"Graphical Representation of Real-Time System Design Elements with Notational Specifications"

Senthil Kumar Janahan[1] Gajanan P.Arsalwad[2] Veeramanickam[3] Arun Sahayaraj[4]

[1,4]Vels Institute of Science Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, India.

[23]*Trinity College of Engineering and Research, Pune, Maharashtra, India-411 048*

skumar.se@velsuniv.ac.in[1], gajanansggs@gmail.com [2], manic.veera@gmail.com[3],

Abstract: System Modeling in Software Engineering involves high level of complexity. The design of a system should be simplified in such a way that it can be replicated accurately in an implementation phase of Software Development Life Cycle (SDLC). Complexity and criticality of Real-Time System are crucial to care, therefore design phase bears great importance. This paper represents three types of Real-Time System design elements in graphical notational format. Action Oriented notation set describes tasks or actions involved in Real-Time System. While Data-Oriented notations set represents the elements related to input and output data elements. The third set addresses to Communication Oriented elements graphically. Using these three sets of graphical notations Real-Time System can be modelled in simplified and unambiguous manner.

Keywords: Real Time System (RTS), Graphical Notations, Modeling, Design, Software Modeling, Action Orientation, Data Orientation, Communication Orientation.

Introduction

A system is called as Real Time System (RTS) where the correctness is measured with timing as well as logical or computational constraints. Real-Time Systems are those where the correctness is checked on the scale of predictability, timeliness, performance and schedulability. Quality of Service, Structural, Behavioral, and Functional aspects of Real-Time System can be described using Model Driven Approach.

Process Control Systems, Automated Teller Machines, Missile Guidance Systems Satellite Data Acquisition Systems, Patient Monitoring Systems, Air Traffic Monitoring Systems, Space Shuttle Systems, Automated Manufacturing Systems, Space Stations are some examples of Real-Time Systems. Failure of such Complex and Critical systems may cause damage in case of Human Beings, Property as well as Important Data. Because of this fact it is important to understand the need for Real-Time System Design and Modeling and that can be done by understanding current Modeling Techniques and its Components.

A generic Real-Time System can be depicted as a controlled subsystem, is any system representing an application environment such as Missile Guidance System, Telephone Switching System, etc which dictates the Real-Time Requirements. The Control Subsystem controls some computing and communication equipment which are used by controlled subsystems. The Operator Subsystem operates the entire activity. The Processes and Resources of such system are governed by a Software System which is known as Real Time System (RTS).

Modelling of Real-Time System: The design of Real-Time System must identify the timing requirements of system performance i.e. logically Correct and Timely. The timing constraint is specified as Deadline, a single time value by which the resulting action must complete. The Timing Constraints can be categorized into two types, Soft and Hard Timing Constraints. Hard constraints include a description of timeliness. A late completion of action is incorrect and constitutes a system failure.Where in Soft Constraints sometimes missing an entire action execution may be acceptable.

Real-Time System characteristics include Action, Concurrency, Resources, Time, Schedulability, Performance, Distributed, Embedded, Controlling and Reactive etc. Representing the behavioural aspect of a system is known as 'Modeling'. To model a system some modelling methodologies are followed. A methodology is the result of long-term determination to make a complete, efficient and correct procedure available to developers. System development Methodology in Software Engineering is a framework that is used to structure, plan and control the process of developing an Information System. A design methodology is composed of design creation and design verification steps. Use of methodology can guarantee of obtaining intended results. It facilitates early evaluation of the application feasibility; increases design productivity and quality and also help in project organization and management.

Deficiency of Notations in Previous Systems: In Structured Analysis and Design Technique (SADT) diagrams are created in a top-down fashion. It offers notes to represent entities and activities. It also uses a variety of arrows to resolve to relate boxes.

Structured Analysis and Structured Design (SA/SD) shows data flow in the system is using Process Model or Data Flow Diagrams (DFDs) and text specifications. It shows control information with control flows in the process model. It uses state models to highlight events, sequence and combinations, actions and models. It shows calling structure between functions with structure model.

Jackson System Development (JSD) method is model's behaviour of Real World over time. Each entity is mapped onto a software process called as 'Task'.

