

Elements of Real-Time Systems, Represented Graphically with Their Notational Specifications

Gajanan Parshuram Arsalwad

Trinity College of Engineering and Research Located in Pune, Maharashtra, India, 411 048

Abstract: Complexity is inherent in software engineering system modeling. To ensure successful replication throughout the implementation phase of the Software Development Life Cycle (SDLC), a system's design should be kept as simple as possible. Due to the vital nature and inherent complexity of Real Time Systems, the design phase is of paramount significance. This article provides a graphical notational representation of three distinct Real Time System design features. A collection of notations called Action Oriented Notation is used to describe the activities of a Real Time System. The elements associated with input and output data are represented by the Data Oriented notations set. The third group is a graphical representation of communication-oriented aspects. These three sets of graphical notations provide for clear and concise modeling of Real Time Systems.

Keywords: Action oriented, data oriented, and communication oriented modeling, design, and software modeling, Real Time System (RTS), graphical notations.

I. Introduction

When time considerations in addition to logical or computational restrictions are used to determine a system's correctness, we refer to it as a Real Time System (RTS). Correctness in Real Time Systems is measured by their ability to execute as expected and to keep to their schedules. The Model Driven Approach may be used to explain the Real Time System's quality of service, structure, behavior, and functionality. Examples include process control systems, ATMs, and missile guidance systems. Some applications of Real Time Systems include satellite data acquisition, patient monitoring, air traffic monitoring, space shuttle systems, automated manufacturing, and space stations. Human life, valuable property, and crucial information may all be jeopardized if such complex and crucial systems failed. The importance of Real Time System Design and Modeling may be appreciated by learning about the tools and methods used in modern modeling.

Any system representing an application environment, such as a Missile Guidance System, Telephone Switching System, etc., that imposes Real Time Requirements may be thought of as a regulated subsystem of a generic Real Time System. The Control Subsystem manages the computers and networks that the other systems rely on. Every aspect of the operation is managed by the Operator Subsystem. A software system known as Real Time System (RTS) controls the processes and resources of such a system.

Modeling of Real Time System: To ensure optimal system performance, the designers of a Real Time System must take into account the constraints imposed by actual time. Deadline is the time limit that must be adhered to in order for the next action to be successful. It is possible to classify the Timing Constraints as either "Soft" or "Hard." Timeliness is a hard limitation that must be described. Incorrectly completing an activity late is a system failure. Under contrast, under Soft Constraints it is often permissible for a whole action to be missed in execution.

Action, Concurrency, Resources, Time, Scheduling Capability, Performance, Distributed, Embedded, Controlling, and Reactivity are only few of the features of a Real Time System. Modeling is the process of representing the operational characteristics of a system. Some modeling approaches are used to model a system. Methodologies are the outcome of persistent efforts to provide developers with a comprehensive, effective, and right set of guidelines. Growth of a System Software engineering methodology provides a framework for organizing and managing the steps required to create an information system. The two main components of any design process are the design creation phase and the design verification phase. The success of an endeavor may be ensured by using the proper approach. It helps with project management, organization, and early assessment of application viability. It also improves the design product's ability and quality. Inadequate Notation in Earlier Systems: Top-down diagramming is used in the SADT methodology, which stands for the Structured Analysis and Design Technique. You may use the available symbols to represent things and actions. Different kinds of arrows are used to connect the dots in the related boxes. Process Model or Data Flow Diagrams (DFDs) and text requirements are used in

Structured Analysis and Structured Design (SA/SD). In the process model, control information is shown alongside control flows. Events, sequences, combinations, and actions are highlighted using state models. Function calling structure is shown using a structural model.

The JSD technique models the dynamic behavior of the real world. In this system, every object corresponds to a 'Task' that runs in the background.

The definition of a graphical description, such as boxes and arrows, is part of Hierarchical Object Oriented Design (HOOD). It offers a high-level solution abstraction with a simple, straightforward notation. The object is simplified while maintaining coherence, and hierarchical refinement and solution comprehension are made possible.

Structured Analysis and Design (SA/SD) has a spinoff specifically for Real Time Systems called Real Time Structured Analysis and Design (RTSAD). It's a mechanism for specifying the software features that will be included in a system. The motivation for this change to Structured Analysis is the need for a more accurate representation of the system's behavior. It applies control transformations to state transition diagrams and utilizes them to govern flows, as well as to combine state transition diagrams with data flow diagrams.

Decomposing an RTS into concurrent jobs and specifying their interfaces is a fundamental part of the Design Approach for Real Time Systems (DARTS) methodology. In order to break down an RTS into independently running components, it offers a set of criteria for arranging activities.

