Representation of Active Rules in Cooperative Work Environment

Hisayuki MASUI
Masakazu NOMOTO
Yahiko KAMBAYASHI
Integrated Media Environment Experimental Lab.
Faculty of Engineering, Kyoto University
Yoshida-Honmachi, Sakyo Kyoto 606-01, Japan

Abstract

Due to the recent development of distributed systems, new application area to support user cooperation is getting popular. Database functions must be utilized for user communication, since there are cases when a user may not be available at the time of message arrival. In few years, most relational database systems will have trigger functions to enforce various kinds of semantic constraints as discussed in SQL-3 standard. In this paper, we assume that ECA(Event Condition Action) rules (a general form of trigger functions) are available, and we will discuss how to realize user coordination utilizing ECA rules. As it is impossible to control user coordination automatically, negotiation among users is required. For this purpose we need to display the situation of the systems in a form which can be understood by users easily. In order to represent active rules, finite automaton model, ECA model and IDEF0(Integrated Computer-Aided Manufacturing DEFinition 0) model are compared and a generalized IDEF0 model is proposed in this paper. Major generalizations are as follows; 1) Control flow and workflow are distinguished, 2) Start conditions are explicitly expressed in order to translate an IDEF0 diagram into ECA rules. Workflow representation by a generalized IDEF0 model can be used for user negation. Dynamic modification of workflow is also discussed.

1. Introduction

Recent development of high-speed computer networks and high-performance workstations are opening up new application areas of computer systems. CSCW(Computer-Supported Cooperative Work) is one of such important applications. Since conventional CSCW systems are developed utilizing video systems, communication facilities including E-Mail and basic window sharing mechanisms, we have been developing new CSCW environment utilizing database technology, called VIEW(Virtual Interactive Environment for Workgroups). Applications of VIEW concept include an office system called VIEW Office(although it had been called *VirtualOffice* [TK93], it was renamed

since the term "virtual office" became a general term) and VIEW Classroom for distance education. In VIEW Office we need to support cooperation among users. We have developed deputy object model[PK95] to realize coordination among users with different views.

Recently active mechanisms are going to be combined with database systems. One typical example is ECA rules developed by Dayal et al.[Da88]. In VIEW systems we are using ECA rules for user coordination. As ECA rules are developed for software systems, there is no strong requirement for modification. In our purpose, it is required to modify rules since a person may not be available, or the rule cannot be applied due to some privacy reasons. Furthermore, it is required that a user can know the meaning of the rules easily. We will use IDEF0 (Integrated Computer-Aided Manufacturing DEFinition 0) model developed by US Air Force to define workflow[MM93]. As detailed description of actions are required to use IDEF0, the description by IDEF0 is more powerful and it is easy for users to understand. The followings are major contributions of this paper.

1) Comparisons of finite automaton model, ECA model and IDEF0 model are given.
2) By the above comparison extensions of IDEF0 model are discussed. The generalized IDEF0 model has high expressive power, which is easy to be understood by users and also easy to translate it into underlaying ECA rules of database systems.
3) Dynamic modification of generalized IDEF0 model is also discussed.

2. Basic Concepts

In this section three methods to describe user communication are defined.

2.1 ECA Model

A trigger mechanism[Esw76] is important in a database system to enforce consistency constraints and also to realize an automatic start of transactions. For example, an update of some values may cause updates of other values, or printing of reports. Such a mechanism is also very much useful for office systems. Recently a version of trigger mechanisms, called ECA rules, is introduced to realize active database systems[Da88, DBM88, MD89]. Here, E, C, and A stand for Event, Condition, and Action, respectively. ECA rules are used to handle actions realized by computer programs. In this section, we first describe the ECA rules and then requirements to realize cooperative work among users are discussed.

Primitive events are defined as follows[MD89]:

- User-specified database operations: data definitions, data manipulations
- Database system operations: periodical checking operations, automatically executed database operations, failure of the system, roll-back operations, transaction control, back-up operations
- Temporal events: absolute time, relative time, periodic
- External notifications: application defined events, user defined events

A composite event is defined as a result of applying Boolean operations and concatenations (sequence operation) on primitive events.

ECA mechanism M is defined as

$$M = (E, C, A, K),$$

where E is a set of primitive events and composite events, C is a set of conditions, A is a set of actions. K shows combinations of E, C and A together with coupling conditions. An element of K is $(e_i, c_i, a_i, d_{1i}, d_{2i})$ called an ECA rule, where e_i is an element in E and c_i is a subset of C, a_i is the action which will be triggered when e_i occurs and one of the conditions in c_i is satisfied, and d_{1i} specifies when the condition is evaluated relative to the transaction in which the event is signaled, d_{2i} specifies when the action is executed relative to the transaction in which the condition is evaluated. There are typically the three modes for d_{1i} and d_{2i}, immediate, separate(evaluation (or start of an action) is realized by a separated transaction) and deferred(evaluation (or start of an action) is done when the transaction terminates). An ECA rule can be expressed by a block shown in Fig. 1.

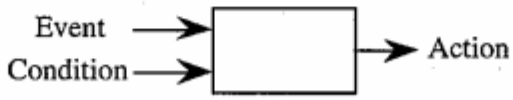

Fig. 1 ECA rule

ECA rules are suitable to describe operations caused by transactions. In our requirement, as communication among users and programs must be considered and some action may take very long time, we need to consider the following problems.

1) Even if a condition is satisfied, the corresponding action cannot be started because the user who is in charge of the action is not available.
2) An equivalent action may be taken by a user who is in charge of the action. Modification of action should be realized also. An action can be divided into several sub-actions.
3) Dynamic change of event-condition-action relationships should be handled. For example, if some condition is satisfied a set of conditions may need to be modified. A simple case is that some condition can be set to be inactive.

