52 pages including cover

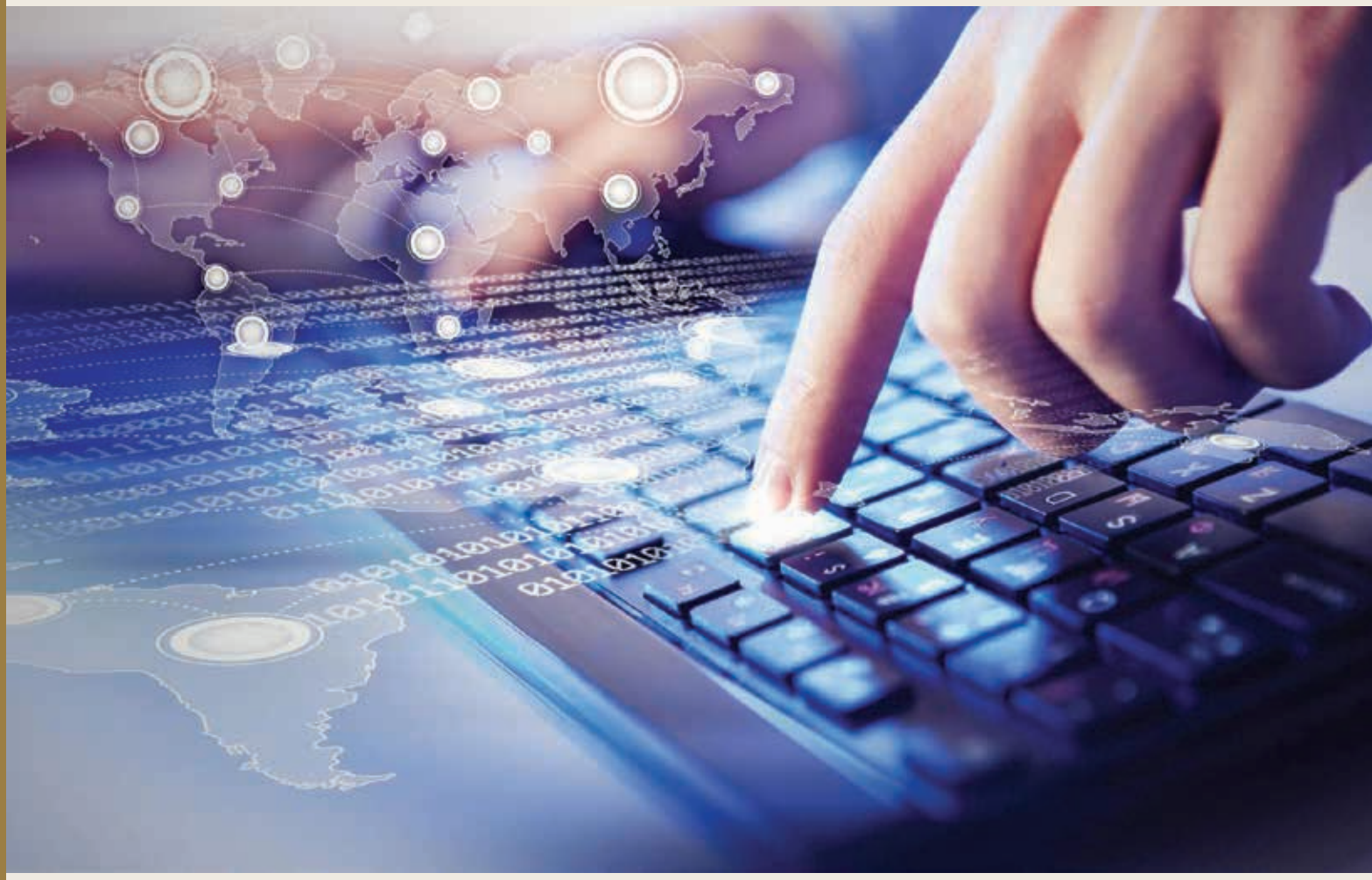
CSI Communications
Knowledge Digest for IT Community

Volume No. 40 | Issue No. 12 | March 2017

₹ 50/-

ISSN 0970-647X

Software Engineering



www.csi-india.org

Sanjay Mohapatra, Vice President, CSI & Chairman, Conf. Committee, Email: vp@csi-india.org

Date	Event Details & Contact Information
MARCH 24-25, 2017	**First International Conference on "Computational Intelligence, Communications, and Business Analytics (CICBA - 2017)"** at Calcutta Business School, Kolkata, India. Contact: **som.cse@live.com**; (M) 94754 13463 / (O) 033 24205209 **International Conference on Computational Intelligence, Communications, and Business Analytics** (CICBA - 2017) at Calcutta Business School, Kolkata, India. Contact (M) 9475413463 / (O) 03324205209, Email id : **som.cse@live.com; www.cicba-2017.in**
APRIL 15-16, 2017	**1st International Conference on Smart Systems, Innovations & Computing (SSIC-2017)** at Manipal University Jaipur, Jaipur, Rajasthan. **http://www.ssic2017.com** Contact : Mr. Ankit Mundra, Mob.: 9667604115, **ankit.mundra@jaipur.manipal.edu**
MAY 08-10, 2017	**ICSE 2017** - International Conference on Soft Computing in Engineering, Organized by : JECRC, Jaipur, **www.icsc2017.com** Contact : Prof. K. S. Raghuwanshi, **hod.it@jecrc.ac.in**, Mobile : 9166016670
JUNE 05-30, 2017	**Workshop on LAMP (Linux, Apache, My SQL, Perl/Python)**, Jaypee University of Engineering and Technology, Raghogarh, Guna - MP, **www.juet.ac.in** Dr. Shishir Kumar (**dr.shishir@yahoo.com**) 9479772915
OCTOBER 28-29, 2017	**International conference on Data Engineering and Applications-2017 (IDEA-17)** at Bhopal (M.P.), **http://www.ideaconference.in** Contact : **conferenceidea@gmail.com**
DECEMBER 21-23, 2017	**Fourth International Conference on Image Information Processing (ICIIP-2017)**, at Jaypee University of Information Technology (JUIT), Solan, India, (http://www.juit.ac.in/iciip_2017/) Contact : Dr. P. K. Gupta (**pkgupta@ieee.org**) (O) +91-1792-239341 Prof. Vipin Tyagi (**dr.vipin.tyagi@gmail.com**)

CSI Adhyayan

A tri-monthly publication for students

Articles are invited for Oct-Dec. 2016 issue of CSI Adhyayan from student members authored as original text. Plagiarism is strictly prohibited. Besides, the other contents of the magazine shall be Cross word, Brain Teaser, Programming Tips, News Items related to IT etc.

Please note that CSI Adhyayan is a magazine for student members at large and not a research journal for publishing full-fledged research papers. Therefore, we expect articles should be written for the Bachelor and Master level students of Computer Science and IT and other related areas. Include a brief biography of Four to Five lines, indicating CSI Membership no., and for each author a high resolution photograph.

Please send your article to csi.adhyayan@csi-india.org.

On behalf of CSI Publication Committee

Prof. A. K. Nayak
Chief Editor

CSI COMMUNICATIONS

VOLUME NO. 40 • ISSUE NO. 12 • MARCH 2017

Contents

Printed and Published by Mr. Sanjay Mohapatra on Behalf of Computer Society of India, Printed at G.P. Offset Pvt. Ltd. Unit-81, Plot-14, Marol Co-Op. Industrial Estate, off Andheri Kurla Road, Andheri (East), Mumbai 400059 and Published from Computer Society of India, Samruddhi Venture Park, Unit-3, 4th Floor, Marol Industrial Area, Andheri (East), Mumbai 400 093. Tel. : 022-2926 1700 • Fax : 022-2830 2133 • Email : **hq@csi-india.org** **Chief Editor: Prof. A. K. Nayak**

Editorial

Dear Fellow CSI Members,

"Before software can be reusable it first has to be usable."

– Ralph Johnson

The theme for the Computer Society of India (CSI) Communications (The Knowledge Digest for IT Community) March, 2017 issue is Software Engineering, the art and science of engineering software systems.

In this issue, Cover Story article is "History of Software Engineering: Status of Software Component, Reusability and Quality" by Munishwar Rai and Kiranpal Singh Virk. The article traces the chronological evolution of the discipline.

Vijay Sharma has contributed to Technical Trends through the article, "Future of Software Engineering". This focuses on challenges and opportunities that lies ahead of the software engineering community.

The Research front is titled, "An Essence of Soft Computing Techniques on Software Development Life Cycle". Here, C Shoba Bindu, E. Sudheer Kumar and K K Baseer have descibed various techniques like Poka-Yoke, Fuzzy Logic, Neural Networks, Particle Swarm Optimization and Genetic Algorithm in the SDLC context.

The Security Corner has S. Hemalatha and P C Senthil Mahesh giving us new insights on Pseudo Code Attack in Software Engineering.

We have several articles which provide us information on various facets and research topics of software engineering and software project management like the "Role of Software Engineering" by Hardeep Singh and Parminder Kaur; "Formal Methods in Software Engineering" by A. Sowmya Mitra; "Risk Management in Effort Estimation of Agile Methodologies" by S Rama Sree and Ch. Prasada Rao; "Real – Time System: A challenge for Testers" by Nancy Goel and Shaily Jain; Logical Hierarchy Requirement Target Planning (Lhrtp) technique to overcome risk on Software Projects" by R. Saranya and "Technological advances in Software Engineering" by V Vetriselvi.

This issue also contains Crossword, CSI activity reports from chapters, student branches and Calendar of events. This issue also covers CSI Regional Student Convention organized by KIIT CSI Student Branch, KIIT University, Odisha

We are thankful to Chair-Publication Committee and entire ExecCom for their continuous support in bringing this issue successfully.

We wish to express our sincere gratitude to all authors and reviewers for their contributions and support to this issue.

The next issue of CSI Communications will be on the theme "Big Data Analytics". We invite the contributions from all CSI members and researchers on this theme. We also look forward to receive constructive feedback and suggestions from our esteemed members and readers at csic@csi-india.org.

With kind regards,
Editorial Team, CSI Communications

President's Message

01 March, 2017

Finally my terms as President of CSI is coming to an end! Now it is time to look back and see what I had set to achieve and has been able to achieve.

CSI 2016 Convention organized by CSI Coimbatore Chapter during January 23-25, 2017 was a resounding success. The conference was very well arranged with high quality papers, and invited talks. The ambience of the conference venue (the Le Meridian hotel) was excellent. The registration had to be stopped as the halls were filled to capacity. The Inauguration function was followed by Award Presentation to outstanding members and IT personalities.

The three categories of awards (Fellowships, Hony. Fellows and Life Time Achievement) were presented. My congratulations to all the winners. It was the first time that CSI-IEEE Education Award was presented. The Award consisting of a plaque and US$500 was presented to Dr. S T Selvi.

The meetings of all statutory committees were duly held and we had fruitful deliberations. I must congratulate all the Members of the organizing Committee led by Mr. P R Rangaswamy, Mr. Ranga Rajgopal, Dr. Nadarajan, Dr. Subramanian and others for working very hard and making CSI 2016 a memorable event.

A number of training programs are being planned for our Members. It includes certification trainings on Enterprise Architecture with The Open Group, Internet Governance with ICANN etc.. We had a meeting with business delegation of Myanmar and we hope to sign an MOU with Myanmar Computer Federation vey soon.

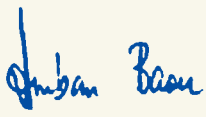
I am glad to inform you that our application to make CSI a Registered Education Provider of PMI (Project Management Institute) has been successful and CSI Chapters will be able to offer PMP trainings.

Attempts are being made to improve different publications of CSI. CSI Journal of Computing will resurface in the month of April 2017 in soft copy form.

CSI Elections are going to start soon. It took time as we had to make some radical changes. When I was elected I had promised to bring in transparency in different CSI activities. CSI elections every year is generating controversy every year. There have been questions about the membership database, issues on the integrity of the vendor who has been entrusted to conduct the elections since 2005 etc.. This year suggestions came up in the Think Tank meeting, National Council meeting, and in the AGM during CSI-2016. Accordingly we have taken steps to incorporate the suggestions. You must have received communication from CSI that a member need to validate his/ her identity by giving PAN or Aadhar number. The process has been completed we will have a clean database for the elections. This year CSI elections will be conducted by NSDL, the most reputed agency in the country who has been conducting e-voting. The agreement with NSDL has been signed and elections will start by March 15, 2017 and results announced by end of this month.

There is no doubt that lot of improvements can be done in other aspects of working CSI. It is not possible to complete in one year. The next President Mr. Sanjay Mohapatra will continue the work and hope the successive leaders continue from where we will leave.

With best wishes to one and all,



Dr. Anirban Basu
President, CSI

Dr. Anirban Basu, Bangalore, president@csi-india.org

History of Software Engineering: Status of Software Component, Reusability and Quality

Munishwar Rai
Ass. Prof., M. M. Inst. of Computer Tech. & Business Management, Maharishi Markandeshwar University, Mullana (Ambala), Haryana

Kiranpal Singh Virk
Ass. Professor, Guru Nanak Khalsa College, Yamuna Nagar, Haryana

1. Introduction

Software engineering as an engineering domain has come a long way from its origin to its present form. It has taken quite a while for software engineering to establish itself as a discipline. It has been an interesting journey all the way. The term 'software engineering' was officially accepted in 1968 when NATO organized a conference with the title 'Software Engineering' [1]. The best brains of the computer science were brought together to discuss the issues faced by the software industry like declining profits, beyond schedule delivery of solutions, over budgeted projects, low quality products etc. These issues were then collectively termed as 'Software Crisis' or 'Software Gap'. In this conference it was widely accepted that software development has lagged behind in implementing good practices as other engineering disciplines do. The solution suggested was a new discipline called 'Software Engineering' in which science and mathematics are applied to make the properties of software useful to people.

2. Programming Languages

Kolence [1], Boehm [2] and Wirth [9] stated that, in the rapid changing software field, there are wide gaps in expectations of the users and deliverance of the software industry. Some gaps are created because the software engineers have not learnt from the past mistakes. Some gaps are created because software engineers have failed to repeat the past successes. Some gaps are created because the software concepts are sometimes over stated. Let's try to understand this 'overstated' thing with a example from history. In 1950 COBOL language was designed to be readable and understandable for non-programmers. In 1990 it was projected that 5th generation languages were designed to solve a problem without writing programming code. In 2004, it was predicted that due to SOA (Service Oriented Architecture), the computing world will not require programmers. However, even in 2017 the success of lightweight programming languages like Erlang, Lua, Forth, Squirrel, newLisp etc negates the above hypothesis of doing away with the programmers. Table 1 clearly shows that the programming languages had been there and would be there in future also.

Before 1990 the focus on improving the software quality was through improving testing techniques and code reviews. Post 1990s the role of analysis and design reviews in improving quality was also emphasized with software models like CMM, ISO etc. However the role was over stated could be traced to year 2000 when Polar Lander of NASA crash landed on Mars surface [4]. The reason of failure was simple - a software bug turned off the jets while the prober was 40 metre high and it simply fell down rather than landing. The bug was eventually a missing control structure 'if'. The detailed inspection of the cause revealed that due to schedule deadline approaching, launch was initiated with incomplete testing. In other words, analysis and modeling substituted for test and validation.

Table 1 : Year-Wise Programming Languages Making their Mark

Language	Year	Language	Year	Language	Year	Language	Year
Fortran	1957	Pascal	1970	Haskell	1990	F#	2005
LISP	1958	C	1973	newLisp	1991	Clojure	2007
Algol 58	1958	Ada	1980	Python	1991	Go	2009
Cobol	1962	Smalltalk	1980	Java	1995	Rust	2010
BASIC	1964	C++	1985	Ruby	1995	Dart	2011
LISP	1964	Objective-C	1986	JavaScript	1995	Julia	2012
SIMULA	1965	Eiffel	1986	C#	2000	Swift	2014
Forth	1970	Perl	1987	Scala	2003	Erlang	2016
Algol 68	1970						

3. Sources of Software Problems

Number of concepts and ideas have faded-in and faded-out since 1950. Every concept has contributed to the growth of software engineering. Where some ideas have created success stories, others have created failures. All failures when analyzed revealed some hidden problems and their sources. The sources which contribute to the various issues of software development are shown in Table 2 and their relationship in Fig 1. The overall software journey has been all about dealing with these four sources. All concepts are suggested, developed or improved while keeping in mind the one or more of these sources.

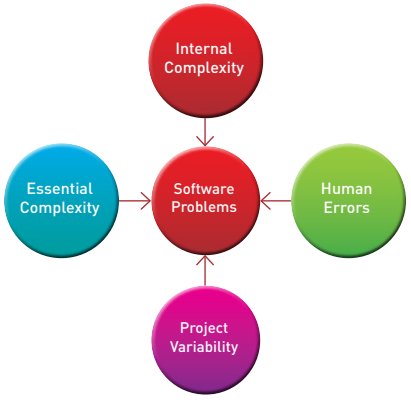


Fig 1 : Software Problems and their Sources

Table 2 : Sources of Software Problems

	Source	Description
1	Internal Complexity	Requirements Specification, Design, Source Code, Maintenance Strategies
2	Essential Complexity	Related with the domain for which software solution is developed
3	Human Errors	Human psychology, behavior aspects
4	Project Variability	Repeatable and Unique aspects of the project

Fig 1 : Software Problems and their Sources

4. Decade-wise History

In this section, we would try to know the milestones in software engineering. With each decade status of software component, reusability and quality is also discussed separately. Starting with 1950, software was considered second to hardware. The journey of over 66 years saw software emerged as a separate independent industry. This transition has seen how the software industry disassociated itself from core practices of other engineering disciplines. In later evolution of software industry, the same practices were incorporated in modified way. The base year for the journey of software could be taken as 1950s. The concentrated and conscious effort to establish software as individual entity could not be traced beyond 1950.

1950-1959 : In this decade computer meant hardware and computer professionals meant electrical/electronic engineers or mathematicians. Software professionals were second rung leaders. The prominent software project was SAGE (Semi-Automated Ground Engineering) for U.S. and Canadian air defense. This project brought engineers from cross disciplines like communications, electrical, radar and computer. The hardware oriented focus of the industry could be gauzed from the fact that leading professional societies of software professionals are IEEE (Institute of Electrical and Electronics Engineers) Computer Society and ACM (Association of Computing Machinery).

Status of Software Component, Reusability and Quality : The software sizes were small and with no predecessors. Therefore, there was no need for component and reuse. The only quality aspect was to deal with the project management.

1960-1969: The decade saw the boundaries between software and hardware being drawn more clearly. The hardware related practices like 'measure twice, cut once' were no longer seem to be applicable on software development. However, the thought of making software development a separate independent entity brought its own sets of problems. First was to assume software was easier to change than hardware. Another misconception was assumed that software unlike hardware does not wear out. Ease of modification lead to 'code and fix approach'. Doing away with civil engineering fundamentals like, 'strong foundation results in stable building', resulted in structure less programs. Critical design reviews of production line manufacturing industry were parked away. As the size and complexity of the software was growing, human-interaction issues gave the psychologist angle to the engineering discipline. In 1963 the first time-sharing system appeared (MIT, Stanford, McCarthy, DEC PDP- 1). It enhanced the user-computer interaction .This resulted in evolution from batch processing systems to time-sharing. This also meant the start of 'Internet' minus 'websites'. However, websites came later in 1989. In 1968 NATO Science Committee [1] organized first conference in Germany on software engineering to discuss the issues being faced by. The top computer brains were invited to discuss the past, present and future of the software industry. It was widely discussed and accepted that there are successful stories like OS/360, sort routines, payroll applications.

Status of Software Component, Reusability and Quality: This decade coined the term 'Software Component'. In the invited address of NATO conference [1], M.D. Mcilroy coined the idea of 'Mass Produced Software Components'. It would be appreciative of the fore-vision of Mcilroy by quoting what he said about software components in 1968: *"My thesis is that the software industry is weakly founded, and that one aspect of this weakness is the absence of a software components sub-industry. We have enough experience to perceive the outline of such a sub-industry"*. E. W. Dijkstra's paper on 'Structure Programming' [6] talked about software components and 'Go To Statement Considered Harmful' [7] were landmarks. Quality attributes like correctness, portability, adaptability, scalability, reliability, completeness were deliberated upon in NATO conferences [1][6].

1970-1979: The NATO conference of 1968 and 1969 became the foundations for initiative for amalgamation of best approaches of different disciplines that were used in 1950s and solutions suggested during 1960s. The decade witnessed the emergence of the various concepts as shown in Table 3.

Status of Software Component, Reusability and Quality : The biggest paradox of 1970 came with the Royce Waterfall Model [8]. Although this model was not sequential as shown in Fig. 2, it was interpreted as sequential process due to two main reasons, Firstly due to government process standards of that time. Secondly, Royce himself have drawn sequential life cycle to forewarn the pitfalls as in Fig 3. In fact Royce had fore-warned about the pure sequential nature of waterfall model in his words: *"I believe in this concept, but the implementation described above is risky and invites failure."* After a good start, the late 1970 saw its own set of limitations. With the growing domain complexity (or essential complexity in Table 1) for which software solutions were being designed, quality factor like scalability and productivity became prominent issues. The metrology aspect of the quality was also suggested by McCall [11] and Boehm [10] in their respective software quality models.

