〈研究論文〉

Journal of the Korean Institute of Industrial Engineers
Vol. 14, No. 2, Dec. 1988

A Tool for Transformation of Analysis to Design in Structured Software Development

Sung Joo Park*
Yang Kyu Lee *

abstract

The primary purpose of this study is to develop an automation tool capable of converting the specification of structured analysis into that of structured design.

Structured Analysis and Structured Design Language (SASDL) is a computer-aided description language based on ERA model and particulariged by ISLDM /SEM. The automation tool utilizes the specifications of data flow diagram described in SASDL to produce their corresponding SASDL specification of structure chart.

The main idea behind the automatic conversion process is to categorize the bubbles in data flow diagram and to determine the positions of the bubbles in structure chart according to their categories and the relative locations in data flow diagram. To make the problem into manageable size, the whole system is broken down into separate parts called activity units.

A great deal of manual jobs, such as checking processes leveling, checking data derivation of processes, deriving structure chart from data flow diagram, checking any inconsistency botween data flow diagram and structure chart and so forth, can be automated by using SASDL and conversion tool. The specification of structure chart derived by conversion tool may be used in an initial step of design to be refined by SASDL users.

1. INTRODUCTION

The whole process of software development may be segmented into a series of successive phases, that is, analysis phase, design phase, implementation phase, testing phase, and maintenance phase.

Of these five phases, analysis phase and design phase are focused mainly in this paper. The main emphasis of analysis phase is placed on what the software is to do and the constraints under which it will perform its required functions. Design is concerned with identifying software components, specifying relationships among components, specifying software structure, maintaining a record of design decisions, and providing a blue print for the implementation phase.

* Department of Management Science. Korea Advanced Institute of Science and Technology

Structured analysis is a systematic, step-by-step approach performing analysis focused on clear and concise communication. Structured design is a refinement of a top-down design, which is an informal design strategy for dividing the large problems into smaller ones [8]. The main products resulted from structured analysis and structured design are data flow diagrams and structure charts respectively.

The objectives of this paper are to develop a computer aided description language well suited for describing the tools of structured analysis and structured design and to devise a tool capable of automating the process of transition from structured analysis to structured design.

Structured Analysis and Structured Design Language (SASDL) is a computer-aided description language based on ERA model [1], and expressed in the ISLDM language of ISDOS[15].

The transition tool produces the specification of structure chart from the specification of data flow diagram by mapping the relations of data flow diagram into its corresponding relations of structure chart.

2. CONCEPTS OF ANALYSIS AND DESIGN

1. Structured Analysis

Analysis usually leads to the specification of a new system which describes how to meet the problem requirements.

To make the specifications of analysis highly maintainable, partitionable, graphic, and logical, structured analysis is equipped with valuable structured tools, such as data flow diagram (DFD), data dictionary (DD), and structured english.

The data flow diagram, one of the structurd tools on which main emphasis of this paper is placed, reveals the processes and the interfaces among these processes. According to the notations of Tom DeMarco [2], data flow diagrams are made up of following four elements, data flows, processes, files, and data sources and sinks.

To accommodate the dynamic features of data derivation of a process, additional notations for selection, iteration, sequence have been used in drawing data flow diagrams.

A file is a temporary repository of data. A source of sink, sometimes called a terminator, constitutes the system's external environment.

Applying structured concepts, DFD's are partitioned effectively from top level to bottem level called functional primitive. In partitioning the bubble into its child bubbles, the data flows into and out of a bubble on a parent diagram are equivalent to net inputs and outputs to and from a child diagram.

According to James Martin, data dictionary is defined as "A Data Dictionary is a repository of data about data". The definition of each data item consists of the data components consisting the data item and the relationships among them.

There are three tools to specify process, structured english, decision tree, and decision table.

2. Structured Design.

Structured Design is a disciplined approach, producing a "blue print" that the programmer can follow in implementing the system.

Structured Design uses two additional tools : pseudocode and structure charts. Pseudocode is a very high level language which is informal and flexible.