Since user's speed is much slower than computers, we need not use coupling conditions(d_{1i} and d_{2i}) when action is realized by some persons.

2.2 Finite Automata Model

A finite automaton is defined by

$$M = (\Sigma, \Gamma, S, \delta, \lambda, s_0)$$

where Σ is the set of input symbols, Γ is the set of output symbols, S is the set of internal states, δ: $S \times \Sigma \rightarrow S$ is the next state function where $\delta(s_1, a) = s_2$ means that a new state is s_2 when input a is applied to state s_1. λ: $S \rightarrow \Gamma \cup \{\varepsilon\}$ is the output function where $\lambda(s_1) = b$ means that output b is generated when the automaton enters state s_1. In this definition we permit states without any outputs(i. e., for s_i, $\lambda(s_i) = \varepsilon$). Instead of λ we can use λ': $s \times \Sigma \rightarrow \Gamma \cup \{\varepsilon\}$ as the output function, where $\lambda'(s_1, a) = b$ means that output b is generated during the transition caused by applying input a to state s_1. s_0 in S is the initial state.

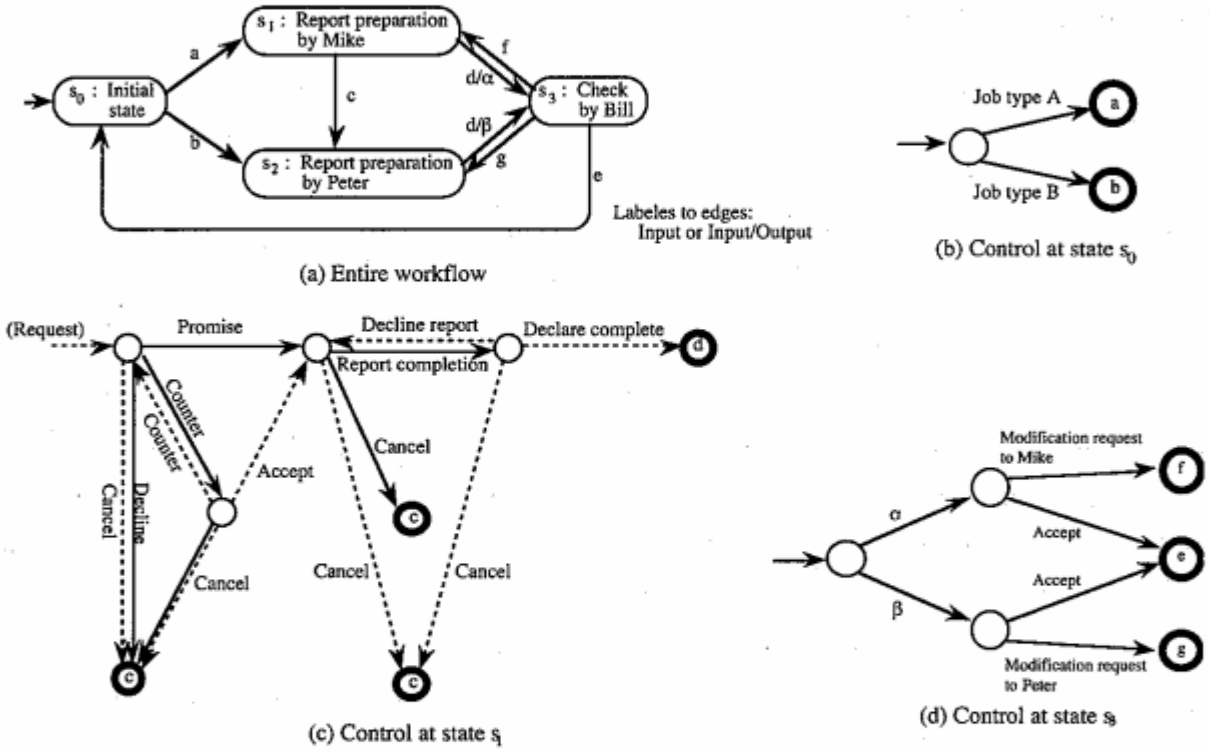

Fig. 2 Control flow and workflow representation by a finite automaton model

A workflow is expressed by a finite automaton as shown in Fig. 2(a). This workflow shows that a report is prepared by Mike or Peter and it is later checked by Bill. If the report is incomplete, the person who prepared the report have to revise until Bill satisfies. In this figure, input alphabet is {a, b, c, d, e, f, g} and output alphabet is $\{\alpha, \beta\}$. λ' is used for the output function. A label like f on an edge shows an input symbol which cause the transition. A label in the form like d/α shows the combination of input and output symbols(in this case input is d and output is α).

State transition conditions are specified by the automata which specify control flow. Examples are

shown in Fig. 2(b), (c) and (d). Here symbols in the states show outputs to be generated at the states(i. e., output function λ is used). Fig. 2(b) shows the control of state s_0. When job type A(B) is given this automaton, a(b, respectively) is generated, which causes the transition from s_0 to s_1(s_0 to s_2, respectively).

Control at s_1 is rather complicated, since it involves some kind of user negotiation. Let Jay be the person who asked the job. The dotted lines show input from Jay and straight lines show input from Mike. The state diagram is exactly same as the diagram shown in [Wi88]. When Jay asks to prepare a report to Mike, he can promise, decline or make counter proposal. If Mike declines or his counter proposal is not accepted, output c is generated, which causes the state transition from s_1 to s_2 in the automaton shown in Fig. 2(a). Then Jay start to ask Peter to do the job. Furthermore, during the job it can be canceled. If the job is completed, the automaton generates output d, which causes the transition from s_1 to s_3 in the automaton in Fig. 2(a).