Hierarchical Object-Oriented Design (HOOD) includes the definition of a graphical description i.e. boxes and arrows. It provides an abstraction of a solution with clear, High-Level and easy-to-understand formalism. It offers a reduced but consistent view of the object and allows hierarchical refinement as well as an understanding of the solution.

Real-Time Structured Analysis and Design (RTSAD) is an extension of SA/SD and dedicated to Real-Time System. It is primarily a specification method addressing the software requirements of the system being developed. The extension to the Structured Analysis is driven by the desire to represent more precisely the behavioural characteristics of the system under development. It uses state transition diagrams to control flows and integrates state transition diagrams with data flow diagrams through the use of control transformations.

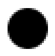
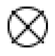
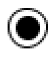
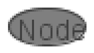
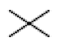
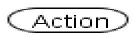
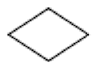
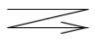
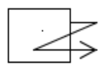
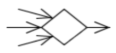
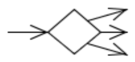
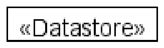
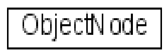
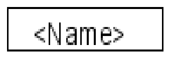
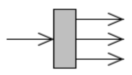
Design Approach for Real-Time System (DARTS) method emphasizes the decomposition of an RTS into concurrent tasks and defining the interfaces between these tasks. It provides a set of task structuring criteria for decomposing an RTS into concurrent tasks.

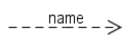
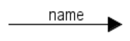
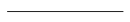
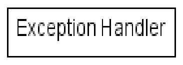
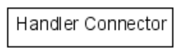
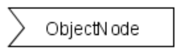
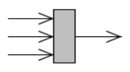
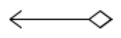
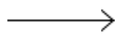
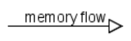
The Notations are intended to support the design approach not to replace them. It is a convenient way to represent software and to make a mental picture of its architecture. Therefore, the graphical description is complemented by the textual description which includes all details. It allows formal expression and refinement of the object characteristics and properties. The textual notations leave provision for both informal and formal texts. Allowing the definition of a documentation structure is which can serve as a framework for step by step integration of advanced notations.

Tools can be used to capture and formally verify the characteristics of an object in the system. The graphical notation recalls the context of the design piece but hides most implementation details. Thus describing the design complexity is while the textual notations help all the details including traceability and control of dependencies between modules with consistency checking. These notations allow using powerful structuring concepts for describing and organizing a system as a set of interconnected hierarchies of objects.

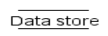
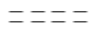
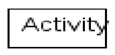
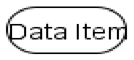
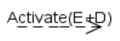
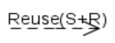
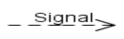
Proposed Work: Three sets of Modeling Notations are represented here.

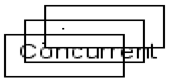
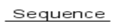
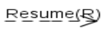
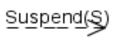
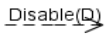
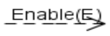
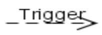
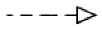
Action Oriented Notations: Action Orientation in System design emphasizes the structure of the system. It is similar to Function Oriented Design methodology. Every action has a state and can be subdivided or integrated. SA/SD focuses on the decomposition of the system functions. In DFD the actions are known as 'processes'. Function trees are used for representing the system functions. After decomposition, function trees are transformed into process diagrams. Use Case Diagrams are action-oriented diagrams which can be coupled with several design methods. While designing Real-time Systems, action-oriented diagrams can be explored to refinement and specification of related actions. Use Case and other diagrams lack notational specifications while representing Real-Time System design components. They need to be rich in number as well as clarity and specification to address most of the design requirements of Real-time System.