Structure chart shows the partitioning of a system into modules, their hierarchy and organization of modules, communication interfaces between modules, and names of modules. Modules are defined as collections of program statements which makes transformation on input to produce output. Predefined modules are provided by the operating system or application library. The connections between modules are shown graphically by arrow connecting them. The module shooting arrow represents a caller module, and the one receiving it a called module. There are two communication types, data and flag. Data is processed by the modules, whereas flag is not. The flag is used solely for making processing decision.

3. DESIGN STRATEGIES

1. Strategies for deriving an incarnation

To enable the essence of the target system to work, you must focus on how to implement that system using the available technology. The essential activities and memory of a system are implemented using processors which are called their incarnation.

In the process of constructing the incarnation of the system to be built, you may follow the following three steps, redefining the system's physical interface with the outside world, allocating pieces of activities to processors, and establishing infrastructural and administrative activitees [10].

An important part of infrastructure activities is the translation activities among processors that reformat the data for either human or computer.

To compensate for the fallibility of the processors and the fallibility of the interprocessor infrastructure, the administrative activities should be established.

As a result of deriving incarnation, each processor is divided into several activity units composed of administrative, infrastructural, transform activity.

2. Relations between Analysis and Design

DFD is a statement for the description of the requirements of the target system, declaring what has to be accomplished. Structure chart is a statement of design, declaring how the requirement shall be met. Contrary to the DFD the structure chart depicts the bosshood.

The essential relationships between the data flow diagram and the structure chart are concisely summarized by Tom DeMarco [2].

The top of the structure chart corresponds to the central transform portion of the DFD. There is one vice-presidential module for each input stream, one for each output stream, and one for the central transform.

For each input process on the DFD, there is one binary substructure ("Get" module and "Transform" Module) on the the Structure chart. For each output process on the DFD, there is one binary substructure ("Transform" module and "Put" module) on the structure chart.

Lower levels of modules beneath a transform module correspond on a one-to-one basis to subordinate processes on the DFD set.

The modules that corresponds to the transaction center process is itself a transaction center, i.e., it manages on subordinate module for each parallel process on the DFD.

Each subordinate module under the transaction center is allocated one of the parallel processes. These subordinates make up the "transaction level." Under each transaction-level module is a set of "action modules," one for each defining action.

3. Transform Analysis

Transform analysis is the major strategy used in designing a program by identifying the primary functional components and making a hierarchy among them. The whole point of transform analysis is to convert the transform centered DFD of analysis to hierarchical type of structure chart of design.

The structure chart must be built such that the top module deals with logical data while the bottom module deals with physical, configuration dependent data. That is what a balanced system is.

The transform part, and its adjacent input and output components are entitled to the top modules. The components of input and output part which deals with the raw data represent the bottom modules.

Next step of transform analysis is refining the first-cut structure chart by adding read and write library modules, error handling modules, initialization and termination processing and so forth.

4. Transaction Analysis

Transaction analysis is valuable in using the system to be built deals with transaction. Transaction is a element or a collection of data that triggers one of the parallel processes, called transaction modules.

The top module, that is transaction-center module controls the transaction module according to the transaction provided.Each transaction module is called by the transaction-center module selectively and exclusively. The transaction modules are placed parallel to each other.

In this type of system, the main module invokes the module for receiving input data, which returns the transaction and its type. After receiving the transaction and its type, the main module invokes the transaction-center module which invokes one of the transaction modules accordibg the transaction and type.

5. Evaluation of the Design

After constructing the structure charts of the target system, it is of great importance to evaluate the quality of them objectively. Among the several design criteria, coupling and cohesion are stressed mainly. The desirable properties of the modules and their interfaces are low coupling and high cohesion.

Coupling is the degree of interactions between modules [11]. Keep the coupling of the system as low as possible in order to make the system highly maintainable, easier to modify, possible to deviate the ripple ffect (a bug in one module appearing as a symptom in another).

Cohesion is the measure of the strength of functional association of elements within a module [11]. Coupling pertains to the "intermodule interface", whereas cohesion to "intra-module interface".

Another important criteria hardly overlooked is information hiding [12]. The concept of information hiding is that each module is designed to hide a design decision from the others. One of the objectives of information hiding is to make the system highly flexible.

4. SASDL AND CONVERSION

1. Tools to Implement SASDL

A Life Cycle Support System (LSS) is a generic name for a computer aided system supporting the activities of system development in one or more phases of life cycle [15]. An LSS receives system description, maintains a data base containing the system description, analyzes the system description stored in a data base, produces documents and output on request based on the system description, and preforms monitoring and other control functions in the development of the software system.