Control automaton for s_2 can be similarly defined. Fig. 2(d) shows the control automaton for state s_3. If Bill satisfies the result it generates e and the automaton will be reset to initial state s_0, otherwise f or g is generated to complete the report.

There are two kinds of interactions among these automata. Signals from control flow automata to the workflow automaton(a, b, c, d, e, f, g) are used to control the automaton. Signals from the workflow automaton(α, β) are used for observation of the flow. In this example, depending on who prepared a report, output becomes α or β.

2.3 IDEF0 Model

IDEF0 was developed by US Air force to describe workflow in detail to be understood by nonprofessional users[MM93].

The fundamental description of IDEF0 includes five elements as shown in Fig. 3, Subject described in the block, Inputs to be processed and operated by Subject, Control which gives influence to subject's activities, Mechanisms which are used to perform the activity, and Outputs which show outcomes of the subject activity. Subject is described in a rectangle and Input, Control, Mechanism and Output are described as arrows. Except Control and Output, these arrows are not always required.

In order to describe a diagram in detail, each subject can be further replaced by a diagram. These diagrams form a hierarchical tree structure.

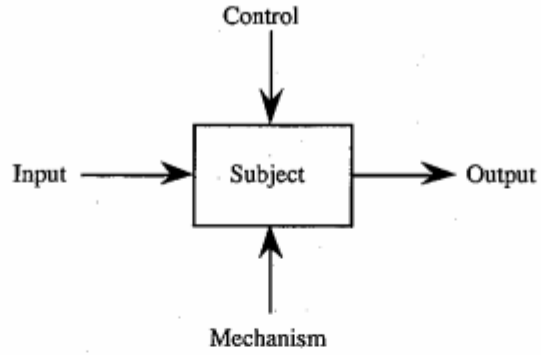

Fig. 3 A unit block of IDEF0 expression

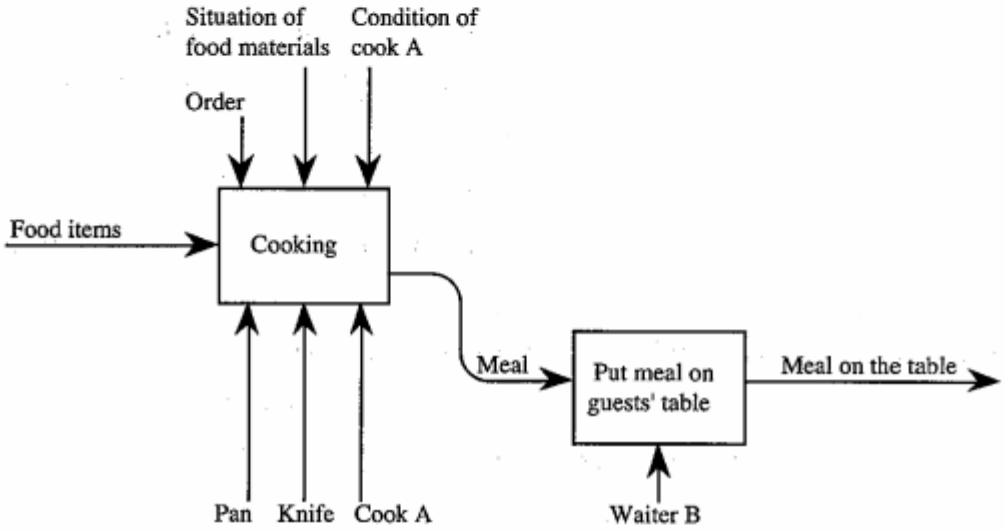

Fig. 4 An example of a process in a restrant

A simple example of IDEF0 diagram is shown in Fig. 4, which shows processing of an order at a restaurant. There are two major actions, "cooking" and "put meal on guests' table" which are shown as subjects in the two blocks. By cooking, food materials(input) are converted into the meal (output).

Cook A is assigned to prepare the meal and thus he is one of the mechanisms. Cooking instruments(in this figure, a pan and a knife are shown) are also mechanisms. Cooking will be stopped if some of the mechanisms are not available. For example, after cutting food materials, cooking cannot continue if pan is used by another cook.

Details of cooking is determined by the order from the guests, situation of food materials (varieties, quantities, quality, etc.), and conditions of cook A(for example, if he has a lot of work to do, he may change the process of cooking). These are shown as controls. When meal is ready, waiter B will bring the meal to the table of the guests. The second block shows this action.

