International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2020 IJSRCSEIT | Volume 6 | Issue 1 | ISSN : 2456-3307
DOI : 10.32628/CSEIT206114

Model Driven Methodology for JAVA

Karishma J. Karande[1], Prakash S. Prasad[2]

[1]M.Tech Scholar, Department of Computer Science and Engineering, Priyadarshini College of Engineering, Nagpur, Maharashtra, India

[2]Assistant Professor, Department of Information Technology, Priyadarshini College of Engineering, Nagpur, Maharashtra, India

ABSTRACT

Real-time systems are getting expanding attention with the rising application situations that are wellbeing basic, complex in usefulness, high on timing-related execution necessities, and cost-delicate, for example, self-governing vehicles. Improvement of real-time systems is blunder inclined and exceptionally reliant on the refined space mastery, making it an exorbitant procedure. There is a pattern of the current programming without the real-time thought being re-created to realize real-time highlights, e.g., in the huge information innovation. This paper uses the standards of model-driven building (MDE) and proposes the main procedure that naturally changes over standard time-sharing Java applications to real-time Java applications. It opens up another exploration bearing on advancement automation of real-time programming dialects and motivates many research addresses that can be mutually examined by the inserted systems, programming dialects just as MDE communities.

Keywords : Real-Time Programming Languages, Real-Time Specification for Java, Model-Driven Engineering

I. INTRODUCTION

Real-time systems frequently encase stringent worldly prerequisites, where a real-time application must respond to boosts from the earth (counting the entry of physical time) inside time interims directed by nature [10]. Such systems have been all around rehearsed in numerous fields, and their application areas continue developing with rising situations [15].

In spite of the fact that planning prerequisites are classified as nonfunctional necessities, they are fundamental to wellbeing related systems. In [26], the creator groups framework disappointment modes into irregular disappointments and efficient disappointments, where precise disappointments add to framework risks which could prompt episodes with cataclysmic results. Orderly disappointments can be additionally arranged into utilitarian disappointments and timing disappointments. Ensure that a wellbeing related framework has the right planning prerequisites and while its planning conduct fulfills these planning necessities. In this way, showing real-time properties structures key proof in guaranteeing the wellbeing of a security-related framework. Because of the high efficiency, conveyability and moderately low support cost, the Java programming language has gotten broad consideration in the real-time and security basic spaces [21, 45]. For example, Java was embraced in [31] and [30] to lessen circulated figuring inactivity in a brought together could-based stage for independent vehicles. Notwithstanding, these works have been created concentrating on usefulness with restricted thought of timing and security ensure, particularly when the perplexing discernment capacities are included.

As commanded by wellbeing guidelines, for example, the ISO 26262 for car systems and IEC 61508 for practical security, hard real-time limitations are fundamental to ensure the wellbeing of the framework (e.g., the vehicle) and its encompassing condition. Along these lines, there is a need to push these current signs of progress in the direction of the real-time system. There is a pattern that developed Java methods (which were created without the idea of real-time) are re-created to have real-time ensures (e.g., real-time large information systems [18] and real-time stream preparing procedures [32]). The significant explanation is that those basic and moderate techniques (like leaving enormous security edges) that were sent by and by are losing ground, with the always convoluted usefulness, higher planning related execution necessities what's more, constrained assets on the rising real-time applications [3, 12–14].

In spite of its ubiquity, standard Java can't be legitimately applied to create real-time programming because of the absence of offices, for example, string booking, asset sharing control, memory the board, and so on., which are basic to accomplish consistency [11] as far as fleeting conduct. This has propelled the improvement of the Real-Time Specification for Java (RTSJ) [8]. RTSJ holds the characteristic points of interest of Java and gives a lot of real-time offices to ensure the framework transient conduct, and yet is more earnestly to be utilized by programming engineers.

Contrasted with the conventional time-sharing applications in Java, growing real-time applications utilizing RTSJ depends exceptionally on the ability in the structure of the real-time system and requires careful comprehension of the determination. It is additionally mistaken inclined because of the intricacy. These above make improvements in real-time applications an expensive procedure. Despite the fact that there have been framework investigation and confirmation systems [35] to guarantee accuracy in the planning stage, regarding both sensible and worldly conduct, it stays an open and testing issue how to dispense with human-related wrong factors (e.g., brought about by constrained comprehension of the real-time ideas and lacking involvement in RTSJ offices). The security basic nature in some real-time systems areas enhances the effect of such concerns.