1980-1989 : The quality and standardization saw the compliance of standard maturity models (SW-CMM, ISO-9001) in software processes due to mandatory requirement of government for software bidding. The domain complexity resulted in various

Table 3 : The Concepts of 1970s

Concept	Author	Concept	Author
Mathematical Proof	C. Floyd	Psychology of Computer Programming	Weinberg
Programming Calculus	E. W. Dijkstra	Mythical Man Months	Brooks
Top-Down Structured Programming	Mills	Pascal	Wirth
Coupling & Cohesion	Constantine	Software Evolution Dynamics	Lehman and Baldy
Information Hiding	Parnas	Reusable Product Line	Toshiba
Abstract Data Types	W.A. Wulf J.V. Guttag W.C. Lim	Structured Design & Programming	Jackson
Boehm Quality Model	B. W. Boehm	Waterfall Model	W.W. Royce
McCall Quality Model	J. A. McCall		

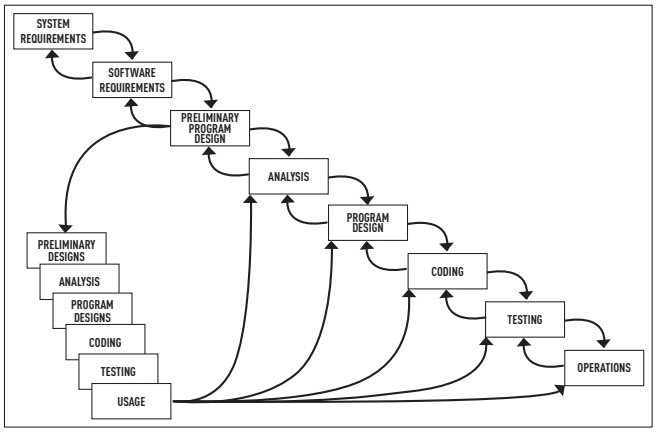


Fig. 2: The Royce Waterfall Model (1970)

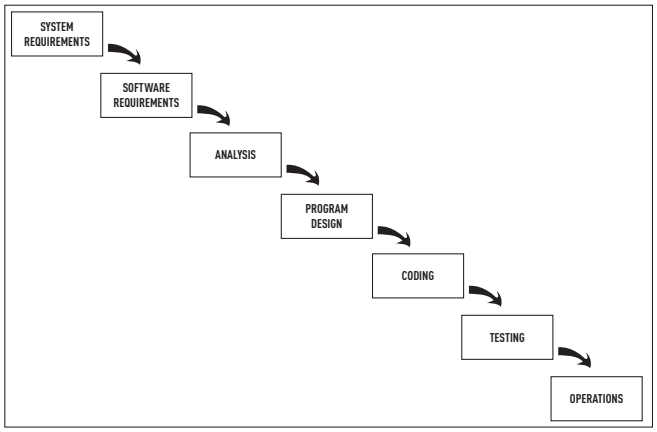


Fig. 3 : The Royce Waterfall Model (1970)

software tools like IPSE (Integrated Programming Support Environment), CASE (Computer-Aided Software Engineering) Other achievements of this decade were object oriented methods, software factories [5], 4GL, CAD/CAM. In 1985 FSF (Free Software Foundation) and GNU-GPL (General Public License) were established by Richard Stallman which became the milestone for Open Source Software in 1990s.

Status of Software Component, Reusability and Quality: In 1986, Brooks [3] suggested there is no 'silver bullet' in software engineering that will solve all problems. Another interpretation of Brooks' thought could be that till now the overall objective of the improving the software was to eliminate one or more source of problem (refer Table 1). Instead of waiting for technological breakthrough we must go for Software Reuse, Rapid Prototyping and Incremental development. In 1988, Barry Boehm suggested a risk-driven model known as 'Spiral Model' to introduce concurrent engineering. Training, reuse, prototyping, and process improvement were suggested to improve quality.

1990-1999 : The late 1980 and earlier 1990 witnessed work of three influential authorities on object oriented aspects namely- Grady Booch's OOAD (Object Oriented Analysis and Design), James Rumbaugh's *OMT* (Object Modeling Technique) and Ivar Jacobson's OOSE (Object-Oriented Software Engineering). The mid 1990 saw the amalgamation of OOAD, OMT and OOSE into UML (Unified Modeling Language). The emergence of 'world wide web' further strengthened the OO methods. Concurrent engineering along with FSF and GNU-GPL of 1980s contributed in Open Source Software. The famous 'dot-com bubble' speculation came to the end in 1999-2001. The famous Y2K (Year 2000) problem of converting the legacy software compatible with twenty first century has been a turning point. The main issue in Y2K problem was that the legacy applications stored only the last two digit of the year. At the turn of the century the year would have been 00 instead of 2000.

Status of Software Component, Reusability and Quality: 1990s was also reuse intensive with COTS (Components Of The Shelf) intensive software development, agile methodology and eXtreme programming. The software quality models namely- FURPS, Dromey and ISO 9126 were suggested during this period. The component based architecture were concretely documented as COM, DCOM and CORBA.

2000-2010: The 'google search engine' and the cloud computing wre the main successful contributions of this decade. The other ones were Model Driven Architecture, Concurrent risk driven process, Domain specific software architectures, hybrid Agile and plan driven methods – which took under its umbrella the initiatives of 1990 like eXtreme Programming. Object oriented methods of 1980s were further improved with design patterns and UML. The concept of web services and

aspect oriented programming were widely applauded. The Y2K problem of previous century initiated the idea of integration process and enterprise level development.J2EE, .NET were the prominent examples.

Status of Software Component, Reusability and Quality : Components were taken to a new level with SOA (Service Oriented Architecture) in 2009. Scalability and Security came up as major quality concerns. Software Fortress Model was suggested in 2003 where in every aspect of software was in analogy of a fortress.

2010-2016 : The sustained growth of cloud computing saw every service, software, database to be cloud oriented. The supply chain management remains the core competency issue for commercial applications. Machine Learning, IoT (Internet of Things), product line engineering and light weight programming are the happening things of this decade.

Status of Software Component, Reusability and Quality : Software quality model ISO 25010 has taken under its fold the previous works on software quality models especially the work of Barry Boehm, and James McCall.

5. Future Challenges

One of the major challenges to software industry is to provide high degree quality and functionality at low cost and that too with-in the shortest possible time. There is still an open challenge in building system with software components to accurately predict the quality attributes by the produced system. Today's typical application is a result of integration of various technological advancements that may include:

- Service-Oriented Component-based Applications
- Multi-Language Programming
- Components Distribution via Remote Services
- Heterogeneous environments
- Dynamic Isolation & resilience
- Development & Monitoring Tools

Service-oriented Component-based Applications provides a framework to construct modularized applications consisting of software components that uses software services provided by other components. Component-based applications SOA are slowly and sturdily making "ubiquitous computing" a success. Security issues in reusable components and the use of vulnerable third-party components and code in new applications has become such a major security issue [12]. Except for the white papers of technology organizations like IBM, Cohorte, Infosys, Stelligent etc, very few research papers are available that considers the concurrent status of the software component reusability in the industry.

6. Conclusions

There have been two approaches of software development. One approach is to eliminate the four sources that contribute to software problems. This is not even theoretically practical as domain complexity would increase in future also with software engineering travelling in unchartered waters. Second approach is to accept the fact that the four sources cannot be eliminated and word around it. The history clearly points to the fact that new domains will be more complex, human interaction could be minimized but not eliminated, newer methods, architecture, languages and technologies would be developed. The only front of improvement lies in good quality models with focus on metrology aspects.

7. References

[1] P. Naur and B. Randell, "Software Engineering: Report of a conference sponsored by the NATO Science Committee," NATO Softw. Eng. Conf., no. October 1968, p. 231, 1968.

[2] B. Boehm, "A View of 20th and 21st Century Software Engineering," Proc. 28th Int. Conf. Softw. Eng. SE - ICSE '06, pp. 12–29, 2006.

[3] F. P. Brooks, "No Silver Bullet — Essence and Accident in Software Engineering," *IEEE Computer* 4, No. 2, pp. 10-19, 1987.

[4] N. G. Leveson, "The Role of Software in Recent Aerospace Accidents," 19th International System Safety Conference, 2004

[5] J. Greenfield and K. Short, "Industrializing Software Development", http://softwarefactories.com/index.html

[6] J. N. Buxton and B. Randell, "Software engineering techniques," Rep. a Conf. Spons. by NATO Sci. Comm., April, 1969.

[7] E.W. Dijkstra, "Go To Statement Considered Harmful," Communications of the ACM, Vol. 11, No. 3, pp. 147-148, 1968.

[8] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," Proceeding of WESCON, 1970.

[9] N. Wirth, "A Brief History of Software Engineering," *IEEE Ann. Hist. Comput.*, vol. 30, pp. 32–39, 2008.

[10] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative evaluation of software quality," in *2nd Int. Conf. Softw. Eng.*, pp. 592–605, 1976.

[11] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in software quality: Concept and Definitions of Software Quality," RADC AFSC, Griffis Air Base, New York, RADC-TR-369, Vol I (of three),. November 1977.

[12] Cobb M. (2013, Nov.), "Open source code reuse: What are the security implications?" [Online], Available: http://searchsecurity.techtarget.com/answer/Open-source-code-reuse-What-are-the-security-implications/

About the Authors

Dr. Munishwar Rai [CSI-00133605] is currently working as Associate Professor at MMICT&BM (MCA department) of MM University, Mullana, India. He has around 19 years of teaching experience. His area of interest is software Reusability, Data mining, Computer Organization and computer networks. He is a life member of CSI. He has published more than 20 papers in both international and national levels. He is involved in various academic and automotives administrative activities at MM University, Mullana. (e-mail: munishwar.rai@mmumullana.org)

Kiranpal Singh Virk [CSI-I0175624] is working as Asst. Professor with Dept. of CS, Guru Nanak Khalsa College, Yamuna Nagar, Haryana, India. Currently he is pursuing Ph.D. from Maharishi Markandeshwar University, Mullana (Ambala), Haryana, India. He is author of one book, and more than 10 papers. He is a life member of CSI, and APG (Association of Punjab Geographers), and member of IPDA (International Professional Development Association). His research interest includes Software Engg., IT, and Wireless Sensor Networks. (e-mail: kpsvirk@yahoo.com)

Future of Software Engineering

Vijay Sharma
VP, SilverOakHealth, Engineering for Bangalore based Digital Healthcare company

Our dependence on software permeates nearly every aspect of our lives, it is a good time to ask ourselves where this relationship is headed. But one thing is sure here that the softwares of tomorrow will work completely on new challenges encountering altogether different constraints. This article focuses on challenges and opportunities that lies ahead of the software engineering community. The future is running at us with unprecedented pace and the best we can do is to embrace it with open mind and accelerate our learning so that we are better prepared for tomorrows challenges.

1. Software Trainers

With all the Machine Learning and Artificial Intelligence coming into the main stream, machines will need to learn from vast amounts of data. Some time in the future they will self train too, but that is at least a decade away, till then we will need humans to train machines. Among us there will emerge a new bread of software professionals known as Software Trainers, who will tell programs to exists on the world and how to recognize objects. Trainers will tell programs what objects they would see, in exact shape, color and size. They state exact sequence of actions, what to process and when to process etc. Data engineering will play a bigger role then compared to software engineering. Software trainers don't need to understand machine learning techniques, but apply them. The programs will learn how to put the pieces together using through feedback and digital playgrounds, thus creating a path of self induced learning.

2. More code will Created and much more will be Destroyed

With more open source projects coming up and more kids learning programming at school level. Software development is becoming highly accessible to everybody, enabling anybody to create software or apps. With the introduction of Cloud, the infrastructure for hosting data and software has virtually dropped to near zero, many instances of software can be created in a few hours, potentially scaling to millions of users, then discarded a few days later. We will see billions of disposable apps being created and destroyed very soon, most even lasting for just few days. As in the case mobile apps for events, product launch specific website we will see lost of code will be used just for few days and discarded later and never being used again.

[Image Source : https://cdn2.hubspot.net/hubfs/326641/CAPA.jpg]

3. Binary is not the past but it is the future

With the entry of Internet of Things in future in everyday life more and more devices and machines will communicate with each other, this communication will surpass the machine to human communication in the order of both scale and magnitude. Software code for tomorrow will be used more for machine to machine communication than for human to machine. Passing data back and forth in JSON packets with REST protocols with lots of overhead of tags will make the communication and thus ultimately performance slow. If both ends of the interface are machines then why would they need to communicate using text and pay for overhead of JSON & XML when in the receiving end it needs to converted back to binary? Thus why not ship the bits directly? We will see introduction of new tools and software procedures which are targeted towards producing machine understand programs. The application code and the transmission packets will become more and more binary efficient signifying leap in the communication protocols. While maintainability and human readability are important today, it might get trumped by the need for extreme performance tomorrow.

4. Smartphones will do everything

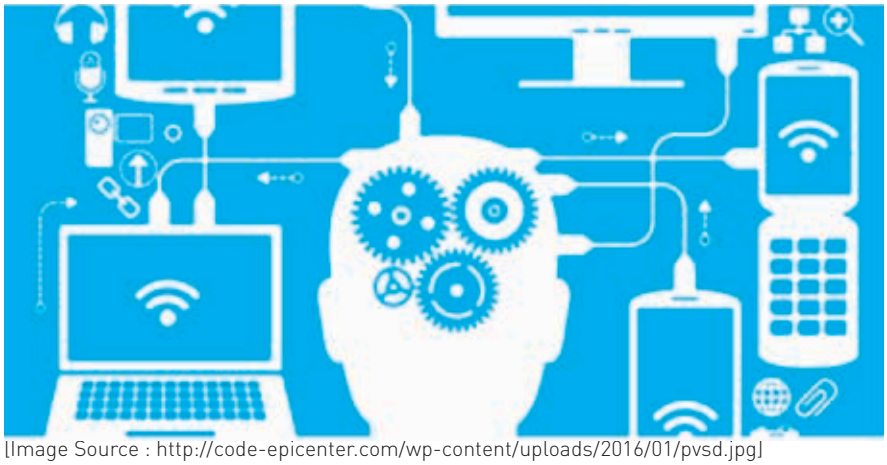
The little rectangular screens have been revolutionizing every part of our lives for more than a decade, and the changes are beginning to get interesting. As we have added more sensors we are able to collect and report much more detailed information. There are various instances available where data collected from these sensors is being used to provide medical assistance when needed. The microphone can pick up your heartbeat. The camera can look at the back of your throat. The accelerometers can track your exercise. All of these can be linked to a cloud full of doctors who can pass your case onto someone who specializes in what ails you. The navigation apps are morphing into route reservation and planning apps that do everything but steer the car. The exercise tracking apps are becoming tools that track all of the rhythms of our body from sleep to work and even more.

The gyroscope, accelerometer, magnetometer, and so forth are starting to get more friends in the neighborhood. Some new phones for instance are even coming equipped with sensors to measure pressure, temperature, and humidity. Yes, the next generation of smart devices will make the current set look basic. In the coming years we will see a lot of software engineering going into smartphones enabling them to do everything and giving the device ultimate powers. Smartphones of the future could use sensors and machine language to recognize other objects around them. These devices will not gather data but will also make apt decisions and choices based on the data. What if leaving your phone in the car for 4 hours triggers a reaction where the mobile itself make a phone call or e-mails to tell you where it is?. What if the mobile automatically start playing some mood elevation music when it senses you are stressed?

The point is hat the smartphones aren't going anywhere. But instead of a focusing on the world within the phone's screen, the smartphone of tomorrow will be tuned more than ever before to the world around you and the programmers of today will play a great role in shaping the future of smartphones.

4. Databases will perform increasingly sophisticated tasks

been almost two decades since when we have moved business systems to Database based storage system from the file base storage. Our decision has been good as databases have proved to be most efficient way of handling data ever known to us. The most common architecture around database is that you have an Application Server handling all business logic and you have set of Databases dumped with all kind of data ranging from texts, numbers, passwords etc. The data of interest is fetched from database processed and results are saved back to database. Simply extracting the information from the database and handing it to a separate "big data" package will become increasingly time consuming and require much more programming. Leaving the data in the database and letting its engine perform the analysis will be much faster because it will limit the overhead of communication, as well as decrease the amount of programming necessary to extract value from the data store. The current generation of database are becoming more and more intelligent and consistently proving that they are capable of doing much more then just storing data. This trend will continue and we will see more and more sophisticated operations to be pushed to the database layer. The future databases will be equally capable of performing aggregations, finding patters and even doing complex statistical functions. This capability will provide immense boost to over application performances as it will lower the overhead of moving data around.



[Image Source : http://code-epicenter.com/wp-content/uploads/2016/01/pvsd.jpg]

5. Crowd-Sourced Documentation will become the norm

Community of developers like Stack Overflow will become the new norm for reference while books and official documentation will become the thing of the past. Programmers are often able to find the answer they are seeking quickly using Stack Overflow then alternatives like reviewing documentation, seeking assistance from a co-worker, etc. The recent trends show that Stack Overflow questions are visited 2x-10x more often than official documentation. This is driven by the fact that Stack overflow documentation has more depth and breadth both in terms of numbers and advancement then compared to any official documentation available. The immense knowledge available in crowd sourced documentation will not only help us in better situate our self in tackling massive software challenges, but also enable long-tail developers to quickly create software for reuse and remix of other developer's efforts. Having access to immediate solution to the problems faced by developers will be helpful in diagnose and resolve the issue at hand, significantly boosting the engineering productivity.

6. Software will direct Policies

As more and more business and consumer workflows will be captured and executed in software, software will finally become the way to establish new policies and not politics. As in the case of Uber, no government had any policy or legislative acts pertaining to app based transportation earlier but with the introduction of Uber in their respective

states, governments came forward to formulate a policy around this method of transportation to regulate it. World of software is changing so fast that no government can keep with the pace, thus we will ultimately see more and more policies and acts revolving around consumers directed by app.

Conclusion

Information technology is about to grow dramatically with significant impact to both the global economy and everyday life. Computing power will increase rapidly as the costs of hardware and communications are dropping, making it increasingly more economical to implement systems in software rather than hardware. Mechanical devices in automobiles, airplanes, and power plants will be replaced by software components because software is more adaptable, can provide more functionality, and can be upgraded more easily to accommodate future needs. All industries including banking, insurance, telecommunications will use software to automate and personalize the services they offer to their customers. The opportunity to add net new value within the software ecosystem will decrease due to shift of analytics and cognitive capabilities from humans to machines. To stay relevant we don't only have to be good software developers but we also need to engage our self in leveraging the full software ecosystem encompassing rapid iteration, instant distribution and insightful monitoring. We are at an inflexion point and we have to embrace the future with an open mind while being ready for new set of challenges. ■

About the Author

Vijay Sharma [CSI-I1510163] Currently Vijay works a VP - Engineering for Bangalore based Digital Healthcare company SilverOakHealth. In this role Vijay leads the Technology Team at Silver Oak Health. Vijay and his team are responsible for design and delivery of next generation, world-class products, services and technologies related to mental healthcare.

An Essence of Soft Computing Techniques on Software Development Life Cycle

C Shoba Bindu
Professor, Department of CSE, JNTUA

E Sudheer Kumar
Assistant Professor, Department of CSE,
Sree Vidyanikethan Engineering College, Tirupati

K K Baseer
Asso. Prof., Dept. of IT,
Sree Vidyanikethan Engineering College, Tirupati

Software Engineering is a discipline that aims at producing high quality of software through systematic, well planned approach of software development. To accomplish high quality software it is indispensable to produce defect free product. Defect is the unexpected or undesired conduct that occurs in the product. Anything related to defect is a recurrent process not a particular state, whereas enlightening the qualities of software like reliability, usability, availability, maintainability, cost, time and energy is key role in software development. These qualities are more loyal on the requirements, analysis, architecture, design, development, testing and deployment.

In order to improve the software qualities on these cycles, shunning mistakes or making alarm for each activity may be suitable choice. Each and every element of software is diligently related to software quality. The software quality decreases when the software complexity increases. Therefore, understanding, measuring, managing, controlling and minimizing the software complexity is a big challenge in software engineering. It is important to focus on quality, monitoring the product and system performance. On the other hand, usability is important to safeguard the software quality and to increase the speed and accuracy of the range of tasks carried out by the users of a system because in software industries, the performance of the software is mostly improved through usability. In order to accomplish all of these needs in software development environment, some of the Soft Computing Techniques (SCT's) will helps in a better way. These are Poka-Yoke, Fuzzy Logic, Neural Networks, Particle Swarm Optimization, Genetic Algorithm, etc. This study gives an essence and impact of SCT's on Software Development Life Cycle (SDLC).

1. Introduction to SCT's

The following are the suitable SCT's used for the development of defect free software in improving the quality:

Poka-Yoke: In software development processes, Poka-Yoke concept is one of the methods to enrich usability and quality. Poka-Yoke (PY) which is a Japanese term, Poka means mistake and Yoke means prevent which is mistake preventing or mistake proofing technique. HP introduced PY into their Common Desktop Environment software and prevents hundreds of defects before it reaches to their customers. Shigeo outlines a method that uses sensor or other devices for hooking errors that may pass by operators or assemblers and it is said to be PY. A finest example of Poka-Yoke design from manufacturing industry is SIM card slot in cell phones. The seamless example of Poka-Yoke process in software application is Gmail email attachments feature.

Fuzzy logic: This theory was developed by Lofti A. Zadeh in the 60's and is based on the theory of fuzzy sets. It deals with the vagueness, uncertainty and imprecision of many real-world problems and also to simulate human reasoning and its ability of decision making based on not so precise information present in the early phase. Some of the promising key application areas of Fuzzy Logic (FL) which have been recognized are - Project Planning, Software Reliability Prediction, Software Usability, Software Quality Assessment, Performance Analysis of Software, Test case Allocation, Software Reusability, Software Fault Prediction and Size Estimation.

Neural Networks: It was developed to model the neural architecture and computation of the human brain. A Neural Network (NN) consists of simple neuron-like processing elements. Processing elements are interconnected by a network of weighted connections that encode network knowledge. NNs are highly parallel and exercise distributed control. NNs have been used as memories, pattern recall devices, pattern classifiers, and general function mapping engines. A classifier maps input vectors to output vectors in two phases. Neural Network (NN) can be used to build tools for software development and maintenance tasks, it can perform better in estimations & predictions, and it is used in integrating security at the design level of SDLC and also can be used across a variety of testing criteria.