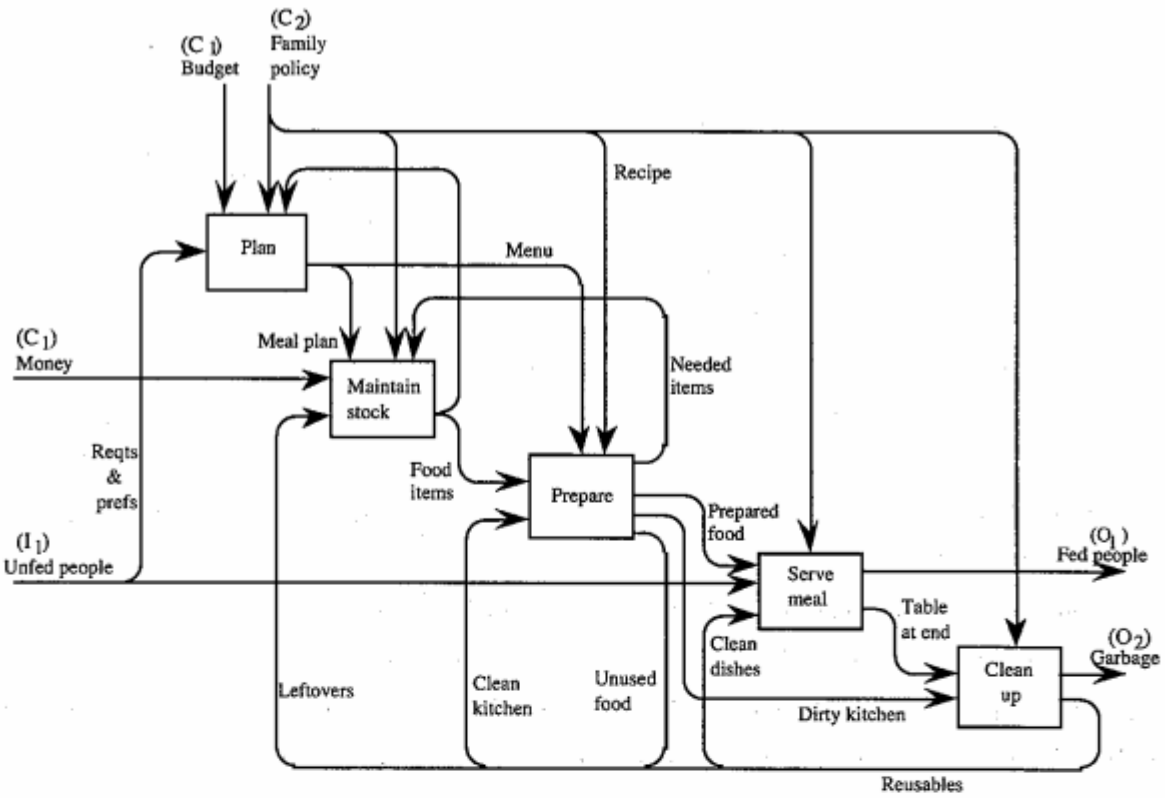

Fig. 5 Feed family example

A more complicated example is shown in Fig. 5 which is a Feed Family model given in [MM93]. In this example, input(I_1) is unfed people, controls(C_1, C_2) are money & budget and family policy, outputs(O_1, O_2) are fed people and garbage. The workflow shown in Fig. 5 is easy to understand. For simplicity mechanisms are omitted. There are mutually constraining activities, details that require hiding, exception handling, and error conditions. This diagram is decomposed to hierarchical diagrams adequate for writing detailed specifications. This example is selected, since the complexities of feeding a family are rich enough to represent any systems analysis methodology.

3. Comparisons on the Models

Since each model has different advantages and disadvantages, we will compare the models. The results will be used to generalize IDEF0 model.

A. Problems of ECA models

Since ECA model is developed for database/transaction processing, starting time of actions is very much important. Events and conditions determine when to start the corresponding action. Details of the action is described by a program and it is not easy for users to understood its operation. Composition of ECA rules is also difficult to be understood, since it is a combination of

programs/rules. As a conclusion, ECA model is suitable for database transactions and not suitable for user interface.

In ECA model when an event occurs then conditions are checked, and at that moment if there is no satisfied conditions the corresponding action will not be started and the rule will be canceled. In order to handle cooperation of users we need to consider the following case.

After 9 a. m.(Event), we will start a job(Action) when all the members of a group come (Condition). In this case, when Event occurs the condition may not be satisfied. The job may be able to be started at 9:30 a. m., for example, when the last member comes. We can eliminate Event by converting it into Condition(at 9 a. m. is an Event but after 9 a. m. is a Condition). The following condition can be used.

We will start a job(Action), when all the member of a group comes(Condition 1) during 9 a. m. and 10 a. m.(Condition 2).

Action will be started when both Condition 1 and Condition 2 are satisfied.

For user cooperation we will use this kind of Condition-Action rules. Since ECA rules are also required to cope with transactions, we will translate Condition-Action rules to ECA rules. Advantage of ECA model is that some version of the model will become available in most database systems in near future.

B. Advantages and disadvantages of finite automata models

As shown in Fig. 2, we can represent both workflow and control flow by a finite automaton model. More general control is possible. For example, the control such as "Bill cannot ask rewriting of a report if it is prepared by Peter" can be also expressed by a finite automaton model. It is realized by defining a part of paths by defining regular expressions.

Another advantage is that branching condition can be explicitly expressed. For example, there are two transitions to s_2 and s_3 from s_1 and selection of one transition is explicitly stated by control signals c and d.

On the other hand, in IDEF0 model selection is expressed by control part and no explicit selection is possible.

Fig. 6 shows IDEF0 diagram corresponding to the finite automaton model in Fig. 2. It is rather easy to understand, but the information on which branch should be taken is not expressed. Since each block can be replaced by an automaton, hierarchical representation is possible by both finite automaton and IDEF0 models. In IDEF0 model, detail description block "Report preparation by Mike" corresponds to the automaton shown in Fig. 2(c). Workflow and control flow are always mixed in IDEF0 model.

One serious drawback of the finite automaton model is that it cannot express parallel execution. There is a nondeterministic automaton model which can handle parallelism. In that model all possible paths are realized. In order to express a subset of all possible paths we need to define states, each of which corresponds to one subset of the states of the original finite automaton. Since the number of

such states is $(2^n - 1)$, it is practically impossible to use such a model.

Use of Petri nets[Pe81] is one possible solution. We will not discuss further since it is easy to translate diagrams in Fig. 2 into Petri nets.

