December 1993

A Basis of Information Systems Analysis and Its Application to Prototyping

Ryo Sato
University of Tsukuba

A Basis of Information Systems Analysis And Its Application to Prototyping

Ryo Sato
Institute of Socio-Economic Planning, University of Tsukuba,
Tsukuba ciy, Ibaraki 305, Japan

rsato@sk.tsukuba.ac.jp

Keywords

business transaction system, information systems methodology, data flow diagram, entity life cycle history, state space representation, static prototype specification, analysis repository system

Abstract

In this paper a general model of business systems is proposed. The model describes the whole mechanism of typical routine processing of slips and business papers as a processing of transactions. Furthermore it provides a rigorous basis for two of the structured systems analysis tools.

The model proposed is called a business transaction system. It consists of both static and dynamic structures. The static structure depicts interconnection among transaction names and a file system for routine transaction processing. The dynamic structure is constructed as a state space representation by introducing a state space, and then the whole dynamic system is a discrete event system. The state space consists of a file system and a schedule of internal transactions to finish. The file system defines conditions on which internal transactions start processing. That is, it is a mediator of interaction of transactions and then logically synchronizes internal transactions.

In order to demonstrate that a business transaction system provides a theoretical basis for information systems analysis, we investigate the meaning of tools such as the dataflow diagram(DFD) and the entity life history(ELH). It is shown that the DFD is a static model of business processing and that the ELH partially describes dynamic behavior of the system.

Based on the consideration above we have designed a CASE tool whose primary concern is requirements definition. It is called an analysis repository system and implemented on a 4GL(4th generation language). The data structure of the repository will provide a minimal model of static structure of a business transaction system and then makes prototyping of the structure possible through manipulation of SQL tables.

I. INTRODUCTION

There are many information systems methodologies which are used in analysis of information systems in organizations. Each of them has its own concepts and technical tools which are used in making documents. In a typical analysis process of a methodology, a business system is described as dataflow diagrams, specifications of information systems that are supposed to support the business system are decided, a data model and a process specification are specified, then designed, and an information system is implemented by a database management system on a computer network. However, each methodology does not seem to possess a formal model of business system which information systems are supposed to support, except formatted documents. We need a model that provide a formal and rigorous basis to business analysis and requirements identification in information systems methodologies.

Olle(1988) and Brough(1992) have similar orientation to remedy this situation. Olle has proposed a framework to describe concepts used in many information systems methodologies in a unified way. The framework is a variant of the entity relationship model and describes concepts, techniques and their relations in tables. Brough provided the $F^{I}T$ framework to give a perspective of tools of methodologies.

In this paper we will show a general model of business system and the way how to provide a definite conceptual basis for tools of the structured analysis.

The model is called a business transaction system and is a discrete event system whose state space representation has a file system as part of its state space. The file system defines conditions on which internal transactions start processing. That is, it is a mediator of interaction of transactions and then logically synchronizes internal transactions. It will be understood that one of the fundamental reasons why different analysts have different designs of a file system is a manifestation of the fact that there are infinitely many state representations for an input-output system such as a discrete event system, as shown in the mathematical genaral systems theory (Mesarovic and Takahara, 1975). In order to demonstrate that a business transaction system provides a theoretical basis for information systems analysis, we investigate the meaning of tools such as the dataflow diagram(DFD) and the entity life history in sections 5 and 6, respectively. It is shown that the DFD is a static model of business processing and that the entity life history partially describes dynamic behavior of the system).

Based on the consideration above we have designed a CASE tool whose primary concern is requirements definition. It is called an analysis repository system and implemented on a 4GL(4th generation language). The data structure of the repository will provide a minimal model of static structure of a business transaction system. We will show the structures of of SQL tables which make prototyping of the structure possible.

II BUSINESS TRANSACTION SYSTEM

2.1 Data model

The DAE data modeling facility is used to describe the structure of file systems. DAE is a simplification of the TH data model (Hotaka, 1989). The TH model has composite domains but DAE dose not. DAE is an abbreviation for Domain-Attribute-Entity. A file system is a set of files which satisfy its consistency condition defined by referential relations and inheritance.

The DAE has three kinds of entity types: domains, controlled entity types and file types. Each file type can be represented by a table. A table consists of two parts. As in Fig. 1 one is used for the definition of the table itself and the other for data in it.

y			
a_1	a_2	a_3	a_4
v_{11}	v_{12}	v_{13}	v_{14}
v_{21}	v_{22}	v_{23}	v_{24}
v_{31}	v_{32}	v_{33}	v_{34}

definition of this table

data in this table at present

Fig 1 Table

File type y and its attributes a_1, a_2, a_3, a_4 are needed for the definition of the table. A row in the table is called an occurrence. The set of file types is denoted by FTS and the set of controlled entity types KS. Each controlled entity type corresponds a file type. When a file type y corresponds to a controlled entity type x we write as FT(x) = y. The fact that a_1, a_2, a_3, a_4 are the attributes of a file type y is written by A(y) = {a_1, a_2, a_3, a_4}. The set of all attributes in DAE is denoted by AS. FS is the set of all file contents functions. Each file contents function f gives to any file type y its data contents of the table. That is, f(y) is the data part of y's table representation. If one of insert, modify or deletion is executed to a file type y, then the resultant data part of the table is given as f'(y) by some appropriate file contents function f'. f(FT(x)) is sometimes written as f(x) for notational simplicity. DAE data modelling facility urges each file type to have primary key attribute, and it is used to define referential relation and inheritance one. The two relations, with KS and their attributes, make the file system.

Definition 1 Primary key

Let x be an arbitrary key entity type. Let key:FTS →AS be a function which
satisfies the following condition:

For arbitrary file contents function f and occurrences z_1, $z_2 \in$ f(FT(y)), we have $z_1 = z_2$, whenever $V_f(z_1, key(FT(x))) = V_f(z_2, key(FT(x)))$ holds.
Then key(FT(x)) is called a primary key of x, or simply a key of x.