Model-driven building (MDE) is a contemporary programming advancement worldview, which advances models as first-class curios. In light of models, engineers can play out a progression of model administration tasks in a robotized way, and in the end produce programming curios, for example, documentation and working code. This decreases the measure of time required to build up a framework and accordingly improves the profitability of programming engineers, by at any rate a factor of 10 much of the time [23, 25]. Embracing MDE likewise decreases the number of mistakes all through the advancement procedure and improves consistency [51]. What's more, MDE can be applied to any space to accomplish automation, because of the idea of area explicit demonstrating and the interoperability gave by model administration activities, which can be executed in a computerized way.

In this paper, we apply the standards of MDE in the area of real-time programming with Java. We propose the primary procedure that can naturally change over existing time-sharing Java applications to real-time applications in RTSJ, through a progression of model administration activities. The yield programming is in full consistence to the RTSJ determination, with conditions to the RTSJ runtime condition supporting booking, memory the executives, asset sharing, asynchrony, and so on. This empowers the designers with the constrained real-time foundation to perform the fleeting examination on their non-real-time base code and convert it to source code written in RTSJ. Because of the use of MDE systems, profitability and consistency all through the advancement. Human blunders are disposed of in the automation. We depict a robotized toolchain related to the proposed philosophy. All the practical squares in the toolchain and the included specialized methodologies are clarified. The logical difficulties tended to and concealed issues found towards the programmed age of real-time applications with MDE systems are talked about. We likewise bring up future research bearings past this paper.

II. RELATED WORK

Demonstrating is a fundamental piece of any framework designing procedure. Specialists of all orders develop models of the systems they mean to work to catch, test and approve their framework structure thoughts with different partners before focusing on a long and exorbitant generation process.

MDE is a product building technique that plans to decrease the unplanned multifaceted nature of programming systems by advancing models that emphasis on the basic intricacy of systems, as the top of the line relics of the product improvement process. Rather than those conventional programming advancement techniques, where models are mostly utilized for correspondence and after death documentation process, in MDE models are the primary living and advancing ancient rarities from which solid programming improvement curios can be delivered in an analysable and computerized style.

MDE was proposed when object-situated systems arrived at a point of fatigue [7, 37]. MDE comprises the most recent change in perspective in programming designing as it raises the degree of deliberation past that gave by third era programming dialects. In ongoing examinations, MDE has been appeared to build efficiency by as much as a factor of 10 [23, 25], and altogether improve significant parts of the product advancement procedure, for example, viability, consistency and recognizability [33].

There are two significant parts of MDE —

(i) Domain specific demonstrating, where area specialists make their very own space explicit displaying dialects (DSMLs) to catch the ideas in their space (and make examples of their DSMLs to show their systems);

(ii) Model administration activities, which are programs performed on models in a robotized way to produce programming building antiquities. Model administration activities commonly incorporate, yet are not constrained to:

- ✓ Text-to-Model Transformation (T2M): to change over content, (for example, source code) into models dependent on parsing rules characterized in the change;
- ✓ Model Validation: to check the well-formedness of models, just as custom limitations against the components in models;
- ✓ Model-to-Model Transformation (M2M): to interoperate between various demonstrating advances, where one kind of model is changed into another sort;
- ✓ Model-to-Text Transformation (M2T): to produce content dependent on the substance of the model (e.g., documentation age and source code age);
- ✓ Model Comparison: to contrast various adaptations of a model with discover what is changed;
- ✓ Model Merging: to coordinate models characterized by various gatherings yet share model components.

MDE has been applied to an assortment of spaces, with demonstrated advantages. In [28] MDE is applied to change model question dialects to MySQL inquiries to diminish the exertion and blunder rates in physically making MySQL inquiries. In [51], MDE is applied to consequently produce completely useful graphical editors for UML profiles. In [5], MDE is applied to change characteristic dialects to database question dialects to frame complex inquiry utilizing basic normal language syntaxes.