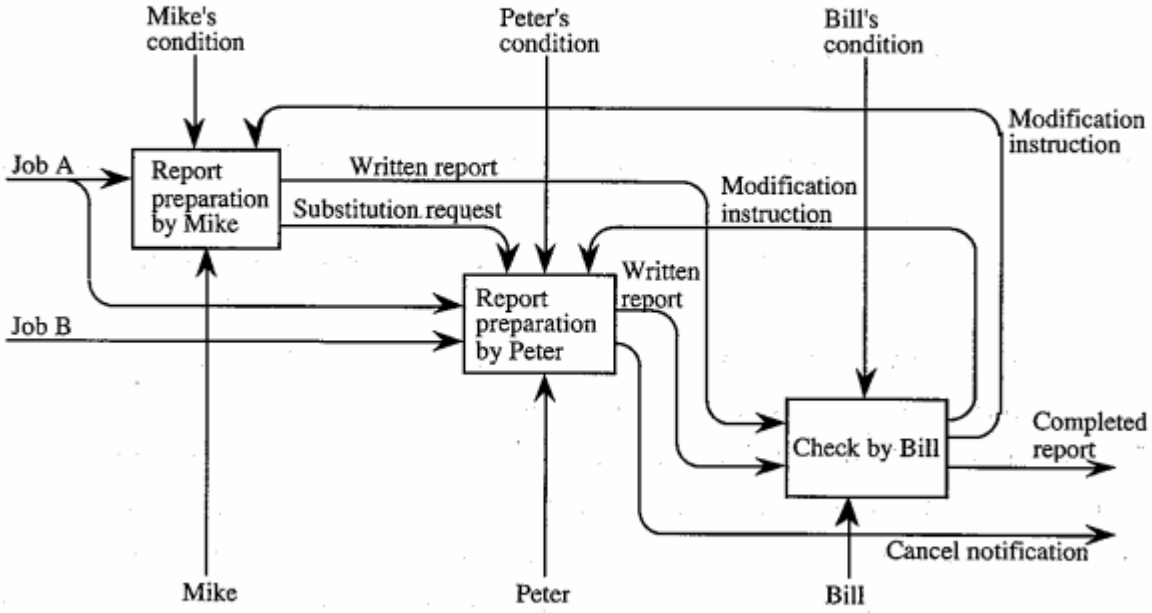

Fig. 6 IDEF0 representation for control and workflow of Fig. 2

C. Advantages and disadvantage of IDEF0 models

IDEF0 model put emphasis on the representation of the action. All of input, mechanism, control, subject and output are related to the action. Event and condition of ECA model are only a part of control in IDEF0 model. Control by automaton model also corresponds to control which selects the output which really used. Control of IDEF0 model is used for the following three purposes.

1) Start condition like ECA model.
2) Branch condition like finite automaton model.
3) Parameter of the action.

In finite automaton model only one action is defined. On the other hand in IDEF0 model action can be modified by control. It can be regarded as parameters for action definition. In IDEF0 model, these three kinds of control signals are not distinguished.

IDEF0 model can describe details of actions. Combination of actions can be expressed by combining a diagram, which can be easily understood by users.

As shown in Fig. 6, IDEF0 model can express both control flow and workflow in one diagram. It is easy to understand(see Fig. 2 for comparison), but one problem is that control flow and workflow are mixed. Although clean separation of the both may not be possible(see the example in the following paragraph), it is important to distinguish them in order to understand the meaning of a

diagram.

In a company some manager may be in charge of making a plan to do a job. His work is actually making control flow for other people. Senior managers must determine workflow of such managers. Thus there will be a hierarchy of control flows.

4. Generalized IDEF0 Model

In the previous section we compared ECA model, finite automaton model and IDEF0 model. It is required to generalize IDEF0 model due to the following reasons.

1) We must express starting conditions and events for actions in order to translate IDEF0 diagram into ECA rules.
2) Separation of control flow and workflow should be achieved as much as possible.

Further generalization is motivated by the following reasons.

3) Explicit expression of workflow branch control as shown in Fig. 2(a) should be realized: In Fig. 6 the condition when "substitution request" will be issued, is not explicitly shown.
4) Schedule on mechanism assignment should be required: For example, if there are only two pans and they are required by four actions, proper assignment of pans is required. We need to add proper scheduling procedures for mechanisms.

For the purposes of 1) and 2), we will define a generalized block as shown in Fig. 8.

Control has two kinds, one for start condition and the other for parameters and branch conditions. It is recommended to use separate arrows for parameters and branch conditions. We will use bold arrows to express start conditions and events. There are two kinds of outputs, result of the action and control for other blocks. We will distinguish the both by using dotted line for control output. In conventional IDEF0 start condition is not clearly expressed. It is assumed that actions defined by Subject will be started when all inputs are ready. We can start some actions if explicit statement for starting condition is given. If there is just one input, then action is assumed to start when the input is ready. In order to translate IDEF0 diagram into ECA rules we must explicitly show the start conditions. Fig. 8 shows a diagram using generalized IDEF0 model, which corresponds to Fig. 5.

As there are too many control arrows we will use the following simplified notation.

If there is no control corresponding to start conditions, we assume that the action must be started when the all inputs are ready.

By this assumption, we can remove most bold arrows in Fig. 8, still keeping the property that it can be translated into ECA rules.

For explicit expression for workflow branch control, we can use labels for outputs. Fig. 9 shows an example. Conditions a and b are given as Control. Output A will be produced if a is satisfied. Output B is produced if a · b(both a and b) is satisfied.

We can use another IDEF0 diagram which determines schedule for items of Mechanisms. Details

of workflow branch control and scheduling of Mechanisms are omitted here.

