Sci. J. Fac. Sci., Minufiya Univ., Vol. XIV,2000, 283-302

Informal and Formal Specification for Methodology Information Exchange Service (MIES) Problem.

*** Fatma A.Omara**

**** Reham A.Mahmoud**

*Ass. Prof in Computer Science & Eng. Dept., Faculty of Electronic Eng. Menouf, Minufiya University.

**Research Student, Mathematics Dept., Faculty of Science, Minufiya University.

ABSTRACT

The differences between informal and formal requirements specification languages are noted, and the issue of bridging the gap between them is motivated and discussed. Structured Analysis (SA) and the Vienna Development Method (VDM) have been used as surrogate for informal and formal languages, respectively. Two approaches will be presented for integrating them. In the first approach the SA model of a system is used to guide the analyst's understanding of the system and the development of the VDM specifications. The second approach proposes a rule-based method for generating VDM specifications from a set of corresponding SA specifications. These two approaches are illustrated through a simplified Methodology Information Exchange Service (MIES) problem.

1. INTRODUCTION

The informal and formal specifications for the MIES problem is considered the goal for this paper. The MIES problem in the Multimedia environment is transmitting the pictures, voices; especially pictures in the form of stream of frames using Open Distributed Processing Systems (ODP) [1]. According to this problem three processes are used : Generating stream of frames for pictures, Monitoring stream of frames for pictures which belongs to the Quality of Service Violation (QoS-Violation) phase, and finally the Receiving stream of frames for pictures. The determination and specification of requirements are key activities in the development of computer-based information system [1]. It is recognized that "the better adapted the definition of functional requirements to organizational needs, and the more explicit and complete the statement of requirements-the greater the system success" [2].

Therefore, it is crucial that the users' needs and problems are well understood and analyzed. The system requirements reflect these needs, and the requirements are translated into an adequate set of specifications for system construction.

There are three rules associated with the requirements engineering process :

1- At the front end of the process, the first role is that the user's needs provide the underlying rationale for the system under development [3][4].

2- At the other end of requirements engineering process is the designer/ constructor whose function is to translate the users' requirements into an implemented software system.

3- In between is the role of the information systems analyst as the requirements engineer who mediates between the users and the designer/constructor. According to this role the systems analyst requirements, the software engineer elicits the users' requirements, analyzes and documents them, and finally, transmits them to the designer/constructor [1][5].

2. Formal and Informal Requirements Specification Languages:

The requirements specification languages could be classified into two major classes :

I- Formal Specification Languages.

II- Informal Specification Languages.

I- Formal Specification Languages:

One may have a mathematical (usually formal logic) basis and employ a formal notation to model the system requirements [6], [7]. On the other hand, precision and conciseness of specifications can be achieved by using the formal notation, because the formal notation can be analyzed and manipulated using mathematical operators, and the mathematical proof procedure can be used to test the internal consistency. The VDM can ensure completeness in the sense that all of the users' requirements have been met. Finally, as the final problem solution–the implementation of the system will be in a formal language. It is easy to avoid ambiguities in crossing the dividing line from formal specification to formal implementations [6].

II- Informal specification languages :

A combination of graphics and semiformal Textual – Languages could be used to describe and specify system requirements. Given the graphical and "English – like" nature of these languages, one provides an ideal vehicle for eliciting user requirements and communicating the analyst's understanding of the requirements back to the user for verification. The informal language specifications tend to be imprecise and ambiguous. So, there is a possibility that the user, requirements engineer, and the designer/constructor will read their own understanding into the specifications [6]. Furthermore, the available operators for

reasoning and analysis of graphical and textual descriptions are limited and semistructured, human reasoning is the primary mechanism used for the analysis and verifying of informal language requirements specifications.

2.1 Structured Analysis and Vienna Development Methodology Approaches:

Structured Analysis (SA) and the Vienna Development Method (VDM) will be used as representatives of informal and formal requirements specification languages.

According to Structured Analysis Approach, the graphical and semiformal notation are used to describe the flow, processing of data, the data and process structures, and the data processing logic of an information system.

The Vienna Development Methodology Approach, has a well– developed mathematical system and proof rules for formal proofs of the specifications based on formal predicate logic and set theory.