Growing real-time systems through a model-based methodology aren't novel in the network [24, 46]. The thought proposed in this paper is incompletely enlivened by them. None of these works study the relocation from standard Java to real-time Java. Furthermore, a large number of past endeavors depend on the idea of model-driven

engineering, which is an obsolete MDE practice and has an absence of hardware support. By applying MDE methods, as recently portrayed, Real-Time framework designers can profit by the efficiency gain from MDE, just as the consistency and practicality through automation gave by MDE.

RTSJ, initially created as Java Special Request 1 under the Java Community Process in 2001 1, has been well-practiced in a wide scope of utilization spaces, including car, producing control, flying and data systems [22, 43, 46, 47]. For example, RTSJ has been applied to the auto-pilot arrangement of an unmanned flying vehicle, which is the main Java-based framework that fulfills all Boeing's operational necessities and flew in tests [1]. Jcoap, realized by RTSJ, gives real-time interchanges to IoT systems [29]. In [17], RTSJ has been applied in a piece of realtime enormous information preparing systems with FPGA-based equipment quickening. In industry, JamaicaCAR created by both Acis and Perrone Robotics2 gives a lightweight application structure to vehicle head units and in-vehicle data systems. Likewise, Acis and CLAAS3 present arrangements (in particular Jamaica-IoT) for computerized plant and assembling, which empowers sending and activity of information investigation and control rationale at the system's edge.

The RTSJ is intended to help both hard and delicate real-time applications. This particular comprises of two significant segments —

i. expansions from the Java programming language; and
ii. adjustments on the semantics of the standard Java Virtual Machines (JVM) [8].

This area quickly surveys the programming determination of RTSJ, together with its reference usage just as the supporting Virtual Machines (VM). Point by point portrayals of each RTSJ office and the application models can be found in [10].

Altogether, there are seven augmentations from the standard Java language that are given in the bundle javax.realtime, including task planning and dispatching, memory the executives, shared asset control, offbeat occasion taking care of, and so on.

One significant office gave in RTSJ isjavax.RealtimeThread, which takes a lot of planning-related parameters (e.g., need, period and cutoff time) determining a real-time string's discharge, execution and timing properties. Three kinds of strings are gotten from this element: intermittent, sporadic and aperiodic, contingent upon the information discharge parameter. Moreover, a lot of non-concurrent occasion handlers are given to permit client characterized activities in the instances of cutoff time miss or move overwhelm. As a matter of course, a preemptive fixed-need scheduler books the real-time strings, however, client characterized planning and dispatching strategies are likewise conceivable.

Another significant expansion of the real-time memory of the executives model. In RTSJ, a lot of memory the executive's offices are given in RTSJ (e.g., Immortal Memory and ScopedMemory) to permit the development of self-characterized memory models. Notwithstanding, RTSJ forces a lot of memory getting to decides that confine memory-getting to practices to anticipate dangling reference (i.e., references that point to objects in recovered memory squares). With memory the executives model characterized, the standard Java city worker is never again required so its erratic impedance is abstained from during run-time. Afterward, a real-time trash specialist is bolstered by JamaicaVM, which permits the utilization of Heap memory and facilitates the improvement of RTSJ applications by abstaining from building complex memory models.

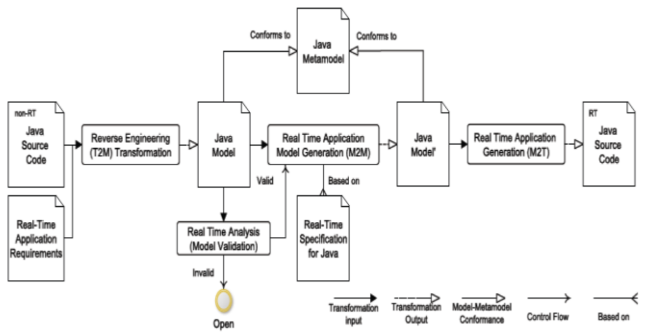
Conforms to
Java Metamodel
Conforms to
non-RT
Java Source Code
Reverse Engineering (T2M) Transformation
Java Model
Real Time Application Model Generation (M2M)
Java Model'
Real Time Application Generation (M2T)
RT
Java Source Code
Real-Time Application Requirements
Valid
Based on
Real Time Analysis (Model Validation)
Real-Time Specification for Java
Invalid
Open
Transformation input
Transformation Output
Model-Metamodel Conformance
Control Flow
Based on

Figure 1. Time-sharing applications to real-time applications migration

Within the sight of shared articles, RTSJ gives a few asset sharing arrangements like need Inheritance [40] and Priority Ceiling Protocol (PCP) [36]. Among these conventions, the PCP yields the limited blocking time (i.e., one basic area in particular) and assurance stop free asset gets to. Likewise, asynchrony is all around taken care of by means of a lot of offbeat occasion taking care of offices. At long last, a lot of time-related offices (e.g., real-time framework clock and High Resolution Time with the granularity of nanoseconds) are bolstered.