In order to meet the need for various LSS's well suited for specific application areas, the LSS generator was developed. The LSS generator is a computer-aided tool capable of developing a specific LSS from the formal specification of LSS [15].

In this paper the LSS generator called ISLDM/SEM was used to develop an LSS capable of specifying the tools of structured analysis and structured design.

ISLDM/SEM consists of Information System Language Definition Manager (ISLDM) Ststem and System Encyclopedia Manager (SEM). ISLDM System is the LSS generator Processor in ISLDM/SEM. The functions of ISLDM are language processing, global analysis, documentation generation, and table generation [5].

SEM is the generalized software for generating the LSS processor by particularizing generalized table-driven software by means of a table.

The LSS generator is based on Entity-Relationship (E-R) model [2] because it is natural and close to the view of the LSS user, successful in other LSS in the Past, and consistent with the general system theory.

2. Definition of SASDL

SASDL contains 21 objects and 54 relationships of which 11 objects and 34 relationships are used for description of the analysis part (containing connection part), and the rest are for the design part.

2.1 OBJECT

PROCESS performs transformation on the input data, producing output data. DATAFLOW is data transported between PROCESSes. TERMINATOR is a source of a sinker, which represents the system's external environment. STORE is a repository of data, namely file in the DFD.

GROUP is a composition of ELEMENTs or other GROUPs. It is one or more data elements in which a higher level data can be decomposed. MODULE is a collection of program statement with four basic attributes : input and output, function, mechanics, and internal data. LIB-MODULE is a predefined MODULE. INC-MODULE is a MODULE coded internally to the caller MODULE.

CALL-CONDITION limits the calling condition of a caller MODULE. CALL-LOOP provide the calling MODULE with the possibility of iterative call. ACTIVITY-UNIT is an unit where the conversion is applied. TRANASCTION-CENTER object is used for identifying the transaction centered parallel processes.

2.2 RELATIONSHIPS AND CONVERSION

Transform relation and transaction-center relation are the key relations in conversion process. Transform-relation identifies input data and its resulting output data of a process. Transform-relation can be correlated to simple-call-relation in the design part.

Transaction-center-relation identifies the relation of transaction center process's transmitting output data to its parallel processes. In the design part, transaction-call-relation corresponds to it.

Selective-input-relation and selective-output-relation represent the relation of a process's accepting or transmitting one of its three exclusive data parts. The relation can be converted to call-either-or-relation in the design part.

Conditional-input-relation an conditional-output-relation represents the relation that a process accepts r transmits its data depending on the specific condition. Conditional-imput-relation, and conditional-output-relation can be converted to call-dep-on-relation in the design part.

Conditional-input-relation and conditional-output-relation represents the relation that a process accepts or transmits its data depending on the specific condition. Conditional-input-relation and conditional-output-relation can be converted to call-dep-on-relation in the design part.

Compose-relation is for identifying which processes are the part of an activity unit. The top modules in the activity unit are under the direct control of the activity unit. The candidates for the top module are the transform processes, and their adjacent logical input, and logical output process. The logical input processes adjacent to transform process are enligible for the top modules. The logical output processes having adjacent transform processes which transmits their input data are also enligible for the top modules.

Sequential-input-relation and sequential-output-relation represent the relation that a process receives or transmits its data sequentially. These relations may be converted into call-with-seq-relation in the design part.

Iterative-input-relation and iterative-output-relation represents that the process accepts or transmits data repeatedly for required number of times. Call-loop-relation in the design part may be used when converting these relations.

Manages-relation identifies which processes are the transaction centered parallel processes.

3. Connection Tool

The unit of analysis to which the conversion procedure is applied is an activity unit. For each activity unit, every process in it is identified and classified. According to the property and the relative position in the activity unit, the process is allocated its location in the structure cart.

A. Main part

For each activity unit, every process in it is identified. According to the property of the process, the main module invokes other relevant module. By checking rules, no process is converted twice.

B. Input part

If the property of a given process is logical input or administrative input, this part is invoked. If the process is converted into modules for receiving input data, for making transformation on the input data and for receiving transforned data.