In the above definition $V_f(z, a)$ is a function that gives the correspondent value of the attribute a in the occurrence z. In DAE each key entity type has exactly one key, and it is always assumed that dom(key(FT(x))) = x holds.

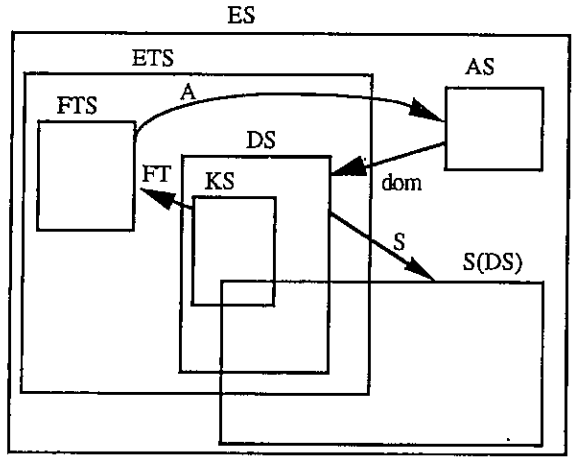


Fig. 2 The DAE data model

Definition 2 Referential key

Let x_1 and x_2 be arbitrary elements of KS and f∈ File arbitrary file contents function. An attribute a ∈ A(FT(x_2)) is called a referential key of FT(x_2) to FT(x_1) in f or FT(x_2) refers to FT(x_1) in f if and only if

(1) dom(key(FT(x_1))) = dom (a), and

(2) $V_f(f(FT(x_2)), a) \subseteq V_f(f(FT(x_1)), key(FT(x_1)))$ holds.

Referential keys are used to express the referential integrity which have to be held among attributes of files. The integrity defined by referential keys is called the referential integrity. The set and functions in DAE are depicted in Fig. 2. For the full definition of DAE, see Sato(1993).

2.2 Static structure of a business transaction system

There are three kinds of transactions. External, internal dispatch and internal finish transactions.

Definition 3 External transaction

E is the set of external transaction names. E is finite.
RE is the set of external roles.
f_{RE} : RE → E. The function f_{RE} is a bijection and represents what role causes to happen an external transaction.
f_{EK} : E → P(KS) - A. This specifies the key entity types that have records of occurrence of external transactions.

Definition 4 Internal finish transaction

I is the set of internal finish transaction names. I is finite.
RI is the set of internal roles.
f_{RI} : RI → I. The function f_{RI} is a bijection and represents what role causes to finish an internal transaction.
f_{IK} : E → P(KS) - A. This specifies the key entity types that has records of occurrence of internal finish transactions.

Definition 5 Internal dispatch transaction

D is the set of internal dispatch transaction names. D is finite.
f_{DK} : D → P(KS) - A. The key entity types specified by this function change when dispatch transactions occur.
f_{DI} : D → I. This specifies an internal finish transaction f_{DI}(d) which shows the end of the process started by the dispatch transaction d∈ D.

The occurrence of dispatch transactions means the start of some internal tasks. Those tasks cannot start until some event arrives or appropriate data are provided, such as orders from customers or the finish of a machine operation at a stage of production.

We assume that E, I and D are mutually disjoint.

Definition 6 Static structure of a business transaction system

A static structure of business transaction system is a class of file system, external transactions, internal finish transactions and internal dispatch transactions.

The static structure defined above is a description of a business system from the point of view that a business system is a transaction processing system used together with corresponding roles and structured data(file). Consideration in Section 2.3 shows that the static structure defines an outlook of the dynamic mechanism of a business transaction system.

The static structure can be depicted in simple way, though it does not show the attributes of controlled entity types. Fig. 3 and 4 show an example and the meaning of pictorial components, respectively.