Structured Analysis could be used as cognitive guide to develop the VDM specifications. In this approach, the actual interaction between the SA and VDM specifications is done by a human analyst, who uses the SA specifications as a cognitive guide to the understanding and structuring of the system. The primary tool used in SA to describe and analyze a system is the Data Flow Diagram (DFD). The DFD at the highest level, describes the system as a data transformation process which receives input data flows and control signals from its environment, then processes or transforms them into output data flows and control signals which flow back to the environment. The highest level DFD is called " Context – diagram" [8]. The more detailed model of the overall system is built by partitioning the context diagram, i.e. the overall data transformation process, into subprocesses. Each of the subprocesses may be further partitioned into their lower level subprocesses until the bottom level processes called "functional primitives" are reached. Functional primitives are not partitioned further; instead they are described using transform descriptions. The partitioning is carried out by tracing either the input data flows forward or data flows output backward into the system. A subprocess is recognized at the point where a data flow is operated upon or changes either its Quantitative or Qualitative attributes. The points of transformation of the data flows provide the heuristics for partitioning the higher – level transformation process. Finally, a process of "Topological Repartitioning" may be applied to the hierarchically partitioned set to minimize the number and complexity of interfaces between the partitioning [9].

The hierarchically partitioned Data Flow Diagrams (DFD) have a set of syntactic and semantic rules which need to be observed to ensure their completeness, correctness, and consistency. The definitions of the data structures

can also be hierarchically successively partitioned into lower – level definitions until "self-defined" terms are reached [8].

2.2 The Transformation of SA into VDM Specification:-

The formal language uses essentially auxiliary functions [6], [7]. Auxiliary functions are used to identify the logical properties of the PRE, and POST conditions of specifications. The specifications in the VDM are usually developed by repeatedly refining complex specifications into subspecifications until they close to an implementation specification [8]. The cognitive technique to guide this stepwise specification refinement is based on the hierarchial partitioning techniques of Data Flow (DF) analysis [8]. This cognitive approach might be considered similar to the use by Andrews and Gibbins [6] of the bottom level of an informal flowchart to develop formal specifications.

The used technique in this research for guiding operational decomposition with transform partitioning consists of:

1. Representing the input and output data flows and externals in the Data Dictionary (DD) in an abstract syntax.
2. Giving a specification for each transform in the Data Flow Diagram set.
3. Combining (using the VDM combination constructs sequence If- Then–Else) the specifications according to the architecture provided by the leveled DFD set.

2.3 The Generation of VDM Specifications From SA Models:

In this section we sketch a rule – based method for generating VDM specifications. In this method, the analyst therefore needs to recognize the structure inherent in the control flows and processes where necessary, restructure them to conform to the structured constructs [9], [10], [11]. This method can, however, be extended to include other control structures (such as repeated rule If – Then – Else) and other forms of process descriptions using structured english. The method consists of mapping decision table process description into VDM specifications using the decision table conversion rule, and then composing these specifications in a bottom–up fashion using the sequence composition rule (based on the precedence of the DFD [12]).

2.3.1 VDM Specification Generation Rules:-

(A) Decision Table Conversion Rule

The associated rule with column J of the decision table is the implication $C \Rightarrow A$ where :

C is a conjunction of conditions.

A is a conjunction of actions specified in column J.

The decision table conversion rule from a decision table to the structure for a VDM specification of an operation OP is given by :

(a) **Obtain ext–OP** : the **rd** operands of **ext–OP** are the condition data or control flows and the **Wr** operands are the action data or control flows [8].

(b) **Obtain Pre–OP :** is disjunction of the conditions.

(c) **Obtain Post–OP:** is conjunction of the decision table rules [13][14].

(B) Sequence Composition Rule:

Given two VDM Operations in Sequence A ; B, the specification of A ; B is obtained in the following two steps :

Step 1 : Ext ext – A U ext – B

Step 2 : { Pre – A} (A ; B) { Post – A ⟹ (Pre – B ⟹ Post –B)}

In step 2, means that if **Pre – A** is true for an initial state and **A** ends in a state for which **Pre – B** is true, then we may conclude that **Post – B** (as well as **Post – A**) is true. If, on the other hand, **A** ends in a state for which **Pre – B** isn't true, then no conclusion on **Post – B** is implied [15].

2.4 The SA approach for requirements Specification of The Methodology Information Exchange System (MIES) Problem.

The SA of the MIES Problem can be described as follows :

I- The Essential Model:

The essential Model is a model of what the system must do (fig. 1) in order to satisfy the users' requirements, with as little possible said about how the system will be implemented. The components of this model are the environmental model and behavioural model.

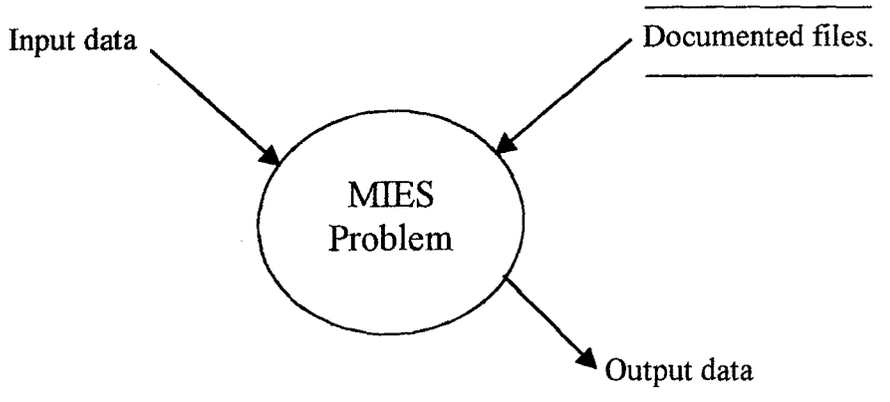

Fig.1 The Essential Model of the MIES problem.