Particle Swarm Optimization: Swarm Intelligence (SI) is an innovative distributed intelligent paradigm for solving optimization problems that originally took its inspiration from the biological examples by swarming, flocking and herding phenomena in vertebrates. PSO is a robust stochastic optimization technique based on the movement of intelligent swarms. PSO applies the concept of social interaction to problem solving. The basic concept of PSO lies in accelerating each particle towards its Pbest and Gbest locations with a random weighted acceleration at each time. PSO will be used to estimate the parameters for predicting software reliability, helps controlling the quality and predicting cost of software, it will help the project managers to efficiently plan the overall SDLC of the software product, can perform performance prediction, trades-off between architectural designs alternatives, and also it can be used to effectively generate alternatives in spanned design

space and facilitate the design decision during the development process.

Genetic algorithm: These are often used for optimization problems in which the evolution of a population is a search for a satisfactory solution given a set of constraints. Genetic algorithms are one of the best ways to solve a set of problems for which little information is given. Genetic algorithms use the principles of selection and evolution to produce solution for various complex problems. The Genetic Algorithms also out performs the exhaustive search and local search techniques, so they will work well in any search space. GA can be applied to automatic test data generation for path coverage which is an undecidable problem, this will perform autotuning strategy that can be used for optimizing performance or energy or a combination thereof, it can be used as architectural selection in response to the current situation of various environment and also it focus on the real world problem in the SDLC such as Organizations understaffed, Separation of duties.

2. Usage of SCT in SDLC

In order to eradicate and capture issues the PY process implementation mainly depends on tester's ability. Using Poka-Yoke Integration in Software Engineering (PYISE) helps stakeholders of software development to reckon success of work in progress [1]. More effort was spent towards requirements, project conceptualization and coding phases to triumph better achievements based on Phase scores and Phase wise achievements. Obligating validation PY in kit is always a good suggestion for developers so that the validation mistakes should be handled in your code. The HQLS-PY model [2] is to inject and integrate product monitoring at the right place to ensure the quality focus of the software based on the user experience and helps in alerting at the right time. The PYISE tool has additional two phases which are rollout and post implementation support. Each and every phase is having its own PY integration activities and related scores along with weights score need to dispense by the project manager and achievements scores need to assign by team lead or equivalent followed by phase scores will be evaluated by both project manager and team leader as well. In Fuzzy-based Poka-Yoke Model (FPYM) [3] by using four metrics called, UGAM, IoI, SM and SSM software performance attributes are defined and performance is analyzed to avoid and identify human mistakes in the process of life cycle.

Project managers who are using project management models early in a project provide numerical inputs which are not always feasible. Providing a more expert knowledge based approach to build the model using FL technique will help to represent the imprecision in inputs and outputs. FL could also be used in size estimation which can be done in advance to make better plans, and succour in tracking progress. In the model [4], the defects are predicted using metrics Requirement Defect Density (RDD), Requirement Specification Change Request (RCR) and Requirement Inspection and Walkthrough (RIW) at requirement phase followed by Cyclomatic Complexity (CC) and Design Review Effectiveness (DRE) to predict the defect at the end of design phase and Programmer Capability (PC), Process Maturity (PM) metrics at the end of coding phase will get the total number of defect predicted for the software before testing phase. Using fuzzy-based methodology the model [5] helps to assess usability software risk by taking into account the risk factors, risk likelihood and risk severity. Each of risk factors is evaluated using Fuzzy-AHP (Analytic Hierarchy Process) to probe multi-criteria decision making from hierarchy criteria.

NNs are the most common software estimation model-building technique used as an alternative to mean least squares regression [6]. NN on Cost estimation with cluster analysis increases the training efficacy of the network. NN has also been used to determine the most suitable metrics for effort estimation. Deriving a metric using a NN has several advantages. The developer needs only to determine the endpoints (inputs and output) and can disregard the path taken. A three-layered feed-forward back-propagation neural network is used software design [7]. The back-propagation neural network is a well-known type of neural network commonly used in pattern recognition problems. The McCall hierarchical software quality measurement framework (HSQF) model expresses the intuition of developers in selecting particular factors and criteria leading to overall software quality measures. NN can be used across a variety of testing criteria, test case and coverage metrics, and fault severity levels. Another area of software Testing in which NN has found attention is its use as Test Oracle. Neural Networks has been one of the technologies used during software implementation and testing phase of SDLC for software defect detection in order to intensify software reliability and it has also been used in area of application security and network security in technologies such as authentication system, cryptography, virus detection system, misuse detection system and intrusion detection systems (IDS).

The below figure 2.1 will depicts you the broad way of understanding the integration of soft computing techniques in various phases of SDLC which will help to deliver defect free, quality, reliability and high usability product to customers.

The Particle Swarm Optimization to tune parameters in Software Effort Estimation which reduces the Mean Absolute Relative Error. The combination of particle swarm optimization (PSO) and bagging technique for improving the accuracy of software defect prediction. The integration of Quality Function Deployment (QFD) technique and PSO method to develop more precise software cost estimation model [8]. The use of PSO in the field of performance prediction will systematically support the creation and evaluation of new architecture candidates [9]. PSO is used to compare and find the minimum software test cases for testing the software. PSO out performs primarily for complex functions with big search spaces. The use of Particle Swarm Optimization (PSO) technique to estimate the parameters of software reliability models [10].

In [11] energy benchmarking shows that there is a strong correlation between performance and energy requirements.

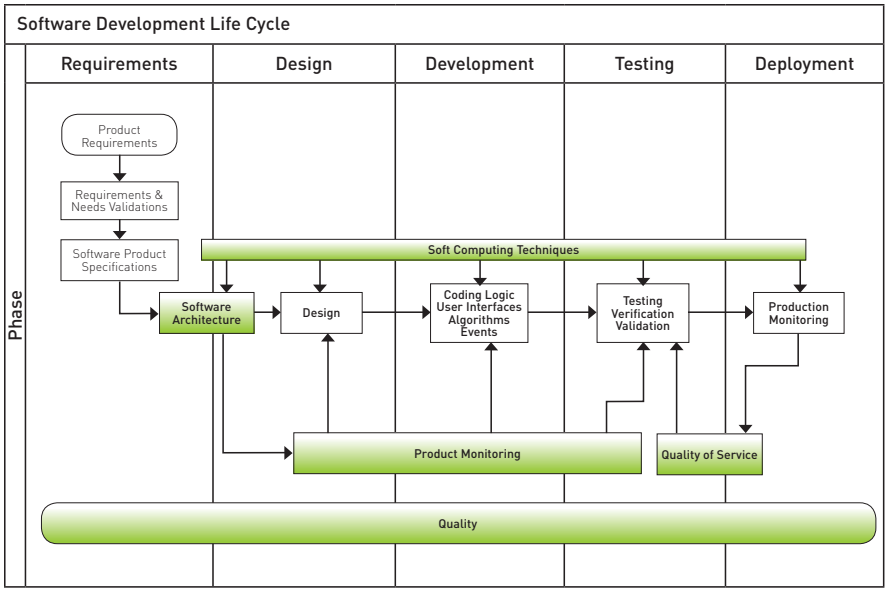


Fig. 2.1 : Integration of SCT in SDLC Phase

Genetic algorithms are also amenable to complex objective functions that take performance and energy both into account for optimization. Additionally, they can be easily modified to include architectural parameters such as the line size and associativity of a cache. GA based approach [12] is discussed for the architectural selection problem which enables a software system to autonomously search possible architectural instances (the search space). GA has been used as one good solution for automatic test data generation for path coverage is an undecidable problem, which has a great influence on path coverage of software testing. In MOGA [13] focused on the real world problem in the systems development life cycle such as Organizations understaffed, Separation of duties.

3. Impact of SCT on SDLC

Avoiding post release issues which in turn help us to entrap defects in early phases making it cheaper to fix. The human resource monitoring precedes to product monitoring which is highest in requirement phase followed closely by coding, conceptualization and design thereby achieving quality focus in software development. Validation PY in kit can help developers to revisit and update periodically on validation issues. PYISE tool is to upsurge the quality of the software by plummeting the number of defects and errors during the development of software and also can be applied to appraise the project in progress which helps in making process systematic.

The benefit of fuzzy logic for software engineering project management is the flexibility available in terms of the types of input and output variables which will not lead to over commitment. The concept of fuzzy logic in size estimation is gathering size data on previously developed programs and dividing the historical product size data into size ranges which will help to compare the planned product with these prior products. Different specified metrics used in FIS will help you to find out the defects before testing phase in software development process. Fuzzy-based methodology allows to calculate the magnitude of risk on software usability by using fuzzy-AHP. The below Fig. 3.1 showcase about usage of various techniques of soft computing on different levels of SDLC phases to achieve several parameters in the process of software development.

NN with cluster analysis for software development cost estimation will be a promising approach to provide more accurate results on the forecasting of software development costs. Unlike the traditional approach, where the developer is saddled with the burden of relating terms, a NN automatically creates relationships among metric terms. To identify the possible attacks from the design and also to evaluate scenarios from software designs NN is used. The introduction of fuzzy sets into

Parameter / Technique	Effort, Size & Cost Estimation	Architecture Selection	Defect Prediction	Developers on Quality Measure	Performance Prediction and Optimization	Success of work Progress	Security Concern & Reliability	Test Data Generation	Quality of Services
Poka-Yoke		☑		☑		☑		☑	☑
Fuzzy logic	☑		☑			☑			☑
Neural Network	☑			☑			☑		
Particle Swarm Optimization	☑		☑		☑		☑	☑	
Genetic Algorithm		☑			☑			☑	

Fig. 3.1 : Achievable Parameters with SCT on SDLC

the McCall hierarchical software quality measurement framework (hsqf) makes it possible to capture information granules, to give meaningful "shape" to otherwise daunting collection of factors, criteria and metrics their relationship and relative importance. The neural net formalizes and objectively evaluates some of the testing folklore and rules-of-thumb that is system specific and often requires many years of testing experience.

This unique combination will help the project managers to efficiently plan the overall software development life cycle of the software product. A larger search space is explored and architects could be able to select better alternative architecture model in the field of performance prediction. Software test is the main approach to find errors and defects assuring the quality of software. The estimated model parameters were used to predict the faults in a software system during the testing process. The PSO technique provided sets of parameters with each software reliability model evaluated.

Usage of GA for the target routines, improving performance result in simultaneous reduction in energy requirements and also will effectively explore a hardware software co-design. Using of GA on architectural selection problem to find an optimal instance to the current situation and requirements within a short time. With two phase approach is used in order to generate test data which can cover the paths having the lowest coverable probability. Multi objective genetic algorithm (MOGA) is suggested for recruitment of the staff, allocation of Jobs based on the skills they possess. MOGA has been used to minimize the cost associated, minimize the time involved and to maximize the efficiency by proper usage. The automatic test case generation will create random population of solutions, evaluate the solutions, select parents, create child solutions, evaluate children, swap some of the existing solutions with some of the better solutions, repeat until the termination criterion is met and return best solution.

Conclusion:

This study is to produce defect free product and to improve the software qualities. Focusing on quality and monitoring the product to increase the system performance is very important. Also on the other hand, at the time of tasks performing by users of a system it is important to safeguard the software quality and to enrich the speed and accuracy of the range of tasks. All of these needs in software development environment can be accomplished with Soft Computing Techniques (SCT's). In future using of new metrics such as software medium, software system measurement, inclusion of architectural parameters to effectively explore a hardware software co-design, reduce information loss and provide greater design alternatives at later stages, security capabilities into design, cost benefit analysis of models will drives in greater extent for defect free and quality product.

References:

[1] Baseer K.K, Rama Mohan Reddy.A, Shoba Bindu.C, "Quantifying Poka-Yoke in HQLS: A new approach for High Quality in Large Scale Software Development", 49th Convention of CSI on Emerging ICT for Bridging Future (CSI-2014), December 12-14, 2014, CSI, Hyderabad, proceedings in Springer-Advances of Intelligent Systems and Computing Vol.337, ISBN No: 978-3-319, pp.293-301, 2014.

[2] K. K. Baseer, A. Rama Mohan Reddy and C. Shoba Bindu, "Monitoring tool for mistake proofing quality software", IP India Patent: 2007/CHE/2015 A, The Patent Office Journal, Issue No. 19/2015, May 8, pp. 33133,2015.

[3] K. K. Baseer, A. Rama Mohan Reddy, C. Shoba Bindu, "FPYM: Development and Application of a Fuzzy based Poka-Yoke Model for the Improvement of Software Performance", InderScience-Special Issue on Soft Computational Approaches for Risk Exploration in Global Software Development. (Accepted-Under Production stage).

[4] H. B. Yadav and D. K. Yadav, "A Fuzzy Logic based Approach for Phase-wise Software Defects Prediction using Software Metrics", Information and Software Technology, vol. 63, pp. 44-57, 2015.

[5] AfriyantiDwi Kartika and Kridanto Surendro, "A fuzzy-based methodology to assess software usability risk", in Proceedings of the IEEE International Conference: Information and Communication Technology (ICoICT), 2016.

[6] Yogesh Singh, Pradeep Kumar Bhatia, Arvinder Kaur, OmprakashSangwan, "Application of Neural Networks in Software Engineering: A Review", in Proceedings of Third International Conference, ICISTM 2009, Ghaziabad, India, March 12-13, 2009.

[7] A. Adebiyi, JohnnesArreymbi and Chris Imafidon, "Applicability of Neural Network to Software Security", in Proceedings of the IEEE International Conference on Computer Modelling and Simulation (UKSim), 2012.

[8] Divya Kashyap, A. K. Misra, "Software cost estimation using Particle Swarm Optimization in the light of Quality Function Deployment technique", International Conference on Computer Communication and Informatics (ICCCI), 2013.

[9] Adil. A. A. Saed, Wan M.N. Wan Kadir, "Applying Particle Swarm Optimization to Software Performance Prediction An Introduction to the Approach" Malaysian Conference in Software Engineering (MySEC), 2011.

[10] Alaa Sheta, "Reliability Growth Modeling for Software Fault Detection Using Particle Swarm Optimization" IEEE Congress on Evolutionary Computation, CEC 2006.

[11] Tania Banerjee and Sanjay Ranka, "A Genetic Algorithm based Autotuning Approach for Performance and Energy Optimization", Sixth International Conference on Green Computing and Sustainable Computing (IGSC), 2015.

[12] Dongsun Kim and Sooyong Park, "Dynamic Architectural Selection: A Genetic Algorithm Based Approach", International Symposium on Search Based Software Engineering, 2009.

[13] D. Sundar, B.Umadevi and Dr. K. Alagarsamy, "An Optimized approach for the Improvement of CMMI in Human Resource Management Using Multi Objective Genetic Algorithms", Second International conference on Computing, Communication and Networking Technologies, 2010. ■

About the Authors

Dr. C Shoba Bindu [CSI-01174921], received her Ph.D degree in CSE from JNTUA, Anantapuramu, Andhra Pradesh. She is currently extending her services as Professor in department of CSE, JNTUA. Her research interests are in the areas of Mobile and Adhoc Networks, Network security, Data Mining and Cloud Computing. She can be reached at shobabindhu@gmail.com.

E. Sudheer Kumar received B.Tech and M.Tech in Information Technology from JNTUA, Ananthapuramu, India. Currently, he is Assistant Professor in the department of CSE, Sree Vidyanikethan Engineering College, Tirupati, India. His research interests include data science, software engineering, software architecture, and other latest trends in technology. He can be reached at sudheerkumar.e@gmail.com

Dr. K K Baseer [CSI-I1182542] received his PhD from JNTUA, Ananthapuramu, India. Currently, he is working as Associate Professor in the department of Information Technology, Sree Vidyanikethan Engineering College, Tirupati, India. His research interests include Software Engineering, Software Architecture, Data Science, Information Retrieval System and other latest trends in technology. He can be reached at kamalapuramkhajabaseer@gmail.com

Pseudo Code Attack in Software Engineering

S Hemalatha
Professor/CSE , Panimalar Institute of Technology

P C Senthil Mahesh
Professor/CSE , Jeppiaar Institute of Technology

In recent technological development of hardware and software are relies on the software development. Developing of software is depends on programming languages inventions. Programming languages invention is adaptable to software metric calculation like line of code, code reduction, reusability etc. During this metric calculation stages the different attacks can be performed to reduce the metric values and delayed the proposal of software packages. There are many attacks are identified and prevented in networks, this pseudo code attack is the novel method to perform increasing the metric calculation in software coding. This article proposed the technique to introduce the pseudo code attack and identifying the pseudo code attack present in the source code.

1. Introduction

Most software development associations have some goal or desire to produce secure software that assures their products robustness and availability. During the software development, there is either flaws or faults are introduced from the design of the software or from the implementation. Failures and particularly vulnerabilities are maximizing the cost for the developers and need them to spend more time on software maintenance instead of new features. Most of the software developers depend on testing to minimize their maintenance cost and to generate software with high availability and robustness. Unfortunately, testing mostly focuses on validating the proposed functionality and not on identifying the vulnerabilities.

There are several DDOS attacks are in the network field. These kinds of attacks are trying to reduce the system performance, in order to achieve degradation of system aspects with normal aspects.

The Pseudo code attack is the new kind of DDoS attack. This pseudo code attack is possible in software development process. This article discusses about the pseudo code attack in DDOS. DDOS prevents the intended usage of the software development. The common DOS attacks are network based which attempt to exhaust the system resources. The source code which does not constantly releases a system resource might be explored in the similar manner, resulting in a consumption of resources.

The software development process mainly focuses on developing of the new and innovative software's to the society. When a new version is developed by the software team, a new version is put in to the functional point analysis. This functional point analysis calculates the number of read and writes cost involved in the new version. This will be compared with the older version to check the quality of the new one. If the calculated cost is less or equal to the old version, then the developed team will release the software.

From that, the Pseudo code attack is new type of DDoS attack. This attack is automatically installed in the system memory. Whenever the functional point analysis calls, the newly generated segment code will be added and moved to the original code. This results cause in increasing the number of read and write operations on memory. This process will lead to show the FFP in more cost. So the newly developed segment is put it to wait for release and evolutes the development state again. This causes the delay in releasing of software. This pseudo code is not like a virus or any malware program. Hence it cannot be identified by any antivirus detection software.

2 Effects of Pseudo Code Attack

The pseudo code attacks are installed in the system memory. It affects the system performance in diverse ways. With the arrival of pseudo code attack, the system operations are deviated from their normal working procedure. The effects of pseudo code attack are as follows:

The pseudo code attack reduces the server performance as it is installed in the system memory. It leads to increased cost of Full Functional Points. The pseudo code takes additional storage in the system memory. Hence, the disk storage space is also increased. Consequently, this attack reduces the disk speed of the system and causes delay in reading.

3. Research Issues and Challenges

Developing a software security requirements are complicated and have some issues which maximizes the difficulty of developing such requirements. Some of the current issues and challenges for software developments are discussed as follows:

- **Security is continually changing:**

The procedure for generating requirements should be precise and clear. Hence, the future security requirements should research provides insight into a new appearance of software threat.

- **The security requirements should be declared in a positive tone:**

Generally, Security requirements are declared in a negative tone

that formulates the validation and verification more difficult . For instance, a negative requirement might be "The program shall not permit remote exploits", which is complex to validate. The process of validating and verifying that requirement might require testing the program could not do, not what the program should do.

- **Security is continually changing:**

The procedure for generating requirements should be precise and clear. Hence, the future security requirements should research provides insight into a new appearance of software threat.

- **The security requirements should be declared in a positive tone:**

Generally, Security requirements are declared in a negative tone that formulates the validation and verification more difficult. For instance, a negative requirement might be "The program shall not permit remote exploits", which is complex to validate. The process of validating and verifying that requirement might require testing the program could not do, not what the program should do.

The security requirements for developing the software should be language and platform independent:

Any technique intended to provide common requirements becomes less practical when associated with the particular language or platform. Providentially, the software requirements can be written with a common tone to cover up all the possible situations that a development might face. Some of the requirements will be less observable in certain languages, for example, any language that abstracts system memory management will not have necessities for unsuccessful memory allocation attempts, etc.

- **The security requirements should be verifiable and testable for the development process to work:**

The standard idea behind the requirements-driven software development contains generating the requirements that can be established at each stage of the development process. It can be validated or tested during the testing phase of software development .

Verification is done at each stage after the requirements phase, and assures that the criteria of each requirement are correctly being attained to that point in the development. The requirement testing is done at the end of the development stage during the testing stage, and assures the inclusion of the requirement. A security requirement should be both verifiable and testable for it to be probable to track the progress of the requirement throughout the development stages, and test to assure the requirement was incorporated into the software development.

A development may only requisite some software security requirements, but not all:

The requirements will cover up everything from system memory to cryptography. But, some of the applications may only require a subset of those security requirements. A procedure should exist so that the security requirements desired for a development can be selected based on the non-security associated requirements.

Due to these kind of attacks, there are lot of issues are occurred during the software development. In order to formulate secure and scalable software, an appropriate model required to be developed. To overcome these issues, the research focuses on developing a model based on the cosmic FFP and energy points analysis. It minimizes the unwanted read and writes operations occurred on the system. Based on these approaches, a vulnerable DDoS attack is prevented while developing the software.

4. Micro Motivation

The software sizing and energy points applied in software development process. Also, the research issues and challenges in this software sizing are also discussed.

4.1 Software Sizing

Software size estimation is important in providing a credible software cost estimate. Hence, selecting the suitable method by which to estimate size is significant. In most of the cases, the estimation risk depends more on precise size estimates than on some other cost-based parameter.

Therefore, it is significant that software sizing can be done as accurately and consistently as possible, given the uncertainties intrinsic in size estimation. Though, software sizing is complex due to the number of reasons.

1. It is executed in a number of different circumstances, some with a huge deal of knowledge regarding the system and some with about no knowledge at all.

There are many alternatives for the structure and language utilized to articulate the design and requirements.