III. CONCLUSION

This paper proposes a model-driven procedure that consequently changes time-sharing Java applications to realtime applications in RTSJ. This approach facilitates the improvement of real-time systems by permitting programming architects to build real-time Java applications without essential information on the RTSJ programming particular. What's more, the proposed technique is good to those associations with a need to re-build up their items to have real-time highlights. The proposed philosophy gives a real-time framework improvement arrangement that diminishes programming advancement cost, builds profitability and wipes out human-related blunders. In this paper, a total standard Java to RTSJ change automation engineering is given required activities during every change stage portrayed in detail. What's more, change rules are exhibited for producing major RTSJ offices and the RTSJ run-time condition dependent on the JamaicaVM with the given data sources.

IV. REFERENCES

[1]. Austin Armbruster, Jason Baker, Antonio Cunei, Chapman Flack, David Holmes, Filip Pizlo, Edward Pla, Marek Prochazka, and Jan Vitek. 2007. A real-time Java virtual machine with applications in avionics. ACM Transactions on Embedded Computing Systems (TECS) 7, 1 (2007), 5.

[2]. Neil Audsley, Alan Burns, Mike Richardson, Ken Tindell, and Andy J Wellings. 1993. Applying new scheduling theory to static priority pre-emptive scheduling. Software Engineering Journal 8, 5 (1993), 284-292.

[3]. Neil C Audsley, Yu Chan, Ian Gray, and Andy J Wellings. 2014. Real- Time Big Data: the JUNIPER Approach. (2014).

[4]. Jason Baker, Antonio Cunei, Chapman Flack, Filip Pizlo, Marek Prochazka, Jan Vitek, Austin Armbruster, Edward Pla, and David Holmes.

2006. A real-time java virtual machine for avionics-an experience report. In 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06). IEEE, 384-396.

[5]. Konstantinos Barmpis, Dimitrios Kolovos, and Justin Hingorani. 2018. Towards a framework for writing executable natural language rules. In European Conference on Modelling Foundations and Applications. Springer, 251-263.

[6]. John Barnes. 1997. High integrity Ada: the SPARK approach. Vol. 189. Addison-Wesley Reading.

[7]. Jean Bézivin. 2005. On the unification power of models. Software & Systems Modeling 4, 2 (2005), 171-188.

[8]. Gregory Bollella and James Gosling. 2000. The real-time specification for Java. Computer 33, 6 (2000), 47-54.

[9]. Hugo Bruneliere, Jordi Cabot, Grégoire Dupé, and Frédéric Madiot. 2014. Modisco: A model driven reverse engineering framework. Information and Software Technology 56, 8 (2014), 1012-1032.

[10]. Alan Burns and Andy Wellings. 2016. Analysable Real-Time Systems: Programmed in Ada. CreateSpace Independent Publishing Platform.

[11]. Alan Burns and Andrew J Wellings. 2001. Real-time systems and programming languages: Ada 95, real-time Java, and real-time POSIX. Pearson Education.

[12]. Wanli Chang and Samarjit Chakraborty. 2016. Resource-aware automotive control systems design: A cyber-physical systems approach. Foundations and Trends in Electronic Design Automation 10, 4 (2016), 249-369.

[13]. Wanli Chang, Dip Goswami, Samarjit Chakraborty, and Arne Hamann. 2018. OS-aware automotive controller design using non-uniform sampling. ACM Transactions on Cyber-Physical Systems 2, 4 (2018), 26.

[14]. Wanli Chang, Dip Goswami, Samarjit Chakraborty, Lei Ju, Chun Xue, and Sidharta Andalam. 2017. Memory-aware embedded control systems design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 36, 4 (2017), 586-599.