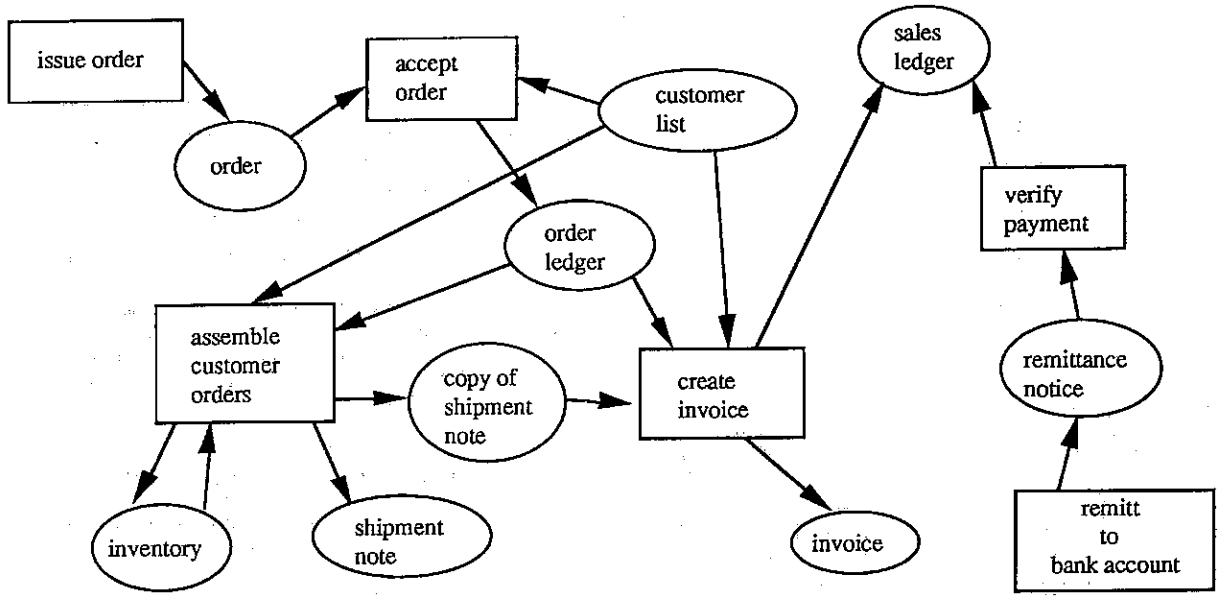


Fig. 3 Sales and invoicing

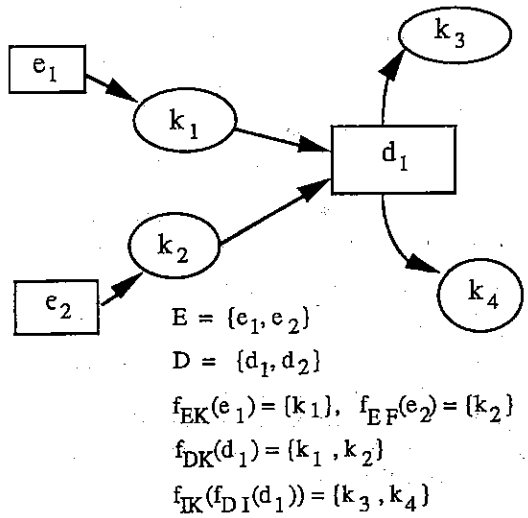


Fig. 4 Static structure of a business transaction system

2.3 Dynamic mechanism of a business transaction system

A system is a relation between input and output. If input and output are respective sets of time functions then the system is called time system. State space representations are widely employed to describe causal behavior of time systems. Let x be a function from a time set T to a some set. For any t, t'∈ T, t ≤ t', the restriction of domain of x to [t, t') is written as $x_{tt'}$. State space is a key concept by which we can grasp the behavior of the dynamic system.

Definition 7 State space representation(Mesarovic et al,1974)

Let S⊆ X×Y be a time system and C an arbitrary set. C is a state space for S if and only if there exist a family of functions $\Phi = \{\phi_{tt'} \mid \phi_{tt'} : C \times X_{tt'} \rightarrow C$ and t, t'∈ T, t ≤ t'} and a function $\mu : C \rightarrow B$ such that

(i) S = {(x, y) | there exists some c∈ C such that for any t∈T $y(t) = \mu(\phi_{0t}(c, x_{0t}))$}

(ii) for any t, t', t''∈ T

(α) $\phi_{tt''}(c, x_{tt''}) = \phi_{t't''}(\phi_{tt'}(c, x_{tt'}), x_{t't''})$

(β) $\phi_{tt}(c, x_{tt}) = c$

(γ) $\phi_{tt'}(c, x_{tt'}) = \phi_{0\tau}(c, \sigma^{-t}(x_{tt'}))$

where $x_{tt''} = x_{tt'} * x_{t't''}$ and $\tau = t' - t$.

The pair <Φ, μ> is called a (time invariant) state space representation of S, and Φ a time invariant transition family.

Mesarovic and Takahara(1975, 1989) has shown that a time system is causal if it has a state space representation.

A discrete event system is a special time system defined as follows:

Definition 8 Discrete event system (Sato1991a)

If a time system S⊆ X×Y satisfies the following conditions then it is called a discrete event system:

(i) S is stationary.

(ii) The input alphabet A of the input space X is P(A'), the power set of a finite set A'.

(iii) For any x∈ X and t, t'∈ T, t< t', event(x)∩[t, t') is finite, where event(x) = {t | x(t) ≠ A}.

(iv) For any y∈ Y and t, t'∈ T, t< t', F(y)∩[t, t') is finite and ∪ (F(y)∩[t, t')) =[t, t'), where F(y) = {[r, s) | y(r) = y(t'') for any t'', r ≤ t'' < s, and y(r) ≠ y(s)}.

Input set X which satisfies the above conditions (ii) and (iii) is called a discrete event input space.

Now taking the sate space as $File \times F_S$, state transition functions $\Phi = \{\phi_{tt'} \mid \phi_{tt'}: File \times F_S \times X_{tt'} \rightarrow File \times F_S$ and t, t' ε T, t ≤ t'} and the projection Π_F, where Π_F is the projection on $File \times F_S$ along File and $F_S = \{s \mid s:RI \rightarrow T^{\infty}\}$, form a state space representation. Though we cannot show the exact mechanism here

because of the limits of pages, Fig. 5 and 6 reveals very briefly the state transition. The formal definition is in Sato(1993).

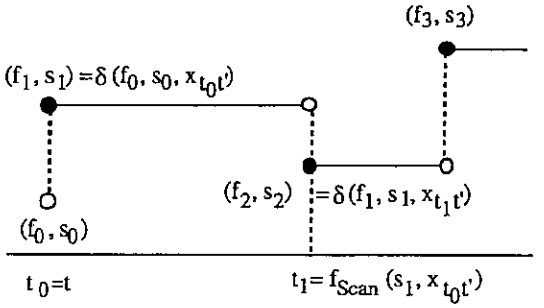


Fig.5 Behavior of a business transaction system

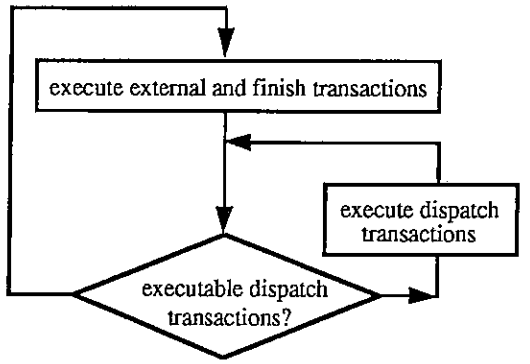


Fig. 6 Flow chart of processing of a dynamic structure

The following theorem shows that the resultant system of the dynamic mechanism of business transaction system is a discrete event system.

Theorem 1

Let $<\Phi, \Pi_F>$ be a dynamic structure of a business transaction system with a static structure. Then $Res(<\Phi, \Pi_F>)$ is a discrete event system.