Usually, software projects are a combination of reused, new, and modified mechanisms. A sizing method should be able to include all three phases, even when the modification and reuse take place in the design and requirements as a substitute of just in the program code.

Software sizing measures, Software sizing measures are utilized to normalize the other measures so that suitable comparisons can be prepared within or across the systems. Productivity statistics cannot be calculated, without a sizing measure. A sizing measure is basic to any software measurement program. During the estimation of cost and schedule, the common use of sizing measure is probable; there are many other potentially precious applications, together with earned value, risk identification, change management and progress measurement. Nowadays, there are two software sizing measures are greatly widely used: They are,

- Source Lines Of Code (SLOC) and
- Function Points (FP). Even though, each measure has diverse things and has very diverse characteristics.

SLOC – It is a measure of the size of the software system that is formulated. It is extremely dependent on the software technology used to formulate the system design, structure and how the programs are coded. Also, there are numerous well-documented issues and problems with SLOC. In fact, Capers Jones declared that anyone using SLOC is "consigning professional malpractice." In spite of these issues, SLOC is still frequently used by very professional and reputable

organizations.

In contradiction to SLOC, **Function Points (FP)** is a measure of distributed functionality which is comparatively independent of the software technology used to expand the system. At the same time as FP faces and solves many of the issues inherent in SLOC, and it has introduced a loyal following, it has its own set of problems. Since, LOC and FP have been the only broadly accepted ways to size a software system; a developer's dimension choices have been very inadequate. The SLOC and FP measures are exclusively powerful and useful. Each measure has its own merits and drawbacks.

4.2 Energy Points

Most of the computer software and hardware developers are focuses on solving problems with minimum storage space and maximum speed. But energy use for computing is an increasing concern. Some of the steps to minimize the energy points in software development are:

1. Run multiple application on shared servers
2. Log less
3. Delete historic data
4. Compile interpreted languages
5. Reduce data translation among components,

Here, Functional Points obtained are mapped to Energy consume in order to obtain the Energy Points.

The formula used to derive for Energy Point Mapping is as follows:

ER = 13.3 μ w/k

EW = 6.67 μ w/k

EP = (R * ER + W * EW)

Where R is each read operation in Function Point and ER is the energy required for read operation.

Where W is each write operation in Function Point EW is the energy required for write operation.

5. Research Issues and Challenges

Software sizing and software energy point's process face many challenges to attain an accurate and proper estimate for many reasons. Since, software is insubstantial rather than the estimation procedure in nature is not simple particularly with the insubstantial products. One of the major difficulties in the software sizing is the availability of the data that is required to verify the usefulness of any suggested models, measures and functional sizing approaches. Most of the software sizing techniques was based on the small amount of data. Some of the techniques for instance compute the size and cost based on 30 UML files. Hence, the resulted approach in this situation does not have a high reliability and it cannot be generalized. There is an issue in the estimation process in preserving the suitable and proper dataset to check any kind of sizing techniques, measures or any cost models. It results a real problem in enhancing the software estimation. process. The nature of the development process where all the software requirements are known, moreover to the requirement creep issue also the correlation among the cost factors and how each factor may affect the result of the software sizing. Another challenge is that there is no precise rules and standard for the whole process of software development. It is still in adhoc stage that does not restricted to the certain standard. Currently, the functional size measurement is performed manually and hence it is time consuming. In order to speed up the process and reduce the probability of human errors, numerous attempts are done to automate the functional size measurement techniques. A set of COSMIC related tools on the software market and research community. But still, it is necessary to identify the tool vendors to fill the research gap analysis. Hence, the proposed research focused on developing a system based on the cosmic FFP energy points for secure software development.

6. Proposed Method for Energy Point Estimation

In the software development projects, estimated size and time is the important measures of quality assurance .Quality of software by means of mean time between failure and a number of bugs per line of code. The measures are classified into direct and indirect. Direct measures are Cost, line of codes, speed memory size and number of errors. Indirect measures are Function (size), quality, complexity, efficiency, reliability. In the software development process these metric is also getting affected by means of denial of service attack through micro motivation. This kind of attack in software project level can be identified by analyzing the energy point. The energy point is calculated using the functional point .Functional point plays a vital role in estimating the quality of the software. Developed software is segmented into modules, each module are taken for energy calculation.

This research work is mainly focuses on between energy point to functional point. Developed modules are the main sources in functional point. Functional point measurement is used to identify the entry, exit , read and write operation of the modules. This output is in numerical value, which are passed to the energy point calculation. Energy point calculation is produces the read and write energy in developed modules. This read and write energy. is used to identify the pseudo code DDoS attack present in the modules or not. MCRose cosmic FFP measurement is used for analysis the energy point calculation. The modules are passed to MCRose in real time , the entry, exit and number of read /write of modules is taken for energy calculation.

Joule Meter : Joule meter estimates the energy usage of a VM, computer, or software by measuring the hardware resources (CPU, disk, memory, screen, etc.) being used and converting the resource usage to actual power usage based on automatically learned realistic power models. Joule meter can be used for gaining visibility into energy use and for making several power management and provisioning decisions in data centers, client computing, and software design.

The technology is especially helpful for IT leaders managing power management settings, PC users who wish to get fine grained visibility into their computing energy use, and enthusiast developers who wish to leverage power measurer for optimizing their software and hosted service design for power usage.

The Joule meter modelling tool can be used to optimize power use in

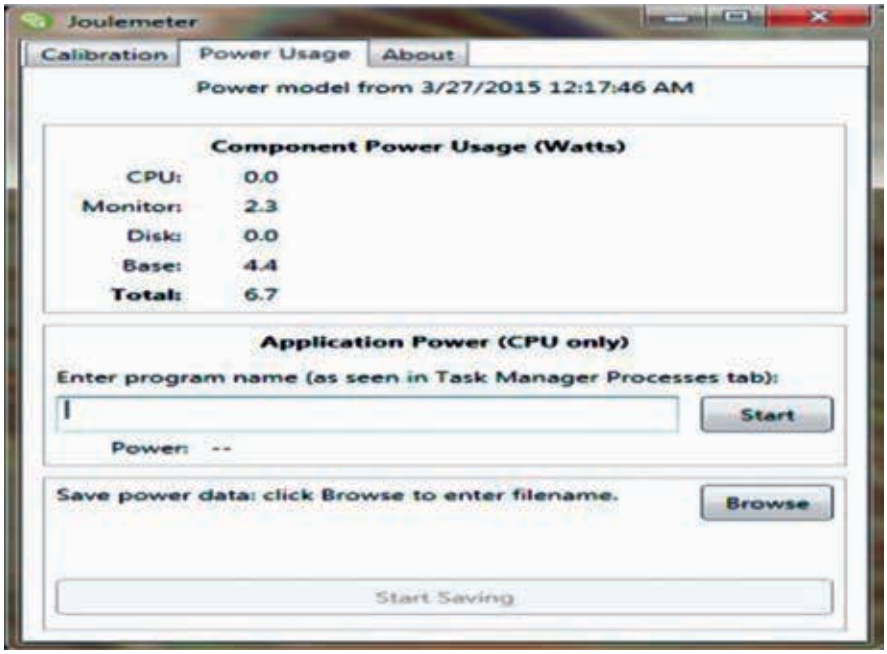


Fig. 6.1 : Joule Meter

multiple scenarios. The measurement of VM power allows developing power budgeting techniques for virtualized data centers.

Managing and tracking PC sleep, combined with remote wakeup, allows optimizing desktop power consumption in enterprise buildings. Separating the impact of hardware components on battery life allows .Users to trade-off power management settings for improving battery life and enables developers to make appropriate design trade-offs for their software applications. Figure 6.2 shows the snapshot of the Joule meter.

The read and write energy calculation from the joule meter can be used to identify the pseudo code attacks present in the system or not.

7. Conclusion

Among the various threats to networks, Distributed Denial of service (DDoS) attacks have evolved to be a major threat to the availability, accessibility and operations of the many Internet based Services. Recent researches claim that thousands of such attacks are launched every week, causing severe damage to both commercial and governmental sites. In this thesis a new technique is described to detect and prevent DDoS.Pseudo code attack is also p[lay a vital role in Software engineering to overcome the DDoS Attacks, which can be detected using joule meter.

References

[1] Yau, DK, Lui, J, Liang, F & Yam, Y 2005, 'Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles', IEEE/ACM Transactions on Networking (TON), vol. 13, no. 1, pp. 29-42.

[2] Ye, B Qing, S, Okamoto, T & Zhou, J 2001, 'Defeating Denial-of- Service Attacks on the Internet', in Information and Communications Security, Springer Berlin Heidelberg, vol. 2229, pp. 304-315.

[3] Yu, S, Zhou, W, Doss, R & Jia, W 2011, 'Traceback of DDoS attacks using entropy variations', Parallel and Distributed Systems, IEEE Transactions on, vol. 22, no. 3, pp. 412-425.

[4] Yu, Y, Li, K, Zhou, W & Li, P 2012, 'Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures', Journal of Network and Computer Applications, vol. 35, no. 3, pp. 867-880.

[5] Zahariadis, T, Leligou, H, Karkazis, P, Trakadas, P, Papaefstathiou, I & Vangelatos, C 2010, 'Design and implementation of a trust-aware routing protocol for large wsns', International Journal of Network Security & Its Applications (IJNSA), vol. 2, no. 3, pp. 52-68.

[6] Zayaraz, G, Thambidurai, DP, Srinivasan, M & Rodrigues, DP 2005, 'Software quality assurance through COSMIC FFP', ACM SIGSOFT Software Engineering Notes, vol. 30, no. 5, pp. 1-5.

■

About the Authors

Dr. S Hemalatha Professor/CSE , Panimalar Institute of Technology

Dr. P C Senthil Mahesh Professor/CSE , Jeppiaar Institute of Technology

The Role of Software Engineering

Hardeep Singh
Dept of Computer Science, Guru Nanak Dev University, Amritsar
hardeep.dcse@gndu.ac.in

Parminder Kaur
Dept of Computer Science, Guru Nanak Dev University, Amritsar
parminder.dcse@gndu.ac.in

1. Introduction

Development of software was a simple activity in earlier days. But as technology improves, software becomes more complex and size of software projects becomes larger. To deal with complexities, Software Engineering (SE) provides various methods and enables the development of a reliable product with desired quality. The notion of SE was first introduced in 1968, defined as 'the application of a systematic, disciplined and quantifiable approach to the development, operation and maintenance of software [IEEE][1,2]. The main aim of the SE is to follow a systematic approach to build high quality software within specified time and budget. The absence of SE leads to:

- Complex Code.
- Inconsistent User Interface.
- No Systematic Testing Strategies.
- Lack of methodical, quantifiable methods.
- Requirement gathering remains incomplete.
- Lack of systematic methods further leads to weak or too complex architecture.

2. SE Layers

SE works in different layers i.e. Process Layer, Method Layer and Tools Layer as shown in Fig. 1.

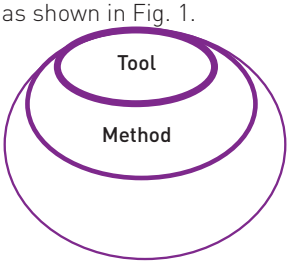


Fig. 1 : Layers of Software Engineering

The *processlayer* defines an outline for a set of key process areas that must be acclaimed for effective delivery of software on time. The *method layer* covers requirements analysis, design, coding, testing, and maintenance phase of the software development. The *tools layer* provides computerized or semi-computerized support for the process and the method layer. Tools can be integrated in such a way that one tool can use information created by another tool, known as Computer-Aided Software Engineering (CASE). CASE tools helps in designing and documenting traditional-structure programming techniques. For example, two prominent technologies using CASE tools are PC-based workstations and application generators that provide graphics-based interfaces to automate the development process.

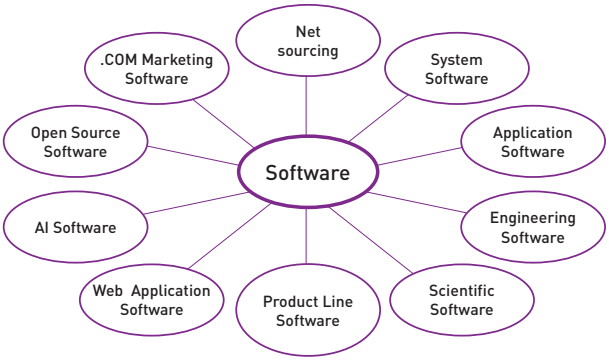


Fig. 2 : Types of Software

3. Types of Software

Now-a-days, variety of software is available in the market as shown in figure 2. System Software, Application Software, Engineering/Scientific Software, Embedded Software, Product-Line Software, Artificial Intelligence Software, Web Applications, Ubiquitous Computing, Netsourcing, Open-Source Software are few types of the software.

Software development takes place in various phases such as:

- Requirement's Engineering
- Software Design
- Software Coding
- Software Testing
- Software Implementation
- Software Maintenance

The above said various SD phases can be accomplished by selecting one of the available software development process models as shown in Figure 3. These process models can be selected according the type of application which is going to develop. The applications, which are small in size and with well-defined requirements, are well handled by Waterfall Model [3-4]. When developer is not sure about the requirements, then one of the Prototyping Models either Evolutionary or Incremental Prototyping is used [5]. Spiral model is used when risk identification as well as risk evaluation is required in each phase of the development [4][6-7]. Large as well as complex programs are also handled by Spiral Process Model. Today, development is done with Iterative and Incremental Model i.e. every new release of software development takes care about change in demand. Agile Development [8], RAD Process Model [9], Extreme Programming [10], Component-based development [11] are some software development methodologies through which one can develop a high quality product in a desired time schedule.

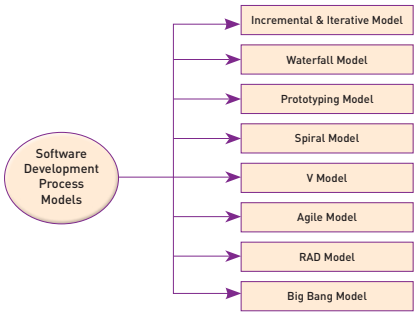


Figure 3: Different Types of Software Development Process Models

SE also deals with two basic objectives- High Quality and Low Cost. In order to control quality and cost, the processes leading to the development of software need also be controlled. However, as in other engineering disciplines, anything which cannot be measured, cannot be controlled. Therefore, a very significant aspect of the Software engineering is the software measurements. It is dealt with in two dimensions i.e. Software Metrics and the quantitative measurements. Most often, these two terms are interchangeably (Fig. 4).

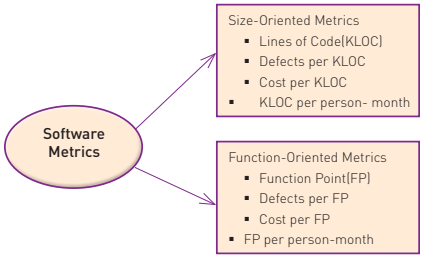


Fig. 4 : Software Metrics

In Literature, there are various software metrics suites discussed by different Software Engineers e.g., "C.K. Metric Suite" by Chidamber and Kemerer [12], "The MOOD Metrics Set" by Abreu [13], Lorenz and Kidd Object-Oriented Metrics suite [14], McCabe Cyclomatic Complexity [15].

4. Software Engineering: The Future

As the usage and the demand of the software increase, the size and the complexity of the software systems are likely to increase as well, most probably in the exponential manner. This possesses a tremendous challenge to the practitioners and researchers of software engineering. The major difficulty encountered is that the software processes are not scalable. As the size and complexity increase, the software processes need to be expanded in a nonlinear fashion. Moreover, not many professional are either ready or trained to handle such processes.

Another situation which is likely to emerge is the integration of systems with one another leading to a situation which is termed as System of Systems (SoS) or Ultra Large Systems (ULS). This is multi-dimensional challenge as we need to have new processes, methods and tools to deal with these types of systems.

The changing computing environment presents a challenge for SE Computing. As steadily the computing is being shifted to cloud platform, the demand for scalable, reliable and secure applications is on the rise. The Service-oriented Architecture (SOA) is going to play an important role in determination of the nature of the application on this platform in the times to come.

The integration of new technologies like **S**ocial Networks, **C**loud Computing, Big Data **A**nalytics and **M**obility is giving rise to new IT infrastructure (SCAM). Since this infrastructure is going to be employed by most of the organisations, the applications need to be built and maintained and this SCAM Stack, for which, again, we need a change in the orientation with which we develop and maintain our application.

References

[1] "IEEE Standard Glossary of Software Engineering Terminology," IEEE std 610.12-1990, 1990

[2] Ian Sommerville, "Software Engineering (9th Edition)", Pearson Education, 2010

[3] Royce, Winston (1970), "Managing the Development of Large Software Systems" (PDF), Proceedings of IEEE WESCON, 26 (August): 1–9

[4] Jalote, Pankaj, "An Integrated Approach to Software Engineering, Third Edition, Springer, Narosa Publishing House, 2016

[5] Smith MF, "Software Prototyping: Adoption, Practice and Management", McGraw-Hill, London, 1991

[6] Boehm, B, "Spiral Development: Experience, Principles, and Refinements", Special Report CMU/SEI-2000-SR-008, July 2000

[7] Boehm B, "A Spiral Model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes, ACM, 11(4):14-24, August 1986

[8] Larman, Craig (2004). Agile and Iterative Development: A Manager's Guide. Addison-Wesley. p. 27. ISBN 978-0-13-111155-4

[9] Martin, James (1991). Rapid Application Development. Macmillan. pp. 81–90. ISBN 0-02-376775-8.

[10] "Extreme Programming" USFCA-edu-601-lecture

[11] Crnkovic, Ivica, et al. "A classification framework for component models." Software Engineering Research and Practice in Sweden (2007)

[12] Chidamber, S. R. and Kemerer, C. F., "A Metrics Suite for Object Oriented Design," IEEE Transactions on Software Engineering, vol. 20, 1994.

[13] Abreu, F. B. e., "The MOOD Metrics Set," presented at ECOOP '95 Workshop on Metrics, 1995.

[14] Lorenz, M. and J. Kidd, Object-Oriented Design Metrics, Prentice- Hall, 1994.

[15] McCabe, T., "Cyclomatic Complexity and the Year 2000," IEEE Software, May 1996.

■

About the Authors

Hardeep Singh is working as a Professor in the department of Computer Science at Guru Nanak Dev University Amritsar, India. He has around 100 research papers in the International/ National Journals as well as Conferences. His research interests include Software Engineering, Open-Source Systems, Web Engineering, Information Systems, Service-oriented Architecture and Cloud Computing.

Parminder Kaur is working as an Asst. Professor in the Dept. of CS at Guru Nanak DevUniversity Amritsar, India. She has completed her Ph.D. from Guru Nanak Dev University Amritsar in the year 2011. She has around 35 research papers in the International/ National Journals as well as Conferences. Her research interests include Software Engineering, Web Engineering, Semantic Web, Open-Source Software and Software Security.

Formal Methods in Software Engineering

A Sowmya Mitra
Scientist, DRDO, Hyderabad

Software based applications are becoming ubiquitous in several mission / safety critical systems and main challenge is to provide formalism, techniques and tools to optimize rigor despite system complexity. In safety critical / mission critical systems failure might result in serious impact to an organization and even can cause catastrophes. Rigorous verification and validation is indispensable. Conventional V&V methods include code walkthrough/code inspection, static and dynamic testing. However, these traditional methodologies when carried out rigorously can "Detect the presence of Bugs" but never "Prove the absence of Bugs". As a practical alternative, systems / subsystems need to be verified with mathematical proofs termed as "Formal Methods". Formal methods provide a foundation for special environments leading to models that are complete, consistent and unambiguous. Formal methods can prove "Always or Never" and major approaches include Theorem Proving and Model Checking.

1. Introduction

Formal Methods : Formal methods are the use of mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems. By **mathematically rigorous** we mean specifications are well formed statements in mathematical logic and formal verification are rigorous deductions in that logic. The various phases in which Formal Methods can be applied to the chosen software are as follows:

- ✓ Defining Requirements
- ✓ Modelling: Mathematical representation of a man-made system most suitable for the application
- ✓ Formal Specification: Characterization of an existing system expressed in formal specification language
- ✓ Formal Analysis / Formal Verification: Model Checking & State Exploration / Theorem Proving & Proof Checking
- ✓ Documentation

The principal distinction between the two approaches stems from the choice of formalism used in reasoning process.

Theorem Proving: Theorem proving is one of the key approaches to formal verification. Theorem proving reasons about program P correctness in terms of pre and post conditions based on Hoare Logic, i.e. **{ϕPRE} P {ϕPOST}**. This formula can be read as "if property ϕ PRE holds before program P starts, ϕ POST holds after the execution of P. In Hoare's calculus, axioms and rules of inference are used to derive ϕ POST based on ϕ PRE and P.

These techniques are most powerful where properties are written in first order predicate logic. Correctness of properties is established through set of stored theorems. There are many theorem provers in active use today. The popular ones include ACL2, Coq, HOL, Isabelle, PVS, FRAMA-C etc. A common aspect of all these theorem provers is that they support complex logics and are highly expressive. However, the expressive power lacks automation and there is no fully automatic procedure to deduce a given logic. In such scenarios, user intervention is required in terms of assistance to theorem prover in its search for a proof. The guidance takes the form of setting intermediate lemmas, axioms, logical predicates, selecting heuristics and strategies at various steps of the proof which makes theorem proving technique a highly human intensive job.

Model Checking: Model checking is an algorithmic method for determining if a model M of a system satisfies a correctness specification P. A model of a program consists of *states* and *transitions*. A specification or *property* is a logical formula capturing the design intent. A model checking algorithm has two main inputs – a formal property in a property specification language, and a finite state machine representing the implementation. The role of the algorithm is to search all possible paths of the state machine for a path which refutes one or more properties. If one exists, then the path trace is reported as the counter-example. Otherwise the model checker asserts that the property holds on the implementation. This kind of verification involves establishing that the model semantically entails the specification i.e. **M⊨ P**.