[15]. Wanli Chang, Alma Pröbstl, Dip Goswami, Majid Zamani, and Samarjit Chakraborty. 2015. Reliable CPS design for mitigating semiconductor and battery aging in electric vehicles. In IEEE International Conference on Cyber-Physical Systems, Networks, and Applications. 37-42.

[16]. Robert I. Davis and Alan Burns. 2011. A survey of hard real-time scheduling for multiprocessor systems. Acm Computing Surveys 43, 4 (2011), 1-44.

[17]. Ian Gray, Neil Cameron Audsley, Jamie Garside, Yu Chan, and Andrew John Wellings. 2015. FPGA-based acceleration for Real-Time Big Data Systems. In 9th HiPEAC workshop on Reconfigurable Computing.

[18]. Ian Gray, Yu Chan, Jamie Garside, Neil C. Audsley, and Andy J. Wellings. 2015. FPGA-based hardware acceleration for Real-Time Big Data systems.

[19]. Les Hatton. 2004. Safer language subsets: an overview and a case history, MISRA C. Information and Software Technology 46, 7 (2004), 465-472.

[20]. Florian Heidenreich, Jendrik Johannes, Mirko Seifert, and Christian Wende. 2009. Closing the gap between modelling and java. In International Conference on Software Language Engineering. Springer, 374-383.

[21]. Thomas Henties, James J Hunt, Doug Locke, Kelvin Nilsen, Martin Schoeberl, and Jan Vitek. 2009. Java for safety-critical applications. In 2nd international workshop on the certification of

safety-critical software controlled systems (SafeCert 2009).

[22]. Erik Yu-Shing Hu, Eric Jenn, Nicolas Valot, and Alejandro Alonso. 2006. Safety critical applications and hard real-time profile for Java: a case study in avionics. In Proceedings of the 4th international workshop on Java technologies for real-time and embedded systems. ACM, 125-134.

[23]. Ari Jaaksi. 2002. Developing mobile browsers in a product line. IEEE software 19, 4 (2002), 73-80.

[24]. A Juan, Jorge Garrido, Juan Zamorano, and Alejandro Alonso. 2014. Model-driven design of real-time software for an experimental satellite.

[25]. IFAC Proceedings Volumes 47, 3 (2014), 1592-1598.

[26]. Juha Karna, Juha-Pekka Tolvanen, and Steven Kelly. 2009. Evaluating the use of domain-specific modeling in practice. In Proceedings of the 9th OOPSLA workshop on Domain-Specific Modeling.

[27]. Timothy Patrick Kelly. 1999. Arguing safety: a systematic approach to managing safety cases. Ph.D. Dissertation. University of York York, UK.

[28]. Dimitrios S Kolovos, Richard F Paige, and Fiona AC Polack. 2008. The epsilon transformation language. In International Conference on Theory and Practice of Model Transformations. Springer, 46 60.

[29]. Dimitrios S Kolovos, Ran Wei, and Konstantinos Barmpis. 2013. An approach for efficient querying of large relational datasets with oclbased languages. In XM 2013-Extreme Modeling Workshop. 48.

[30]. Björsn Konieczek, Michael Rethfeldt, Frank Golatowski, and Dirk Timmermann. 2015. Real-time communication for the internet of things using jcoap. In 2015 IEEE 18th International Symposium on Real-Time Distributed Computing. IEEE, 134-141.

[31]. Shaoshan Liu, Jie Tang, ChaoWang, QuanWang, and Jean-Luc Gaudiot. 2017. Implementing a Cloud Platform for Autonomous Driving. arXiv preprint arXiv:1704.02696 (2017).

[32]. Shaoshan Liu, Jie Tang, ChaoWang, QuanWang, and Jean-Luc Gaudiot. 2017. A unified cloud platform for autonomous driving. Computer 50, 12 (2017), 42-49.

[33]. HaiTao Mei, Ian Gray, and Andy Wellings. 2016. Real-Time stream processing in java. In Ada-Europe International Conference on Reliable Software Technologies. Springer, 44-57.

[34]. Parastoo Mohagheghi and Vegard Dehlen. 2008. Where is the proof?-A review of experiences from applying MDE in industry. In European Conference on Model Driven Architecture-Foundations and Applications. Springer, 432-443.