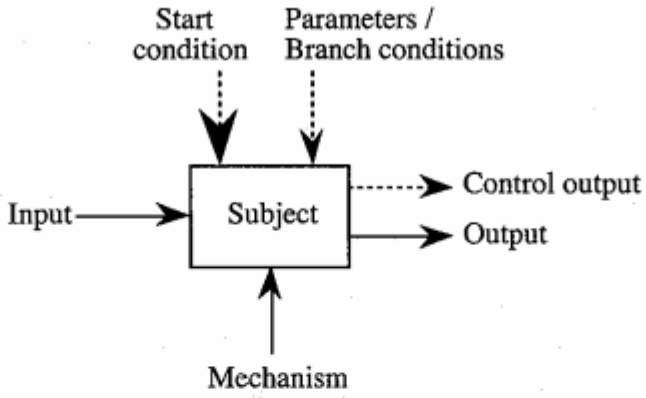

Fig. 7 A generalized IDEF0 block

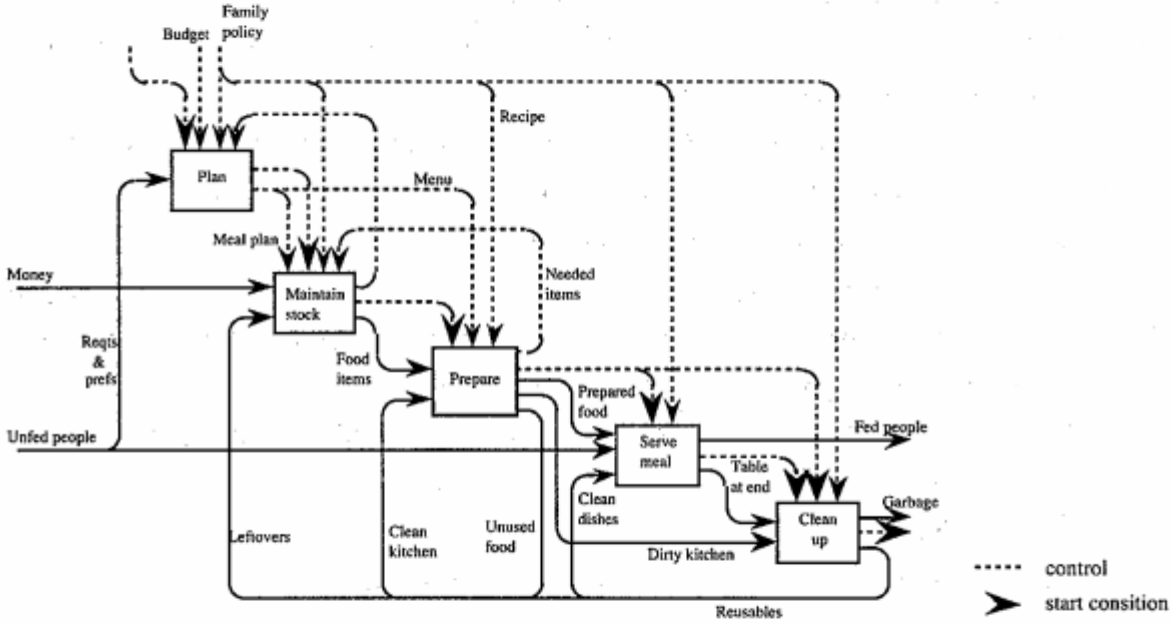

Fig. 8 Representation of feed family example using generalized IDEF0 model

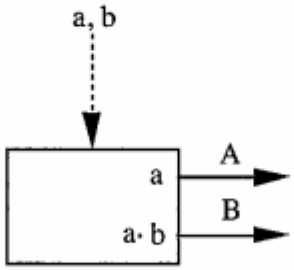

Fig. 9 Explicit expression of branch condition

5. Use of Generalized IDEF0 Model for User Coordination

Since it is not possible to determine or modify workflow automatically, negotiation among users is required. IDEF0 or generalized IDEF0 can be used to show the status to users in easily understandable form. Because users are involved, we need to handle dynamic modification of diagrams. For example, a user to whom a job is assigned may not available because he is out of the town.

There are typically the following two methods to handle dynamic modification.

1) Prepare all possible paths beforehand.
2) When some conditions are satisfied, the rule to modify the diagram is applied.

If there are many possibilities, method 1) is not practical. We will use method 2) in this paper.

There are the following cases for IDEF0 diagram modification.

1) Workflow diagrams are not modified. Only details of conditions are modified. Change of deadline dates is one example.
2) A part of a diagram is removed. For example, in Fig. 6, if Job A is processed by Mike, the block "Report preparation by Peter" will not be used, thus it can be removed. This kind of modification can be used in the following two cases.
    2-1) Simplification of a diagram. By removing blocks which are not used, only really used job flow will be shown. Furthermore, complicated diagrams can be simplified by removing less important control flows and/or mechanisms.
    2-2) Generation of user's view. Workflow related to a specific user is required to make his own schedule.
3) Addition of blocks and edges.
    3-1) Addition and deletion of edges. For example, new control is added or some mechanism is replaced.
    3-2) Replacement of partial diagrams. If some condition is satisfied, we need to replace a part of a diagram by another diagram. A simplified case is as follows.

        If condition C is satisfied, replace block B by diagram D.

We will consider the case when some job cannot be finished before the deadline.

1) Change of deadlines: In this case we need not modify the diagram. Only deadline values are modified.
2) Giving up the job: In this case the part of the diagram related to the job will be removed.
3) Modification of diagrams:
    3-1) Assign additional people to perform the job by adding these people as Mechanisms.
    3-1) Replace a block by a diagram which corresponds to a modified job or users' decision.

We will show an example corresponding to case 3-2). Consider the case to prepare a program

shown in Fig. 10 without ( ) parts(it will be explained later).