Some of the software Model Checking tools are SPIN, BLAST, SATABS, CBMC etc. Figure 1 illustrates an overview of Formal Methods.

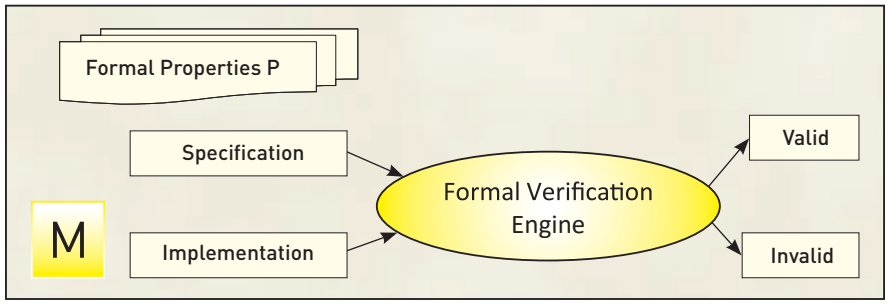


Fig. 1 : Formal Methods – Overview

2. Methodology:

Using formal methods to prove the functional correctness of any logic as per the specifications, a step by step procedure is mandatory. These steps can be generalised irrespective of the techniques used to reason about the correctness. Methodology at work assumes that the model of the implementation is auto generated by the tool under consideration and hence modelling the program under test is not considered here. The three basic steps are as follows:

- Step 1: *Requirement Analysis*: Detailed requirement analysis is the first and the foremost important task where sound domain knowledge about the system is a pre-requisite. Requirement analysis can be represented in any semi-formal notation. The outcome of this step results in identifying the mapping between the input state vector and the expected outputs. This forms the basis to model formal specifications.
- Step 2: *Modelling Formal Specifications*: This step involves specifying the requirements analysed in the previous step using formal notation as per the syntax and semantics of the specification language used by the tool under consideration.
- Step 3: *Reasoning the correctness of the code w.r.t the formal specifications*. Given the code and the specifications, the tool carries out reasoning about the correctness of code for its specifications.

The subsequent sections describe Theorem Proving and Model Checking techniques with specific tools used.

3. Theorem proving using FRAMA-C

FRAMA-C: FRAmework for Modular Analysis of **C** is a tool for static analysis of C programs. FRAMA-C platform gathers several analysis techniques into a single collaborative framework. It is built upon a set of plug-ins like Weakest Precondition (WP), Value Analysis (VA) to perform static verification. The WP plug-in uses weakest precondition computations to generate proof obligations. FRAMA-C uses its own formal specification language namely ANSI/ISO C Specification Language (ACSL) to express the behavioural properties of C programs. To formally prove the properties, proof obligations are submitted to external automatic theorem provers.

ACSL is used to model function contracts. A contract is an "***opaque***" specification of function behaviour. Contract of a function defines

- ✓ What the function ***requires*** from the outside world
- ✓ What the function ***ensures*** to the outside world
- ✓ provided the "requires" part is fulfilled

3.1 A small example

ACSL is used to model function contracts for given C code. The below snippet describes pre and post conditions modelled for foo function.

The function takes an integer value as an argument and returns a negative or positive value based on value of x. The precondition states the range of x (-2^{n-1} to $2^{n-1}-1$) with "requires" function contract. The post condition describes result w.r.t integer value using "ensures" contract.

3.2 Support for Modular Verification

Theorem proving strongly supports modular verification which can be used to model hierarchical function contracts. As per the principles of assume guarantee reasoning, in modular verification, the pre and post conditions of caller and callee have dual roles in callers proof. Figure 2 illustrates the principle used in modular verification and Fig. 3 analysis flow in FRAMA-C environment.

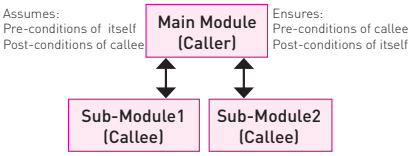


Fig. 2 : Modular Verification

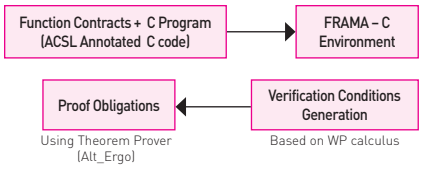


Fig. 3 : Analysis Flow

4. Model checking using CBMC

CBMC, a bounded program analysis tool for C programs formally verifies ANSI-C programs and checks the properties like pointer safety, division by zero, array bounds and user-provided assertions. CBMC forms a transition relation for a program and transforms it into SSA (Static Single Assignment) form and unbound to a fixed length. The specifications are also handled similarly and the final formula consisting of the conjunction of the program SSA and the specification negation is fed to a SAT (Satisfiability) solver. The SAT solver checks for satisfiability, if the resulting formula is satisfiable, a counterexample is produced, else verification success is reported. It allows user input to be modelled using non-determinism (nondet()) so that a program can be checked for a set of inputs rather than a single input. Figure 4 shows the Analysis flow for CBMC.

4.1 CBMC constructs for property verification

CBMC uses CPROVER primitives for modelling user defined assertions.

Nondeterminism: It is desirable to analyze C programs for any choice of inputs. In CBMC, inputs are therefore modelled by means of nondeterminism, where the values of input are not specified. Example: int nondet_int();

Assumes & Assertions: CBMC assumptions, expressed as CPROVER_assume, restricts the program traces that follow the assumptions. It takes a Boolean expression.

The assert statement, expressed as CPROVER_assert takes a Boolean expression as argument. CBMC checks

```
int foo (int x)  // C Code                  /*@ requires (x >= -2147483647); // precondition
{ if (x <0) return –x;          ⟶           ensures \result >= 0;          // postconditions
                                            ensures x < 0 ➞ \result == -x; // postconditions
  else return x; }                          ensures x >= 0 ➞ \result == x; // postconditions
```

whether this condition is true for each run of the designed code. If the condition is true for each run, then it reports Verification Successful.

4.2 A simple example

```
int main()
{
  int a ,b;
  __CPROVER_assume( 0 <= a <= 5);
  __CPROVER_assume( 0 <= b <= 6);
  __CPROVER_assert( 0 <= a+b <= 11, "Success");
    return 0;
}
```

In the above code value of 'a' lies between 0 to 5 , b from 0 to 6 and hence assertion 'a+b' displays verification successful.

5. Conclusions

Formal method approaches namely Theorem Proving and Model Checking as an alternative approach to conventional testing methods greatly enhance software reliability. Corner case bugs for critical modules of Mission / Safety critical software could be detected and could prove the absence of errors.

However, in spite of advances in the two key methodologies, reasoning about the correctness of floating point programs is still challenging. Inherent large input space of floats leading to state space explosion stands as a bottle neck for model checking methodologies. On the contrary, lack of automated theorem provers for discharging proof obligations for floating point properties makes it tough to apply Theorem Proving methods for floating point program verification.

6. References

[1] The Science of Programming – David Gries

[2] Principles of Model Checking – Christel Baier & Joost-Pieter Katoen

[3] A survey of automated techniques for formal software verification – Silva, Vijay D., Daniel Kroening & George Weissenbacher

[4] CPROVER User Manual – Kroening, D., & E. Clarke

[5] Practical Introduction to FRAMA-C – David M ENTRE

[6] ACSL by Example – Hans Pohl, Jens Gerlach

■

About the Author

Mrs. A Sowmya Mitra (CSI-1510343) pursued her B.S in Information Systems from BITS, PILANI and is presently working as a scientist in Software Quality Assurance group in DRDO. Her areas of interest are Software Engineering, Software testing, IV&V and Formal Methods.

Real – Time System: A challenge for Testers

Nancy Goel
Assistant Professor in Chitkara University, Himachal Pradesh

Shaily Jain
Associate Professor in Chitkara University, Himachal Pardesh

Testing of a Real-Time System (RTS) is a big challenge for skilled and experienced Testers. Design issues majorly affect the testing strategies and testability of the system. This article gives a brief introduction to some of these issues and challenges faced by testers during testing Real-Time System than testing non Real-Time System(non RTS).

I. Introduction:

Time constraints are a big parameter in testing RTS though non RTS principles are also applicable for RTS. There are other issues related to hardware and software design decision. As we know that Hardware and Software design decisions have an intense effect on testing strategies of the system. There are twenty key areas in Test Process Improvement (TPI) and one of them defines that to involve testers at the earliest stage of project results in focus on testability [1]. So, it is required that testers and designers should work together as early as possible in the development process, resulting in high quality product.

II. Terminology used and basic concepts:

Real Time System: Many systems such as computer, including most RTS can be viewed in Figure 1. In this figure event stands for input and task stands for initiating a computation. This produces a result after abort. A task or job with respect to Real-time is a task that must be executed or accomplished at meant time [2] and completes before the deliberate point of time known as deadline.

We include the value domain and the time domain both as an important factor to define RTS. We have different classes (soft, firm, hard essential and hard critical) of RTS based on cost of missing a deadline [3]. While designing RTS the type of events or different events and its frequency should be considered. An event can be of periodic type, sporadic or aperiodic type. A load hypothesis is also calculated on the basis of terms of types and frequencies of the events [2].

Testing: Dynamic execution of test cases hold assessing and increasing reliability [4] as its two main aspects [5] . Testing itself defines assessing the reliability, based on selecting the test cases; executing them on operational distribution and supervising the number of confront failures. The failures are analyzed and the reason behind it known as fault [5] is removed thereby the reliability is supposed to increase. There are different testing methods (e.g. Boundary value analysis, State-base testing, Equivalence partitioning, Syntax testing [6]) which helps in generating test suites containing many test cases primarily before uncovering failures.

A good example where same strategies has been used for testing the value domain in RTS as used in non RTS is the DOI187b standard for testing avionics systems[7]. RTS need to be tested in both temporal domain and value domain [8] which is the main challenge in testing. Testing in temporal domain means input should be given to the test object at an accurate moment and we need to control the temporal state of the test object at the beginning of test execution. The result timing must be examined thoroughly because there is a probability of non-determinism also.

Testability depends upon two central concepts [8] known as Observability i.e. utility provided by the system to detect what the system does, how and when it does it, and the other concept known as Controllability i.e. utility accessible to the user to command the execution or re-execution of a test case. A system's testability characterize the features of software needed for validating the software [9].

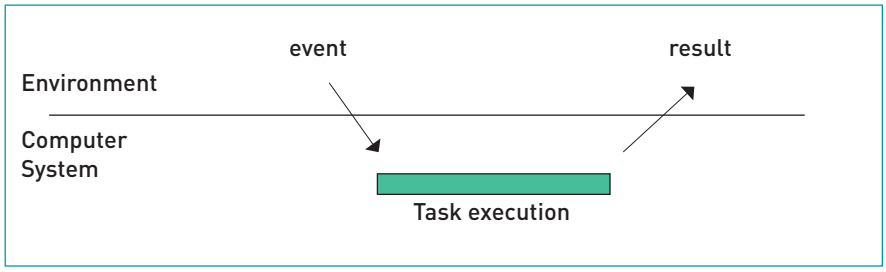


Fig. 1 : Simple model of a computer system.

III. Factors affecting the design of a RTS:

1. **Scheduling:** Sometimes more than two jobs or tasks will be possible to execute simultaneously in RTS. So there must be a sequence of job execution based on some set of rules known as Scheduling. It can be either dynamic where there is execution done without any prior calculation and conflict of jobs could be solved with set of rules using priority approach known as preemption; or it can be static where execution of job is done with prior calculation and is cyclic [10].
2. **Design pattern:** For RTS we

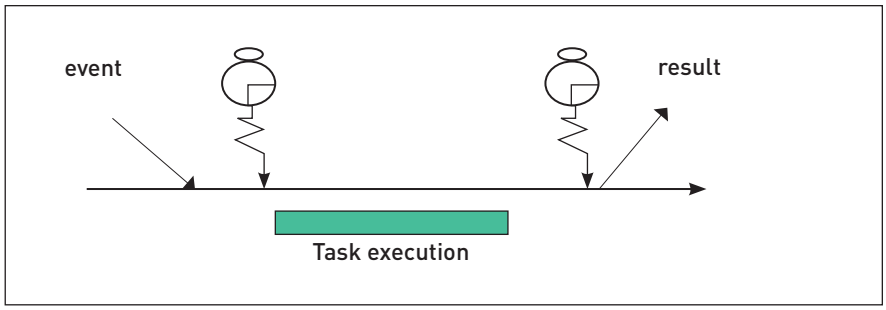


Fig. 2 : Observation and reaction to an event in a time-triggered system.

have two patterns such as Time-triggered and Event-triggered [8]. The main difference between these two is the performance which occurred when communication between RTS and environment takes place. A Time-triggered system (T-t system) is explained in figure 2 given below that it is implemented by polling and uses static scheduling. Here overburden conditions cannot be handled because communication of systems with its environment is done at already defined areas in time. In this system evaluated outcome will not be moved back to the environment until next communication point.

In an Event-triggered system (E-t system) as shown in figure 3 given below, an event can be done at any time, and an evaluated outcome can be moved to the environment at the same time. It may face overburden conditions so it is designed to handle such conditions dynamically. Designers should keep it in notice that deadlines can be missed, so design the system accordingly to decrease the damage level. It is critical to promise for a low level of service in such systems.

A. There are many contradictions in values of various properties for Time-triggered and Event-triggered systems both. An overview of how these two are in conflict with each other is shown in figure 4 given below.

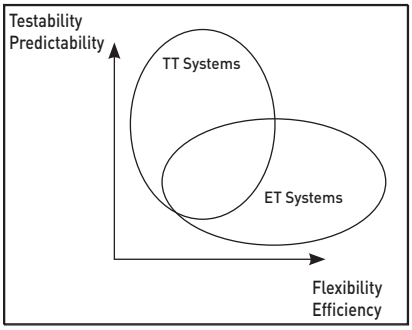


Fig. 4 : Trade-off between testability/ predictability and flexibility/efficiency for the two design paradigms

While performing testing in T-t systems, Controllability is improved as the tester only need to know about for which observation time period an event is to be initialized to the system due to static scheduling (jobs to be executed in same order). Whereas in E-t systems, dynamically order of job execution in done which makes testing more hard and results may vary. So, we can say that here testing should not be done with systematic coverage exploration and instead we must adopt statistical testing approach with customize loads.

Flexibility can be obtained in E-t systems because of its ability to handle overburden and any change in the system or its parts successfully disables all prior test outcomes. Whereas any type of modification in static scheduling is not sensible. So, T-t systems are not a good choice as it may results in re-calculation, re-testing, and sometimes needs to redesign the systems.

Efficiency of R-T system can be achieved by measuring the difference between worst case and average case execution time. Here we consider sporadic jobs. These jobs are scheduled as periodic in T-t system. So we need to measure the difference between its worst case and average case execution time. If it is less or differ more then there is resource wastage and that remaining time is not useful. But in case of E-t systems sporadic jobs are scheduled on their occurrence. Here either an execution of more critical job is done by preempting the currently executing job or less critical job has to wait than the current job resulting in difference between its worst case and average case execution time. Then this will give outcome as higher efficiency and the remaining time may be used for execution of less critical job by dynamic scheduling.

Predictability of a RTS has been increased in T-t system due to negligible order of events and static scheduling. Here execution of job is pre-determined and this property is very important with respect to hard critical RTS as these systems require more confidence regarding its working in all conditions. Whereas due to dynamic scheduling, E-t system decreases the predictability results in making regression testing harder. That is why statistical test methods are adopted for E-t system.

B. Various Tools attributes:

It is very difficult to obtain a trading testing tool for automatic test execution of RTS. One of its reason is the system might be under Black Box testing where support on time is must. Embedded systems, Specialized application fields with particular requirements are reasons resulting hard to exist such tool that can bear the new technology [11].

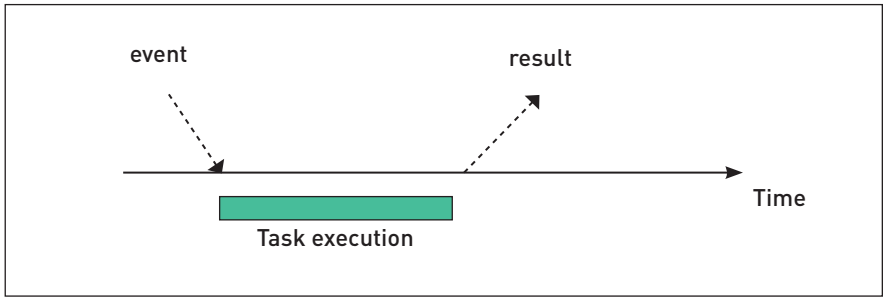


Fig. 2 : Observation and reaction to an event in a time-triggered system.

- A tool must have an attribute of measuring how much part has been taken by it in examining when the system was under testing.
- The timing of the input is important for RTS and an introduction of an event is the required feature of a testing tool.
- The input should be released to the system under test at a predefined time.
- Tool either supports time-stamped or timers for the events to take place.
- There should be clock synchronization in case of a tool for automatic test execution. Having a real-time network, synchronization can be achieved either with a global clock used as a master clock frequency, or the local clocks are used with a clock synchronization protocol.
- Advanced tools must support resynchronization of the test execution whenever a failure has been detected. Such tools can take actions to re-install the system under test into a familiar state and restart the test case execution at a point from test suite after the failed test case.

IV. Conclusion:

Testing team must participate in design decisions to make system more flexible.

This is a big challenge for non Real-Time Systems and Real-Time Systems both to diagnose and re-execute the system under test after failure [12].This means testing in RTS is more challenging due to the reason of temporal and value domain test cases which makes Observability and Controllability both harder to get. Temporal domain's testing of RTS also affects prospective tools for automatic test execution which is essential for an efficient test.

References:

[1] T. Koomen and M. Pol, Test Process Improvement A practical step-by-step guide to structured testing, Addison-Wesley and ACM Press, ISBN 0-201-59624-5, 1999

[2] H. Kopetz and P. Veríssimo, *Real Time and Dependability Concepts*, Chapter 16 in Distributed Systems, second edition, Addison-Wesley, edited by S. Mullender, ISBN 0-201-62427-3, 1993

[3] C.D. Locke, *Best-Effort Decision Making for Real-Time Scheduling*, Technical Report CMUCS-86-134, Department of Computer Science, Carnegie-Mellon University, USA, 1986

[4] P.G. Frankl, R. G. Hamlet, B. Littlewood, and L. Stringini, *Evaluating Testing Methods by Delivered Reliability*, IEEE Transactions on Software Engineering, Vol. 24, No. 8, Aug., 1998

[5] BS 7925-1 *Software Testing Vocabulary*, British Standardisation Institute, 1998

[6] B. Beizer, *Software Testing Techniques*, second edition, Van Nostrand Reinhold, ISBN 0-442-20672-0, 1990

[7] DO-178B Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc, 1828 L Street NW, Suite 805, Washington, DC 20036, 1992, URL: http://www.rtca.org

[8] W. Schütz, *The Testability of Distributed Systems*, Kluwer Academic Publishers, ISBN 0-7923-9386-4, 1993

[9] International Standard ISO/IEC 9126. *Information technology – Software product evaluation – Quality characteristics and guidelines for their use*, International Organization for Standardization, International Electrotechnical Commission, Geneva

[10] B. Lindström, J. Mellin, and S.F. Andler, *Testability of Dynamic Real-Time Systems*, Proceedings of Eighth International Conference on Real-Time Computing Systems and Applications (RTCSA2002), Tokyo Japan, 18-20 March 2002, pp. 93-97

[11] L. Hayes, *Automated Testing Handbook*, Software Testing Institute; ISBN: 0-970-74650-4, 1995

[12] R. Iorgulescu and R.E. Seviora. *A Method for Continuos Real-time Supervision*. Software Testing, Verification and Reliability, Vol. 7, pp 69-98, 1997

■

About the Authors

Ms. Nancy Goel is an Assistant Professor in Chitkara University, Himachal Pradesh. She can be reached at nancy.goel@chitkarauniversity.edu.in.

Dr. Shaily Jain [CSI-N1270613] is an Associate Professor in Chitkara University, Himachal Pradesh. Her interest areas are data mining, networks, IOT and embedded systems. She can be reached at shaily.jain@chitkarauniversity.edu.in

Risk Management in Effort Estimation of Agile Methodologies

S Rama Sree
Prof. in Dept. of CSE & Vice Principal at
Aditya Engg. College, Surampalem, Andhra Pradesh

Ch. Prasada Rao
Assoc. Prof in the Department of CSE at
Aditya Engineering College, Andhra Pradesh

Risk management is one of prominent responsibility of the software project management. Software Risk Management is very essential from the inception of the software project to the deployment and retirement of the project. A software project is considered to be risky due to uncertainty of stakeholder's requirements, in accurate effort estimations, poor communication, etc. Most of the projects fail or get cancelled due to non-identification of uncertain future events called risks and in accurate effort estimations. There is a direct correlation between effort estimation and risk management. Effort estimation should provide information on likelihood of project risks. It takes the input as output of other risk analysis activities. The objective of this article is to provide some possible risk driven factors while estimating the software effort in agile methodologies and try to provide the mitigation techniques.

Agile Methodologies:

Now-a-days, most of the companies have been transforming from traditional process to modern process models. The main goal of any process model is to deliver the project with in time, budget, high quality and which can meet all specifications given by the customer. The traditional or heavy weight process model like waterfall , spiral, prototype, unified process models etc., are not able to deliver the product to customer with in triple constraints of software project management and it indicates the failure of the software project. There could be a lot of other reasons for the failure of software projects developed by conventional process models where in poor management, lack of customer involvement, in accurate effort estimations, fail to identify or predict the risk are the most frequent fit falls. To improve the success rate of the software projects, 70% of the companies are adopting Agile Methodologies. Agile Methodologies are light weight process models which are Scrum, Kanban, XP, DSDM, etc. Scrum and XP are very popular among them and which are intended to develop the software projects quicker, faster and with high quality. The agile models have been inculcated with iterative development and incremental delivery of the applications. In Agile Approaches, planning and documentation need not be highly constructive and at the same time customer collaboration and interaction is also highly improved. This results in improvement of feedback.