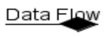
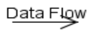
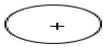
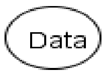
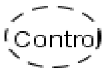
Diagram: Action Oriented	Details
	This symbol is used to denote the start of the activity
	This symbol is used to stop the activity or process
	This symbol is used to denote the end of the activity
	This symbol is used to denote the node of an activity
	This symbol is used to denote to stop of activity
	This symbol is used to denote the action
	This symbol is used to denote the decision
	This symbol is used to denote the con activity
	This symbol is used to denote the Exception handling
	This symbol is used to denote the merging of the decision
	This symbol is used to denote the distribution of decision
	This symbol is used to denote the data store
	This symbol is used to denote the object node
	This symbol is used to denote the object with the name
	This symbol is used to denote the fork

	This symbol is used to denote the discrete flow
	This symbol is used to denote the end message
	This symbol is used to denote the connection
	This symbol is used to denote the exception handler
	This symbol is used to denote the exception handler
	This symbol is used to denote the object node
	This symbol is used to denote the
	This symbol is used to denote the aggregation
	This symbol is used to denote the connection or message
	This symbol is used to denote the memory flow

Data-Oriented Notations: In the procedural approach to programming, the focus is on functions or procedures. While in case of Object Orientation focus is on real-time entities acting as an 'Objects'. When it comes to the Data-Oriented Design the emphasis is transferred from functions and object to 'Data'. This approach targets mainly the type of data which is an input to an output from the system. An input and output data requirement defines the design of a system. This approach can suite where the performance and criticality of the system are assessed using quality and timeliness of processed data. Real-Time applications involving databases like Banking Applications, an interactive application such as games, systems like Weather Forecasting Systems could come under this category. While modelling such systems following Data-Oriented Notations are useful which can represent system graphically.

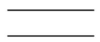
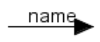
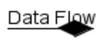
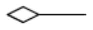
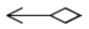
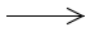
Diagram: Data oriented	Details
	This symbol is used to denote the Datastore
	This symbol is used to denote the buffer
	This symbol is used to denote the activity
	This symbol is used to denote the data item
	This symbol is used to denote the activate signal
	This symbol is used to denote the pause signal
	This symbol is used to denote the signal

	This symbol is used to denote the concurrent process
	This symbol is used to denote the sequence
	This symbol is used to denote the resume signal
	This symbol is used to denote the suspend signal
	This symbol is used to denote the disable signal
	This symbol is used to denote the enable signal
	This symbol is used to denote the trigger signal
	This symbol is used to denote the discrete signal

	This symbol is used to denote the continuous data flow
	This symbol is used to denote the data flow
	This symbol is used to denote the selection process
	This symbol is used to denote the discrete data transformation
	This symbol is used to denote the control element

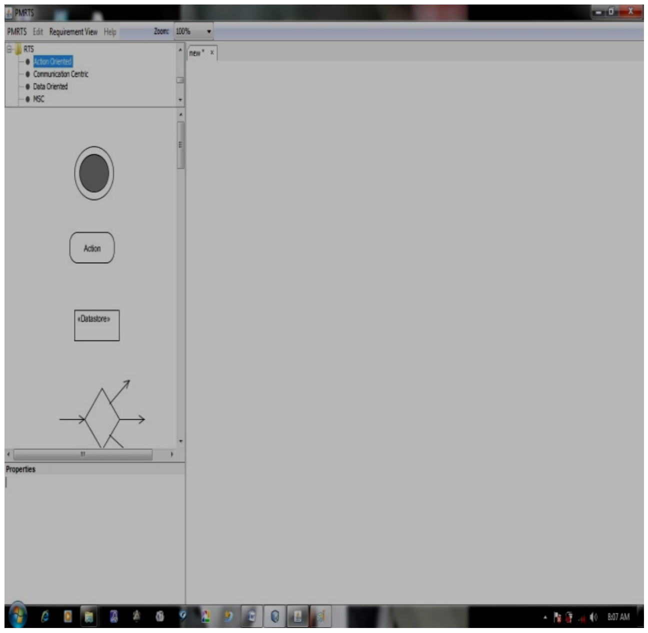
Communication-Centric Notations: This approach focuses on the interaction between or among different elements of the system. It illustrates the dynamic behaviour of the system. It emphasizes on the structural organization of the objects.