To preserve the number of input data, output data and the name of the data, this procedure invokes allocate corner procedure, producing n-array variables assigned the values of input and output data of the process.

The module for receiving transformed data become the caller module and the module

for receiving data and transformation become the called module.

The input process adjacent to the transform process is under the direct control of activity unit control module. It passes its output data to the activity unit control module.

The procedure for subpart search is invoked to check the subpart search relation for each process. Every subpart process is placed on the same level below the transform module.

If transaction-center-relation is used to convert the input data, the input part process invokes allocate-transaction-center-procedure. It allocates the names of the transaction centered process to the two dimensional variables, and the input data and the output data of each process is allocated to three dimensional variables, then invokes procedure for making input module. It makes the hierarchical relations among modules. The top module is for receiving final output data of the group of transaction centered processes. The values of final output data had been assigned to the three dimensional variables. The next module below the top module is transaction control module which calls the transaction centered parallel processes and processes receiving input data.

For each input leaf, data decomposition procedure is invoked to decompose the data according to the data structure.

C. Transform part

If the property of a process under conversion is transform, this procedure is invoked. Transform processes are the core of the activity unit. Every central process is under the direct control of the activity unit control module.

To minimize coupling, all central transform modules are located parallel to the activity unit control. The receiving data from the control module is the input data to the process and passing data to the control module is the output data from the process.

Transform process is mapped to a single module, transform module. Modules for receiving and transmitting data are not required, because all the data couples are done through the control module.

If transform-relation is used, locate the process under the activity unit control module and make the data coupling as described above. If the transaction-center-relation is applied, the process to deal this case the same that of input part. The difference is that the transaction control module is controlled directly by the activity unit control, and there is no module for accepting input.

D. Output part

When the property of a process is logical output or administrative output, this procedure is invoked. The sturcture of the resulting modules is very similar to that of input part. The main differences are the data passing type, calling relation, and leaf data to be decmposed further. The relation type of a process is transform-relation, the process is converted to module for putting input, module for transforming data, and module for putting output data.

The caller module corresponds to module for putting input data, and the called module to module for transforming and putting output data.

If the transform relation is transaction-center-relation, allocates the values of transaction centered processes and their input and output data to the multi-dimensional variables. The top module is the module for receiving for first input data, controlling the transaction center control module. The transaction control module contols the transaction centered processes and processes for transmitting output which are located parallel.

5. EXAMPLE

1. Target System

This example was originally prepared by Meilir Page-Jones [11]. It was modified and extended somewhat to increase clarity and to meet the purpose of this study.

The whole system is divided into 8 activity units. The activity unit recording customer orders receives customer purchase orders from the customer. Validating process to check the customer order follows. For each filled order, customer shipment authorization is issued to allow the department of shipping to make a shipment for the products. The insufficient orders are back ordered and recorded into purchase orders waiting for parts file. In case of occurring the insufficient order, insufficient notification is issued to the purchase in charge of purchasing goods in need.

This example is decomposed up to level 3. The level to which the conversion from anslysis to design is applied is level 2.

2. Illustrative Results.

To demonstrate the conversion process, the SASDL specification of an activity unit and some of its derived specification of the structure chart are shown below.

The activity unit recording cust order has been selected among eight activity units. For each process in analysis part, there may be several modules correspond to it in design part.

```
DEFINE ACUN recording-cust-order ;
  COMPOSED OF validate-cust-acct-no,
    validate-cust-part-line,
    collect-valid-cust-purch-order,
    rec-valid-cust-purch-order, format-ship-auth,
    rec-back-order, format-insuff-notif;
DEFINE PROCESS validate-cust-acc-no
  PART OF rec-cust-purch-orders ;
  COMPOSES recording-cust-order ;
  P-TYPE adin ;
  NUMBER "1.3.1" :
  TRANSFORMS cust-acct-no TO valid-cust-acct-no ;
  RETRIEVES cust-acc-no FROM custs ;
```