Effort Estimation in Agile Methodologies:

Project planning is very essential in software project management to make successful projects. Project planning is the process or the series of steps to establish the scope, define the objectives and develop the action steps to obtain them. Project Management Plan includes Scope, Effort, Cost, Time, Quality, Communication, Human Resources, Procurement, Risk, Stakeholder planning etc. One of the prominent activities in software project management is to estimate the effort accurately. In Software Project Management, Effort is measured in Person-Months, Man-Months, Man-Hours, Man-Years, etc. Based on the effort we can calculate the total development time and required cost for the completion of the project. KLOC, FPA, UCP, Object Points, Class points, etc are used for effort estimations in the traditional software development. But none of them have given the accurate effort estimations in software project management when compared to Story Points in Agile Methods. The Standish Group reported in 2015 where 39% of software projects developed by Agile Methods are Successful and only 11% of the Software Projects developed by Waterfall model are Successful. Even though Agile methods performs better than traditional models, there is an evidence of certain risk factors that can impact the accuracy of effort estimation.

Risk Management:

Whether it is traditional process or the modern process, Risk management will be playing a prominent role in software project and process management. Risk management constitutes risk planning, identifying, analyzing, prioritization, monitoring and controlling. The objectives of Risk Management are to reduce the expensive of rework, minimize the probability or likelihood of impact of negative risks . Risk is an uncertain future event or condition that can influence the success of software operation. If the risk occurs, it could be a negative impact (lost) or the positive impact (opportunity). Risk can influence any of the software development activity like project planning, customer

relations, environment establishment, etc,. Risk management is an ongoing activity from the inception of the project to deployment and retirement of the software project. Risk Management team should always be a proactive rather than reactive. Proactive teams start of thinking the potential future harms and also try to fix it . The Standish Group 2015 chaos reported that only 29% projects are successful, 52% are challenged and 19% are failed. One of the reasons to get fail is non-identifying high risk factors while estimating the effort in the software development. The main objective of this article is to address the categories of risk factors that influence the effort estimation in Software Agile Development. The estimations are always go hand in hand with risks.

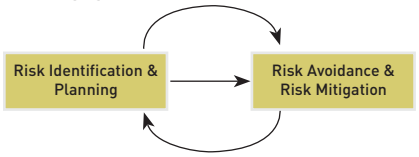


Fig. 1 : Risk Management

Risk Management for effort estimation in Agile Approaches:

The objective of the Risk Management in effort estimation is to identify the probable risk factors and take necessary action plans to reduce the impact of effort related risk in the projects. If the risk occurs there is a probability of over budget, over runs, low quality and increase in rework which leads to failure of the software projects. Common risk categories in the software project management are Software Development Life Cycle (SDLC), Project Management, Group Awareness, External Stakeholder Collaboration, and Technology Setup. Each category of risks is described by risk areas wherein each risk area constitutes risk factors. In the literature, different Effort Factors and Risk Factors are identified. Key Team Capability, Domain Experience, Platform Experience, Communication Capabilities, Requirement Volatile, Customer Involvement....are the relevant effort factors.

Table.1 : Risk Management in Effort Estimation of Agile Methodologies

S. No	Effort Factor	Risk Identification	Risk Mitigation
1.	Domain Experience	Lagging in Domain Experience	Provide domain training by experts in specific domains
2	Requirement Volatile	Unclear and Inadequate Requirements	Obtain clear requirements by frequent communication between team and customer. Construct the prototype of the project to baseline the requirements
3	Customer Involvement	Insufficient customer involvement	Use of rich communication media like video, web conferencing when face-face interaction not possible.
4	Platform Experience	Lack of knowledge about platform in which software is developed.	Include training by experts who are familiar to that platform.
5	Communication Capability	Lack of Communication between team and client	Foster team collaboration by improving communication skills and language skills
		New Teams	Scrum master would not pressurize new teams until 2 or 3 sprints to be completed
6	Inadequate Prioritization of Requirements	Requirement Conflicts	Regular Requirement Prioritization. Tool to give ratings
7	Software Reliability	Results in safety, economic, security and environmental damage	Fault tolerant design, formal validation and testing

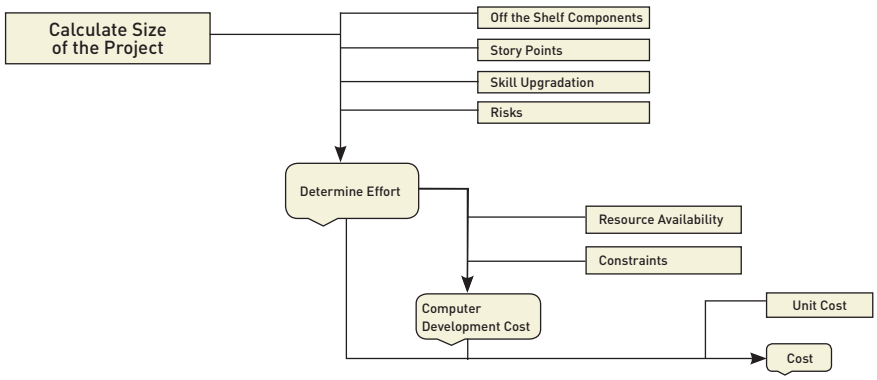


Fig. 2 : Effort and Cost Management

The key responsibilities of Risk Management are to Identify Risk Factors and mitigate the negative impact of the risks. Risk Avoidance and Mitigation approaches are the similar actions to refer the risks. Avoidance is done before the project is initiated where as mitigation would focus on actions during the software development. Mitigate the risk would require that the manger either increases the project budget or the project performance.

Conclusion:

Agile models are producing the more number of successful projects comparatively traditional process models like waterfall model, Unified Process Model, etc. Predicting the effort in the software development is very essential in either traditional or model process models. Accurate effort estimation leads to success of the projects. While estimating the effort so many risk may encounter. This article provides risk assessment and risk control activities associated with effort estimation in Agile software

development. We have noticed that domain experience, communication capability, customer involvement are the most prominent effort factors while estimating the effort. In this article, possible risks were identified while estimating the effort and also suggested the mitigation approach for each risk.

References:

[1] Dr. S V Shrivastava, Dr. Urvasi Rathod, A Risk Management Framework for Distributed Agile methodologies.

[2] Mala.V.Patil, Software Effort Estimation and Risk Analysis-A Case Study, ICTES 2007.

[3] Ekananta Manalif, Luiz Fernando Capretz, Ali Bou Nassif, Danny Ho, Fuzzy-Excom Software Project Risk Management, ICMLA-2012.

[4] Chandan Kumar and Dilip Kumar Yadav, A Probabilistic Software Risk Assessment and Estimation Model for Software Projects, ScienceDirect 2015.

[5] Cinzia Muriana, Giovanni Vizzini, Project risk management: A deterministic quantitative technique for assessment and mitigation, ScienceDirect 2017.

[6] Adam Trendowicz, Software Cost Estimation, Benchmarking, and Risk Assessment, 2013.

[7] Adam Trendowicz, Ross Jeffery, Software Project Effort Estimation, 2013.

■

About the Authors

Dr. S Rama Sree (CSI - F8000836) is a Professor in Department of CSE & Vice Principal at Aditya Engineering College, Surampalem, Andhra Pradesh, India. From the last 15 years she has been involved in teaching the under graduate and post graduate students. She held the administrative position as Head of the Department of CSE for 12 years and currently working as Vice Principal. She published 35 papers in National/International Journals & Conferences. She is a member of several Editorial & Review Boards of International Journals and also member of several professional bodies. She reviewed two text books on C Programming. She received Outstanding Faculty Award and Award for Research Excellence in 2016. Her research interests include Software Cost Estimation, Software Reusability, Software Reliability, Software Prioritization, Software Defect Prediction, Software Maintenance and Soft Computing. She can be reached at ramasree_p@rediffmail.com.

Ch. Prasada Rao (CSI-F8000837) is Assoc. Prof in the Department of CSE at Aditya Engineering College, Andhra Pradesh. He is pursuing Ph.D from K L Universtiy, Vijayawada. His areas of interest are Software Engineering, Compiler Design and Automata Theory. He can be reached at prasadarao.chatla@aec.edu.in.

Logical Hierarchy Requirement Target Planning (LHRTP) technique to overcome risk on Software Projects

R. Saranya

Asst. Professor, Department of Computer Science, Central University of Tamilnadu

Requirements in software are defined to be demand or necessitate. Requirements elicitation for software is able to gather predictable variations clearly over the anticipated duration of the software. Requirement elicitation gather the requirement of system from users, consumers and other stakeholders. The requirement elicitation comprises domain professionals, market experts, and others. Requirements elicitation concentrates on the scope, clearly gathering the predictable difference, and the adoption of use cases that define the variations that are expected to happen over the duration of the software. Requirement elicitation involves identifying and prioritizing requirements as a process is complex to balance large software projects with many stakeholders.

Keywords: Stakeholders, LHRTP, Requirement Elicitation, Mutual Filtering

I. Introduction:

Software engineering concerns with wide use of engineering principles to achieve cost-effective software with potentiality to function on real machines. Requirement engineering in software development is more crucial. Everyone agrees that security is difficult. The requirements engineering principles are framed based on an idea that would engage the community overcoming complex problems. Security is about the prevention of several difficulties due to the presence of attackers behaving malicious activities. Software security is incredible due to the intrinsically complex task and the problem happens because of three main reasons such as networks are everywhere, systems are easily extensible and system complexity is rising. The principle objective of requirement engineering research is not just to point out the fact about security risks keep on rising every day, but rather to defeat attack, just as any other system property. Security should be tackled at the beginning of the software lifecycle.

Requirement engineering is a processes used to find, analyze and validate system requirements. The objective of requirement engineering is to describe the principal requirements engineering activities such as elicitation, specification and validation. Requirement elicitation is the process of determining, understanding, reporting, and realizing the user requirements and constraints for the system. Requirements specification is the process of documenting the user's needs and constraints unmistakably and accurately. Requirements verification is the process of promising the system requirements are absolute, accurate, reliable, and understandable. The security at the requirements stage is most essential concept for understanding not only level of secure software but also a secure way to guarantee user satisfaction with end product. In order to facilitate better understanding, each of elicitation specification and validation phases considers a significant aspect of the requirements engineering stage. Section1 shows the three commonly used methods for Requirement elicitation and their limitations, Section 2 describe the Logical Hierarchy Requirement Target Planning (LHRTP) technique which overcomes the exciting method's limitations, Section 3 gives the experimental evaluation result of LHRTP.

2. Existing methods for Requirement Elicitation:

(i) Stake Rare Method

Stake Rare [1] is a method that is able to balance the large software projects with many stakeholders using social networks and collaborative filtering to identify and prioritize requirements. Collaborative filtering is utilized to prioritize the requirements based on the stakeholder ratings. Stake Rare identified stakeholders and request identified stakeholder to suggest other stakeholders and stakeholder actions. Constructs a social network with stakeholders as nodes and their suggestions as connections, and prioritizes stakeholders using a range of social network calculations to decide software project authority. Using the information gathered from surveying and interviewing 87 stakeholders, the

report demonstrated that Stake Rare forecasts stakeholder requirements accurately.

Stake Rare provides a more absolute and correctly prioritized list of requirements using collaborative filtering. But Stake Rare involves certain challenges regarding collaborative user requirements and Fails to improve the objectiveness and scalability of the larger system

(ii) Security Requirements Engineering (SRE)

Here a process is proposed to address the challenges of collaborative user requirements elicitation. The process promotes stakeholder agreement, through exploiting team dynamics to fetch a better understanding of requirements. However, the process needs a further work to evaluate the process within a larger-scale project. A framework for security requirements elicitation and analysis [2] is build based on constructing a context for the system, denoting security requirements as criteria, and emerging satisfaction metrics for the security requirements. The system context is expressed using a problem-oriented information, then is validated against the security requirements during building of a satisfaction metrics. The accuracy of security requirement is achieved using satisfaction metrics. Furthermore, framework needs to solve risk analysis and understanding of formal arguments. Constraints satisfies the argument for security requirements. System context using problem oriented notation is validated against security requirements.

SRE Methods did not impose structure on the requirements analysis process and fail to adequately provide for the description of Stake holder activities

(iii) Model-based Oracle Software Generation (MOG) method

A framework is presented to elicit the software requirements and also to prioritize the software requirements. The proposed framework ranked the requirements by the qualified level of threat related with each requirement through AHP. As a further enhancement, AHP is only utilized to only determine the importance weight of the requirement and not to prioritize requirement .Then prioritization is performed to manage the requirements. Prioritization through AHP results using threat level analysis is not more suitable for facing DOS attacks. Define an automatically generated partial, passive oracle from the agent design models. Not yet develop for preventing fault at initial stage of software requirement specification.

3. Logical Hierarchy Requirement Target Planning (LHRTP)

Stakeholder is an individual or group who influence the success of failure of a project.

This approach handle the suggestions of the multiple stakeholders' and developed an effective framework for larger system. The problem of information overload is overcome by prioritizing according to the requirements of the stakeholder.

The LHRTP approach uses to recognize and prioritize stakeholders based on the influential factor in larger project. Stakeholders are identified from the list of stakeholders namely stakeholders 1, 2, 3....n as they form the source of requirements. Furthermore, they are prioritized, according to their level of influence (i.e.,) scores in the larger projects.

Fig. 1. Shows top-down approach which assesses the virtual importance of assessment criteria. The assessment criteria compare and substitute with respect to each criterion, and resolve the problem related to the prioritization of larger projects for every decision substitute to reach the effective objective.

The first step in LHRTP approach is to construct the chain-of-command in such a way that the decision goal is at the top level, assessment criterion and sub-criterion are in the middle levels followed by the assessment alternatives at the bottom. Once the chain-of-command is constructed, LHRTP approach provides a structured framework for setting software requirements priorities on each level using paired comparisons.

As shown in Fig. 2, the top-down approach represented with top, sub-criterion and bottom level. In order to identify the criterion and sub-criterion on larger projects, discussion were conducted on separate academic, consulting, and governmental stakeholders. Each stakeholder from multiple groups has experts who were asked for their input requirements regarding the criterion and sub-criterion levels for removing the risk factor. After thorough and intensive planning with the experts, a refined list of software criterion and sub-criterion were obtained.

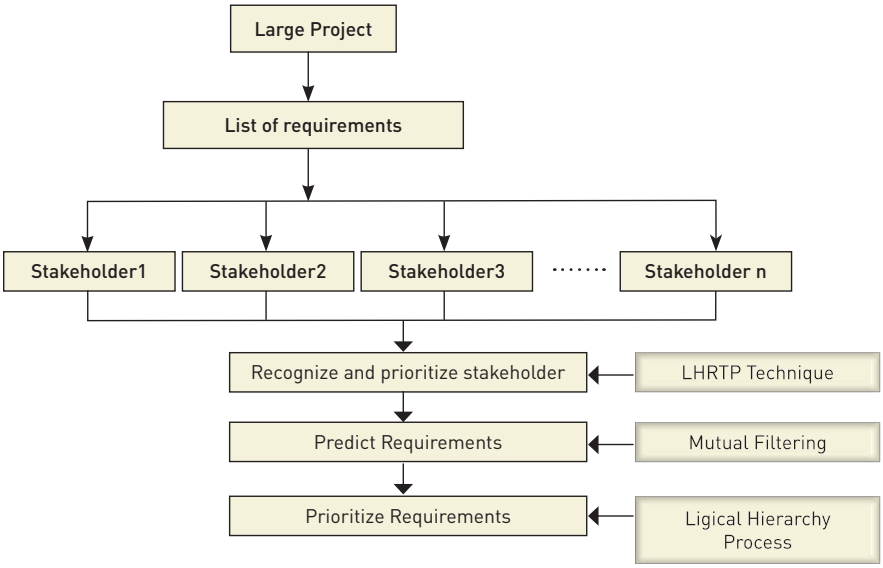


Fig. 1 : Flow Diagram of LHRTP approach

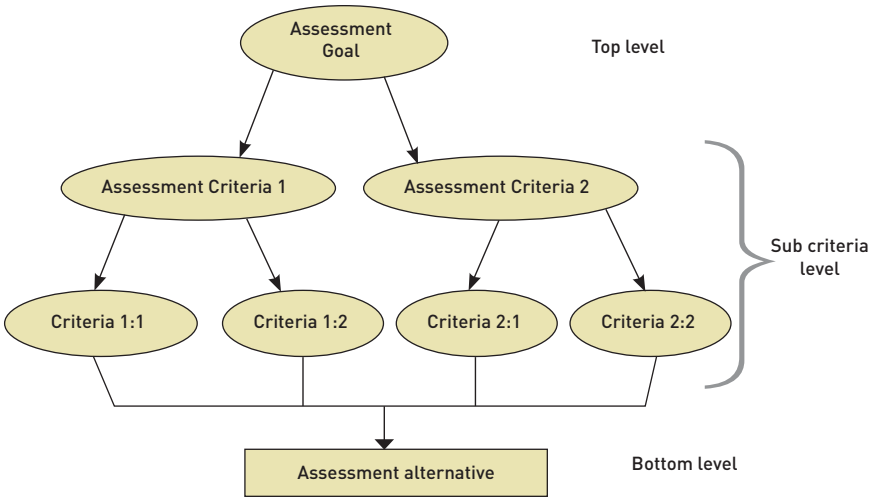


Fig. 2 : Hierarchy Process Representation

Mutual Filtering in LHRTP Technique

Mutual filtering performs the software prediction in addition to their scores in order to forecast each user's preference for un-scored items. Mutual filtering recommender software systems produce recommendations for a given user on one or more items. In mutual filtering, users are the individuals who provide scores to a system and receive recommendations from the system. A scoring is a statistical demonstration of a user's preference for an item. Filtering using Top down approach is the set of score that a particular user has provided to the system.

The requirements that are highly significant are recommended to stakeholders in order to keep away from information overload. Mutual filtering filter large sets of data projects for removing the inconvenience for end users. By collecting information from many users, prediction of user interest is made very easily. The scoring from the stakeholders' and the priority of the stakeholders are used to prioritize the requirements in LHRTP approach. To compute the importance of a requirement in a large project, the influence of the stakeholder's role in the project is identified, and then the control of the stakeholders in their roles is determined.

The algorithm for LHRTP approach is:

Begin

Step 1: Identify criterion, sub-criterion from the list of requirements of multiple stakeholders

Step 2: Stakeholders of count 'N' is used to identify the substitute

Step 3: Build the top-down approach for each stakeholder

Step 4: Top-Down approach follows with global load to recognize and prioritize the stakeholders

Step 5: Mutual Filtering in TD-REP approach predict the requirements

Step 6: Analyze the global load for sub-criterion and performance of the substitute with respect to the sub- criterion

Step 7: Obtain the global weights of sub-criterion using the top-down approach with objective function. End

4. Experimental Evaluation of Requirements Elicitation Using LHRTP

Performance experiments are conducted with various conditions using

Tabulation of System Success Rate

Software Requirement Identifier	System Success Rate (success %)		
	StakeRare method	SRE Framework	LHRTP technique
1	78	82	86
2	79	84	87
3	82	87	92
4	84	88	93
5	84	89	92
6	86	92	95
7	87	93	97

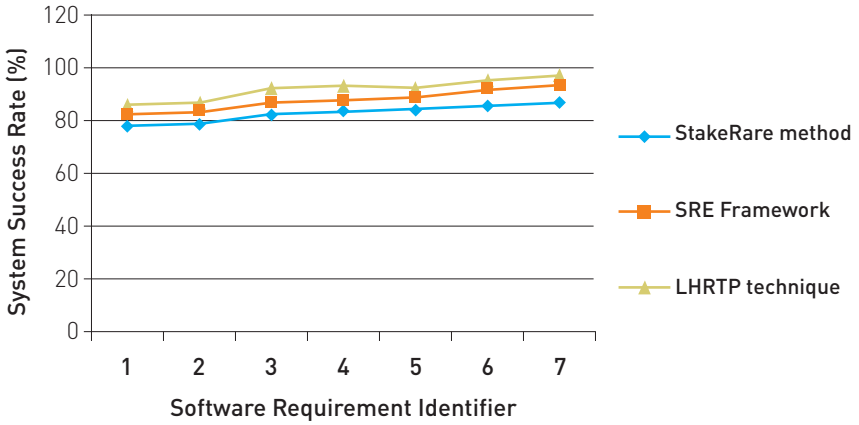


Fig. 3 : Measure of System Success Rate

JAVA platform for software requirements elicitation. RALIC Dataset is used for performing the experiment on TD-REP which contains the various datasets of stakeholders and their requirements on a real software project. The RALIC datasets consist of 61 stakeholders from OpenR, 50 stakeholders from ClosedR, 76 stakeholders on 10 RateP-Obj objectives, 76 stakeholders on 48 requirements (RateP-Req) 76 stakeholders on 104 precise necessities (RateP-SReq), 79 stakeholders on 10 project objectives (RankP-Obj), and 79 stakeholders on 51 requirements (RankP-Req).

The Top-Down approach based on Requirements Elicitation target Planning (TD-REP) approach is compared against the StakeRare method and Security Requirements Engineering (SRE) framework.

Fig. 3 illustrates the system success rate based on the software requirement identifier. The software identifier starts with '1' in LHRTP and success rate improved by 9-12% when compared with the StakeRare method [1] and 3-5% success rate improved when compared with the SRE Framework [2]. The topdown approach engages the multiple stakeholders in the process of assessing the software requirements in a chain of-command form. The chain-of-command form of structure improves the success rate in TD-REP approach.