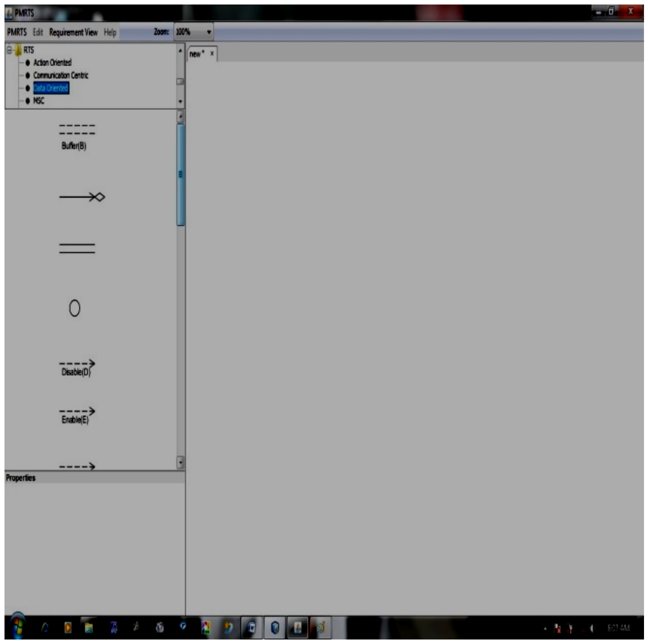
This approach shows the object organization. The flow of control can be handled with or without considering time sequence while using communication centric diagrams. The overall flow of control and behaviour of the whole system can be viewed by these diagrams.

Diagram: Communication Centric	Details
	This symbol is used to denote the connection equivalent connection
	This symbol is used to denote the association end bound
	This symbol is used to denote the composition not bound
	This symbol is used to denote the composition two end bound
	This symbol is used to denote the aggregation part end bound
	This symbol is used to denote the connection or association end bound

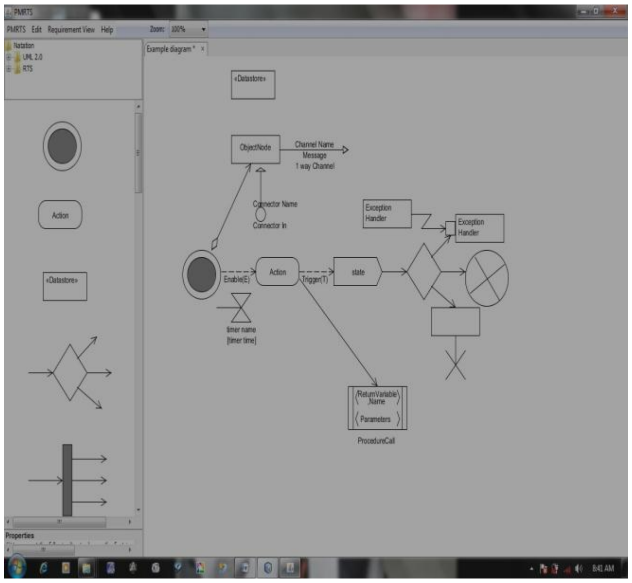
Designing Real-Time communication between system components, man-machine interaction like in interactive games these diagrams are very useful. Following Communication Centric Notations can be considered under this design approach.

Implementation: The implementation of these three sets of notation is done using NetBeans IDE Version 7.2 and Windows 8. The final output is presented in the form of a Tool with Graphical Interface to the user who is System Designer. The following images show the Screenshots of the same.





The following screenshot shows the model designing using this too for any Real-Time System.



Conclusion: System Modeling is a critical task and it should be carried out with utmost care. While designing Real-Time System Graphical Notations are very useful, since notational specification can extricate ambiguous elements from the system design. These notations also help in describing system elements in textual format.

Different modelling techniques are available for modelling but when it comes to Real Time System modelling, they lag the on the scale of the notational specification. This paper discusses different types of graphical notation which are very useful while designing Real-time System. Action Oriented Notation set explores the details of tasks in Real-Time System. It maps tasks or functions from the system and represents it into the graphical format on the scale of its Structure and Behavior. The system used to take data as an input and intend to produce some useful output in the form of data again. These elements can be explicitly represented by a second set of notations i.e. Data-Oriented notation Set. The behaviour of the whole system or individual components can be represented by communication in between two or more components of a system. This concept of a Real-Time System Model can be better addressed by Communication Centric Notation Set. To summarize, this paper addresses three kinds of notation sets to represent Real-Time System Model in an unambiguous manner. The system designer can clearly represent Real-Time System graphically using these set of notations.