Instead than serving as a replacement for the design process, the Notations are meant to supplement it. It's a practical method of representing software and seeing its structure. As a result, the visual depiction is accompanied with a comprehensive written description. Characteristics and attributes of objects may be formally expressed and refined in this way. Both casual and formal writing styles are accommodated by the textual notations. providing the means to define a documentation structure that may act as a basis for the gradual incorporation of more complex notations.

Capturing the properties of objects in a system and formally verifying them is possible with the help of the right tools. The graphical notation is evocative of the design's setting, yet abstracts away the nitty-gritty of its execution. Thus, the textual notations aid in explaining the design complexity while helping with all the nuances, such as dependency traceability and control and consistency checking across modules. To describe and organize a system as a series of interrelated hierarchies of

objects, these notations make use of potent structuring notions.

II PROPOSE D WORK






Three sets of Modeling Notations are studied and represented here with Tool for Graphical Representation

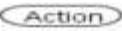







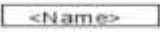




(1) **Action Oriented Notations:**

System architecture is given primary importance in Action Oriented Design. It's a procedure not unlike the FOD approach. Every process may be broken down into its constituent parts or combined into a whole. Function decomposition is fundamental to SA/SD. 'Processes' are what we call the steps taken in DFD. The capabilities of the system are represented by means of function trees. As a result of the breakdown, function trees become process diagrams. Practical Application Action-oriented diagrams, which may be used with other design approaches, are called diagrams. Action oriented diagrams may be used to further specify and refine linked activities during the design of Real time Systems. Notational standards are lacking for Use Case and other diagrams used to depict elements of Real Time System design. To meet the majority of Real time System design needs, they should be comprehensive in both quantity and quality

A,

05
2018








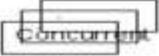



Action Oriented	Details
	This symbol is used to denote the start of activity
	This symbol is used to stop the activity or process
	This symbol is used to denote the end of activity
	This symbol is used to denote the node of an activity
	This symbol is used to denote to stop of activity


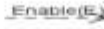
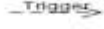






	This symbol is used to denote the action
	This symbol is used to denote the decision
	This symbol is used to denote the con activity
	This symbol is used to denote the Exception handling
	This symbol is used to denote the merging of the decision
	This symbol is used to denote the distribution of decision
	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 name
	This symbol is used to denote the fork
	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

2. **Data Oriented Notations:** The primary units of analysis in procedural programming are the processes themselves. While in Object Orientation, the emphasis is on 'Objects' that exist in the actual world. In the Data Oriented Design, 'Data' takes center stage, replacing traditional design tenets like 'functions' and 'objects'. The focus of this method is on the information that enters and leaves the system. The parameters of a system's architecture are its input and output data needs. Where performance

and reliability are paramount, this method might be a good fit.


3. based on the accuracy and timeliness of the processed data. Database-driven real-time applications like banking software, interactive software like video games, and data-driven systems like weather forecasting models are all examples. Data-oriented notations are helpful for graphical system representation when designing such systems.







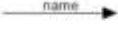



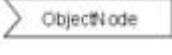
Data oriented	Details
	This symbol is used to denote the Data store
	This symbol is used to denote the buffer
	This symbol is used to denote the activity
	This symbol is used to denote the data item
	This symbol is used to denote the activate signal
	This symbol is used to denote the pause signal
	This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

- 5. Communication Centric Notations:** In this method, the relationships between the various parts of a system are highlighted. It shows how the system behaves dynamically. The emphasis is on the physical arrangement of things.
- 6.** Using this method, the hierarchy of the objects may be shown. Communication-centric diagrams may be used to manage

control flow either chronologically or non-chronologically. These diagrams are useful for observing the overall behavior and control flow of a system. Useful for planning real-time data exchange between various parts of a system, as well as human-machine interaction in the form of interactive games. This design methodology allows for consideration of the Communication Centric Notations.

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
	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

EXPERIMENTAL RESULTS

The implementation of these three sets of notation is done using Java and NetBeans IDE Version 7.2. 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.

Diagram 1: The Tool provides the Menu Bar with Main Menu named as PMRTS, Edit option for existing diagrams, the

requirement view (optional) and help menu. It also provides zoom option for better view of the diagram.