II- The Environmental Model:-

The environmental model defines the boundary between the system and the rest of the world. (i.e. it describes the outside of the system). It consists of a short description of the purpose of the system, A context–diagram and an event – list.

The Description: The MIES problem could be shown as a distributed video player with the observable actions; Generating, Receiving, Quality of Service-Violation (QoS) for the stream of frames.

Context-diagram: The next part of the environment model begins to answer some of the questions raised by the statement of purpose of the system. A context model (Fig. 2) is a special case of the DFD, in which a single bubble represents the entire system, it consists of terminators, data flows, control flows, data stores and a single process represent the entire system.

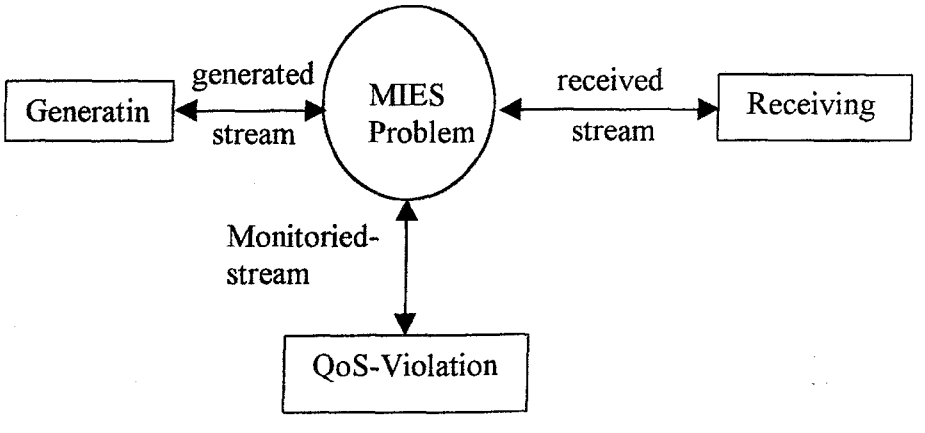

Fig.2 Context model of MIES Problem

III- The Behavioural Model :

This Model describes the required behaviour of the insides of the system necessary to interact successfully with the environment (Fig.3). This model consists of DFDs, Data Dictionary (DD), State Transition Diagrams (STDs) and process specification. In the SA model we concentrate on DFDs and DD only. It is noted that the Data Flow Diagrams of this problem consists of three levels as in (Fig. 4), (Fig. 5), (Fig.6).

The MIES problem shows a distributed video player with the observable action : generating, displaying, QoS-violation (the Quality of the service). The behaviour within this problem consists of a producer who has a role to generating