[35]. Renaud Pawlak, Martin Monperrus, Nicolas Petitprez, Carlos Noguera, and Lionel Seinturier. 2015. Spoon: A Library for Implementing Analyses and Transformations of Java Source Code. Software: Practice and Experience 46 (2015), 1155-1179. https://doi.org/10.1002/spe.2346

[36]. Ben Potter, David Till, and Jane Sinclair. 1996. An introduction to formal specification and Z. Prentice Hall PTR.

[37]. Ragunathan Rajkumr. 2012. Synchronization in real-time systems: a priority inheritance approach. Vol. 151. Springer Science & Business Media.

[38]. Douglas C Schmidt. 2006. Model-driven engineering. COMPUTER-IEEE COMPUTER SOCIETY- 39, 2 (2006), 25.

[39]. Martin Schoeberl, Andreas Engelbredt Dalsgaard, René Rydhof Hansen, Stephan E Korsholm, Anders P Ravn, Juan Ricardo Rios Rivas, Tórur Biskopstø Strøm, Hans Søndergaard, Andy Wellings, and Shuai Zhao. 2017. Safety-critical Java for embedded systems.

Concurrency and Computation: Practice and Experience 29, 22 (2017), e3963.

[40]. Martin Schoeberl, Hans Sondergaard, Bent Thomsen, and Anders P Ravn. 2007. A profile for safety critical java. In 10th IEEE International Chang, Shuai Zhao, Ran Wei, Andy Wellings, and Alan Burns Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07). IEEE, 94-101.

[41]. Lui Sha, Ragunathan Rajkumar, and John P. Lehoczky. 1990. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. 39, 9 (1990).

[42]. Fridtjof Siebert. 2007. Realtime garbage collection in the JamaicaVM 3.0. In Proceedings of the 5th international workshop on Java technologies for real-time and embedded systems. Citeseer, 94-103.

[43]. Fridtjof Siebert. 2010. Concurrent, parallel, real-time garbagecollection. In ACM Sigplan Notices, Vol. 45. ACM, 11-20.

[44]. Rashmi P Sonar and Rani S Lande. 2018. Javolution-Solution for Real Time Embedded System. In 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE). IEEE, 1-10.

[45]. Chris Tapp. 2008. An introduction to MISRA C++. SAE international journal of passenger cars-electronic and electrical systems 1, 2008-01- 0664 (2008), 265-268.

[46]. Kleanthis Thramboulidis. 2007. IEC 61499 in factory automation. In Advances in Computer, Information, and Systems Sciences, and Engineering. Springer, 115-124.

[47]. Kleanthis Thramboulidis and Alkiviadis Zoupas. 2005. Real-time Java in control and automation: a model driven development approach. In 2005 IEEE Conference on Emerging Technologies and Factory Automation, Vol. 1. IEEE, 8-pp.

[48]. Christian Wawersich, Michael Stilkerich, and Wolfgang Schröder- Preikschat. 2007. An OSEK/VDX-based multi-JVM for automotive appliances. In Embedded System Design: Topics, Techniques and Trends. Springer, 85-96.

[49]. Andrew J Wellings. 2004. Concurrent and real-time programming in Java. John Wiley New York.

[50]. Shuai Zhao. 2018. A FIFO Spin-based Resource Control Framework for Symmetric Multiprocessing. Ph.D. Dissertation. University of York.

[51]. Shuai Zhao, Andy Wellings, and Stephan Erbs Korsholm. 2015. Supporting multiprocessors in the ICECAP safety-critical java run-time environment. In Proceedings of the 13th InternationalWorkshop on Java Technologies for Real-time and Embedded Systems. ACM, 1.

[52]. Athanasios Zolotas, Ran Wei, Simos Gerasimou, Horacio Hoyos Rodriguez, Dimitrios S. Kolovos, and Richard F. Paige. 2018. Towards Automatic Generation of UML Profile Graphical Editors for Papyrus. In Modelling Foundations and Applications, Alfonso Pierantonio and Salvador Trujillo (Eds.). Springer International Publishing

Cite this article as :

Karishma J. Karande, Prakash S. Prasad, "Model Driven Methodology for JAVA", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 6 Issue 1, pp. , January-February 2020.
Journal URL : http://ijsrcseit.com/CSEIT206114