In the above theorem $Res(<\Phi, \Pi_F>)$ is called the resultant system of $<\Phi, \Pi_F>$ and defined as $Res(<\Phi, \Pi_F>) = \{(x,y) | y(t) = \Pi_F \cdot \Phi_{0t}(c, x_{0t})$ for any $t \in T\}$. The file system in a business transaction system plays a role so that it makes parallel processing of transactions possible in a certain way in an organization. General systems theory (Mesarovic and Takahara 1975, 1989) has shown that there are unlimited number of state space representations of a system if it has one. The theorem shows the file system is part of a state. Thus it is not surprising that there is broad room in decisions on what controlled entity type are needed, how to allocate attributes into controlled entity types and then what integrity conditions to be defined. That kinds of decision will be taken care of by information systems methodologies.

III. CHARACTERIZATION OF DATAFLOW DIAGRAM

In this section, accroding to Sato(1992), a dataflow diagram depicts is investigated in relation to the business transaction system.

3.1 Formulation of data flow diagram

Below we refer the term data flow diagram (abbreviated as DFD) as the one in the most detail resolution.

Definition 9 Dataflow diagram - DFD (DeMarco 1979, Gane et al 1979)

Sink: a finite set of data sinks.
Source: a finite set of data sources.
Process: a finite set of processes.
Datastore: a finite set of data stores.
The above four sets are mutually disjoint.

Arrow $\subseteq$(Source×Process)$\cup$ (Process×Process)$\cup$ (Process×Sink)$\cup$ (Datastore×Process)$\cup$ (Process× Datastore). Every ordered pair in Arrow represents a DFD. The set Arrow is a finite set of ordered pairs which satisfies the following conditions.

1) For an arbitrary s$\in$ Sink there is a process p such that (p, s)$\in$ Arrow.

2) For an arbitrary s$\in$ Source there is a process p such that (s, p)$\in$ Arrow.

3) Process = {p | f = (j, p) and f'=(p, j') for some f, f'$\in$ Arrow and j, j'$\in$ Source $\cup$Process $\cup$Sink$\cup$ Datastore}.

4) Datastore = { d | f = (j, d) and f'=(d, j') for some f, f'$\in$ Arrow and j, j'$\in$ Process}.

a:Arrow $\cup$ Datastore $\rightarrow$ BusinessPapers, where BusinessPapers = P(Attribute) and Attribute is a set of attributes defined in a DFD. The function a is a correspondence between each arrow and its attributes called the attribute function. The attribute function satisfies the condition that for any f = (p, d)$\in$ Arrow$\cap$(Process× Datastore) a(f) $\subseteq$a(d) holds.

Dataflow = {(f, a(f)) | f $\in$ Arrow}: a dataflow is a pair of an arrow and its attributes.
A dataflow diagram is a group (Sink, Source, Process, Datastore, Arrow, Dataflow).

3.2 Transformation of data flow diagram into static structure

For an arbitrary dfd = (Sink, Source, Process, Datastore, Arrow, Dataflow) we can define a primitive static structure S=(KS, AS, A, E, f_{EK}, I, f_{IK}, D, $f_{D\,I}$, f_{DK}). This correspondence is denoted by F(dfd) = S.

Fig. 7 is an illustration of F. For the formal definition, see Sato(1992). Some of characteristics of the transformation F is as follows. Data sinks are disappeared. There could be many arrows from Source, because a source is often an actual existence in the real world. In F(dfd) each arrow from external source is an external transaction. In the case of branch the arrow will become an one controlled entity type. Also the attributes from or to a data store are merged into the attributes of the controlled entity type that corresponds to the data store.

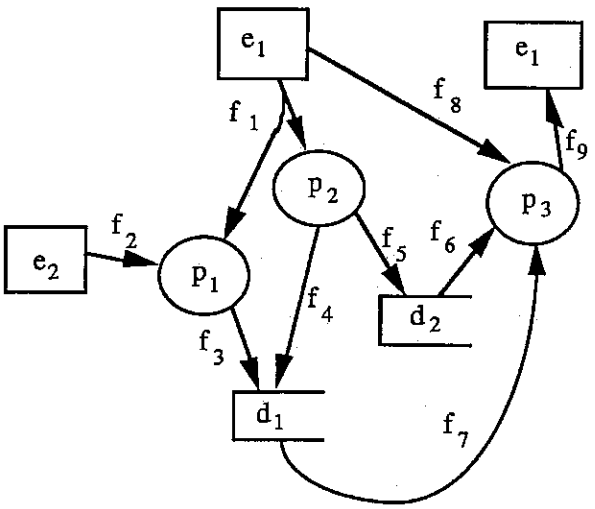
Proposition 2

For an arbitrary dfd, F(dfd) is a primitive static structure.

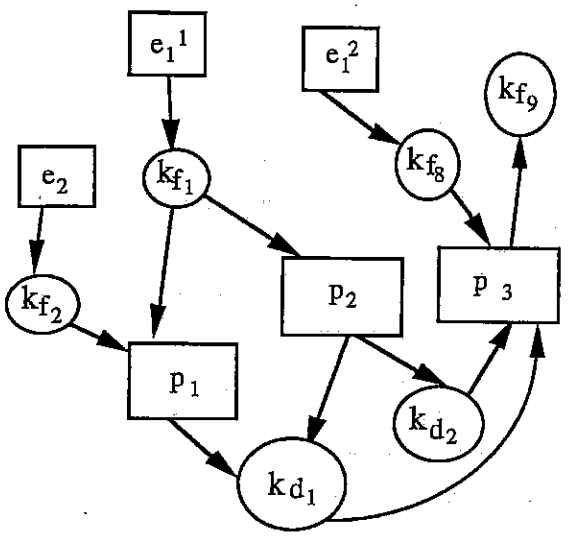
Sato(1992) showed that there is an opposite correspondence G of primitive static structure into DFD. And F and G are virtually mutual inverse correspondence. That fact shows that DFD and primitive static structure convey the almost same

information.