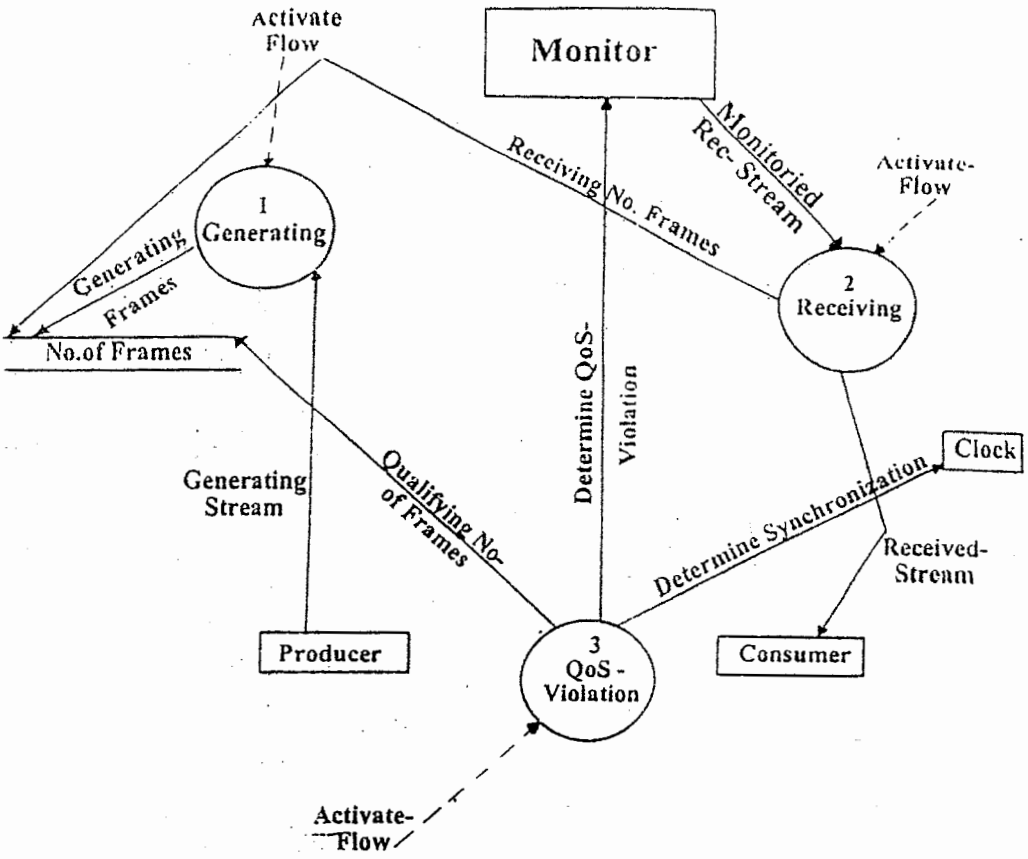

Fig. 3 The Behavioral Model of MIES Problem

and transmitting frames, a consumer receiving and displaying these frames and a stream object modeling a synchronous communication between these two processes. In addition a clock and a monitor object are introduced to carryout the necessary quality of service management [9],[16].

This algorithm illustrates the MIES problem using LOTOS languages [10]:-

```
Process producer [ generate, transmit] : noexit : =
(generate ? frame : picture ; transmit ! frame ;
producer [generate, transmit])
 |[ generate] |
generator [generate] (0)
where
process generate [generate] (frame – no : picture) : noexit : =
(* this process generates an infinite number of frames which
    may then be transmitted by the producer *)
    generate ! frame – no ; stop
    generator [generate] ( succ ( frame – no))
    END proc (* generator *)
    END proc (* producer*)
Process consumer [receive , display ] : noexit : =
Receive ? frame : picture ; display ! frame ;
Consumer [receive , display]
    END proc (* consumer *)
Process monitor [display, QoS-violation] : noexit : =
    Display? Frame : picture ;
    (monitor [display, QoS-violation]
    QoS-violation ; stop
        )
    END proc (*monitor*).
```

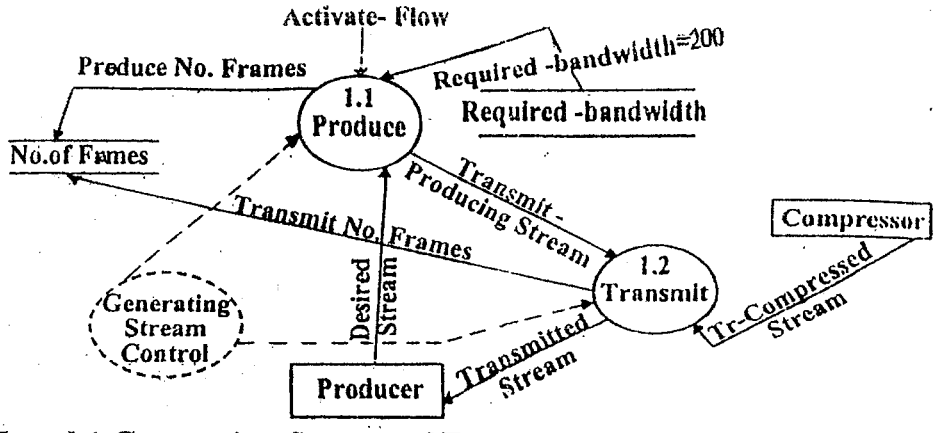

Fig. 4 Level.1 Generating Stream of Frames .

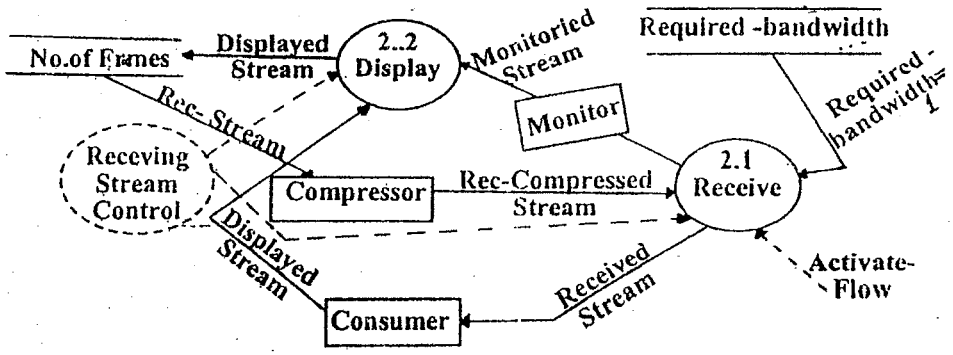

Fig. 5 Level. 2 Receiving Stream of Frames .