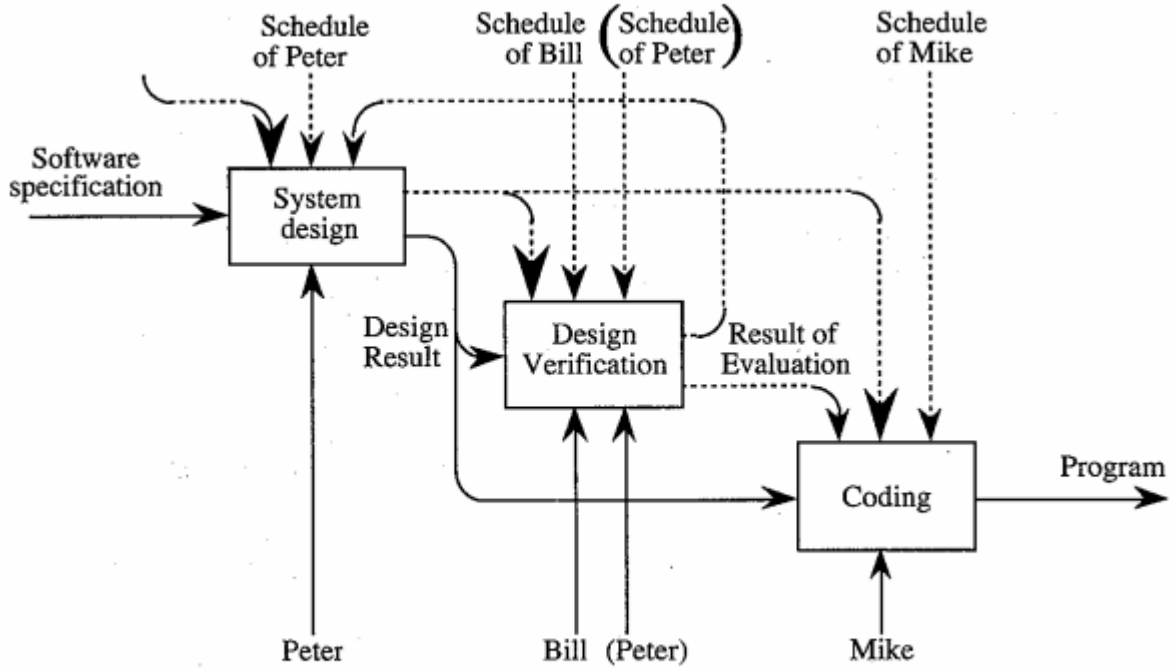

Fig. 10 A Workflow for Program Coding

Here design verification and coding are performed in parallel after system design. During the coding result of verification is given as control and if there is some problem, system design must be modified and coding must be modified accordingly.

If there is a deadline for design verification and it is found that verification may not be able to finish before the deadline, we need to assign additional persons for design verification. A block showing decision for help is generated and other users will help design verification. The modified diagram is shown in Fig. 11.

Fig. 12 shows an example of a diagram modification rule. When a job cannot be finished two days before the deadline, the rule in Fig. 12 will be applied. Newly added block and edges are shown by bold lines. A new decision block is added to determine modification. In parameter control only deadline condition is added to the decision block. Additional members correspond to all possible members who can help the job. Coordinators at the decision block will determine new deadline and people who will help the job, using the conditions of these additional members.

Fig. 11 shows the result of the application of the rule to Fig. 10. After a decision is made, simplification of diagram will be applied. In this case, if it is decided that only Peter will help the Design verification, we can generate a diagram where ( ) parts are added in Fig. 10.