Future Work: This paper represents sets of notations which can be made richer by adding some more notations so that Structure and Behavior of Real-Time System can be precisely represented in the design phase. Few more categories can be identified which will show either aspect of a system. We can add some constraints to these notations so that its behaviour can be tailored to a high extent. On the other side, if some ambiguities are there they can be removed by finding some pinholes in graphical representations. A profiler can be added to each notation to make it more descriptive and clear for Real-Time System design. In the end, no work is complete or perfect, so some more findings will always drive this work for the betterment of Real-Time System Modeling and Design.

Gajanan Arsalwad received the B. Tech and M. Tech degrees in Information Technology and Computer Science and Engineering from the Swami Ramanand Tirth Marathawada University and Pune University Maharashtra, India, in 2008 and 2011, respectively. He is currently working as Assistant Professor in Information Technology Dept. at TCOER, Pune.

Senthil Kumar Janahan received his B.Tech and M.E degrees in Information Technology and Computer Science and Engineering respectively from affiliated institutions of Anna University Chennai, Tamilnadu India in 2006 and 2011 respectively. He is currently working as an Assistant Professor in School of Engineering Department of Computer Science and Engineering at VISTAS, Chennai, India.

M. R. M. VeeraManickam is currently working as an Assistant Professor in Dept. of Information Technology, Trinity College of Engineering and Research, affiliated to SPPU, Pune. He received his B.Tech. Degree in Information Technology from LVEC, Anna University, Chennai, India, in 2006, and M.Tech. degree in Information Technology from Sathyabama University, Chennai, India, in 2011. His main research work focuses on E-learning

References

[1] Daniel L. Moody Patrick Heymans, RaimundasMatulevicius, “Improving the Effectiveness of Visual Representations in RequirementsEngineering: An Evaluation of *I** Visual Syntax”, 17th IEEE International Requirements Engineering Conference, IEEE 2009

[2] Peter Gluchowski, Christian Kurze, Christian Schieder, “A Modeling Tool for Multidimensional Data using the ADAPT Notation”, Proceedings of the 42nd Hawaii International Conference on System Sciences, IEEE 2009

[3] QIAN ZHANG, “Visual Software Architecture Description Based on Design Space”, The Eighth International Conference on Quality Software, IEEE 2008

[4] Xiao He, Zhiyi Ma, Weizhong Shao, Ge Li, “A metamodel for the notation of graphical modelling languages”, 31st Annual International Computer Software and Applications Conference(COMPSAC 2007)
IEEE 2007

[5] Muan Yong Ng, Michael Butler, “Towards Formalizing UML State Diagrams in CSP”, Proceedings of the First International Conference on Software Engineering and Formal Methods (SEFM’03), IEEE 2003

[6] MIGUEL FELDER, MAURO PEZZ`E, “A Formal Design Notation forReal-Time Systems”, ACM Transactions on Software Engineering and Methodology, Vol. 11, No. 2, April 2002

[6] Jose L. Fernandez, “An Architectural Style for Object-Oriented Real-Time Systems”, IEEE 1998

[7] Ami Silberman, Thomas J. “A Task Graph Model for Design and Implementation of Real-Time Systems”, Marlowe Department of Mathematics & Computer Science Seton Hall University Real-Time Computing Laboratory Department of Computer & Information Science New Jersey Institute of Technology Newark, NJ 07065, US, IEEE 1996

[8] H. Lewis Chau, Gary K. Chan, “Visual Language for Behavioral Specifications of Reactive Systems”, Department of Computer Science Hong Kong University of Science *8.1* Technology Clear Water Bay, Hong Kong, IEEE 1994

[9] Alice H, “A Requirements Specification Method for Adaptive Real-Time Systems”, Muntz Hughes Aircraft Company Space and Communications Group Los Angeles, California 90009, IEEE 1991

[10] Benny Hall, “SPIL A NEW GRAPHICAL DESIGN NOTATI0NS, SoftTech, Inc., IEEE 1991.