Conclusion:

The software requirements elicitation overcomes the problems such as the information overload and prioritization of requirements using a LHRTP based on Requirements Elicitation target Planning. This approach initially performed the identification of larger project, analysis of requirements, recognize and prioritize stakeholders, followed by prediction and prioritization. The LHRTP approach for software requirements elicitation process based on the suggestion of multiple stakeholders and mutual filtering overcomes the inconvenience during the interaction of end-users on larger projects. The LHRTP approach balance among the possibly different inputs obtained from individual stakeholders in order to reach optimum result by overcoming the risk. The experiments of LHRTP approach are conducted for different conditions using JAVA platform to attain the maximum objective function value, scalability ratio, maximal success rate, 7.405% improved confidentiality rate with minimal computational cost, and masquerading rate. The LHRTP approach is capable of resolving conflict, fulfillment of both tangible and intangible criteria from different stakeholders' view points, to achieve different goals.

References

[1] Soo Ling Lim., and Anthony Finkelstein., "StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation," IEEE Transactions on Software Engineering, 2012

[2] Haley., Charles B.; Laney., Robin; Moffett., Jonathan D. and Nuseibeh, Bashar., "Security Requirements Engineering: A Framework for Representation and Analysis," IEEE Transactions on Software Engineering,2008

[3] Silvia T. Acuña., John W. Castro., Natalia Juristo., "A HCI technique for improving requirements elicitation," Information and Software Technology., Elsevier journal, 2012

[4] Qasem Nijem., "Software Requirements Elicitation Tools for Service Oriented Architecture: Comparative Analysis," International Journal of Computing Academic Research (IJCAR) ISSN 2305-9184 Volume 2, Number 3 (June 2015), pp. 109-122

■

About the Author

Dr. R Saranya [CSI-I1503352] is working as Assistant Professor, Department of Computer Science, School of Mathematics and Computer Sciences, Central University of Tamilnadu, Thiruvarur. Her Areas of specialization are Cyber Security, Big Data Analytics, IoT. She has been awarded Mother Teresa Sadbhawana Award in recognition of Sterling Merit Excellent Performance and outstanding contribution. E-mail: saranya@cutn.ac.in

Technological advances in Software Engineering

V Vetriselvi

.Sc.,M.Phil.,SET, Asst. Professor ,Department of MCA, Shrimati Indira Gandhi College, Trichy-2. Email id: vetriselviramesh@gmail.com

A major challenge for software engineering today is to improve the software production process. Nowadays, most software systems handcrafted, wade software project management is primarily based on tenuous conventions. Software engineering faces the challenge of replacing the conventional mode of operation by computer-based technology. This theme underlies the Software Engineering Institute that the DoD has established at Carnegie, Mellon University. Among the contributors to software development technology are ideas, such as object-oriented programming, hardware improvements related to personal workstations, and programming environments that provide integrated sets of tools for software development and project management. Facilities and tools are by themselves not sufficient tc achieve an order of magnitude improvement in the software production process. Future directions in software engineering must emphasize a constructive approach to the design of reusable software and to automatic generation of programs. We will briefly explore the promising technology that can be used to implement these ideas.

Introduction:

A major for the software engineering field is to bring about a radical improvement in the software production process which is plagued toy low quality and inflexibility of its products and serious overruns in terms of both cost and time. A decade ago, when software engineering first emerged as a separate sub discipline, the initial focus mere on controlling the production process than achieving a radical improvement by changing the process. The result of this control view has been a number of substantial activities in areas such as measuring system performance and programmer productivity and developing techniques for program testing and symbolic debugging. Today's practice is still largely dominated by this control view and its ensuing analytic approach to improving software production. although useful for better understanding of the software production process and suitable for finding gradual improvements, analytic tools of the kind mentioned above are generally not adequate to achieve an order of magnitude improvement in the Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission. A radical improvement requires a constructive approach that changes the process itself instead of improving on existing practices. The purpose of this paper is to review the current events in software engineering that support this constructive approach and to explore future developments that may lead to a substantial improvement of the software production process. Current events that relate to software production are rooted in the short but dynamic history of software engineering. A brief analysis of the history is followed by a discussion of one of the major events in the area at the present time, which is the mechanization of the discipline into support systems arid tools that assist programmers in the application of software engineering techniques. The observation that tools and support systems are by themselves not sufficient to bring about a radical improvement in the software production process leads to an outlook on the future. The major ideas and concepts for achieving the desired radical improvement in software production are reusability and automation. The last part of tiffs paper is dedicated to a discussion of these subjects and shows how they can be applied in practice. It is the author's belief that the software production process can be improved substantially if we can steer the development of software engineering in the direction of reusability and automation.

The nature of Software Engineering

Engineering is the creation of mechanisms or objects that facilitate the achievement of a goal. Civil engineers build bridges for people to get to the other sides electrical engineers build radios for the purpose of broadcasting news and musical entertainment, programmers write database systems for people to store and retrieve information. The .Oxford dictionary of

the American Language stresses the fact that engineering is the application of scientific knowledge and the control of power to achieve the intended goal. The adjective "scientific" seems unnecessarily restrictive because experience and transfer of know how are substantial factors in engineering without necessarily being scientific. An interesting aspect of engineering is that the goal of an engineering endeavor may be to facilitate achieving some other goal. This is the idea of a tool. The mechanical engineer, for instance, may design a floating crane that is used by the civil engineer to build his bridge. The goal of tile mechanical engineer is not a particular bridge, but the process Of building bridges. The object he creates is not the beam that spans the river, but the tool that enables the civil engineer to put that beam in place.

Software engineering is particularly concerned with the general criteria that determine the quality of a design and of the resulting software product. Criteria frequently discussed in the literature are Correctness the correspondence of specification, design and implementation Reliability the ability to reproduce a result Performance the ability to respond within tolerable time limits without excessive demands on storage capacity Adaptability the ability to modify software to take advantage of hardware improvements or to respond to changing application requirements Extensibility the ability to extend the functionality of a system Friendliness the ability to interact with the user in terms of understandable messages while not requiring irrelevant precision of user input Reusability the use of parts of a system in the design and implementation of another system Fault. Tolerance the protection of information integrity against hardware or power failure Robustness the protection of information integrity against unintentional user mistakes and malicious user acts Privacy/Security the protection of information against unauthorized access and against the effects of modification in someone else's data

The Evolution of Software Engineering

The foundation for software engineering was laid in the sixties with the invention and formulation of basic concepts in programming languages and operating systems. The design of FORTRAN which introduced the concept of procedural abstraction was soon followed by the design of Algol60 which introduced a wealth of new concepts including data types, parameter evaluation 30 modes, recursive procedures, static and dynamic scopes, dynamic data objects and a formal description of language syntax. Later in the decade, SIMULA67 introduced the concept of object-oriented programming through classes and subclasses, while Algol68 and Pascal introduced user defined data types, reference variables and disjunctive type structures. Much of the engineering during this period was concerned with the optimization of parsing and code generation and with the efficient use of hardware resources in timesharing operating systems. Around 1970, the focus of attention shifted from basic concepts in languages and systems and their implementation to the construction of systems out of program modules. Programmers became more ambitious and wanted to construct systems that were hard to express in a single program. At this point in time, the need arose for programming-in-the-large which concerns itself with program interface specifications, the modification process of program modules in the context of an evolving system, and the interaction between programmers in the context of a software production project. This development had the effect that software engineering shifted its focus from the construction of individual programs to the process that controls the creation of software systems. The transition from pure programming-in-the-small to the more ambitious programming in the large is viewed by many as the actual birth of software engineering. The distinction between these two forms of programming was clearly stated for the first time in a seminal paper by DeRemer and Kron [DK76]. Some of the most important initial results of software engineering were the modularity concept and Parnas' hiding principle. Other constructive work in software engineering of that period included the design of system version control and configuration management mechanisms. In addition, a substantial effort was put into measurements of performance and productivity as well as into models for controlling the software life cycle which includes the production process from inception and specification to implementation and successive releases. The waterfall model is the best known among the various models proposed for life cycle management [Le80]. An alternative approach to controlling the complexity of large software systems is taken by the founders of a programming methodology. Their activities give rise to the concept of structured programming and to various approaches to program verification. Structured programming is in fact a philosophy based on the limitations of human beings in dealing with the substance of programs. It builds on our strengths (rather than our weaknesses) by promoting the utilization of three of our abilities in dealing with algorithms, enumeration, induction and abstraction. Enumeration allows us to distinguish between an oversee able number of cases, Induction allows us to make use of iteration and recursion, Abstraction allows us to ignore details at proper moments and to reduce complexity by viewing collections of objects as atomic units. Program verification has been put on a solid basis in the last decade. The axiomatic approach is particularly suitable for proving the correctness of programs based on their control structure. Algebraic verification is particularly well suited for demonstrating the completeness and consistency of a collection of operations defined for an encapsulated data structure. The method of a denotation description of the semantics is particularly suitable for showing the consistency of a language design and for expressing the meaning and interpretation of language constructs. Although program verification is well understood, a major drawback of the state of the art is our inability to apply the various methods to large systems. The attempts in that direction have resulted in some interesting interactive verification systems that can handle

small to medium size programs but not large systems consisting of many components that are not always collectively available. The state of the art in program verification at the end of the last decade was one of the causes for another change in the direction of software engineering leading to the exploration of software development tools and environments. Two other causes were the analytic approach to improving the software production process and the labor-intensive implementation of life cycle support. The analytic approach blocked further progress because of the tacit assumption that the software production process was basically well organized and needed only further local optimization. The approach to life cycle support puts system development and project management entirely in the hands of people with little or no support from software technology. In the next section, we discuss the resulting events of the present that are characterized by a mechanization of life cycle support into integrated programming environments.

Programming Environments

A programming environment is a software system that supports the development and maintenance of software products. The term "programming environment" does not refer so much to the activity of writing programs, but more to the manipulation of programs for the purpose of system generation, configuration and version control, project management and documentation. Althoughthe term "system development environment" is actually more appropriate in this context, we will stay with tradition and stick to the widely used term "programming environment" to denote systems that support the entire spectrum of activities involving software production. The goal is for programming environments to support the entire life cycle and not just the programming fraction of life cycle. Traditional programming environments lack some properties that seem very desirable in modern programming environments. These properties are tool integration and uniformity of the user interface. Tools are integrated when they possess common knowledge that can be applied in each tool. This common knowledge often takes the form of shared data formats or of information stored in a common database. An example of tool integration is the combination of editor, compiler and debugger that all operate on a common syntax tree. The editor shares syntactic knowledge with the compiler and is able to enforce the syntax rules while a program is being written. The debugger shares program structure knowledge with the compiler and is able to translate problems back into source representation through the common database. Tool integration is of great help to create environments that are more specifically task. Oriented than the traditional general-purpose environments which are still most common today. Tool integration is almost totally lacking in the traditional programming environment. Tools such as the text editor, the compiler, the linking. Loader ned the debugger share at best some knowledge of the underlying file system. No information is shared, however, about data formats or data values and no information is exchanged through a common database. A text file is a Pascal program, for instance, because the author believes it is one, not because the text editor checked that it really is.

Reusability and Automation:

Programming environments, software metrics and software engineering methodologies are helpful, but not enough to bring about an order of magnitude improvement in the quality of our software products and in the predictability of the production cost and effort. Constructive and analytic techniques are both valuable and should be further pursued to facilitate the software production process. However, one must expect no more than a gradual improvement from applying these techniques, because none of them necessarily changes the software production process itself. It seems that a major problem in software production is the fact that most software is written from scratch. The reasons why this is common practice are threefold.

- Programs are hard to read
- Programs are strongly tied to their context
- Information on existence of programs is often hard to get.

The fact that the meaning of a program is hard to derive from the source code gives programmers the feeling that one might as well write the program from scratch instead of trying to understand the designer's reasoning behind an existing program text. Although documentation is of some help, its two major drawbacks are its separation hem the source text and its lack of rules that guarantee uniformly and completeness. Formal specifications are in fact far more hopeful for an accurate description of what a program does, but 3re generally even harder to understand than the source text. A possible solution to this problem is to agree on a functional description of programs that does not describe in detail what a program does, but describes the data structures it uses, the input values it accepts and the output values it produces. After going through the effort of understanding someone else's program, good intentions are often rewarded with disappointment because of the program's dependency on the runtime environment. Even if a program is designed to run on a popular operating system such as UNIX rM2, the programmer who wants to make use of it in his environment will discover that the Program does not run because of incompatible peripheral equipment or local operating system extensions. Context dependencies are often hard to detect because documentation on these matters is rarely provided. A programming language such as Ada may alleviate this problem because of its precise description of package dependencies. Explicit description of a program's dependency on other programs is strongly recommended over implicit dependencies that are generated by deep nesting of scopes and by an excessive use of global objects. It is often very hard to find out which programs written by other people can be used again. Many programs are designed as system modules and are burried deep down in a system description. Names of program modules often make little sense outside of the system context, while the purpose

of a program usually is described in relation to the modules it interacts with instead of in terms of its own independent functionality. The result is that a design always seems unnecessarily complicated to an outsider. This phenomenon causes the outsider to think that he could have done a better job than the designer of the existing program. This lack of confidence in your fellow programmer is a major cause of unnecessary duplication of effort. The problem of acquiring information about the existence of programs is solved in part by encouraging the potential user to try harder to find out what is available and how it is used. However, this is not a reasonable proposition without counting on substantial help from the original designer. The only way that one can realistically hope that people will try to reuse software is to demand that designers of original programs take reusability into account from the start. If reusability is adopted as an original design objective, one may expect a program documentation style to emerge that clearly explains a program's independent functionality, its intended use and its dependency on its context. A programming environment can play an important role in making software reusable. It can provide facilities that allow users to browse through libraries that describe existing programs and their usage. The better programming environment will provide, in addition, an engineering environment that is used for transforming an existing program into one needed for a specific application, or for deriving a specific program from a general description. Reuse of software is at this point in time our best hope for improving the software production process and its resulting product. It has the potential of reducing the cost and effort of the process and it has a good chance of increasing reliability through incremental modifications of programs of proven quality. However, the preceding discussion shows that the term reusability must not be given the narrow interpretation of reusing existing programs without change, In fact, reusability spans a spectrum of applications that each makes sense in a particular context. Two direct applications of reusability are the use of program libraries and of shared code. The best example of reused program libraries is that of mathematical subroutines. The IEEE Society has done us a good service by standardizing a set of mathematical routines, including specifications of input/output precision. It would be extremely helpful if similar standard packages were designed and maintained for string processing, window management and namespace management. The Ada language made a useful contribution by including in the language a standard package for file handling and for text I/O. Users of timesharing operating systems are very familiar with the idea of sharing code. Their programs routinely use common operating system facilities for file handling, input/output and memory management. It is in this context immaterial whether or not executable versions of code are shared. Even if programs each use their own copy of a common program, the fact that counts is that the utility program was not written by the user, but taken, as is, from an available pool. Practice has shown that reusability through code sharing is greatly facilitated by eliminating the context dependency factor. This can be done in one of two ways: either by writing a program that is independent of its context (this is basically the Ada Language approach), or by having the various users work in the same context so that context dependencies are irrelevant. Although the latter seems to be a cop out, its usefulness has been firmly established by the success of the UNIX operating system, The reason why many companies are interested in standardizing on UNIX, as universities basically have done over the last decade, is to capitalize on the available software that runs on UNIX while avoiding the problems o| having to translate and rewrite existing programs to run on different operating systems. The Aria language provides another form of reusability through type abstraction [Ad83]. Generic packages can be written in Ada that specify the traversal and updating operations on data structures while leaving the element type unspecified. This facility supports the concept of reusability by allowing a programmer to define the details of a data structure once and for all, independent of what type of objects will be stored in that structure. A generic package for queues, for instance, can be instantiated for messages, for jobs, for arrival and departure schedules, etc. Other practical forms of reusability are through specification and through common design. Memory management and parsing techniques are the typical examples of common design that is applied in many operating systems and compilers. Although this form of reusability has the drawback of requiring implementation, it has the great 2UNIX is a trademark of Bell Laboratories. Advantage of building on a conceptual basis that can be taught in the classroom. Reusability can be greatly enhanced by automation of the program generation process. Automation in this context means using tools that translate a precise program specification into a program form for which a compiler or interpreter exists. The automation tools are commonly known as generator programs, or generators for short. The input of a generator is a program description and its output is a program in a programming language or in some other form that can be translated into machine code. The target of a generator might for example be intermediate code as generated by the front-end of a compiler. This intermediate code is then translated into machine code by the code generator part of a compiler [. The idea of a generator was first proposed for compilers in the form of a compiler-compiler. Reusability and automation can be very effective when applied to the design of a family of systems that have a large part in common. Examples of such families are database management systems, compilers and programming environments. The kind of facilities which members of a system family typically have in common are the facilities for database or file management, for input/output and for maintaining the user interface.

Conclusion:

The main objective of software engineering is to help produce high

quality software systems within reasonable bounds of time and cost. The major factors that determine the quality of a software product in addition to its desired functionality are reliability, performance, flexibility and friendliness of the user interface. Software engineering is right now facing the challenge of solving the serious problems encountered in the software production process which lead to cost and time overruns and products that are lacking in many of the quality factors. Tools most frequently used for improving the software production process are program measurement and software development support tools. Measurement tools are helpful in finding the bottlenecks in the existing software production methodology. Support tools are helpful in alleviating the task of the people involved the production process. Both kinds of tools help to make the production process more effective and more reliable. The original design of isolated support tools is gradually being replaced by integrated programming environments that behave more as intelligent assistants than as toolboxes. Measurements and support tools are designed to correct flaws in an existing methodology, but do not address the more fundamental question of methodology itself, There is a general feeling that current practices are inadequate (and will become more so in the near future) to satisfy the growing demand for reliable software that is produced on time and within budget. The basic flaw.,~ of the current process are its labor intensive approach to project. It seems that a significant improvement can be achieved if we can produce reusable software and automate the generation of new software. Success in the area of reusability may reduce the production of new software to a fraction of what is commonly written today, while automation has the potential of simplifying the production process with an additional gain in reliability. Reusability has no chance of being successful unless taken into account as a major design objective from the start. A major obstacle to overcome is the problem of information dissemination. With current software production practices, it is extremely difficult to find out what is available and how things work. Time is ripe for a major effort to define the concept of reusable software precisely and to develop techniques for creating reusable software.

References :

- Reference Manual for the Ada Programming Language. United States Department of Defense, January 1983. [Ah77] Aho, A. V. and J. Ullman. Principles of Compiler Design. Addison Wesley, 1977.
- Barbacci, M. R.; A. N. Habermann; M. Shaw. The Software Engineering Institute: Bridging Practice and Potential. IEEE Software [2], 6. November 1975.
- Brooker, R. A. and D. Morris, A General Translation Program for Phrase Structure Languages. Journal of the ACM 9, pp. 1-10, 1962.
- Buxton, J. N. Requirements for Ada Programming Support Environments (Stoneman). US Government, Department of Defense, February 1980.

■

About the Author

Ms. V. Vetriselvi, (CSI-N1285695) Asst. Professor, Department of MCA, Shrimati Indira Gandhi College, Trichy has completed M.Sc. in Computer Science from Bharathidasan University, Trichy; M.Phil and is SET Qualified. She has got 26 years experience as a faculty in Arts and Science Colleges. She is a Question Paper Setter for Periyar University, Salem and Bharthiyar University, Coimbatore. She has published a Book Titled "Visual Basic 5.0"

Congratulations!

Brig. S V S Chowdhry (Retd) was recently conferred the prestigious Bharat Gaurav Award by the India International Friendship Society. The Award was presented to him at an Awards Ceremony at New Delhi on 21st December 2016 by Dr Bhisham Narain Singh, former Governor of Tamil Nadu and Assam.

Brig Chowdhry was President of the CSI during 1992-94. He was also President of the IETE during 1994-96. He has been honoured with the Lifetime Achievement Awards by the CSI and the IETE.

CSI express hearty congratulation for the same

CSI Regional Student Convention

Chittaranjan Pradhan
Assistant Professor, School of Computer Engineering, KIIT University, Bhubaneswar

KIIT CSI Student Branch, KIIT University, Bhubaneswar, Odisha organized one Regional Student Convention under CSI banner in region IV during 18-19 February 2017. The theme of the convention was on IoT and Information Security. Mr. Sanjay Mohapatra, Vice President, CSI, Prof. Brojo Kishore Mishra, Regional Student Coordinator, CSI Region-IV, Prof. Samaresh Mishra, Program Chair, Prof. Bhabani Shankar Prasad Mishra, Organizing Chair and Mr. Amit Keshri, Senior Architect, Infosys share the dais during the inauguration ceremony of this convention.

During these two days of the convention, around 200 participants & volunteers were assembled. The following events were organized:

Event 1: Panel Discussion

All the guests were discussing regarding the success of CSI Bhubaneswar chapter and the events to be organized under this chapter in the coming days.

Event 2: Keynote Address

Mr. Amit Keshri, Senior Architect, Infosys, Bhubaneswar has delivered his talk titled, "IOT – Internet of Things: Smart Connected Products". He has discussed the current issues of IoT and discussed the solution approach by Infosys Ltd. Prof. Bhabani Shankar Prasad Mishra managed this event.

Event 3: Keynote Address

Prof. Sudhakar Sahu, Asst. Professor, IMA Bhubaneswar has delivered his talk titled, "Application of Crypto System". He has discussed the security issues in the current era and some solutions they are working at IMA, Bhubaneswar. Prof. Bhabani Shankar Prasad Mishra managed this event.

1st

2nd

3rd

Event 5: ICT Quiz

Students have participated in the Quiz competition based on ICT. Aditya Agrawal, Manabhanjan Pradhan & Mayank Agrawal secured 1st, 2nd & 3rd positions respectively. Prof. Nachiketa Tarasia has managed this event.