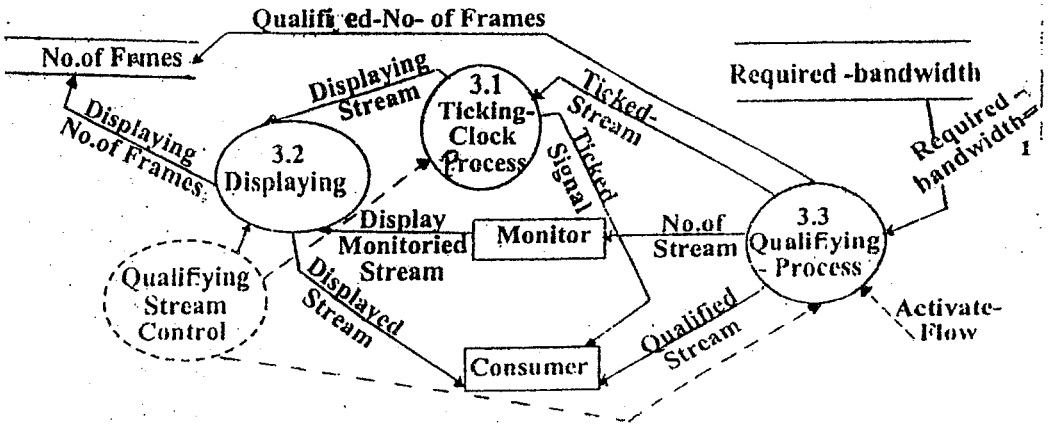

Fig. 6 Level. 3 Qualifying Stream of Frames .

The Data Dictionary of MISE problem is illustrated as follows in (Fig. 7).

The data dictionary of MIES problem (DD):

Structured analysis	**DD**	**VDM Names**	
Activate-flow	= [T / F]	Actv	/* The signal which activate the process * /
Required-bandwidth	= [1 / 200]	Rb	/ * The required bandwidth take the value 200 mb/s and compressed to 1 mb/s during transmission to the stream of frames * /
Monitor-process	= [T / F]	Mp	/* The monitor – process take care to the quality of service during the transmission of the stream * /
Displaying Monitored- stream	= [T / F]	Disms	/* Displaying after the monitoring for the stream of frames * /
Clock-tick	= [1 / 0]	CL	/* The clock tick 1 it means the transmission occurred and tick zero means the transmission failed * /
Structured analysis	**DD**	**VDM Names**	
Required-bandwidth state	= [Norm / Abnorm]	Req	/* Determine if the state of the bandwidth is required according to the nature of process or not * /
Transmitted-stream	= [T / F]	Tr	/* Stream under the transmission process */
Produced-stream	= [T / F]	Pro	/* Produce a number of streams to transmit * /
Received-stream	= [T / F]	Rec	/* Receive a number of streams top display * /
Compress-process	= [T / F]	Compr	/* Compress the stream of frames for picture to reduce the bandwidth * /
Display-process	= [T / F]	Disp	/* Display the streams according to the QoS conditions * /
Rec-compressed stream	= [T / F]	Rece	/* Receive the compressed stream before the decompression process * /

Clocking-stream	= [T / F]	Clo	/* Streams under the clocking process * /
Monitoring-stream	= [T / F]	Monit	/ * Streams under the monitoring process */
Qualifying-process	= [T / F]	Qual	/ * The process determine the quality of the pictures * /
Monitor	= [A : Variable]	Mon	/ * The monitor produce the variables to determine the quality of the picture according to the color, size, density and so on * /
Compressor	= [B : Variable]	Compre	/* The compressor compresses the bandwidth and changes its values according to the nature of the process * /
Consumer	: The person who receive a stream of frames.		
Structured analysis	**DD**	**VDM Names**	
Producer	: The person who generates a stream of frames.		
Desired-stream	= [T / F]	Des	/ * The stream satisfies the all QoS conditions and the transmission process for this stream is succeeded * /
Receive-process	= [T / F]	Recp	/ * Receive a stream of frames after transmitting it through the channel and before displaying it * /
Ticked-Stream	= [T / F]	Tic	/ * Stream under ticking operation * /
Desired-stream state	= [T / F]	Desir	/ * The state of the desired – stream that is required according to the QoS conditions * /

Fig.7 : The Data Dictionary of MIES Problem.

(I) Generating – Process Specification:

(1) Level – Decision Table (Process Description):-

According to fig. 4 the inputs of generating process are activate flow, required bandwidth and desired – stream which have the logical probabilities to occur, where T indicates the occurrence of the event and F indicates the not occurrence of the event. Also the outputs of the same process are produce number of frames and transmit them where Y indicates the occurrence of the event and N indicates not occurrence of the event taking the case of the inputs in the consideration.