Since any sink disappears by F, each sink does not have any dynamic information of business transaction system.



(a) A DFD



(b) Corresponding primitive static structure

Fig. 7 Illustration of the transformation F from a DFD to a pss

Through the correspondence F we can observe that dataflows and data stores decide start conditions of internal processes and then make parallel processing possible. Observing this fact from opposite direction, we can say that in making data flow diagram we can get rid of data which has nothing to'do with start condition of internal transactions.

Not only data stores but also data flows have to be recorded as controlled entity types because they decide behavior of a business transaction system.

IV. ENTITY LIFE HISTORY OF A BUSINESS TRANSACTION SYSTEM

In this section, the entity life history(ELH), a tool for modeling the dynamic aspect of a business transaction processing is formulated. It will be shown that the ELH is actually a set of languages of transactions in a business transaction system which provides partial description of the behavior of business transaction processing.

Some authors have provided similar tools for the dynamic description of business transaction processing. These tools are the entity life history by Cutts(1987), Peters(1987) and McDermid(1990), the Petri net by Cutts(1987) and the state dependency graph by McDermid(1990). Even though these concepts have something in common, they do not seem to be exactly the same. That is the reason why we should start from the definition of the ELH. However, at first, for readability, we show an informal account of the ELH with respect to the behavior of a business transaction system.

Let $<\Phi, \Pi_F>$ be a state space representation of a business transaction system. Let x be an arbitrary discrete event input and (f, s) an initial state. We can think of the state trajectory which is made from x and c = (f, s). Then there is the sequence of transactions in that trajectory which is denoted by $seq_\Phi(c, x)$. We can extract transactions related to each key entity type k. That is, a sequence of transactions is extracted and a new subsequence can be formed such that the following conditions hold:

$e \in E$ is extracted if and only if $k \in f_{EK}(e)$,

$i \in I$ is extracted if and only if $k \in f_{IK}(i)$,

$d \in D$ is extracted if and only if $k \in f_{IK}{}^* f_{DI}(d)$.

This is an entity life history of the key entity type. If we denote this extraction by ψ then we can denoted an entity life history of the key entity type k by $L(k, <\Phi, \Pi_F>) = \{\psi(k, l_1)\psi(k, l_2)\psi(k, l_3)... \mid l_1 l_2 l_3... \in L(<\Phi, \Pi_F>)\}$.

Definition 10 Sequential equivalence

Let $<\Phi, \Pi_F>$ and $<\Phi', \Pi_F>$ be dynamic structures of a business transaction system with the same static structure. They are called sequentially equivalent if $seq_\Phi(c, x) = seq_{\Phi'}(c, x)$ holds for any $x \in X$ and $c \in File \times F_S$.

We can state a basic characterization of a dynamic structure $<\Phi, \Pi_F>$ by its language $L(<\Phi, \Pi_F>) = \{seq_\Phi(x, c) \mid x \in X$ and $c \in File \times F_S\}$.

Theorem 3

Let $(f_{FD}, f_{TraEI}, f_{TraD}, f_{Dis}, \Phi, \Pi_F)$ and $(f'_{FD}, f'_{TraEI}, f'_{TraD}, f'_{Dis}, \Phi', \Pi_F)$ be dynamic structures of a business transaction system with the same static structure. Assume that $f_{TraEI} = f'_{TraEI}$, $f_{TraD} = f'_{TraD}$ and $f_{Dis} = f'_{Dis}$ hold. Then the following statements are equivalent:

(1) $<\Phi, \Pi_F> = <\Phi', \Pi_F>$;

(2) $<\Phi, \Pi_F>$ and $<\Phi', \Pi_F>$ are sequentially equivalent.

The following shows the relation between the equivalence of dynamic structures and that of their languages of key entity types. It shows an ability of the entity life history to describe the dynamic mechanism of a business transaction system.

Theorem 4

Let $<\Phi, \Pi_F>$ and $<\Phi', \Pi_F>$ be dynamic structures of a business transaction system with the same static structure. If $<\Phi, \Pi_F>$ and $<\Phi', \Pi_F>$ are sequentially equivalent then we have $L(k, <\Phi, \Pi_F>) = L(k, <\Phi', \Pi_F>)$ for each key entity type $k \in KS$.

Corollary

Let $<\Phi, \Pi_F>$ and $<\Phi', \Pi_F>$ be dynamic structures of a business transaction system with the same static structure. If there exists a key entity type $k \in KS$ such that $L(k, <\Phi, \Pi_F>) \neq L(k, <\Phi, \Pi_F>)$, then $<\Phi, \Pi_F> \neq <\Phi', \Pi_F>$.

It is easily seen for $<\Phi, \Pi_F>$ and $<\Phi', \Pi_F>$ with the same

static structure that even if L(k, <Φ, Π_F>) = L(k, <Ψ', Π_F>) holds for each key entity type k$\in$KS, still <Φ, Π_F> can not be sequentially equal to <Ψ', Π_F>. This fact, together with the corollary above, shows the ability and limitation of the entity life history.

Cutts(1987), Peters(1987) and McDermid(1990) have explained the entity life history as a set of pictures. Each picture relates the completeness of operations on a file to sequences of access processes. Thus the picture seems to play a role similar to that of a set of rules that generate a language. We have formulated an entity life history as a language of transactions.

V. PROTOTYPE SPECIFICATION

5.1 Static prototype specification

Fig. 8 shows the process of information systems development. It consists of stages and products of them. the "what" axis in Fig. 8 represents the view point from that our primary concern is in required things and required functions. The "how" axis show the technical way how the things or functions can be realized. The horizontal axis in Fig. 8 consists of "real world" and "logical world." The former indicates that the stages are carried out in the real world, and the latter represents that at the corresponding stages one handles logical or abstract objects.