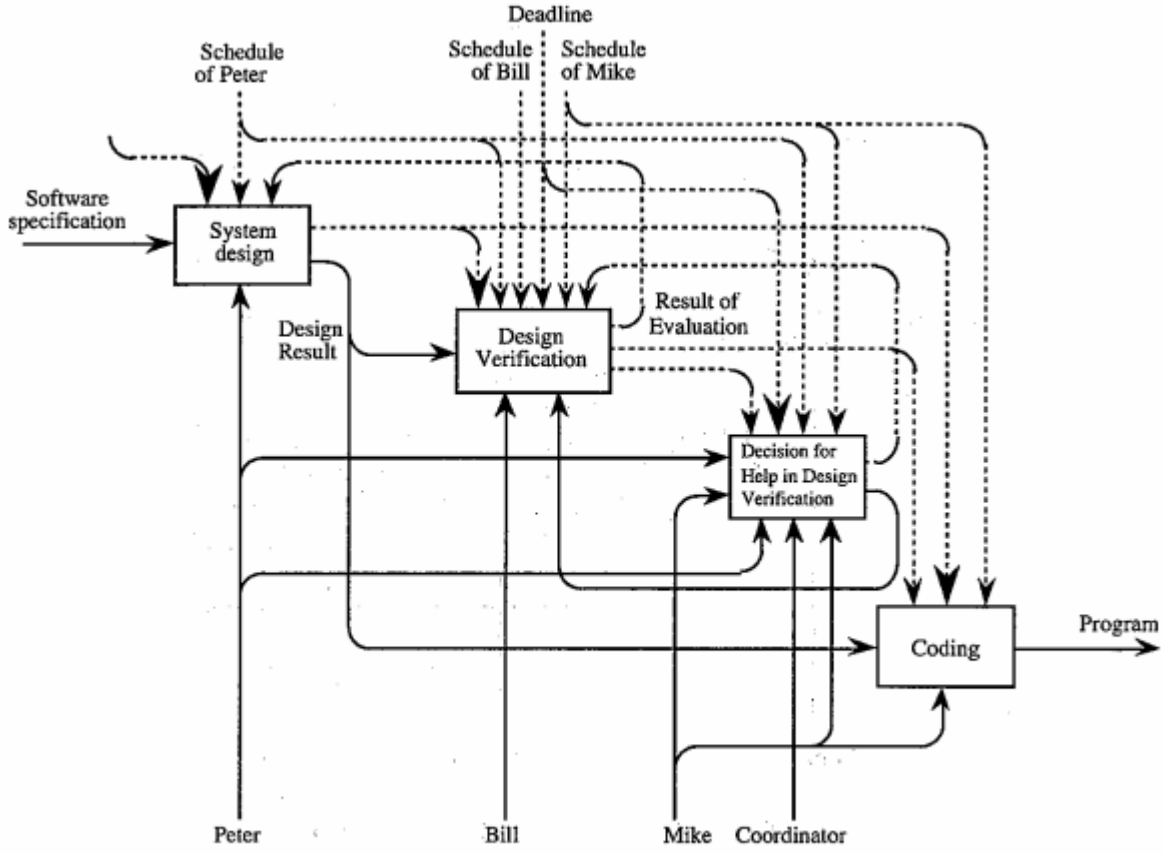

Fig. 11 Modified Workflow for Program Coding

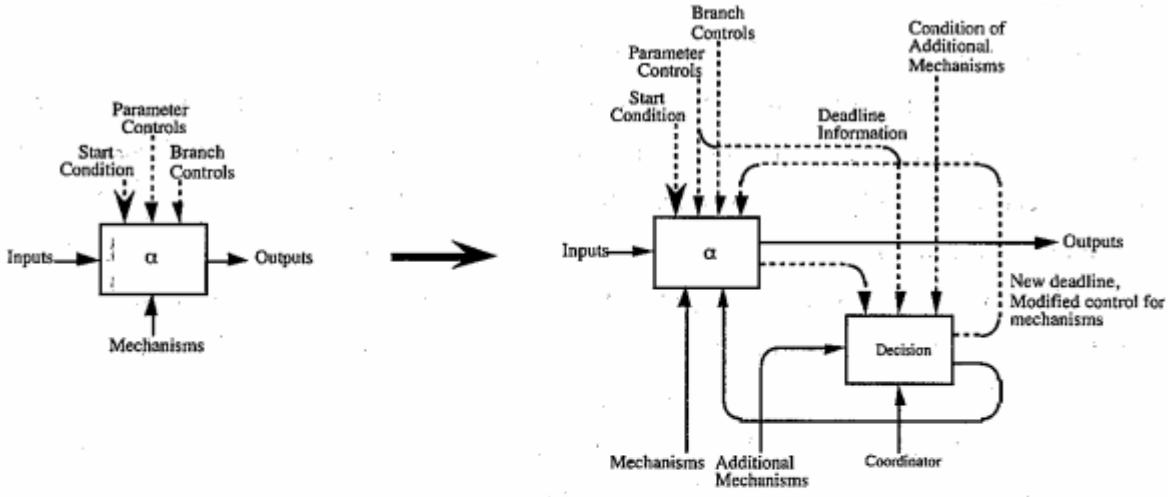

Fig. 12 A rule to modify IDEF0 diagrams

Users have to know the result of modification. We are developing a user interface utilizing generalized IDEF0 diagram which shows the followings.

1) If there is a branch, only the edge which really selected is shown.
2) If some decision is involved as shown in Fig. 11, the result of the decision is shown.

Using hierarchical representation, users can understand the current situation easily. We are

currently designing representative blocks and corresponding ECA rules. Using these blocks we can define workflow without complicated programs.

6. Concluding Remarks

In this paper problems of IDEF0 model are discussed by comparing with finite automaton model and ECA model, and a generalized IDEF0 model is proposed. It can be translated into ECA rules realized by database systems. Furthermore, it has rich expressive power for user decision and negotiation. The system will be used in VIEW Office which has been developed using Distributed Smalltalk and VisualWorks combined with an object oriented database system.

References

[Da88] U. Dayal et al.: "The HiPAC Project: Combining Active Databases and Timing Constraints", SIGMOD Record, Vol. 17, No. 1, March 1988.

[DBM88] U. Dayal, A. P. Buchmann, and D. R. McCarthy, "Rules Are Objects Too: A Knowledge Model For An Active, Object-Oriented Database System", In 2nd Intl. Workshop on Object-Oriented Database Syst., pp. 129 - 143, Springer-Verlag, September 1988.

[Esw76] K. P. Eswaran: "Specifications, Implementations, and Interactions of a Trigger Subsystem in an Integrated Data Base System", Research Report RJ1820, IBM, August 1976.

[KCK92] Y. Kambayashi, Q. Chen, T. Kunishima: "Coordination Manager: A Mechanism to Support Cooperative Work on Database Systems", Proceedings of the Second Far-East Workshop on Future Database Systems, pp. 176 - 183, World Scientific, April 1992.

[MD89] D. R. McCarthy and U. Dayal: "The Architecture of an Active Data Base Management System", In Proc. ACM SIGMOD Intl. Conf. Management of Data, pp. 215 - 224, June 1989.

[MM93] D. A. Marca and C. L. McGowan: "IDEF0/SADT Business Process and Enterprise Modeling", Electronic Solutions, 1993.

[Pe81] J.L. Peterson: "Petri Net Theory and the Modeling of Systems", Prentice-Hall, 1981.

[PK95] Z. Peng and Y. Kambayashi: "Deputy Mechanisms for Object-Oriented Databases", Proceedings of the International Conference on Data Engineering, 1995.

[RS94] W. Reinhard, J. Schweitzer, G. Volksen, and M. Weber: "CSCW Tools: Concepts and Architectures", Computer, Vol.27, No.5, pp.19-27, May 1994.

[TK93] H.Takada and Y. Kambayashi: "An Object-Oriented Office Space Description Model and an Office View Management Mechanism for Distributed Office Environment", Foundation of Data Organization and Algorithms, pp.362-377, Springer Verlag, 1993.

[Wi88] T. Winograd: "Where the Action Is", BYTE, Vol. 13, No. 13, pp. 256A-258, December 1988.