Process validate-cust-acct-no may be converted into 3 modules, module for accepting input data(get-cust-acct-no), module for validating input data (validate-cust-acct-no), and module for retrieving data from file (read-cust-acct-no). If the property of the processes receiving output data is transform, the caller module (get-valid-cust-acct-no) is called by the activity unit control module. Otherwise it may be called by the modules for receiving the next processes' output data.

```
DEFINE MODULE get-cust-acct-no ;
  CALLED BY get-valid-cust-acct-no
    PASSING-DATA cust-acct-no ;
DEFINE MODULE read-cust-acct-no ;
  CALLED BY get-cust-acct-no
    RECEIVING-DATA cust-acct-no ;
DEFINE MODULE validate-cust-acct-no ;
```

CALLED BY get-valid-cust-acct-no
PASSINE-DATA valid-cust-acct-no
RECEIVING-DATA cust-acct-no
PASSING-CONTROL ok ;

6. CONCLUSIONS AND DISCUSSIONS

The objectives of this study are to clarify the relationships between data flow diagram and structue chart, to develop a language capable of expressing data flow diagram and structure chart, and to automate the process of conversion from data flow diagram to structure chart.

The computer-aided specification language is developed using the LSS generator called ISLDM/SEM. The specifications of structure chart can be derived from that of data flow diagram and data dictionary using conversion tool.

By automating the part of analysis phase and design phase, a considerable amount of manual jobs are no more needed. The consistency and completeness of the specification can be increased drastically by utilizing the analysis reports. The graphical interface facility capable of drawing data flow diagram and structure chart using the specifications of SASDL is a valuable area of further research. The development of design simulator capable of producing alternative specifications of structure chart from that of DFD and to evaluate each design by design criteria may be another valuable area for further study.

References

1. Chen, Peter Pin-Shan, "The EntityRelationship Model-Toward A Unified View of Data," *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9–36, Association for Computing Machinery, Inc., Mar. 1976.
2. De Marco, Tom, *Structured Analysis and System Specification*, Yourdon Press, 1979.
3. Freeman, P., "Fundamentals of Design," in *Tutorial on Software Design Techniques*, ed. Peter Freeman and Anthony I. Wasserman, pp. 2–22, IEEE Computer Society Press, Silver Spring, MD 20910, 1983.
4. Gane, Chris and Trish Sarson, *Sturctured Systems Analysis : Tools and Techniques*, Prentice-Hall, 1979.
5. ISDOS, "An Introduction to the Use of Information System Language Definition Manager (ISDLM) and System Encyclopedia Manager (SEM)," ISDOS Ref. #MO320-0, ISDOS, Inc., 325 East Eisenhower Parkway, Ann Arbor, Michigan 48106, Sep., 1981.
6. ISDOS, "Report Specification Interface (RSI) User's Manual," ISDOS Ref. # MO604-0, ISDOS, Inc., 325 East Eisenhower Parkway, Ann Arbor, Michigan 48106, Feb., 1984.
7. Martin, James, *System Design from Provably Correct Consturcts*, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.
8. Martin, James and Carma McClure, *Sturctured Techniques for Computing*, Prentice-Hall, 1985.
9. Martin, Jason, "From Analysis to Design," *Datamation*, pp. 129–135, DATAMATION Poster, New York, NY 10022, Sep. 15, 1985.

10. McMenamin, Stephen M. and John F. Palmer, *Essential Systems Analysis*, Yourdon Press, 1984.
11. Page-Jones Meilir, *The Practical Guide to Structured Systems Design*, Yourdon Press, 1980.
12. Parnas, L.L., "On the Criteria to be Used in Decomposing the Systems into Modules," *Communications of the ACM*, pp. 1053–1058, Association for Computing Machinery, Inc., 1972.
13. Teichroew, Daniel and Ernest A. Hershey, III, "PSL/PSA : A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems," IEEE Transactions on Software Engineering, in *Tutorial on Software Design Techniques*, ed. Peter Freeman and Anthony I. Wasserman, pp. 211–218, Jan. 1977.
14. Wasserman, Anthony I., "Information System Design Methodology," *Journal of the American Society for Information Science*, vol. 31, no. 1, pp. 25–43, Jan. 1980
15. Yamamoto, Yuzo, *An Approach to the Generation of Software Life Cycle Support systems*, Ph. D. Dissertation in the University of Michigan, University Microfilms Internation, Ann Arbor, Michigan, 1981.