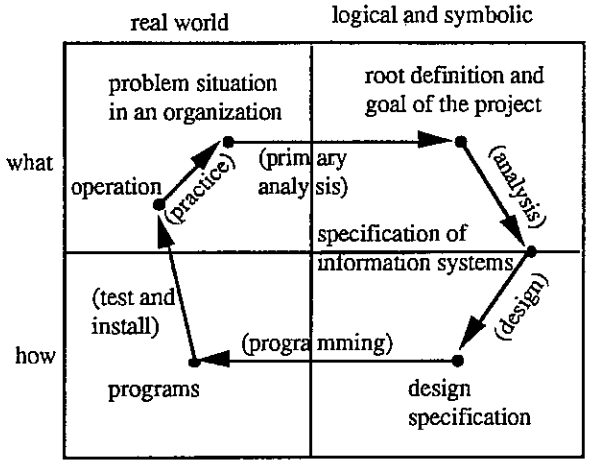


Fig. 8 Process of information systems development

Prototyping approaches are proposed for information systems development. Vonk(1990) insists that it can be used for requirements analysis and that there are three major goals of prototyping approach to requirements analysis. They are designs and constructions of a database, those of dialog structure (menu) and those of business functions.

We divide requirements analysis into two levels. Both correspond to static and dynamic structure of a business transaction system. And prototyping without programming is possible only for the static structure. Therefore a prototype of the static structure of a business transaction system is not only the prototype of an information system but that of the business transaction system. The prototype is called a static prototype specification. In this paper we will show the definition and implementation of that kind of specification. The static prototype specification consists of two kinds of data. They are a static structure of a business transaction system and structured screens of an information system used in it. Since they are modeled in DAE datamodel, we can implement the screens of the information system. That is, we can implement the static relation between business functions, business data and screens of the target information systems directly from analysis products without any programming. In other words, we can use the static prototype of an information system at analysis stage. Fig. 9 shows the process of the analysis where an static prototyping specification is utilized.

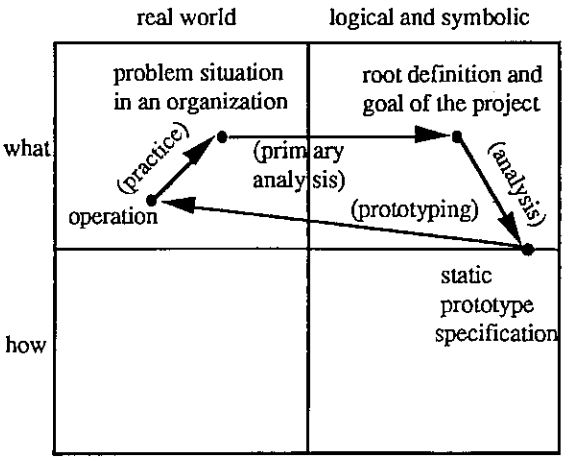


Fig. 9 Process of analysis with static prototype specifications

The product of analysis stage is static prototype specifications which are defined as follows.

(1) Dataflow diagrams. This is a static structure of a business transaction system.
(2) A meta-database. This represents a data model of transaction data in a business transaction system in the form of DAE data model.
(3) A set of screens. Each of the screens consists of its name, the names of business functions, and the names of data that those functions use.

5.2 Meta data for a static prototype specification

In order to make an implementation of a static prototype specification on a 4GL, all data of a static prototype specification are stored into tables. We use four groups of them to store the products of analysis. They are as follows.

(1) Tables for needs of information systems
(2) Tables for the static structure of a business transaction system
(3) Tables for transaction data
(4) Tables for specifications of screens

The relation of these four groups of tables is shown in Fig. 10. Each arrow in Fig. 10 represents an foreign key reference.

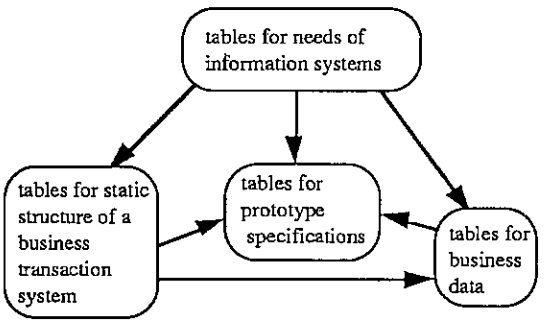


Fig. 10 Referential relation of four groups of tables

(1) Tables for needs of information systems
The root-definition-table stores the names of information systems under concern. The version table has the version numbers used so far.

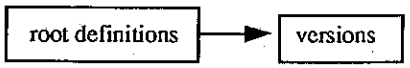
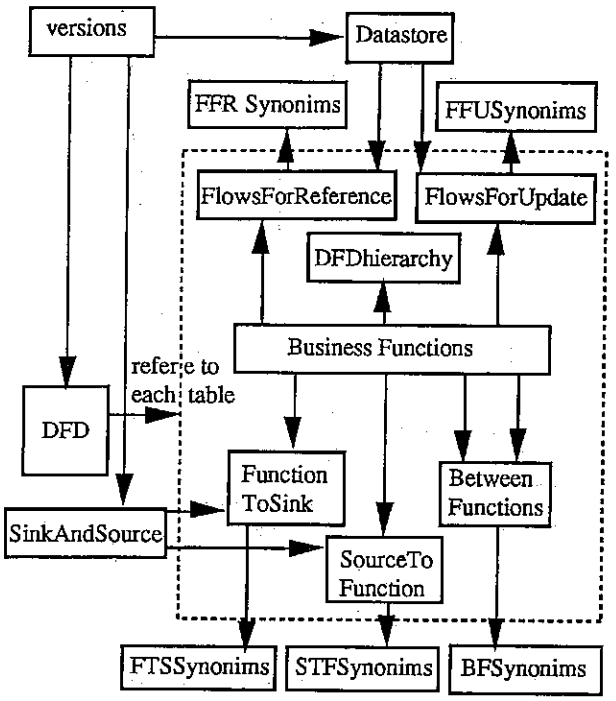


Fig . 11 Tables for needs of information systems

(2) Tables for the static structure of a business transaction system
The tables in this group are depicted in Fig. 12. There are eight tables which represent dataflow diagrams and seven tables to represent hierarchy of the diagrams.