Input	**Activate – Flow**	T T T T F F F F	Logical probabilities
	Required – bandwidth	T T F F T T F F	
	Desired – stream	T F T F T F T F	
Output	**Produce – No of frames**	Y N Y N N N N N	
	Transmit – producing stream	Y N N N N N N N	

(a) Obtain Ext-Generating Stream:-

This table determines the condition data or control flows data from input data.

Generating control flow	Mode	Variable	Mode
Activate-flow	Rd	Actv	Boolean
Required-bandwidth	Wr	Rb	Req
Desired-stream	Wr	Des	Desir

(b) The Resulting Ext Statement is:-

The statements from the previous table according to the mode of the input data are structured.

Ext rd actv : Bool, Wr Rb : Req, Wr Des : Desir

(c) The Postcondition:-

The conditions that make the inputs true are determined.

If actv = true and Rb = 200 then

Des(Desir) = True

(2) The VDM specification of the Generating process:-

The combination of the resulting Ext. statement and the postconditions statements of the inputs create the VDM specification of the process.

Ext rd actv : Bool, Wr Rb : Req, Wr Des : Desir

If Actv = True and Rb = 200 then Des (Desir) = True

(3) The VDM specification for Generating process from the decision table using the decision table conversion rule:-

In this sections, the VDM Specification for the generating process is resulted from the decision table. And determining for all logical probabilities that consider the process is true or false.

Ext rd actv : Bool, Wr rb : Req, Wr Des : Desir

Post (If actv = true and Rb = 200 and Des = True Then

PRO = True and Tr = True)

And

(If actv = True and Rb = 200 and Des = false Then PRO = True and Tr = True)

And

(If actv = True and Rb < > 200 and Des = True Then Pro = True and Tr = false)

And

(If actv = true and Rb < > 200 and Des = false Then Pro = true and Tr = false)

And

(If actv = false and Rb = 200 and Des = True Then Pro = false and Tr = false).

(II) Receiving – process specification:

(1) Level – decision table (process description):-

According to fig.5 the inputs of receiving process are activate-flow, rec-compressed stream and required-bandwidth which have the logical probabilities to occur, where T indicates the occurrence of the event and F indicates the not occurrence of the event. Also the outputs of the same process are receive-process display process and monitored-stream where, Y indicates the occurrence of the

event and N indicates not occurrence taking the case of the inputs in the consideration.

Input	**Activate – Flow**	T T T T F F F F	**Logical probabilities**
	Rec – Compressed stream	T T F F T T F F	
	Required – bandwidth	T F T F T F T F	
Output	**Receive – process**	Y Y Y Y N N N N	
	Display – process	Y N N N N N N N	
	Monitored – stream	Y Y Y Y N N N N	

(a) Obtain Ext – Receiving stream:-

This table determines the rd data which is considered the condition data and the data which is considered the control flow.

Receiving control flow	**Mode**	**variable**	**Mode**
Activate – flow	Rd	Actv	Boolean
Rec – compressed stream	Wr	Rec	Rece
Required – bandwidth	Wr	Rb	Req

(b) The Resulting Ext Statement is :-

The statements from the previous table according to the mode of the input data are structured.

Ext rd actv : Bool , Wr Rec : Rece, Wr Rb : Req

(c) The Postcondition:-

The conditions that make the inputs true are determined.

If actv = True and Rb = 1 Then Rec = Next and Rb = Req (Rec).

(2) The VDM specification of the Receiving – process:-

The combination of the resulting Ext. statement and the postconditions statements of the inputs create the VDM specification of the process.

Ext rd actv : Bool, wr Rec : Rece, wr Rb : Req

The Postcondition:

If actv = True and Rb = 1

Then Rec = Next and Rb Req (Rec)

(3) The VDM specification for Receiving process generated from the decision table using the decision table conversion rule:

In this phase, the VDM specification for the receiving process is resulted from the decision table, and determining for all logical probabilities that consider the receiving process is true or false.

Ext rd actv : Bool wr Rec : Rece, wr Rb : Req

Pre True

Post (If Actv = True and Rec = True and Rb = 1)

Then Recp = True and Disp = True and Monit = True)

And

(If Actv = True and Rec = True and Rb < > 1 Then Recp = True and Disp = False and Monit = True)

And

(If Actv = True and Rec = false and Rb = 1 Then Recp = True and Disp = False and Monit = True)

And

(If Actv = True and Rec = False and Monit = False Then Recp = True and Disp = False and Monit = True)

And

(If Actv = False and Rec = True and Monit = True) Then **Recp = False and Disp = False and Monit = False)**

(III) Qualification–process specification (QoS – Violation process)

(1) Level – Decision table (process description):

According to Fig. 6, the inputs of qualifying process are activate – flow and required – bandwidth which have the logical probabilities to occur, where T indicates the occurrence of the event and F indicates the not occurrence of the event. Also, the outputs of the same process are qualifying – process, ticked stream and displaying – monitored stream where Y indicated the occurrence of the event and N indicates not occurrence of the event, taking the case of the inputs in the consideration.