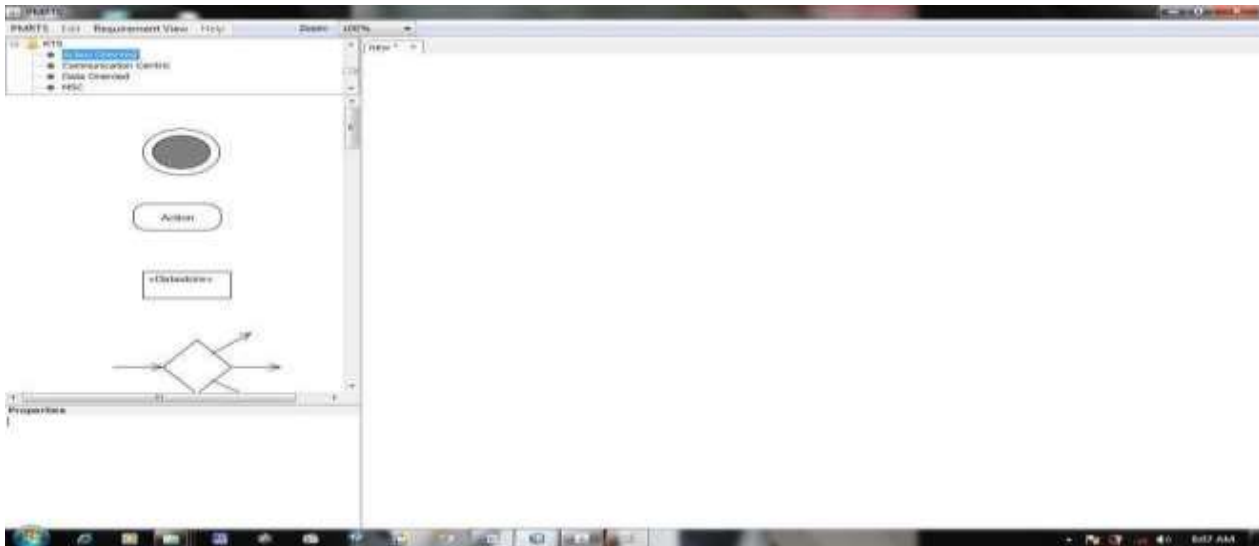


Diagram 2: It provides a work space for designer with three sets of Notations i.e. Action Oriented, Communication Centric and Data Oriented. The Left Panel is divided into two horizontal parts where upper part provides the Notation Set Name and Lower Part provides the Notation available in reselected set with facility to drag and drop in the user working space.

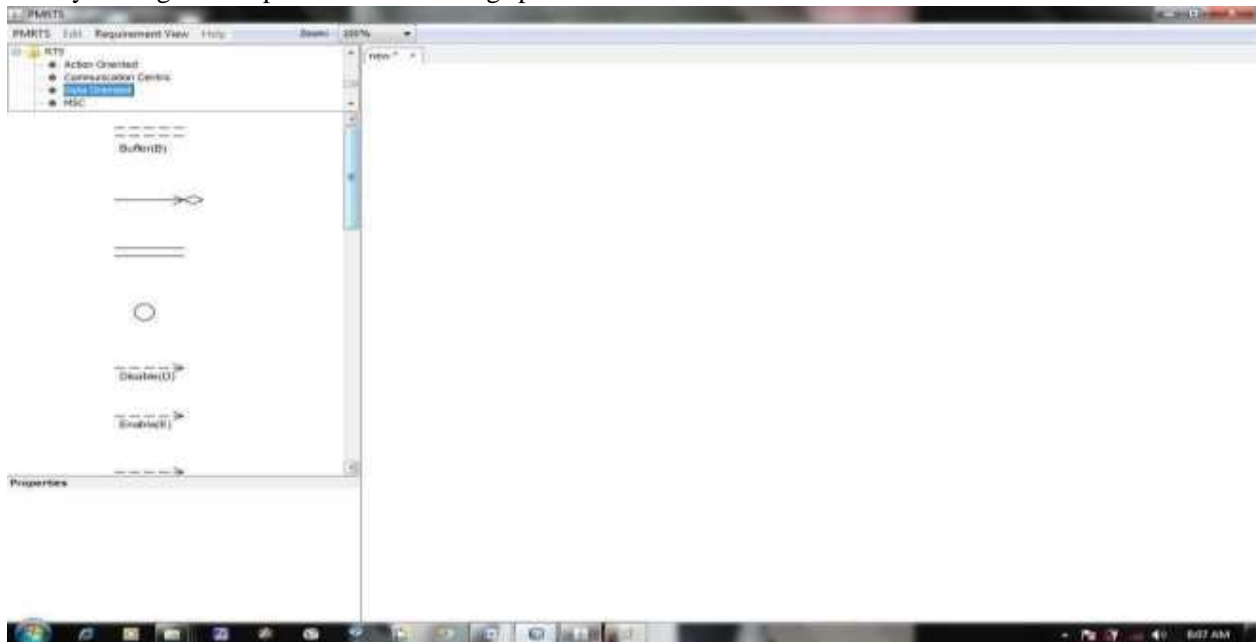
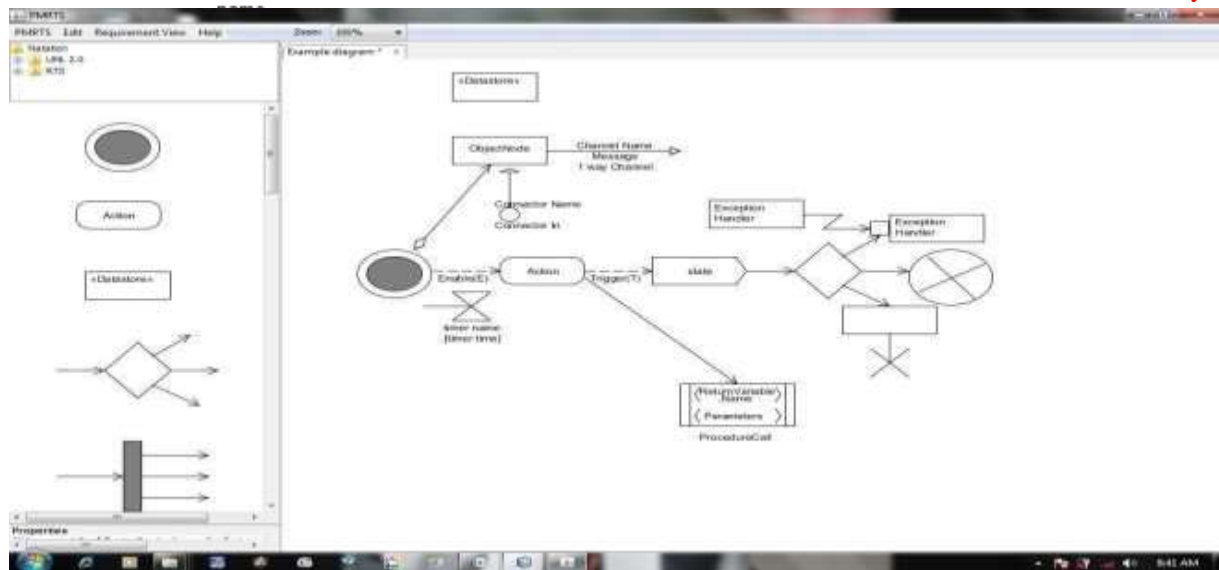