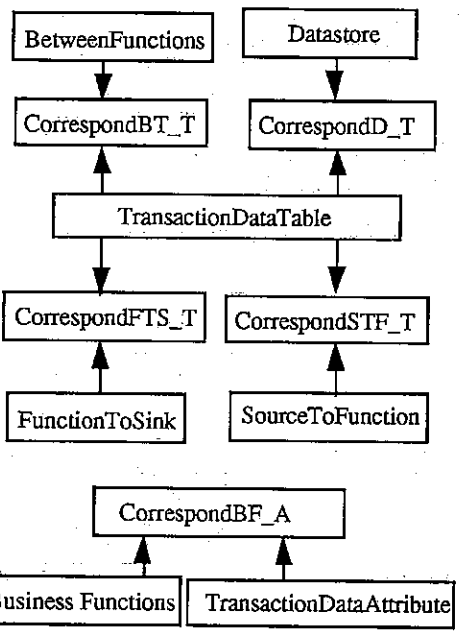


Fig. 12 Tables for the static structure of a business transaction system

(3) Tables for transaction data
The tables in this group are shown in Fig. 13. They are tables for domains, transaction data, attributes of transaction data, and referential relations of these data.

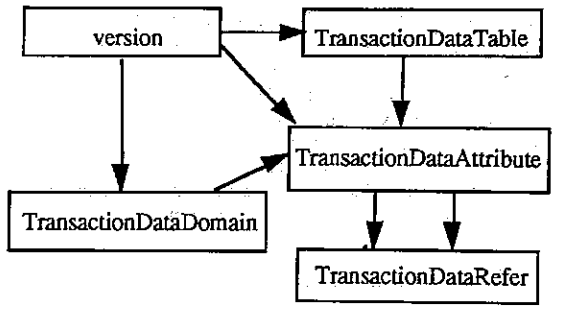


Fig. 13 Tables for transaction data

(4) Tables for specifications of screens
Fig. 14 shows these tables. They are tables for names of screens, links of screens and relations between screens and names of business functions. The data in these tables define names of the target information systems and are used for building prototypes of screens, together with the data in the other groups of tables.

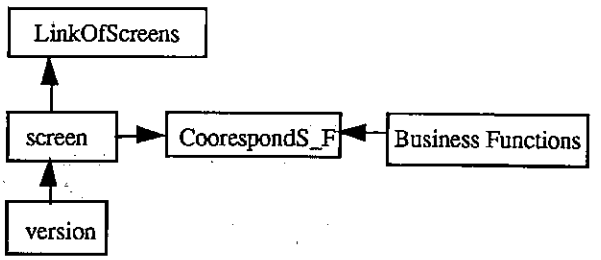


Fig. 14 Tables for specifications of screens

VI. ANALYSIS REPOSITORY SYSTEM

6.1 Functions of analysis repository system

Below we will show the four functions of the analysis repository system we have developed on a 4GL.
(1) Data management of the product of analysis
All of the product of analysis are stored in the form of SQL tables.
(2) Checking rules for integrity
There are some rules in describing dataflow diagrams and DAE datamodel. Some SQL queries are used to check that the tables are consistent with those rules.
(3) Impact analysis
If a part of the analysis product is changes then the influence of it should be examined so that the product as a whole remain meaningful and logically valid. The range of the influence is checked by SQL queries taking advantage of referential relations of tables.
(4) Making prototype specifications of screens
This function enables us to build a prototype specification of a screen of an target information system. You, as an analyst, have to decide how many screens are needed for what business functions, and what data should be accessed through the screens. A prototype specification of a screen consists of the followings.

(a) The name and the identification(key) number of this screen.
(b) The identification(key) numbers of screens to which a user can go from this screen.
(c) The names of business functions which are to be carried out on this screen.
(d) The names of files to which a user can access from this

screen.
A prototype specification is made through manipulation of SQL tables.

6.2 Implementation of analysis repository system

We have used HyperCard(TM) and Oracle for Macintosh(Oracle corporation) to realize the analysis repository system. Oracle is a database management system and has characteristics of 4GL. It has SQL engine to define and manipulate tables. Fig 15 shows the overview of software used for the analysis repository system.

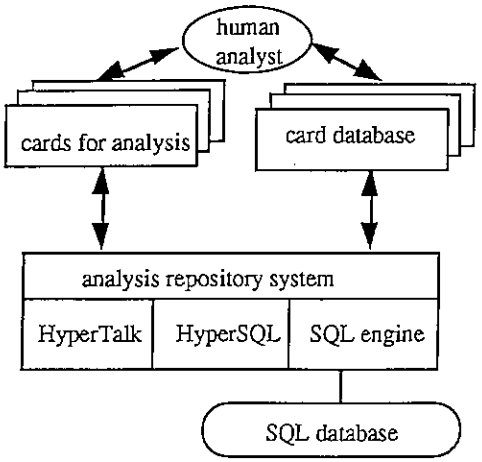


Fig. 15 Overview of analysis repository system

A stack is a group of cards on which a user can manipulate Oracle database. The analysis repository system has five stacks.

(a) Life Cycle Dictionary Stack
(b) DFD Stack: This stores dataflow diagrams as tables and check the integrity between dataflows.
(c) Data modeling Stack: This stores tables of business data and checks the integrity conditions of DAE datamodel.
(d) Prototype Specification Stack
(e) Relation Stack: The correspondence between dataflows and business data is decided on this stack.

6.3 Case study

Inventory control of medical materials in a hospital is analyzed. The product of analysis is stored in the analysis repository system.