Input	**Activate – Flow**	T T F F	Logical probabilities
	Required – Bandwidth	T F T F	
Output	**Qualifying – Process**	Y Y N N	
	Ticked – Stream	Y Y N N	
	Displaying Monitored Stream	Y N N N	

(a) Obtain Ext – Qualification – process:-

This table, determines the condition data **rd** or control flows data **wr** from input data.

Qualifying control flow	**Mode**	**Variable**	**Mode**
Activate – flow	Rd	Actv	Boolean
Rec – compressed stream	Wr	Rec	Rece
Monitoring – stream	Wr	Mon	Monit
Clocking – stream	Wr	Cl	Clo
Required – bandwidth	Wr	Rb	Req

(b) The Resulting Ext statement:-

The statements from the previous table according to the mode of the input data are structured.

Ext rd actv : Bool, Wr Rec : Rece, Wr Mon : Monit,

Wr Cl : Clo, Wr Rb : Req.

(c) The Postcondition:-

The conditions that make the inputs true are determined.

If actv = True and Cl = 1 and Rb = 1 Then Rec = Next

And Cl = Cl(Rec)

And Rb = Req(Rec)

And Mon = Monit(Rec)

(2) The VDM specification of the Qualifying – process :-

This phase is considered the combination of the resulting Ext. statements and the postconditions statements of the inputs data that create the VDM specification of this process.

Ext rd actv : Bool, Wr Rec : Rece, Wr Mon : Monit, Wr Cl : Clo, Wr Rb : Req

If actv = true and Cl = 1 and Rb = 1 Then Rec = Next

And Cl = Cl (Rec)

And Rb = Req (Rec)

And Mon = Monit (Rec)

(3) The VDM specification for Qualifying – process generated from the decision table using the decision table conversion Rule.

The VDM specification for the qualifying process is resulted from the decision table, and determining for all logical probabilities that consider this process is true or false.

Ext rd actv : Bool, Wr Rec : Rece, Wr Mon : Monit, Wr Cl : Clo, Wr Rb : Req.

PRE = True

Post (If Actv = True AND Rb = 1 Then Qual = True and Tic = True and Disms = True)

And

(If actv = True and Rb < > 1 Then Qual = True and Tic = True and Disms = False)

And

(If actv = False and Rb =1 Then Qual = False and Tic = False and Disms = False)

In general, the composition Rule for all processes of MIES problem (generating, Receiving, Qualifying a stream of frames) is combined of all the VDM specifications, where the VDM specification is generated from the decision table using the decision table conversion rule.

3. Discussion And Concluding Remarks:-

In this research, the issue of bridging the gap between informal and formal requirements specification languages is addressed and integrating, the two approaches for doing so. These two approaches are illustrated through a simplified MIES problem.

The cognitive approach initially uses structured analysis (SA) specifications as an aid to human understanding and cognition. This understanding helps the requirements engineer develop top–level specifications of the system using the

formal VDM specification language. The approach then lets SA guides the stepwise refinement of the VDM specifications. Precision, unambiguity, provability, and error detection are the benefits of the approach. This approach, however, requires the use of considerable human effort, thinking and heuristics for coming up with a set of top–level and stepwise–refined VDM specification.

Given the basis of VDM in predicate logic, it may also be possible to automatically convert the VDM specification of a process at a certain level of the DFD into an executable prototype ; e.g., In Prolog. Such a prototype can be useful in human understanding of a process which in turn can help in analyzing and assessing the SA and VDM models of the system. Furthermore, with the current research in translating VDM into Ada. This approach provides a promise of producing Ada code from informal structured analysis requirements specifications.

ISSUES GENERATED FOR FUTURE WORK

From the perspective of the informal specifications community, it provides specification verification mechanisms and the resulting rigour and precision in specification. Additionally, through formal specifications, it moves the informal specification community closer to the possibility of automated code generation.

From the perspective of the formal specifications community, it extends the applicability of formal specifications into the front – end requirements elicitation process. The reason for the slow acceptance of the formal specifications seems to be the high learning time and the lack of communicability of formal requirements specification languages. By developing methods and mechanisms for translating informal specifications into formal specifications, the process of developing and verifying formal specifications can be made transparent to the unsophisticated user. These formal methods could become useful earlier in the requirements specification process.