1st

2nd

3rd

Event 6: Technical Poster Presentation

Students have participated and show case the posters on the theme "Digital India". Shweta Patwari & Kumari Vandana stood the 1st position. Aprajita Singh and Gopal Jana secured the joint 2nd position and shared the prize money. Manabhanjan Pradhan & Swapnil Singh and Rohini Seth & Primula Mukherjee secured the joint 3rd position and shared the prize money. Prof. Chittaranjan Pradhan has manged this event.

1st

Joint 2nd

Joint 2nd

Joint 3rd

Joint 3rd

Joint 3rd

Pre-Convention Tutorial on Data Science

Prof. R.Nadarajan

Session 1

Introduction to Data Science

Speaker: **Karthik Ramasubramanian**:

Data Science – Three Pillars of Data Science (Hacking Skills, Math & Statistical Knowledge and Substantive experience) – Data Structure & Machine Learning – Big Data – Visualization – Cloud – Internet of Things – Statistics – Enlighten the facts and various fields where the Data Science is applied. Educated the problem solving mechanism using data science and predefined algorithms.

Case Study on BMI Calculation

Speaker: **Abishek Singh**

Challenges in insurance – Process of getting life insurance – Prediction of BMI.

Statistics

Data descriptive: Structured, Semi Structured, Quasi structured, Unstructured data.

Machine Learning (ML): ML is an algorithm that can learn this relationship without relying on any rule-bases programming. ML will emphasis on how the final predictions will look like if similar data is supplied in future.

Statistical Learning: Statistical modeling will estimate the relationship based on formal quantification from statistical inferences. The process of statistical inference quantifies the process by which data is generated.

Session 2

Speaker: **Abishek Singh**

Machine Learning Algorithm – K-means, Decision Tree & Neutral Network.

Visualization – GGPlots in R – Boxplot, Histogram, Scatterplot, Sankey Plot, Cohort Charts, Bubble Chart. GGviz – Motion Charts.

Ensembles: Bias vs Variance trade off - Bagging & Booting.

Session 3

Speaker: **Karthik Ramasubramanian**:

Big Data – Ecosystem –

Graph Databases - Graph databases support a very flexible and fine-grained data Model. RDBMS provides results, Graph Databases provides answers.

Session 4

Speaker: Abishek Singh

BMI Calculation -

$$\frac{Weight\ (kg)}{height(m)^2}$$

or

$$\mathbf{BMI} = \frac{weight(lb)x\ 703}{height(in)^2}$$

Image Processing – Face Detection – Facial key Points detection.

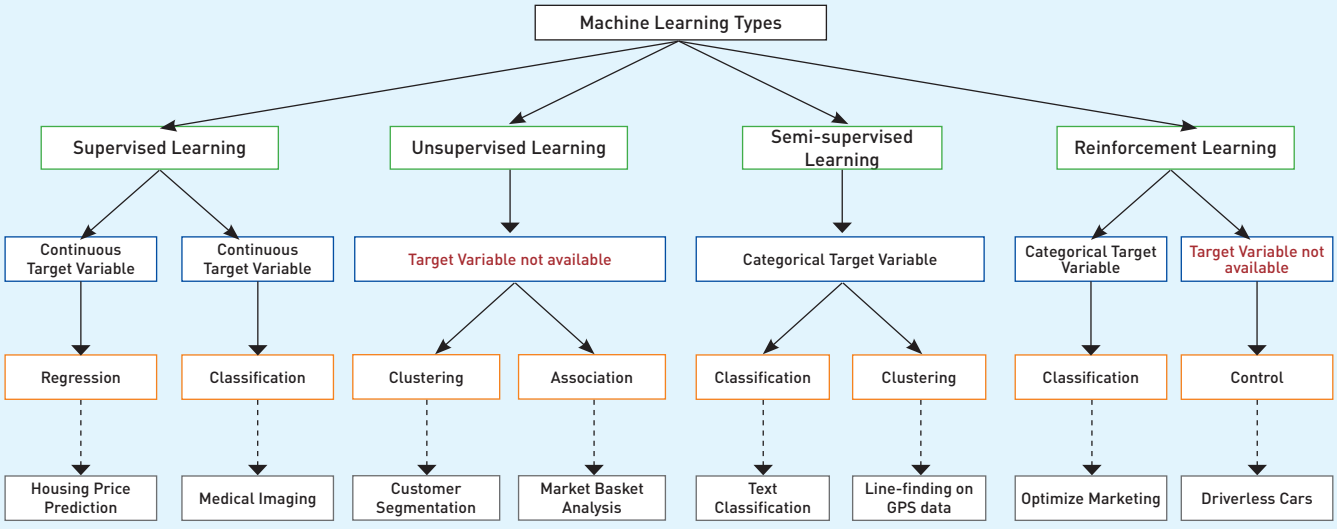
Use case: Predict Emotion, Recognition – verification.

Industry – Academia Collaboration: Fake news detection, Persona identification, Computer vision & Emotion Detection.

Future of Data Science: Deep Learning – Internet of Things – Online Machine Learning (Mobile Devices & Real –Time Performance.

Talent Graph: Technology Stack - Neo4j, Titan, Python, Cloud APIs & MEAN technology stack

Machine Learning Probabilistic Graph Models, Regression Analysis, Natural Language Processing(NLP).



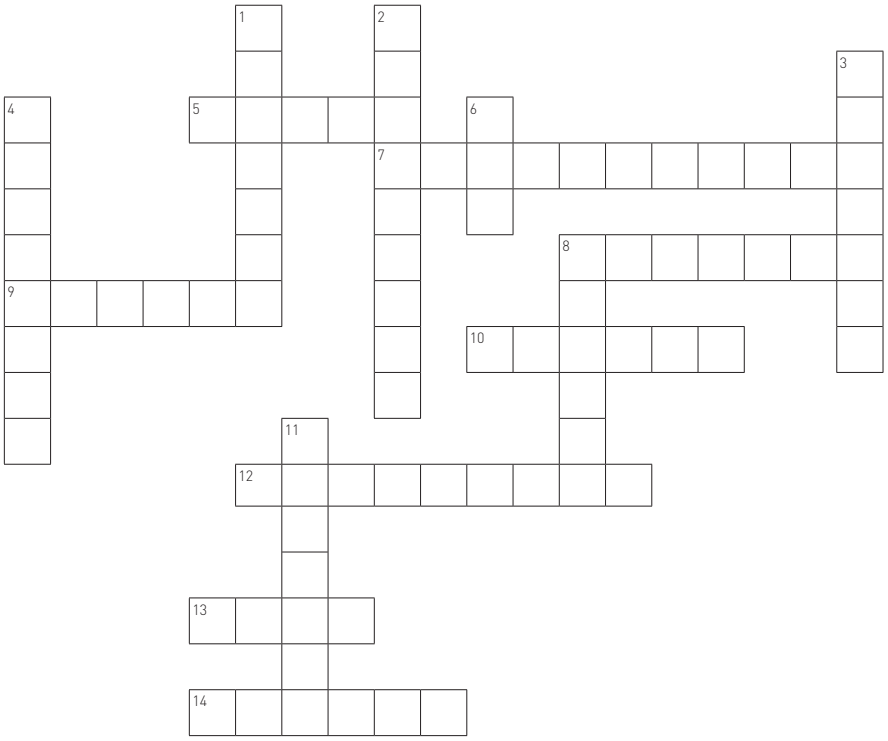
Contd. on next page

CrossWord »

Durgesh Kumar Mishra
Chairman CSI Division IV Communications
Professor (CSE) and Director Microsoft Innovation Center, Sri Aurobindo Institute of Technology, Indore. Email – drdurgeshmishra@gmail.com

Test your knowledge on Software Engineering (Open Source Software)

Solution to the crossword with name of first all correct solution provider(s) will appear in the next issue. Send your answer to CSI Communications at email address csic@csi-india.org and cc to drdurgeshmishra@gmail.com with subject: Crossword Solution – CSIC March 2017 Issue.



CLUES

ACROSS

5. A project acronym for open source software
7. A repository of thousands of open source projects
8. A Linux-based Open source platform for mobile
9. A virtual learning system
10. A tool for automated website hosting
12. Quality control expert of an open source sofware community
13. An open source mail transfer agent
14. Collection of tools which allows running UNIX Applications on Windows Operating System.

DOWN

1. An integrated development environment
2. An open source distributed database system
3. A 3 D graphics and animation package
4. To give a limited version free and charge for premium version
6. An error or flaw in computer program
8. A Freely-available web server
11. Developing a new program from some open source code

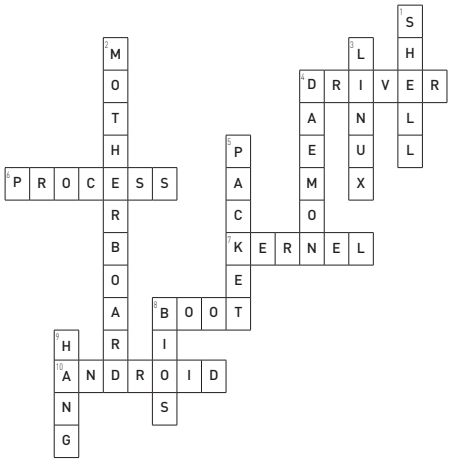
We are overwhelmed by the response and solutions received from our enthusiastic readers

Congratulations!

All Nearby Correct answers to February 2017 month's crossword received from the following reader:

- **Dr. Sandhya Arora,** Professor, Cummins college of Engineering for women, Pune
- **N.Saiteja,** B.Tech 3rd year, CSE, Gokaraju rangaraju Inst. of Engg. and Technology
- **Valli Rajasekhar,** Chitrada, IRS, ONGC
- **Mr. Deepu Benson,** Amal Jyothi College of Engineering, Kerala
- **Chandra Dasaka,** CSI Hyderabad Chapter
- **Prof. Kirti Patil,** Asst. Professor, MET's BKC Institute Of Engineering, Adgaon, Nashik
- **Vishal Sanghai,** 2nd Year, The LNMIIT, Jaipur
- **Shridhar B. Dandin,** Professor, Department of Computer Science, B K Birla Institute of Engineering and Technology BKBIET Campus, PILANI

Solution for February 2017 Crossword



Glimpses of Pre-Convention Tutorial on Data Science

AHMEDABAD CHAPTER

To address the growing cyber crimes and to tackle the internet related frauds, the city's cyber security experts joined hands and organized Logout – 1st ever cyber awareness conference of Gujarat on 22nd January, 2017 in Ahmedabad. The conference was co-organized by the Computer Society of India, ComExpo Cyber Security Pvt Ltd and other cyber security experts of the state where cyber security experts educated attendees with various cyber threats, internet issues and ways to deal with such issues. Logout conference was attended by 650+ students, tech professionals and netizens from around the Gujarat state. Mr. Rakshit Tandon spoke about various cyber crimes and modes operandies used by cyber criminals and educated audience with online safety tips and cyber laws of India. Col Vineet Kumar spoke about tactics used by hackers to commit phone hacking. He also gave an interactive presentation and explained models used by cyber criminals to commit online banking and ATM related frauds. Jayesh Solanki – Chairman Computer Society of India, Ahmedabad Chapter introduced the audience with the awareness campaigns and other work being carried by CSI. He insisted on the needs of such more campaigns by Public-Private partnership to raise the cyber awareness in the state and the country. Ms. Ruzan Khambatta – A social-entrepreneur activist working for women's safety talked about various initiatives that are being carried out by NGOs, Police department and government department to tackle women's safety issues. She highlighted one such activity

AMRAVATI CHAPTER

Dr. Shirish S Sane, Regional Vice President, Region-VI, visited Amravati on 13th February 2017. Dr. G R Bamnote, Chairman CSI Amravati Chapter, Dr. V S Gulhane, Vice Chairman CSI Amravati Chapter, Dr. M A Pund, Hon Secretary CSI Amravati Chapter welcomed him. A meeting was conducted to discuss the issues related with CSI. Dr. Sane guided the Chapter on the issues related with the conduction of various programme and other issues in the betterment of the society and student members. He also appealed the members to update their profile on CSI website and also to increase the number of life members.

BHOPAL CHAPTER

A one week faculty development program on Big Data& Hadoop was organized by Department of IT and School of Information Technology of University Institute of Technology, Bhopal from 31st January 2017 to 4th February 2017 in association with CSI Bhopal Chapter. The expert talks were delivered by eminent speakers like Dr. Vijay Kumar (JNU New Delhi), Dr. Bhaskar Biswas (IIT BHU), Dr. Aruna Tiwari (IIT Indore), Dr. Parag Kulkarni, Pune, Dr. P K Chande (Former Professor, IIM Indore), Dr. Ruchir Gupta (IIITDM Jabalpur), Dr. Syan Ranu (IIT Delhi). Different topics related to Big Data such as machine learning, social networking, trajectories, big data analysis, etc were covered by speakers during the program. The lab sessions were conducted by Mr. Abhishak Gaur in which he demonstrated the twitter analysis, map reduce program etc. The program was Co-ordinated by Dr. Roopam Gupta, Dr. Anjana Pandey and Dr. Mahesh Pawar. More than 70 faculty and Research scholars from affiliated colleges of RGPV, Bhopal have participated in the program and the program was very well appreciated by the participants. The program ended with a valedictory function and Certificate Distribution by the Director of UIT, RGPV Bhopal.

NOIDA CHAPTER

Noida Chapter has organized a one day National Workshop on the topic Information Security, Cyber Forensic & Big Data in the campus of Monad University, Ghaziabad, UP on 26th November 2016. Prof. Ram Chandra, Hon VC, Monad University has given the welcome address and elaborated the aims & objectives. Prof. Manohar Lal has deeply explained the use of computer education in India & its use for protecting the cyber crime in the country. Inaugural session was addressed by the following eminent speakers. Chief Guest Prof. K K Aggarwal in his inaugural address has said that UID is the biggest data base in the world and

it has reversed the old statement. Invention is the mother of necessity in the present internet age. He has also said that we should protect our debit card, credit card & bank credentials by using strong password in the present age of cashless economy. Mr. V K Shukla Add. Advocate UP Government has said that the use of internet has become very important for communication in offices & courts. Cyber crime has also increased in the recent times. Internet has encouraged the field of education, business and knowledge, but at the same time it has also taught new techniques to cyber criminals to hack the sensitive data. Mr. Anuj Agrawal has given hands on training on memory forensic & data forensic. He has also given tips to secure the login id & password from cyber criminals. He has also given various tips to protect from cyber criminals in case of theft of crime. The workshop was coordinated by Dr. Payal Bhardwaj, & vote of thanks was given by Dr. R C Tripathi, Organizing Secretary & Chairman CSI Noida Chapter

TIRUCHIRAPALLI CHAPTER

Tiruchirappalli Chapter conducted the guest lectures on Security in Cloud on 10th January 2017. Speaker for this program was Er. K Vignesh, Assistant Professor, MAM College of Engineering, Tiruchirappalli.

Tiruchirappalli Chapter conducted the guest lectures on Information System and the Entropy on 14th February 2017. Er Parlgyan Singh, Systems Engineer, ICT, BHEL, Trichy was the speaker for the event.

VADODARA CHAPTER

Chapter celebrates The Computer Day on 22nd January 2017. The event was organized in collaboration with Department of Computer Science and Engineering, The M S University of Baroda. Technical Fiesta comprised of total 16 competitions, giving opportunity to students of every age i.e. from class 1st in schools to final year of the University. The highest number of participation was in Programming in C competition, with 45 participants from colleges, and from schools it was 51 teams. The topics for various competitions were IT and socio-IT related theme. The topic of some of the School competitions are pre-announced i.e. "Digital India" for Web Designing, "Internet of Things" for PPT Presentations "Demonitization / Cashless Transactions" for Collage Making, "Clean India, Green India" for Poster Making. Three Prizes for each competition were awarded. Total of 63 prizes were distributed for all the competitions, amounting to ₹ 38,000/- which were covered from Participation Fees and Sponsorship. Several Senior CSI Committee Members and CSI Professionals graced the occasion

VELLORE CHAPTER

Chapter in association with ACM Student Chapter organized a one day Guest Lecturer on Marching Towards Web Wisdom on 28th January 2017 at VIT University. Mr. Sabapathy, Vice President, Technical Operations, Nine Stars Info Technologies Pvt Ltd, Bangalore explained about web data, characteristics of web data and different issues of web data. Discussion about how machine learning techniques can be applied to web data context. He explained the research directions in web data. 80 members participated in the guest lecture. Organized by Prof. H R Viswakarma, Prof. K Govinda, RVP VII.

FROM CHAPTERS & DIVISIONS

Chapter in association SITE, VIT University organized one day Guest Lecturer on Software Analytics on 15th February 2017 at VIT University. Dr. Anjaneyulu Pasala, Senior Research Scientist, Infosys, Bangalore explained introduction to analytical software tools and how these tolls can be used in academia for research, how machine learning techniques can be applied to web data, around 80 members participated in the guest lecture. Organized by Prof. G Jagadeesh & Prof. K Govinda, RVP VII.

FROM STUDENT BRANCHES

REGION-I

The NorthCap University, Gurgaon

7-2-2017 - Awareness Program on competitions and opportunities

The NorthCap University, Gurgaon

9-2-2017 - Event on gaming

REGION-I

Vivekananda Institute of Professional Studies, Delhi

21-1-2017 – Prof. Hoda, Mr. Vyas, Mr. Saurabh Agarwal, Prof. Rattan Sharma, Mr. Jasmeet Sethi and Prof. Vinay Kumar during State Student Convention on Digital India

REGION-III

School of Computer Studies, Ahmedabad

3 & 4-2-2017 - CSI State Level Student Convention (Gujarat) on the them Digital India

REGION-III	REGION-V
Mody University of Science & Technology, Lakshmangarh	**Vasavi College of Engineering (Autonomous), Hyderabad**
	
4-2-2017 to 7-2-2017 - Four day workshop on Cloud Computing	25-1-2017 – Contest on Crossword Puzzle
REGION-V	
NMAM Institute of Technology, Nitte	**Scient Institute of Technology, Hyderabad**
	
28-1-2017 – Workshop on User Centered Design	17 & 18-2-2017 - Workshop on Python Programming
PVP Siddhartha Institute of Technology, Vijayawada	**Geethanjali Institute of Science & Technology, Nellore**
	
4-2-2017 – CSI Student Branch Inauguration	11-2-2017 – One day workshop on MongoDB
Stanley College of Engineering & Technology for Women, Hyderabad	
	
17-2-2017 - Guest Lecture on What is Engineering	20 to 22-2-2017 - Certification on Programming in C

REGION-V

Adikavi Nannaya University, Rajahmundry

14-2-2017 – Prof. M Mutyalu, VC inaugurated the Student Branch

Chalapathi Institute of Engineering and Technology, Guntur

11 & 12-2-2017 - Second National Conference on Recent Advances in Computer Science & Engineering

CMR Technical Campus, Hyderabad

9-2-2017 - Seminar on Study abroad Awareness

16-2-2017 - Guest Lecture on Java App. Development with SDLC

NBKR Institute of Science and Technology, Nellore

30 & 31-1-2017 - Two days Workshop on Web Application Development

17-2-2017 - One Day Workshop on Data Science Research

Amrita Vishwa Vidyapeetham (University), Bangalore

3-2-2017 – Distinguished Speaker Program on Autonomous Mobile Robots Developed at CAIR

GITAM University, Bangalore

18-2-2017 – Student Branch Inauguration

REGION-V	REGION-VI
Sasi Institute of Tech. & Engg., Tadepalligudem	**MGM's Jawaharlal Nehru Engg. College, Aurangabad**
	
11-2-2017 – Prof. Raghavendra Rao during Guest Lecture on Recent Trends in Data Mining	28 & 29-1-2017 – Technical Student Convention
REGION-VI	
Shri Ramdeobaba College of Engg. and Mgmt., Nagpur	**Late G N Sapkal College of Engineering Anjaneri, Nashik**
	
18 to 20-1-2017 – Polaris2K17, a National Level Technical Symposium	6-2-2017 – Prof. Wankhade, Mr. Sudhir Kulkarni, Mr. Rajesh & Mr. Kaith during Microsoft Technology Day Celebration
Kavikulguru Institute of Technology and Science, Nagpur	**All India Shri Shivaji Memorial Society's Institute of IT, Pune**
	
23-1-2017 – Motivational and Patent Registration Seminar by Mr. Vijay Mhaske	5 to 18-12-2016 – Two Weeks Industrial Training Program
Guru Gobind Singh Polytechnic, Nashik	
	
4-2-2017 – Mr. Sudhir Gorade during Guest lecture on Personality development	4-2-2017 – Mr. Ravi Bhave during Guest lecture on Database Management System with SQL

Registered with Registrar of News Papers for India - RNI 31668/1978
Regd. No. MCN/222/2015-2017
Posting Date: 10 & 11 every month. Posted at Patrika Channel Mumbai-I
Date of Publication: 10th of every month

If undelivered return to :
Samruddhi Venture Park, Unit No.3,
4th floor, MIDC, Andheri (E). Mumbai-400 093

REGION-VI

K K Wagh Institute of Engg. Education & Research, Nashik

16-2-2017 - Mr Sanjeev Mishra during Expert Talk on Secure Socket Layer

All India Shri Shivaji Memorial Society's Institute of IT, Pune

7 & 8-2-2017 – Regional Level Student Convention for Region-VI

REGION-VII

S A Engineering College, Chennai

22-2-2017 – Mr Kathiresan is elaborating on the Mission and Benefits of CSI during CSI Awareness Programme

Knowledge Institute of Technology, Salem

17-2-2017 & 18-2-2107 – CSI State Level Student Convention

Valliammai Engineering College, Kattankulathur

10-2-2017 - Seminar on Internet of Things

St Joseph's Institute of Technology, Chennai

2-2-2017 - Young Talent Search in Code Debugging & Poster Design

Kongu Engineering College, Erode

4-2-2017 - Web Designing using HTML and Java Script

Kongu Engineering College, Erode

18-2-2017 – Dr Thangarajan, HOD, CSE inaugurating the Technical symposium ENVISTAS-2K17