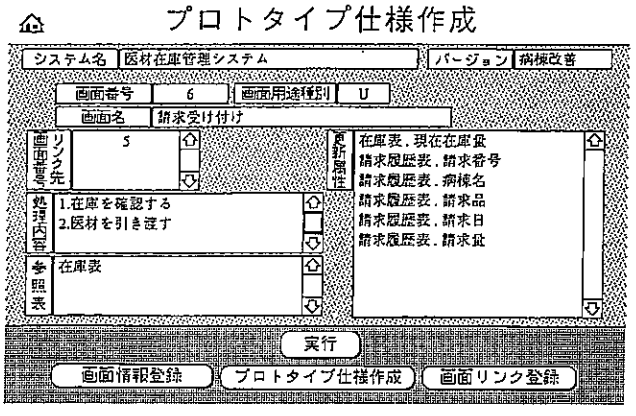


Fig. 16 Screen for building prototype specification

Fig. 16 is an example of a prototype specification of a screen. It is the result of checking a button for "build spec"(-ification) in the screen for "receive request." On that screen a user can specifies the name of the target screen, names of business functions which use the target screen and the other screens linked to the target screen. Clicking the button starts execution of the corresponding SQL queries and the results of the queries are sown in some fields of this card(screen).

VII. CONCLUSION

In this paper we have formulated a business transaction system and applied it to the characterization of tools in information systems analysis.

A business transaction system consists of both static and dynamic structures. The static structure is a description of a file system, external transaction names, internal transaction names and their mutual interconnections. The dynamic structure under the finite resource condition is a mechanism of transaction processing and then a state space representation whose state space consists of the file system and schedule of internal finish transactions.

Since a file system is a part of the state space, its role is not only to record data given by transactions but also to make asynchronous parallel processing possible. Also implied is that the fundamental reason that file systems can be designed in many ways seems to be a systems theoretic one. This is because there are infinitely many candidates for the state space representation of a time system if it has a state space representation. To narrow the possible structures of a file system other conditions, like management aims or performance requirements, might be needed. The point here is that many alternatives of the file system are possible to run a business system consistently. In reality many forms of flips as well as files have been used in firms in the same business area.

The meaning of the dataflow diagram(DFD), which is one of the most important tools of many information systems methodologies, is considered in relation to the business transaction system. It is shown that the DFD describes the static structure of a discrete event system by modeling the relationship between data and processes in a business system. Both the fact that both dataflows and datastores in the DFD correspond to key entity types in a business transaction system and the fact that choice of key entity types is not decided from the behavior (i.e., the set of all input-output pairs) of a business transaction system, explain the reason why the file system must be designed separately from the DFDs. It is not a feature of the procedure of analysis methodology but, again, an implication of the fact that the file system is a part of a state space of a dynamic structure of a business transaction system.

We have formulated the entity life history as the language of transactions related to each key entity type. Theorems 3 and 4 show the possibility and limitation of the entity life history as a tool for modeling the dynamic behavior of business systems.

In the development of an analysis repository system (Suzuki et al, 1992) we have modeled and defined the static prototype specification of a business transaction system. The product of analysis is stored into SQL tables according to the model of the specification. Prototyping on 4GL needs both a static model of a business functions and a complete model of transaction data.

This paper does not insist that the static structure and behavior of a business transaction system have the same flexibility in analysis of information systems. The model, business transaction system, provides a conceptual framework that shows directly the mechanism of dynamic behavior of business system itself. The business transaction system is merely a queuing-type discrete event system except that it uses a file system instead of queues. The answer to the question whether its construction is canonical in any sense or not can only be obtained by realization theory of business systems. Though some

important researches on the realization of discrete event systems have been carried out by Zeigler(1976, 1984), this question is not resolved yet. It is beyond the scope of this paper. It is our hope to provide a logico-mathematical basis of methodologies which have been crystallized from many professionals' intellectual struggle in information systems analysis.

References

[1] Brough, M.(1992): Methods for CASE: A Generic Framework, Advanced Information Systems Engineering: Proceedings for 4th International Conference CAiSE'92, Lecture notes in Computer Science 593, pp. 524-545, Springer.

[2]Cutts, G.(1987): Structured Systems Analysis and Design Methodology, Paradigm.

[3] DeMarco, T.(1979):Structured Analysis and System Specification, Prentice-Hall.

[4] Downs, E., Clare, P. and Coe, I.(1988): Structured Systems Analysis and Design Method - application and context, Prentice-Hall.

[5] Gane, C. and Sarson, T.(1979): Structured Systems Analysis: tools and techniques, Prentice-Hall.

[6] Hotaka, R.(1989):Database System and Data Model, Kyouritu. (In Japanese)

[7] McDermid, D.C.(1990): Software Engineering for Information Systems, Blackwell Scientific Publications.

[8] Mesarovic, M.D. and Takahara, Y.(1975): Mathematical foundation of general systems theory, Academic.

[9] Mesarovic M.D. and Takahara, Y.(1989): Abstract systems theory, (Lecture Notes in Control and Information Science 116), Springer.

[10] Olle, T.W.(1988): Information systems methodologies, Addison Wesley.

[11] Peters, L.(1987): Advanced Structured Analysis and Design, Prentice-Hall.

[12] Sato, R.(1991a): A formulation of simulation modelling methodology, Systems Research, 8-4, 3/19.

[13] Sato, R.(1991b): A general model of business system, J. Japan Soc. for Management Informatics, 2-1,21/31.

[14] Sato, R.(1992): Meaning of data flow diagram, *J. of Japan Society for Management Information*, vol. 1, no. 1, pp. 35 -50, 1992. (In Japanese)

[15] Sato, R.(1993): A foundation of information systems methodology, Institute of Socio-Economic Planning Discussion Paper Series No. 515, University of Tsukuba.

[16] Suzuki, Y. and Sato, R.(1992): A formulation of information systems analysis with prototyping and its implementation into a supporting software, (submitted. In Japanese).

[17] Vonk, R.(1990): Prototyping: the effective use of CASE technology, Prentice-Hall International.

[18] Zeigler, B.P.(1976): Theory of modelling and simulation, John Wiley.

[19] Zeigler, B.P.(1984): Multifaceted modelling and discrete event simulation, Academic.