Finally, the synergy created by the integration of the two approaches can create benefits for the whole requirements process that are greater than the sum of benefits accruing to each of the formal and informal camps. It may become possible to conduct the requirements elicitation and requirements assurance tasks in a truly concurrent and interleaved manner with the informal part of the method eliciting the requirements and the formal part verifying them in an iterative fashion, then the overall quality and efficiency of the requirements elicitation process should be increased.

REFRENCES

[1] I. Shemer, "Systems Analysis: A systemetic Analysis of a Conceptual Model," Commun, ACM, VOL. 30 no. 6, PP. 506 – 512, June 1987.

[2] P. Ein – Dor and E. Segev, A Paradigm for Management Information Systems: Heath / Lexington, 1981.

[3] W.T. Olle, J. Hagelstein, I.G. MacDonald, C. Rolland, H.G. Sol, F.J.M. Van Assche, and A. Verjjin – Stuart, Information Systems Methodologies: A Framework For Understanding : Addison – Wesley, 1988.

[4] R.T. Mittermeir, P. Hsia, and R.T. Yeh, "Alternatives to Overcome the Communication Problems of Formal Requirements Analysis," in Requirements Engineering Environments, Ohno, Ed : North – Holland 1982; see also, Information Analysis : Selected Readings, R. Galliers, Ed: Addison – Wesley, 1987 , PP. 153 – 165.

[5] G. Goldkuhl, "Information Systems Requirement Based on a language Action View – A methodological Outline," in Proc. 6th Scandinavian Research Seminar Systemeering, Bergen, Norway, Aug, 1983.

[6] D. Andrews and P. Gibbins, An Introduction to Formal Methods of Software Development. Milton Keynes, UK : The Open University Press, 1988, Units 1 – 4.

[7] C.B. Jones, Systematic Software Development Using VDM. Engle: Prentice – Hall, 1986.

[8] Martin D. Fraser, Kuldeep Kumar, vijay K. Vaishnavi, "Informal and Formal Requirements Specification Languages: Bridging the Gap," With the Department of Computer Information Systems, Georgia State University, Atlanta, GA 30303, IEEE No. 9143156, 1989.

[9] T. DeMarco, Strucatured Analysis and System Specification.: Prentice–Hall, 1978.

[10] C. Gane and T. Sarson, Structured Systems Analysis, Cliffs, NJ : Prentice – Hall, 1978.

[11] R.O. Jones, Modern Structured Analysis : Prentice – Hall, 1989.

[12] B. Langefors, Theoretical Analysis of Information Systems, 4th ed. Lund, Sweden : Student litterateur, 1973.

[13] M. H, "Development and application of a meta – IV compiler," in Lecture Notes in Computer Science 252 : VDM 87, VDM – A formal Method at Work, D. Bjorner, C.B. Jones, M. Mac an Airchinnigh, and E. Neuhold, Eds: Springer – Verlag, PP. 118 – 140, 1987.

[14] D.O, Neill, "VDM Development with Ada as the target Language," in lecture Notes is Computer Science 328 : VDM, 88, VDM – The Way Ahead, R. Bloomfield, L. Marshall, and R. Jones, Eds : Springer – Verlage, 1989, PP. 116 – 123.

[15] P. Place, Software Eng. Res. Inst., Carnegie Mellon Univ., Pittsburgh, PA, Private Communication, Feb, 1990.

[16] G.S. Blair, L. Blair, and J.B. Stefani, "Specification Architecture for Multimedia Systems in Open Distributed Processing Systems, " Distributed Multimedia Research Group, Computing Department, Lancaster University, United Kingdom. 1997

Fatma A.Omara and *Reham A.Mahmoud*

المواصفات المنهجية والغير منهجية لمشكلة تبادل المعلومات (تشمل نقل الصوت والصورة)

يهتم هذا البحث باختراق الفجوة بين عمل المواصفات المنهجية لمشكلة تبادل المعلومات (تشمل نقل الصوت والصورة) وخاصة نقل الصورة فى بيئة متعددة الإمكانيات من خلال أنظمة الاتصال الموزعة بالطريقة المفتوحة عبر شبكات الحاسب. واستخدام طريقة التصميم الهيكلى كطريقة لبناء مواصفات غير منهجية للمشكلة بينما تستخدم (طريقة فينا المتطورة) كطريقة لبناء مواصفات منهجية للمشكلة. فالطريقة الأولى (طريقة التصميم الهيكلى) تستخدم كطريقة توضيحية لمحلل النظام وتساعد أيضاً فى تطوير وبناء عناصر تكوين المرحلة الثانية. والطريقة الثانية (طريقة فينا المتطورة) تعتمد أساساً على القوانين المتسلسلة منطقياً الناتجة من تخطيط مسارات البيانات الواضحة فى طريقة التصميم الهيكلى السابقة.