Diagram 3: The following screenshot shows the model designing using the tool for any Real Time System. The notations available in left panel can be dragged and dropped in right panel (working space). Once completed the diagram can be saved and reopened for further changes in needed.



CONCLUSION

System modeling is a crucial activity that requires meticulous attention to detail. Notational definition is helpful in Real Time System graphical notation design because it may remove unclear parts of the system design. These notations also facilitate the written description of system components.

Although several modeling approaches exist, notational specifications for Real Time System models are not as advanced. In this article, we'll look at the many graphical notation styles that might come in handy while creating a Real Time System. The Real Time System task specifics are investigated in Action Oriented Notation. The system's duties or functions are mapped onto a graphical representation of the system's structure and behavior. The goal of every system designed to process data is to generate more data that can be used in some way. The second collection of notations, called the Data Oriented Notation collection, provides an explicit representation for these components. Communication between system components may model the behavior of the overall system or of isolated parts. Communication Centric Notation Set is more suited to dealing with this idea of a Real Time System Model. In conclusion, this study discusses three distinct notation sets for accurately depicting Real Time System Models. Using these notations, a system designer may create a graphical representation of a Real Time System.

FUTURE WORK

This article depicts notation sets that might be further enriched by the addition of a few additional notations in order to accurately portray the Structure and Behavior of Real Time Systems during the design phase. There aren't many other groups that will adequately represent each side of a system. These notations may be made to behave in a variety of ways by adding restrictions. However, if

any uncertainties exist, they may be cleared up by locating specific flaws in visual depictions. Each notation may be made more illustrative and understandable for Real Time System design by including a profiler. Since no piece of work is ever really finished or flawless, new discoveries will continually motivate this effort to advance Real Time System Modeling and Design.

Arsalwad Gajanan

In 2008 and 2011, Gajanan Arsalwad earned Bachelor of Technology and Master of Technology degrees in Information Technology and Computer Science and Engineering, respectively, from Swami Ramanand Tirth Marathawada University and Pune University in Maharashtra, India. As of right now, he is an assistant professor in the IT department of TCOER in Pune.

REFERENCES

- [1] Daniel L. Moody Patrick Heymans, Raimundas Matulevicius, "Improving the Effectiveness of Visual Representations in Requirements Engineering: 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 modeling 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 for Real-Time Systems", ACM Transactions on Software Engineering and Methodology, Vol. 11, No. 2, April 2002
- [7] Jose L. Fernandez, "An Architectural Style for Object Oriented Real-Time Systems", IEEE 1998
- [8] 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
- [9] H. Lewis Chau, Gary K. Chan, "Visual Language for Behavioral Specifications of Reactive Systems", Department of Computer Science Hong Kong University of Science & Technology Clear Water Bay, Hong Kong, IEEE 1994
- [10] 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
- [11] Benny Hall, "SPIL A NEW GRAPHICAL DESIGN NOTATIONS, SoftTech, Inc., IEEE 1991.