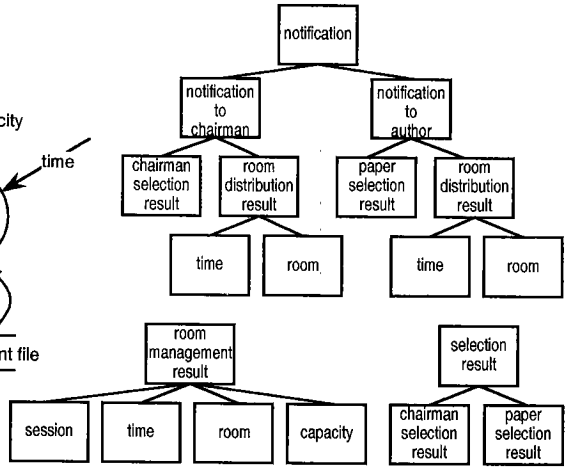

Fig. A·4 Resulting DSD.

Kenji Takahashi is a senior research engineer of NTT Software Laboratories. His research interests include requirements engineering and hypermedia supportfor collaborations in software developement. Takahashi received a BS and an MS in computer science from Tokyo Institute of Technology in 1984 and 1986, respectively. He was a visiting scientist in the College of Computing at Georgia Institute of Technology from 1992 to 1993. He is a member of IPSJ, the IEEE Computer Society and ACM.

Shuichiro Yamamoto is a senior research engineer, supervisor, of NTT Software Laboratories. He is currently engaged in the methodology development for distributed information systems. He contributed significantly to the design and implementation of CASE tools for the information systems of NTT. Mr. Yamamoto received a B.S. in information engineering from Nagoya Institute of Technology in 1977, and an M.S. in information engineering from Nagoya University in 1979. He is a member of IEEE, IPSJ and JSAI.